Chapter 2

DATA MODELS

- Data Model is the modeling of the data description, data semantics, and consistency constraints of the data.
- It defines the logical design and structure of a database and defines how data will be stored, accessed, and updated in a database management system.
- It is a simple representation of complex real-world data structure.
- It facilitates interaction among the designer, application programmers and the end users.

There are three levels data models used for understanding the structure of the database:

1. Conceptual data model: - It defines WHAT the system contains

**2. Logical data model: -** Defines HOW the system should be implemented regardless of the DBMS.

3. Physical data model: - This Data Model describes HOW the system will be implemented using a specific DBMS system

1. Conceptual Data Model:

- Identifies the highest-level relationship between different entities.
  Features:
  ✓ Includes important entities and the relationship among them.
  ✓ No attributes and primary key are specified.
  ✓ Relationship among entities are abstract too. (That is even relationship details are hidden)
  ✓ No other information is shown through the conceptual data model.
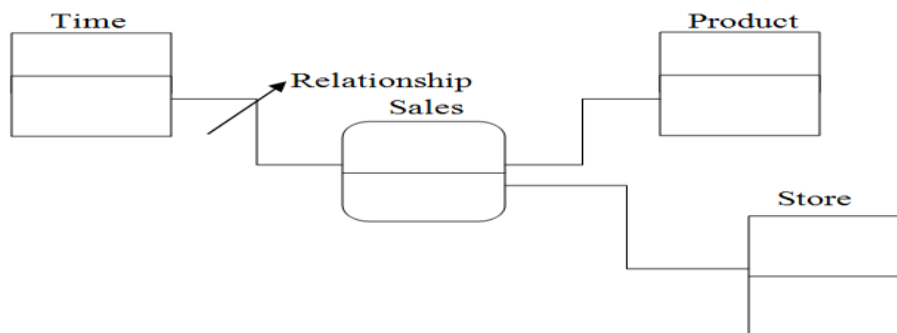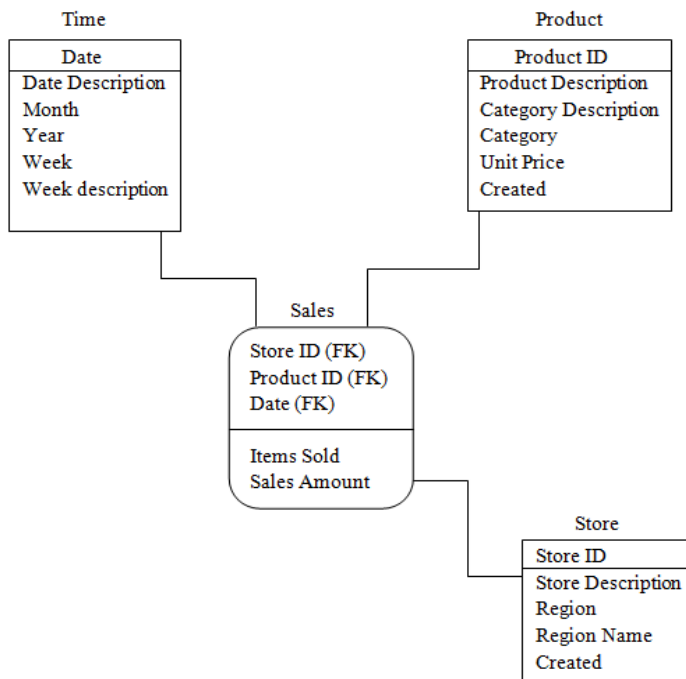
2. Logical Data Model

- A logical data model is an extension of a conceptual model with more details.
- It is independent of technology and technology changes do not impact the model.

Features:

✓ It includes all the entities and relationship among them.
✓ All attributes for each entity are specified.
✓ Primary key and foreign key are also specified.
✓ Normalization takes place.
✓ It's the blueprint that guides and helps in navigating the rapid changes in technology and customer needs.
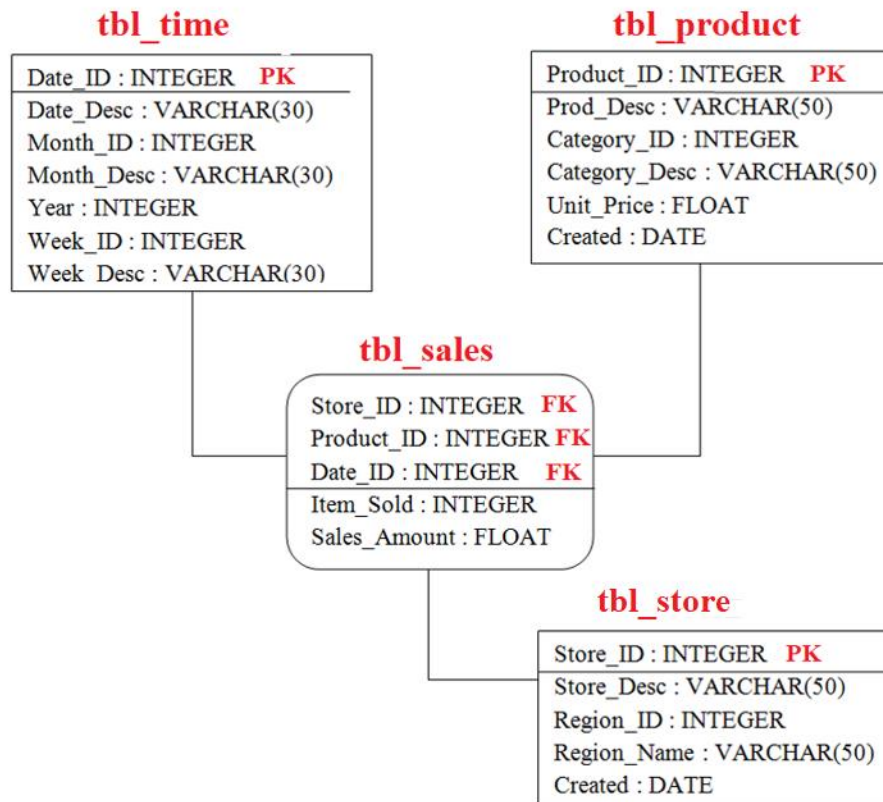✓ The logical model lays the foundation for the physical database.

Time		Product	
Date		**Product ID**	
Date Description		Product Description	
Month		Category Description	
Year		Category	
Week		Unit Price	
Week description		Created	

Sales

Store ID (FK)
Product ID (FK)
Date (FK)

Items Sold
Sales Amount

Store	
Store ID	
Store Description	
Region	
Region Name	
Created	

3. Physical Data Model

✓ Physical Data Model describes a database-specific implementation of the data model.
✓ It represents the how the model will be built in the database.
✓ This model depicts the actual database where the actual data is stored and is accessed in the production environment when users use the application.
✓ It is based on technology and the choice that has been made for the database and the version of the database i.e., ORACLE, mysql etc.

Features:
✓ Show all the table structures, including column name, data type constraints, primary key, foreign key and relationship between tables.

tbl_time

Date_ID : INTEGER	PK
Date_Desc : VARCHAR(30)	
Month_ID : INTEGER	
Month_Desc : VARCHAR(30)	
Year : INTEGER	
Week_ID : INTEGER	
Week_Desc : VARCHAR(30)	

tbl_product

Product_ID : INTEGER	PK
Prod_Desc : VARCHAR(50)	
Category_ID : INTEGER	
Category_Desc : VARCHAR(50)	
Unit_Price : FLOAT	
Created : DATE	

tbl_sales

Store_ID : INTEGER	FK
Product_ID : INTEGER	FK
Date_ID : INTEGER	FK
Item_Sold : INTEGER	
Sales_Amount : FLOAT	

tbl_store

Store_ID : INTEGER	PK
Store_Desc : VARCHAR(50)	
Region_ID : INTEGER	
Region_Name : VARCHAR(50)	
Created : DATE	

Logical Versus Conceptual model

- ✓ In a logical data model, all attributes are specified within an entity. No attributes are specified in a conceptual data model.
- ✓ In a logical data model, primary keys are present, whereas in a conceptual data model, no primary key is present.
- ✓ Relationships between entities are specified using primary keys and foreign keys in a logical data model. In a conceptual data model, the relationships are simply known that two entities are related, but we do not specify what attributes are used for this relationship.

Logical Versus physical model

- ✓ Entity names in Logical model are now table names in Physical model.
- ✓ Similarly, Attributes are now column names.
- ✓ Data type for each column is specified. Data types can be different depending on the actual database being used.

Conceptual vs logical vs physical model

Feature	Conceptual Model	Logical Model	Physical Model
Entity name	✓	✓	
Entity Relationship	✓	✓	
Attributes		✓	
Primary keys		✓	✓
Foreign Keys		✓	✓
Table name			✓
Column name			✓
Column data type			✓

Advantages of data modeling

Structure and Organization: Data modeling helps in structuring and organizing data in a systematic and logical manner. It allows us to define the relationships between various data elements and entities, ensuring that data is stored efficiently and accurately.

Clear Representation: Data modeling provides a clear visual representation of the database structure, including entities, attributes, and their relationships. This visual representation simplifies the understanding of complex data structures and aids in effective communication between stakeholders, developers, and users.

Data Consistency: By defining relationships and constraints, data modeling helps maintain data consistency within the database. It ensures that data is accurate and valid by enforcing rules and constraints, such as data type, primary key, foreign key, uniqueness, and referential integrity.

