

Chapter 2

Data Model

A data model refers to the logical inter-relationships and data flow between different data elements involved. It also documents the way data is stored and retrieved.

Every database and database management system is based on particular database model. Data model consists of set of rules and standards that define how the data is organized in a database. A data model also provides a set of operation for manipulation of the data stored in the database.

Different types of Data Model:

1. Record Based Models

These models specify logical structure of database with records, fields and attributes and are further classified into

1. Physical Data Model
2. Logical Data Model
3. Conceptual Data Model

2. Object Based Models

These models specify logical structure being based on the entity or object. The object based models are further classified into

1. Object Oriented Model
2. E-R Model

1.1 Physical Data Model

The physical data model describes the way how data is stored in the computer by representing records format, record ordering and access path. The physical data model describes the way data are saved on storage media such as disks or tapes. The physical model requires the definition of both the physical storage devices and the access method required to reach the data within the storage device.

1.2 Logical Data Model

These models are used to specify the overall logical structures of database.

Three most widely used record based data model are

- i. Hierarchical data model
- ii. Network data model
- iii. Relational data mode

1.2.1 Hierarchical data model

In hierarchical model, data is organized into a tree like structure with each record is having one parent record and many children. The main drawback of this model is that, it can have only one to many relationships between nodes.

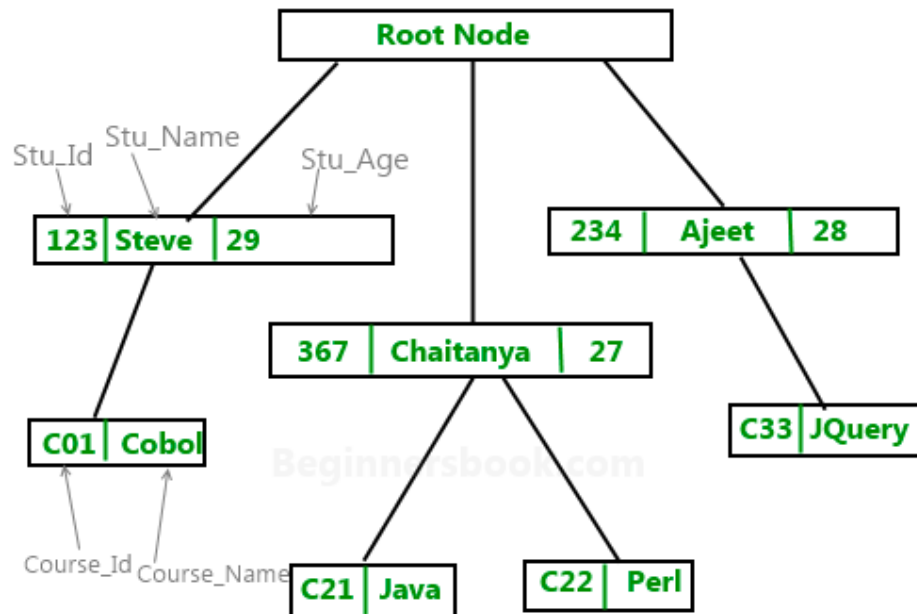


Figure: Structure of Hierarchical Data Model

The above hierarchical model can be represented as relational tables like this:

Student_Table:

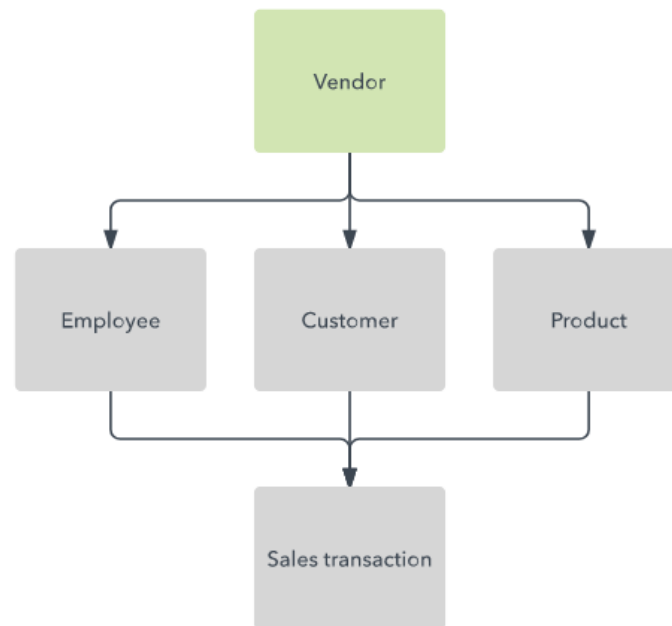
Stu_Id	Stu_Name	Stu_Age
123	Steve	29
367	Chaitanya	27
234	Ajeet	28

Course_Table :

Course_Id	Course_Name
C01	Cobol
C21	Java
C22	Perl
C33	JQuery

1.2.2 Network data model