Efficient Queries and Performance: With data modeling, we can design the database schema in a way that optimizes query performance.

Scalability and Flexibility: Data modeling facilitates scalability and flexibility in the database design. By properly defining entities and relationships, you can accommodate future changes and modifications to the system. It allows you to add new attributes, entities, or relationships without affecting the existing structure, minimizing the impact on the application.
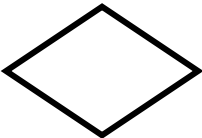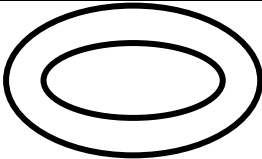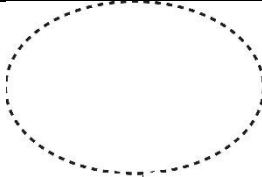
Improved Development Process: Data modeling acts as a blueprint for the development process. It helps developers understand the requirements and design the database schema accordingly. By providing a clear understanding of data dependencies and relationships, it reduces the chances of errors and inconsistencies during development, resulting in a more efficient and effective development process.

Decision Support and Analysis: Data modeling supports decision-making processes and data analysis. With a well-designed data model, you can easily identify and retrieve relevant data, perform complex queries, generate reports, and analyze trends and patterns within the database, leading to better decision-making and insights.
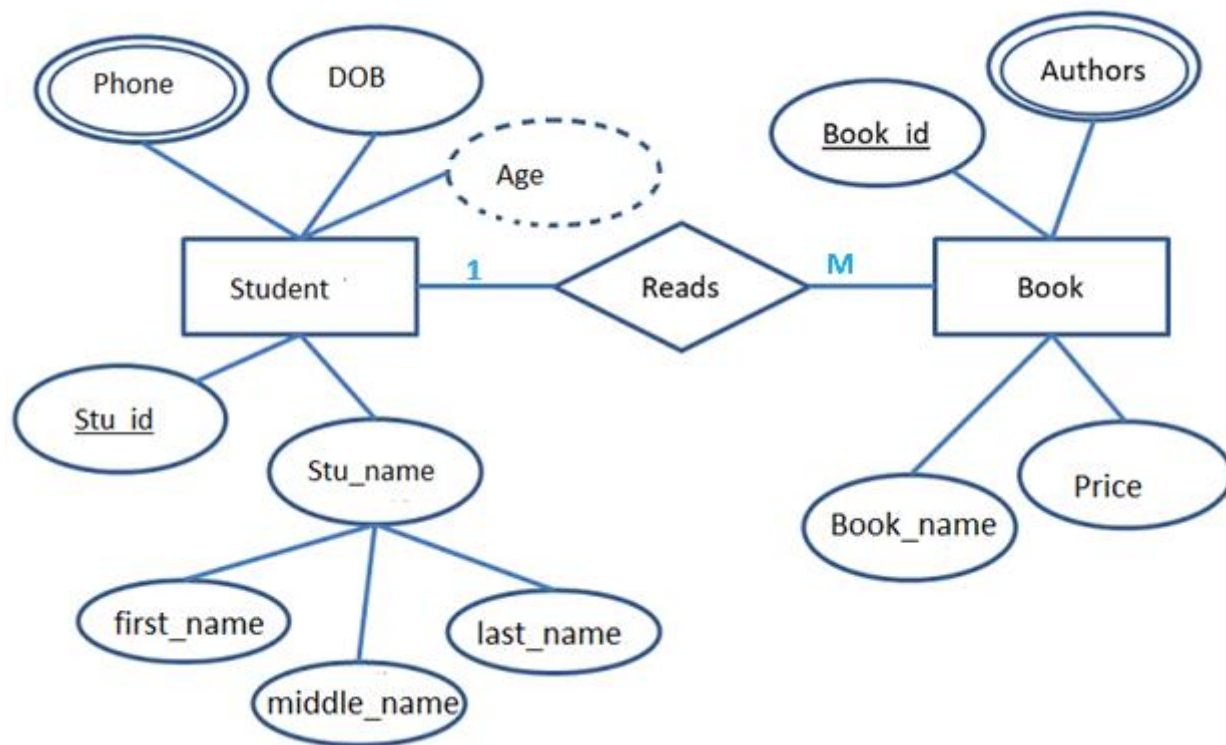
Entity relationship model

- In 1976, Peter Chen developed the Entity-Relationship (ER) model.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- An Entity-Relationship Model represents the structure of the database with the help of a diagram containing entities, relationships among them, and attributes of the entities.
- Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs.
- Creation of an ER diagram, which is one of the first steps in designing a database, helps the designer(s) to understand and to specify the desired components of the database and the relationships among those components.

Symbols used in ER-diagram

Rectangle:-		to represent entity set
Ellipse:-		to represent attributes
Diamonds :-		to represent Relationship among entity set
Lines:-		to Link attributes to entity sets and entity sets to relationship.
Double ellipse:-		To represent multivalued attributes
Dashed Ellipse:-		To represent derived attribute

Double Rectangle:		To represent weak entity set

Example

- ✓ Here ER-diagram shows the relationship named reads in between two entities Student and Book.
- ✓ This is a one-to-many relationship.
- ✓ Student entity have attributes such as key attribute of stu_id, a composite attribute of stud_name with the components such as first_name,middle_name and last_name multivalued attribute of phone and derived attribute of age with its base attribute of DOB.
- ✓ Similarly, the book entity has a key attribute of book _id, a multivalued attribute of authors and other two attributes such as book_name and price.

Basic concept

This model is based on three basic concepts:

1. Entity and Entity set
2. Relationships and Relationship sets
3. Attribute

1. Entity and Entity set

Entity

- An entity is a "thing" or "object" in the real world that is distinguished from all other objects.
  An Entity may be

    - ✓ **concrete** such as person, book, driver, product, employee etc.

    - ✓ it may be **abstract** such as course

- An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.
- For example, employee may have employee_id property whose value uniquely identifies that employee. Thus, the value 677-89-9011 for employee_id would uniquely identify one particular employee in the organization.

Entity Set

- An entity set is a set of entities of the same type that share the same properties.
- **Example:**

    - ✓ The set of all employees of an organization can be defined as the entity set employees.

    - ✓ Similarly, the entity set projects represents the set of all projects undertaken by the organization.

- An entity set is represented by a set of attributes. Possible attributes of the employee's entity set are emp_id, emp_name, address, sex, date_of_birth. Possible attributes of the project entity set are project_id, Project_name.

- For each attribute there is a set of permitted values, called the domain of that attribute, which can be assigned to the attribute.

One instance of the entity set **employee** and **project** are shown as:

Employee

Eid	Name	Sex	Address
E1	Ram	Male	Pokhara
E2	Shyam	Male	Butwal
E3	Sita	Female	Chitwan
E4	Krishna	Male	Kathmandu

Project

Project_id	Project _Name
P1	MIS
P2	Image processing
P3	Linux
P4	OS

2. Attributes

- Attributes are descriptive properties possessed by each member of entity set.
- There exists a specific domain or set of values for each attribute from where the attribute can take its values.
- Eg. The student entity might have attributes like id, name, age program, semester etc.
- In ER diagram attributes are represented by an ellipse.

Attribute

Attribute types

❖ **Simple Attribute:**

✓ Simple attributes are those attributes which cannot be divided further.

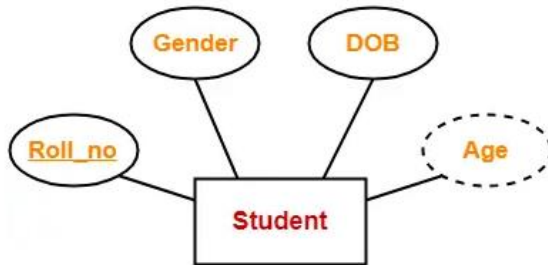✓ Here, all the attributes are simple attributes as they cannot be divided further.

class

Roll_no

Age

Student
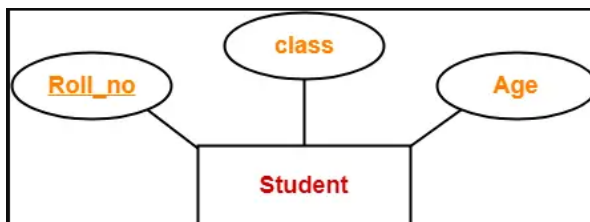
❖ **Composite Attribute:**
✓ Composite attributes are those attributes which are composed of many other simple attributes.
✓ Here, the attributes "Name" and "Address" are composite attributes as they are composed of many other simple attributes.
✓ In ER diagram, composite attribute is represented by an ellipse comprising of ellipse.
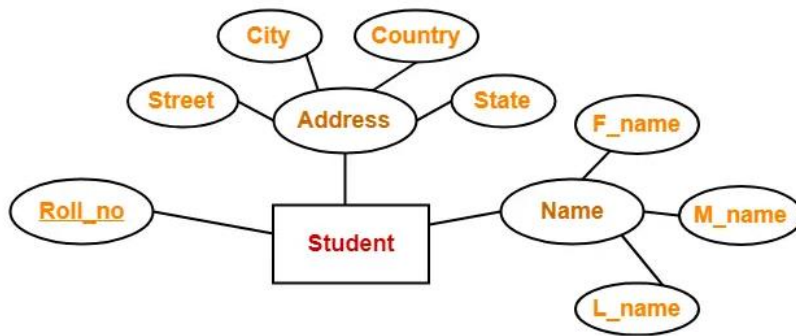
❖ Derived Attribute:

✓ Derived attributes are those attributes which can be derived from other attribute(s).

✓ Here, the attribute "Age" is a derived attribute as it can be derived from the attribute "DOB".

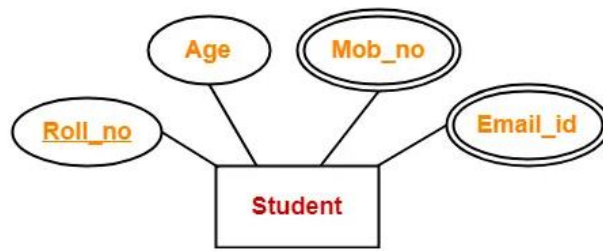✓ In ER diagram, the derived attribute is represented by dashed ellipse.

❖ Single valued Attribute:

✓ Single valued attributes are those attributes which can take only one value for a given entity from an entity set.

✓ Here, all the attributes are single valued attributes as they can take only one specific value for each entity.
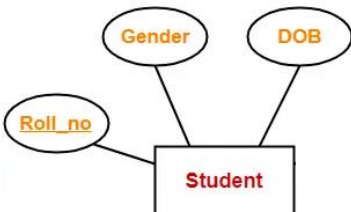
❖ Multivalued attribute

✓ Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.

✓ Here, the attributes "Mob_no" and "Email_id" are multi valued attributes as they can take more than one values for a given entity.

✓ In ER diagram, multivalued attribute is represented by double ellipse.

❖ Key attributes

✓ Key attributes are those attributes which can identify an entity uniquely in an entity set.
✓ Here, the attribute "Roll_no" is a key attribute as it can identify any student uniquely.
✓ In ER diagram, key attribute is represented by an ellipse with underlying lines.
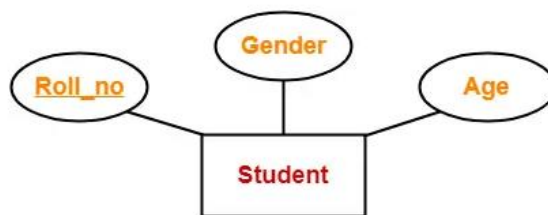
Relationship and relationship sets

- A relationship is an association among several entities.
- It is represented using a diamond shape in the entity-relationship diagram.
  Example: Publisher supplies a book.

  Here In the above example, the publisher and the book are entities whereas the word supplies is a relationship between those entities. Moreover, the attributes of the publisher entity can be that is a pub_id,Pub_name,address and the attributes of book entity can be book_id,book_name.

- A relationship set is a set of relationships of same type. It is a mathematical relation of n>=2 (possibly non distinct) entity sets.
- If E1, E2, E3,...., En are entity sets, then a relationship set R is a subset of {(e1, e2, e3, ..., en) | e1 ∈ E1, e2 ∈ E2, ..., en ∈ En} Where (e1, e2, e3, ..., en) is a relationship.
- The association between entity sets is referred to as participation; that is the entity sets E1,E2,....En participate in a relationship set R.
- A relationship instance in an E-R schema represents an association between entities in the real world that is being modeled.

Consider the two-entity set publisher and book. The relationship set supplies to denote the association between publisher and book that publisher supplies.

Publisher entity set   **Book entity set**

Pub_id	Pub_name	Address
P1	Asmita book suppliers	Newroad
P2	Goodwill	Kalanki
P3	Pariabi prakasan	Biratnagar

Book_id	Book name
B1	Dbms
B2	Operating system
B3	C++
B4	Math

As an illustration, an individual Publisher entity Asmita book suppliers who has pub_id and the Book entity who has Book_id B1 and B2, participate in a relationship instance of supplies. This relationship instance represents that Asmita book suppliers supplies two books i.e DBMS and Operating System.

A relationship may also have attributes called descriptive attributes. Consider a relationship set supplies with entity sets Publisher and Book. We could associate the attribute date with that relationship to specify date when supplier supplies book.
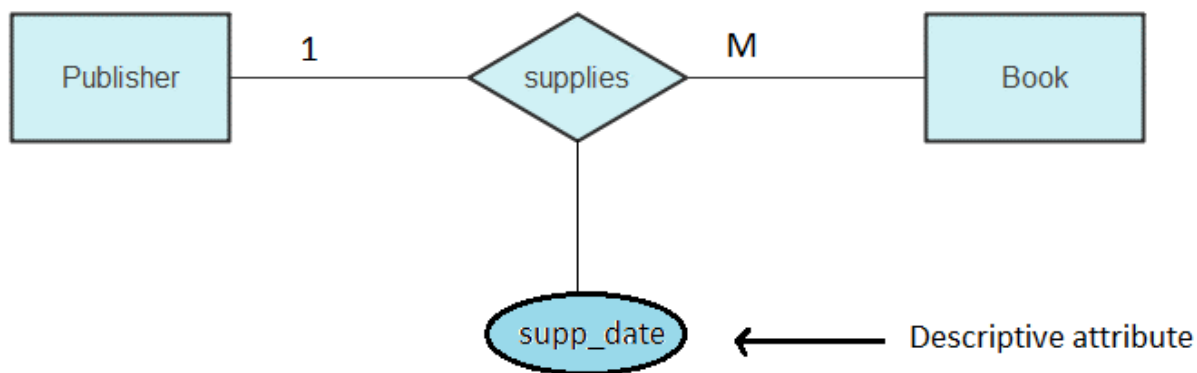

Degree of Relationship

- It refers to the number of entities sets that participate in a relationship.
- Unary relationship is of degree 1.
- Binary relationship is of degree 2.
- Ternary relationship is of degree 3.
- Relationships are often binary.

On the basis of degree of a relationship set, a relationship set can be classified into the following types:
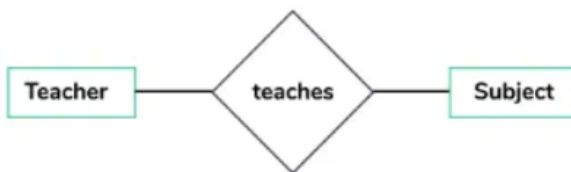
Unary Relationship set

- Unary relationship set is a relationship set where only one entity set participates in a relationship set.
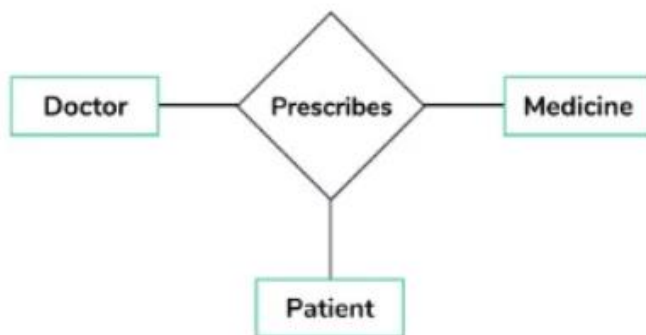
- Eg. One person is married to only one person.
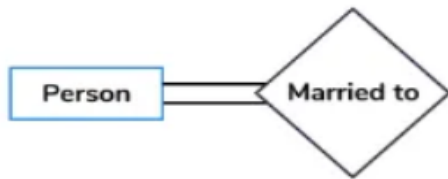
Binary Relationship set

- Binary relationship set is a relationship set where two entity sets participate in a relationship set.

- Eg. Teacher teaches a subject here 2 entities are teacher and subject for the relationship teacher teaches subject

Ternary Relationship set

- Ternary relationship set is a relationship set where three entity sets participate in a relationship set.

- Eg. In the real world, a patient goes to a doctor and doctor prescribes the medicine to the patient, three entities Doctor, patient and medicine are involved in the relationship "prescribes".

N-ary relationship set

- Ternary relationship set is a relationship set where n entity sets participate in a relationship set.

Constraints in E-R models

An ER enterprise schema may define certain constraints to which the contents of a database must conform. We have

i. Mapping cardinalities
ii. Participation constraints

Mapping cardinalities

- Mapping cardinalities, or cardinality ratios, express the number of entities with which another entity can be associated via a relationship set.
- Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

One to One: 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.
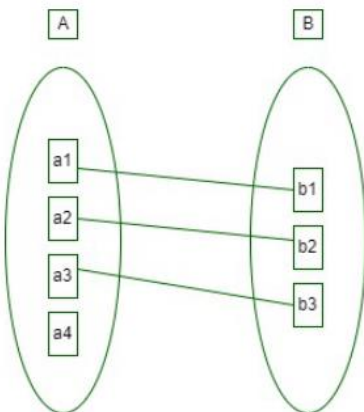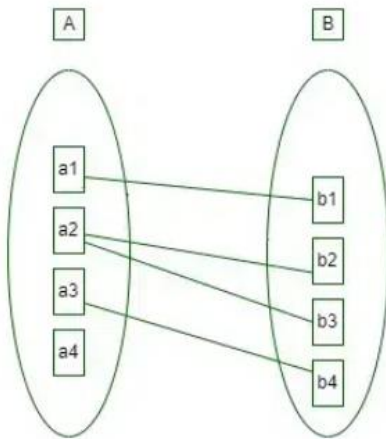

For example, a person has only one passport and a passport is given to one person.
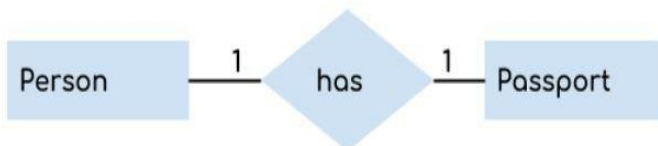
One to many:

An entity in A is associated with any number (zero or more) of entities in B, and an entity in B, however, B can be associated with at most one entity in A.
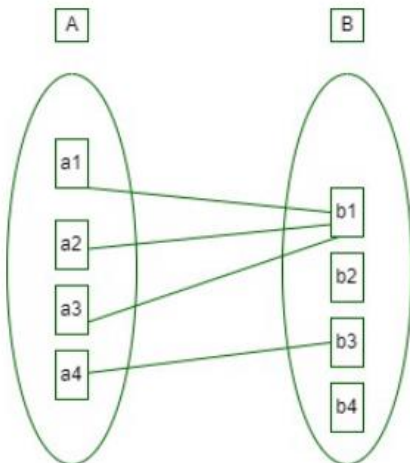

For example, a customer can place many orders but a order cannot be placed by many customers.


Many to One:

An entity in A is associated with at most one entity in B, and an entity in B, however, can be associated with any number (zero or more) of entities in A.

For example, many students can study in a single college, but a student cannot study in many colleges at the same time.
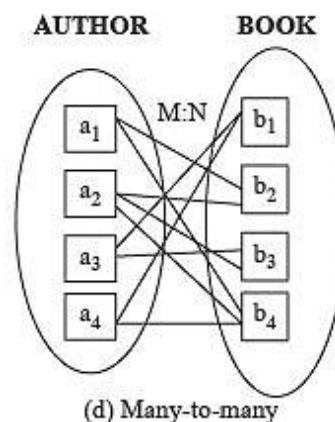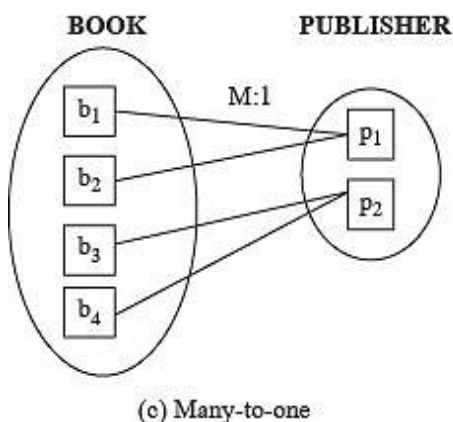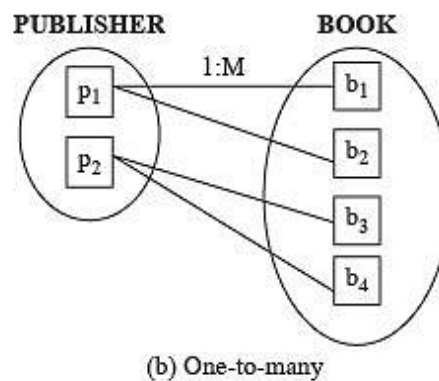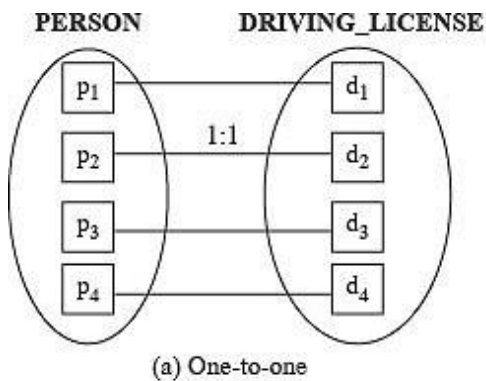
Many to Many: An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.
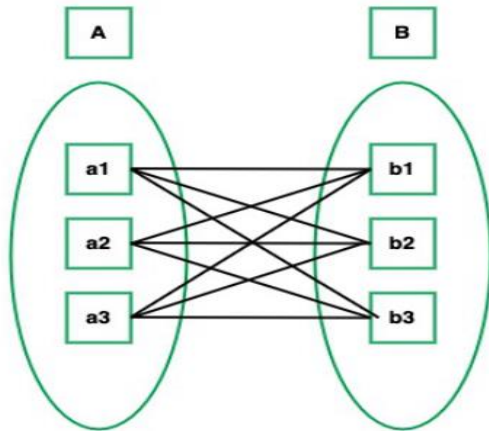
Eg. Student can be assigned to many projects and a project can be assigned to many students.

Alternative Example

(a) One-to-one

(b) One-to-many

(c) Many-to-one

(d) Many-to-many

Participation Constraints

Participation Constraint is applied to the entity participating in the relationship set.

1. Total Participation – Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.
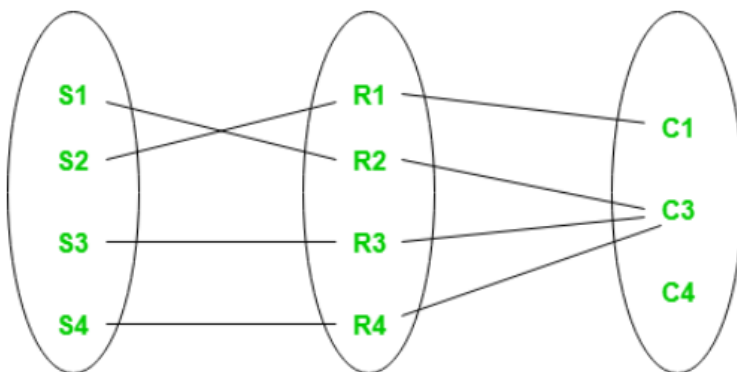
2. Partial Participation – The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

Total Participation and Partial Participation

The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.

Using Set, it can be represented as

*Set representation of Total Participation and Partial Participation*

Every student in the Student Entity set participates in a relationship but there exists a course C4 that is not taking part in the relationship.

Strong and weak entity

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.
- A strong entity set is a set that is made up of many strong entities.
- It may or may not participate in a relationship between entities.

Example:
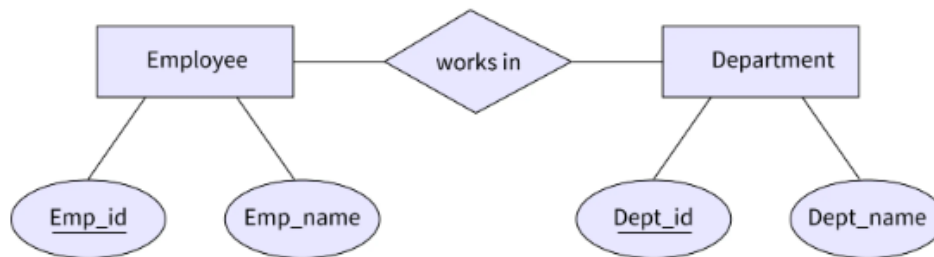
- In the given image, we have two strong entities namely Employee and Department hence they are represented using a single rectangle.
- The relationship between them is works in i.e it gives information about an employee working in a particular department hence it is represented using a single diamond.
- In the above image, if we remove the relationship between the two entities then also the two entities will exist i.e Employee as well as Department will exist since they both are independent of each other, this explains **the independent nature of strong entities.**

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. It instead has a partial discriminator key.
- A weak entity is represented by a double rectangle.
- The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The relation between one strong and one weak entity is represented by a double diamond. This relationship is also known as identifying relationship.
- It always participates in the relationship between entities since its existence is dependent on other strong entities and it always has total participation.
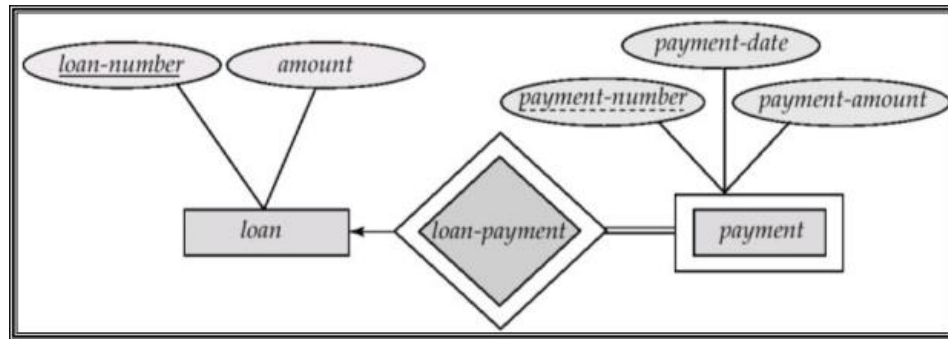
- We underline the discriminator of a weak entity set with a dashed line.
- payment-number – discriminator (partial key)of the payment entity set
- Primary key for payment – (loan-number, payment-number)

Keys

- ✓ Key is a way to specify how entities within a given entity set are distinguished.
- ✓ Conceptually, individual entities are distinct; from a database perspective, however, the difference among them must be expressed in terms of their attributes.
- ✓ Therefore, the values of the attribute values of an entity must be such that they can uniquely identify the entity. In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.
- ✓ A key allows us to identify a set of attributes that is enough to distinguish entities from each other.
- ✓ Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

Role of keys in relational database

- KEYS in DBMS is an attribute or set of attributes which helps us to identify a row(tuple) in a relation(table).
- In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys in RDBMS ensure that you can uniquely identify a table record despite these challenges.
- They allow us to find and establish the relation between two tables.
- Help us to enforce identity and integrity in the relationship.

Let us consider the table named **Employee_info**

Employee_id	Name	Address	Pan_no	Age
1	Pradip	Kathmandu	55821	43
2	Hari	Pokhara	55837	32
3	Nikita	Chitwan	55237	24
4	Pradip	Butwal	55345	43
5	Sita	Lalitpur	55432	48
6	Ramesh	Chitwan	55755	25
7	Hari	Pokhara	55896	32

- Now to fetch any particular record from such dataset, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the wrong data. How many trials before you get the right data?
- To avoid all this, Keys are defined to easily identify any row of data in a table.

Various types of keys are:

1. Super key
2. Candidate key
3. Primary key
4. Alternate key
5. Foreign key
6. Composite key

1. Super key
✓ Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.
✓ In above example following are the examples of super keys

{Employee_id}

{pan_no}

{Employee_id,name}

{pan_no,name}

{Employee_id,address} etc.

2. Candidate key

✓ A set of minimal attributes that can identify each tuple uniquely in the given relation is called a candidate key.
✓ The value of candidate key must always be unique.
✓ It is possible to have multiple candidate keys in a relation.
✓ In above example following are the examples of candidate keys

{Employee_id}

{pan_no}

3. Primary key

✓ The primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.
✓ In the above example {Employee_id} can be chosen are primary key.

4.Alternate key

✓ Candidate keys that are left unimplemented or unused after implementing the primary key are called as alternate keys
✓ In this example {pan_no} act as alternate keys.

5. Foreign key

✓ Foreign keys are the column of the table used to point to the primary key of another table.
✓ Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee_info table. That's why we link these two tables through the primary key of one table.
✓ We add the primary key of the DEPARTMENT table, Department_id, as a new attribute in the employee_info table.
✓ In the EMPLOYEE table, Department_id is the foreign key, and both the tables are related.
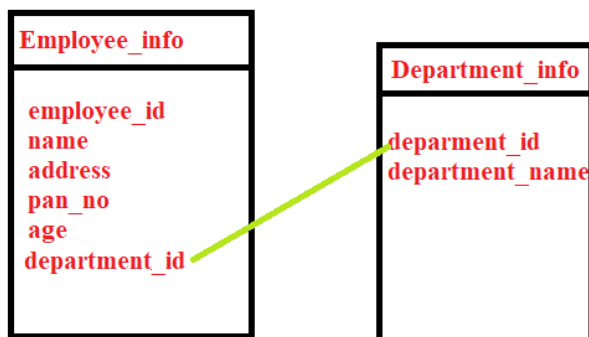

6.Composite key

✓ Key that consists of two or more attributes that uniquely identify any record in a table is called Composite key. But the attributes which together form the Composite key are not a key independently or individually.
✓ In the above picture we have a Score table which stores the marks scored by a student in a particular subject.
✓ In this table student_id and subject_id together will form the composite key
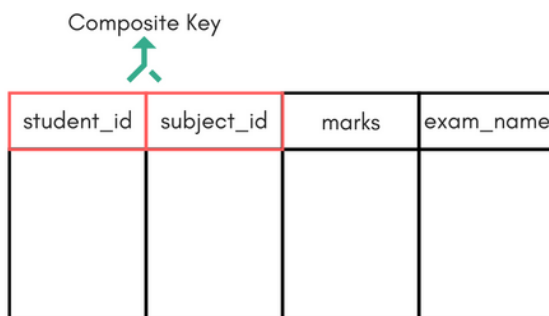
Score Table – To save scores of the student for various subjects.

Roles in E-R diagram

- It is a function that plays(relationship) in an entity called its role.
- Roles are normally explicit and not specified.
- Useful when the relationship meaning needs to be clarified.
- Example, the relationship works-for define in employees as manager or worker. The labels "manager" and "worker" are called roles; they specify how employee entities interact via the works-for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional and are used to clarify the semantics of the relationship.
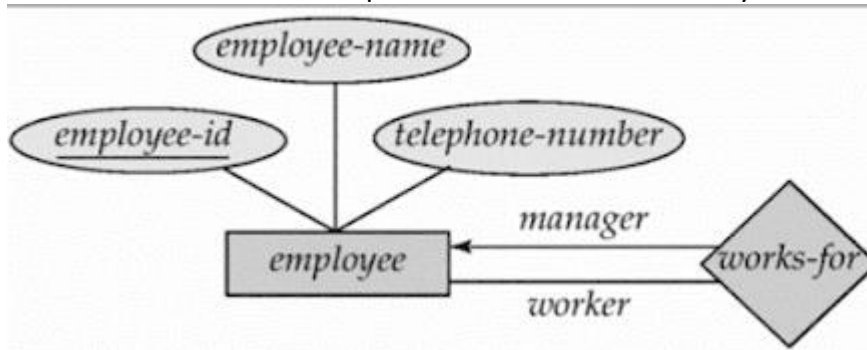
EER(Extended / Enhanced ER model)

- Today the complexity of the data is increasing so it becomes more and more difficult to use the traditional ER model for database modeling.
- To reduce this complexity of modeling we must make improvements or enhancements to the existing ER model to make it able to handle the complex application in a better way.
- Enhanced entity-relationship diagrams are advanced database diagrams very similar to regular ER diagrams which represent the requirements and complexities of complex databases.
- It is a diagrammatic technique for displaying the Sub Class and Super Class; Specialization and Generalization; Aggregation etc.
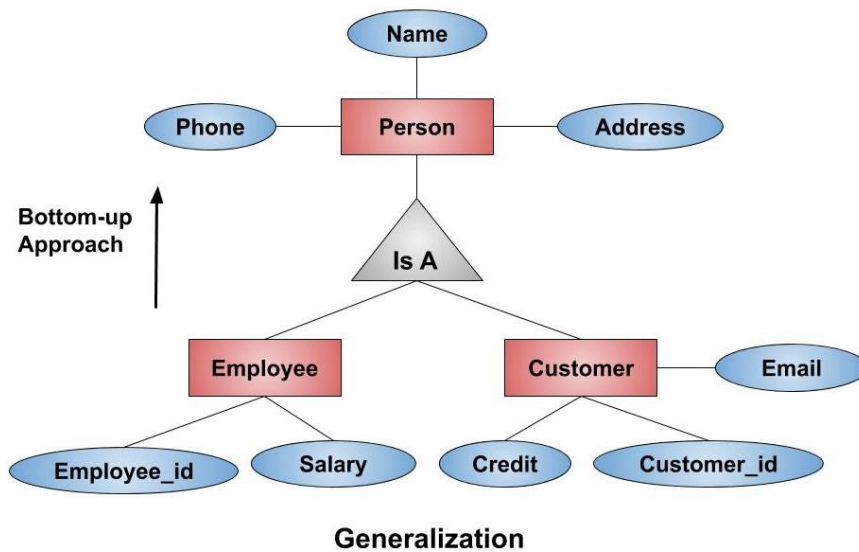
Specialization

- The process of designating sub grouping within an entity set is called specialization.
- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a triangle component labeled ISA (E.g. customer "is a" person).
- Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Specialization
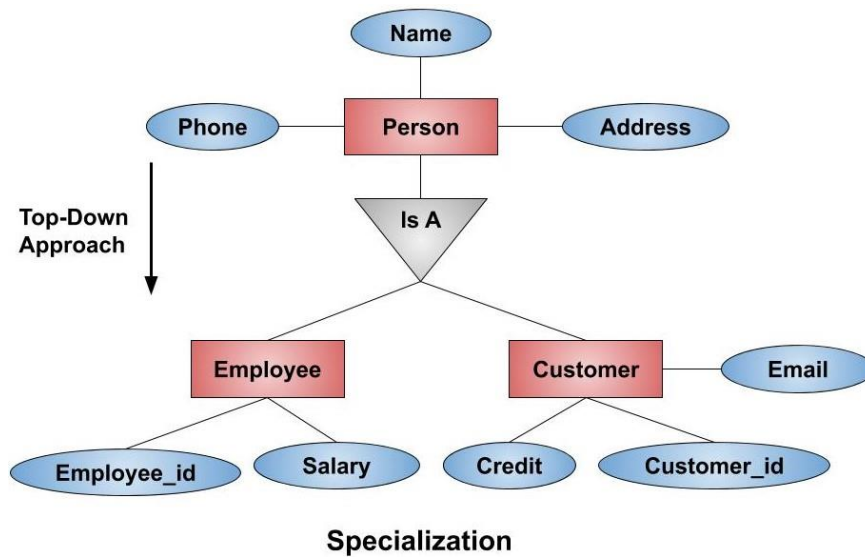
Generalization

- A bottom-up design process – combine a number of entities sets that share the same features into a higher-level entity set.
- Generalization is a containment relationship that exists between a higher level entity set and one or more lower level entity sets.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.


Generalization

Generalization vs specialization

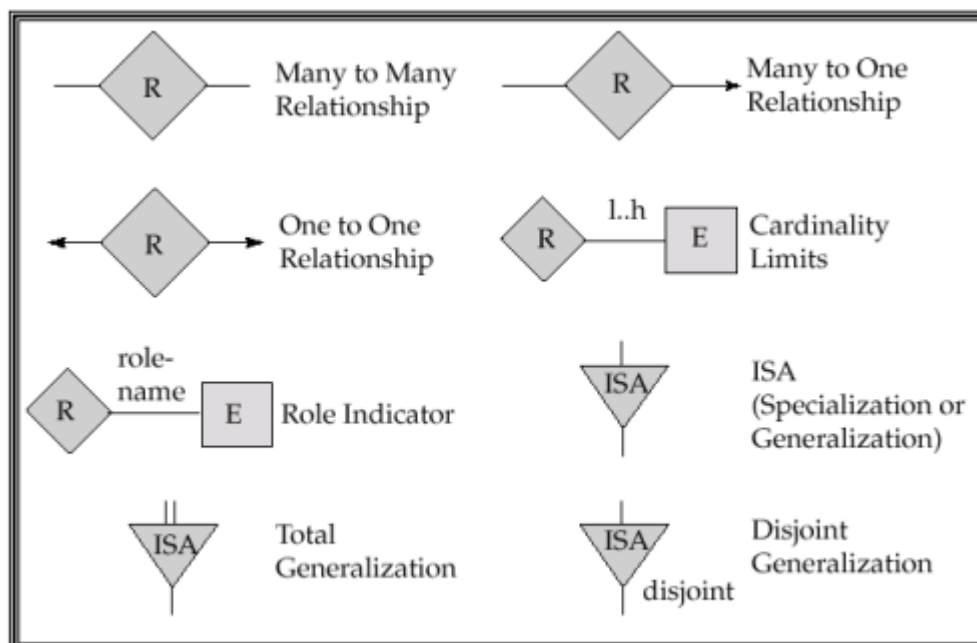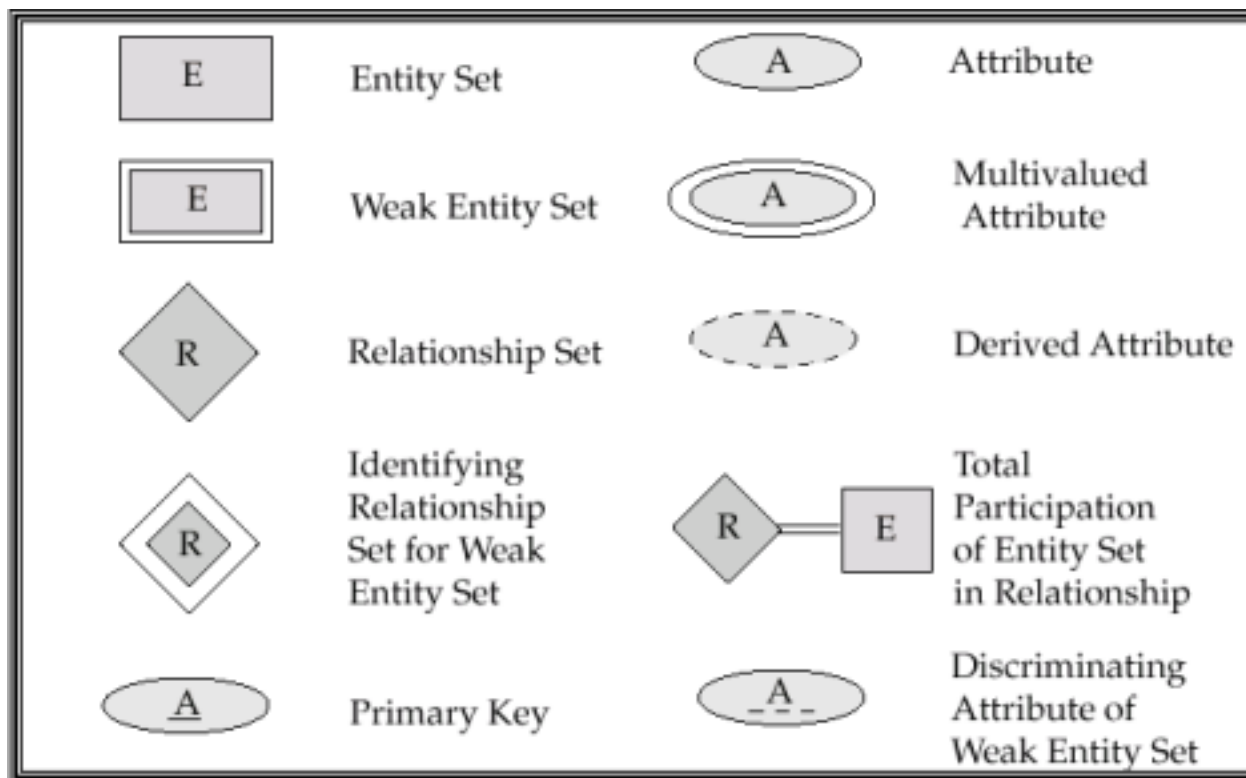| Generalization	Specialization
Generalization works in Bottom-Up approach.	Specialization works in top-down approach.
Generalization can be defined as a process of creating groupings from various entity sets	Specialization can be defined as process of creating subgrouping within an entity set
In Generalization process, what actually happens is that it takes the union of two or more lower-level entity sets to produce a higher-level entity sets.	Specialization is the reverse of Generalization. Specialization is a process of taking a subset of a higher-level entity set to form a lower-level entity set
Generalization process starts with the number of entity sets and it creates high-level entity with the help of some common features	Specialization process starts from a single entity set and it creates a different entity set by using some different features.
Generalization is normally applied to group of entities.	We can apply Specialization to a single entity.
In Generalization, size of schema gets reduced.	In Specialization, size of schema gets increased.

Aggregation

Aggregation in DBMS (Database Management System) is a process of combining one or more entities into a single one that is the more meaningful entity. Also, it is a process in which a single entity does not make sense in a relationship, and one cannot infer results from that relationship so the relationship of one or more entities acts as one entity.
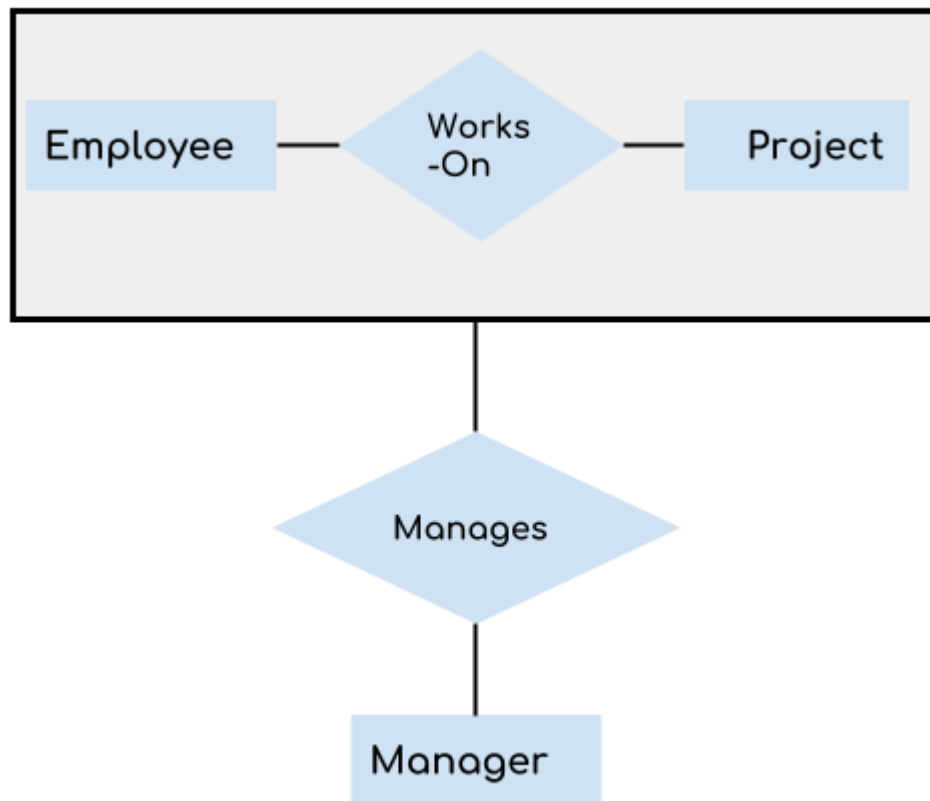
In the real world, we know that a manager not only manages the employee working under them, but he has to manage the project as well. In such scenario if entity "Manager" makes a "manages" relationship with either "Employee" or "Project" entity alone then it will not make any sense because he has to manage both. In these cases, the relationship of two entities acts as one entity. In our example, the relationship "Works-On" between "Employee" & "Project" acts as one entity that has a relationship "Manages" with the entity "Manager".

Summary of Symbols Used in E-R Notation

E	Entity Set	A	Attribute
E	Weak Entity Set	A	Multivalued Attribute
R	Relationship Set	A	Derived Attribute
R	Identifying Relationship Set for Weak Entity Set	R—E	Total Participation of Entity Set in Relationship
A	Primary Key	A	Discriminating Attribute of Weak Entity Set

|			
R	Many to Many Relationship	R	Many to One Relationship
R	One to One Relationship	R—l..h—E	Cardinality Limits
R—role-name—E	Role Indicator	ISA	ISA (Specialization or Generalization)
ISA	Total Generalization	ISA disjoint	Disjoint Generalization

Alternative ER notations
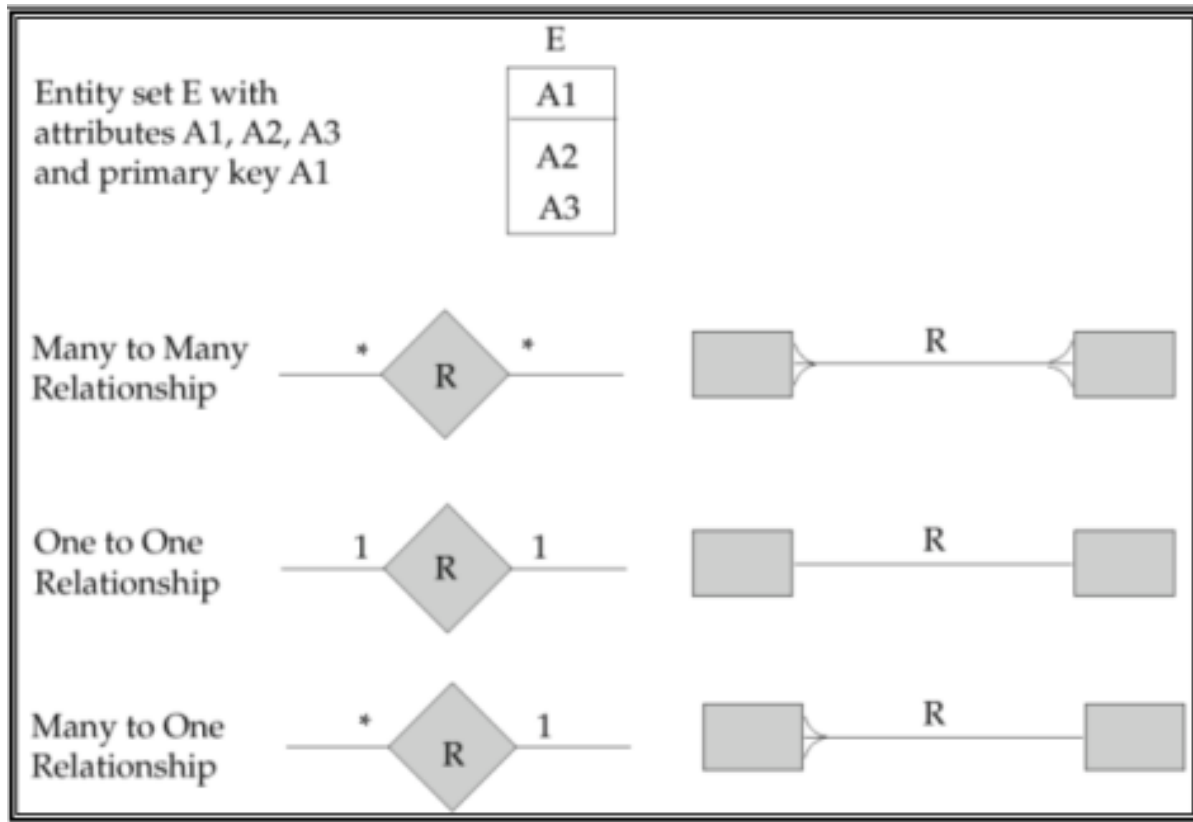
<u>Alternate Data models</u>

Hierarchical Data Models

- It has a root or a parent node and then follows a tree-like structure with other child nodes.
- Only one parent is allowed for a child node, and a parent node can have multiple child nodes.
- The navigation path to a child node is always through a parent node, so is slow and difficult.
- If a parent is deleted, then all child nodes get deleted.
- This model efficiently describes many real-world relationships like index of a book, recipes etc.
- This model could handle one-to-many and one-to-one relations only.
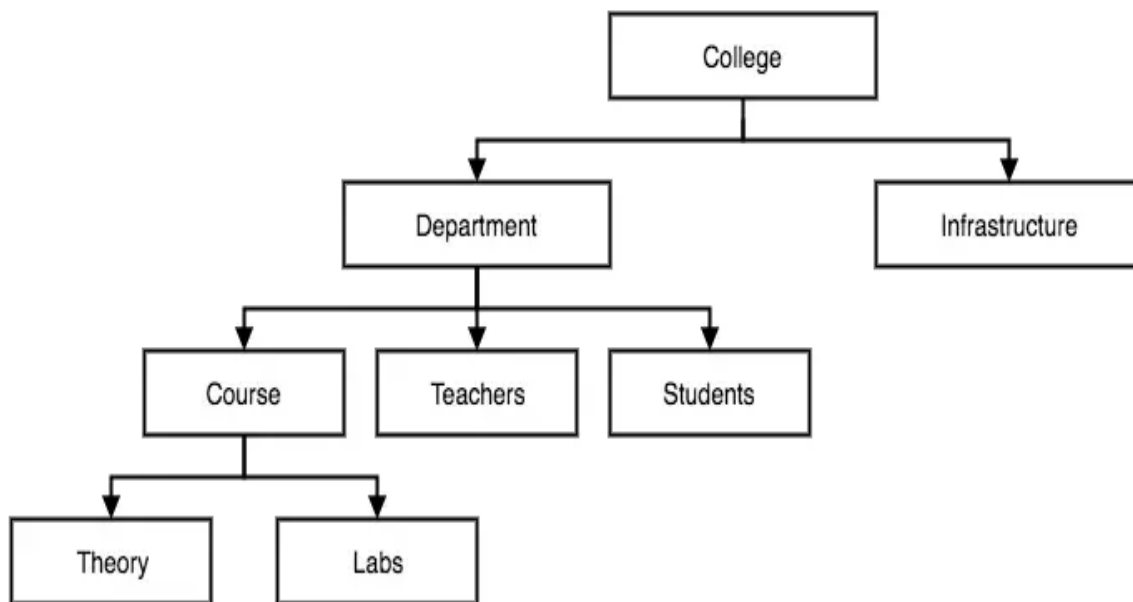
Advantages

- The model allows you to easily add and delete new information.
- Data at the top of the hierarchy can be accessed quickly.
- It supports systems that work through a one-to-many relationship. for example, one department can have many courses, many professors and of-course many students.

Disadvantages

- Difficult to implement
- There is lack of structural independence because when we change the structure then it becomes compulsory to change the application too.

Network data model

- It is a extension of hierarchical model
- Here, a child node could have multiple parent nodes.
- This model could handle many-to-many relations in addition to one-to-one and one-to-many.
- Navigation is faster in this model, as there are multiple paths to reach a child entity.
- This model also gets very complex when data is large.
- The data update is more complex than the hierarchical model
- This was the most widely used database model, before Relational Model was introduced.
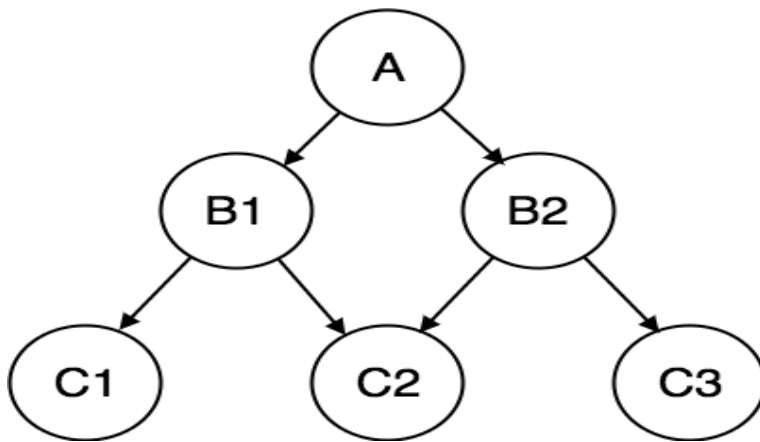

Advantages:

- It supports many to many relationships.
- Simple and easy to design.
- Searching is faster because of multidirectional pointer.

Disadvantages:

- The network model is a very complex database model, so the user must be very familiar with the overall structure of the database.
- Updating the database is a quite difficult and boring task. We need the help of the application programs that are being used to navigate the data.

Draw an ER-diagram for Banking enterprise that keeps the information about the employee, customer, loan, account and payment.
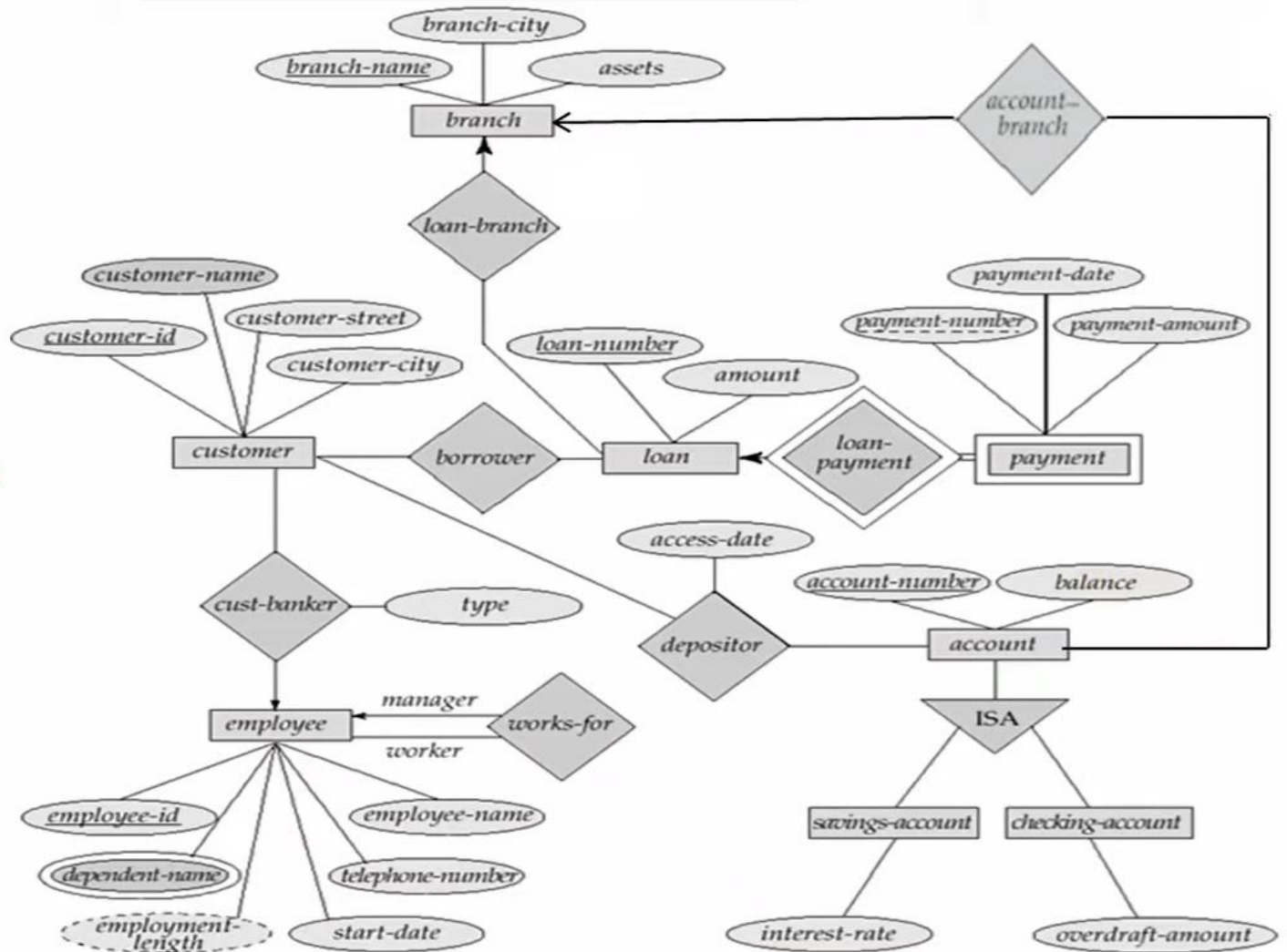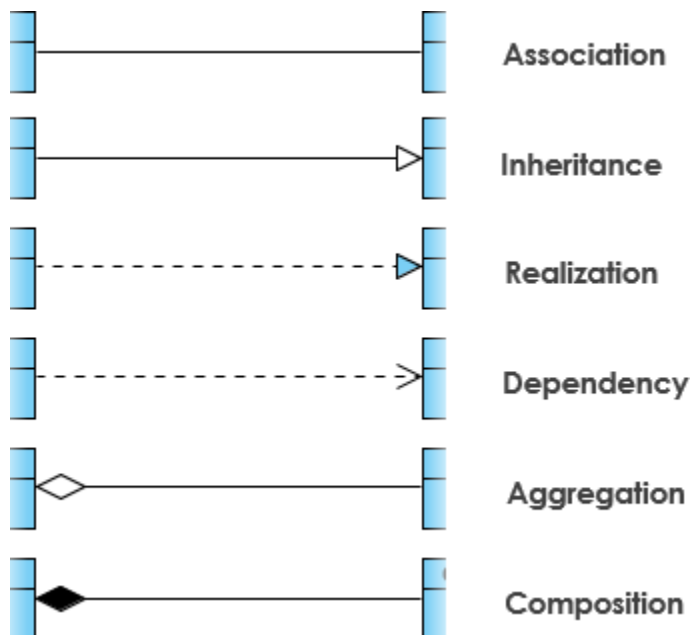
UML Class Diagrams

✓ UML (Unified Modeling Language) is a standardized family of notations for modeling and design of information systems.
✓ It was derived from various existing notations to provide a standard for software engineering.
✓ It comprises of several different diagrams representing different aspect of the system, and one of them being a Class Diagram that can be used for **data modeling.**
✓ Class diagrams are equivalent of ERDs in relational world and are mostly used to design classes in object-oriented programming languages (such as Java or C#).

In class diagrams architects define:

➢ **Class Name** (equivalent of entity in relational world)
➢ **Attributes** (same as in an ERD) including data type

     ✓ The attributes are written along with its visibility factors, which are public (+), private (-), and protected (#)
     ✓ The accessibility of an attribute class is illustrated by the visibility factors.

➢ **Methods** representing its behavior (in relational world those would be stored procedures)
➢ **Relationships:** A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

Association

Inheritance

Realization

Dependency

Aggregation

Composition

✓ We can use class diagrams to design a tabular data (such as in RDBMS), but were designed and are used mostly for object-oriented programs (such as Java or C#).

Example:
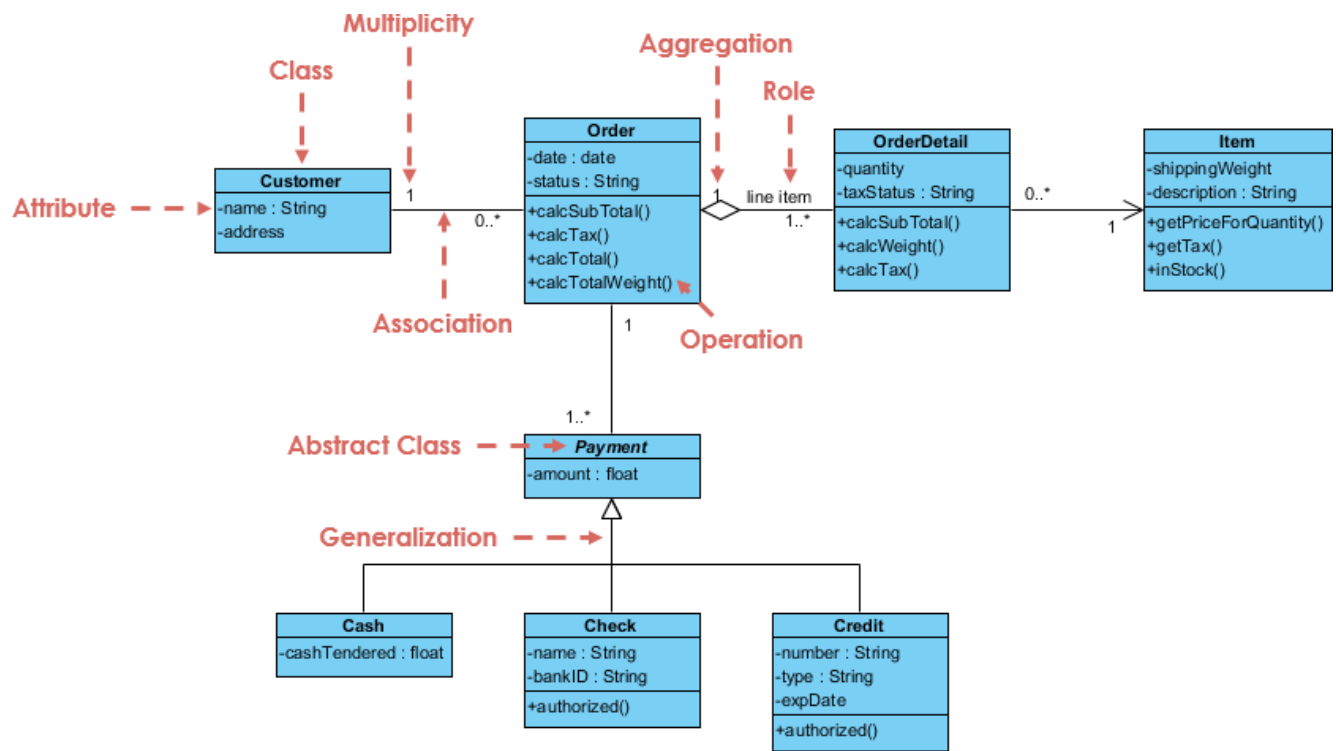
How UML diagram assist in data modeling? [PU: 2019 fall]

UML diagrams can assist in data modeling in a number of ways, including:

- ✓ **Communicating the data model to stakeholders**. UML diagrams are a visual way to represent the data model, which can make it easier for stakeholders to understand. This is especially important for business users who may not be familiar with technical terms or concepts.
- ✓ **Identifying and documenting data requirements**. UML diagrams can be used to identify and document the data requirements for a system. This can help to ensure that the system meets the needs of the users and that the data is properly organized.
- ✓ **Modeling complex data relationships**. UML diagrams can be used to model complex data relationships, such as one-to-many, many-to-many, and self-referential relationships. This can help to ensure that the data is properly organized and that the relationships between the data elements are clear.
- ✓ **Generating code from the data model**. Some UML tools can generate code from the data model. This can save time and effort in the development process.

Here are some specific UML diagrams that can be used for data modeling:

Class diagrams: Class diagrams are used to represent the classes of objects in a system and the relationships between them. They can be used to model the data elements in a database.

Object diagrams: Object diagrams are used to represent the objects in a system at a specific point in time. They can be used to show how the data elements are related to each other.

Sequence diagrams: Sequence diagrams are used to show the interactions between objects in a system over time. They can be used to show how the data elements are used by the objects.

Overall, UML diagrams can be a valuable tool for data modeling. They can help to communicate the data model to stakeholders, identify and document data requirements, model complex data relationships, and generate code from the data model.

References:

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/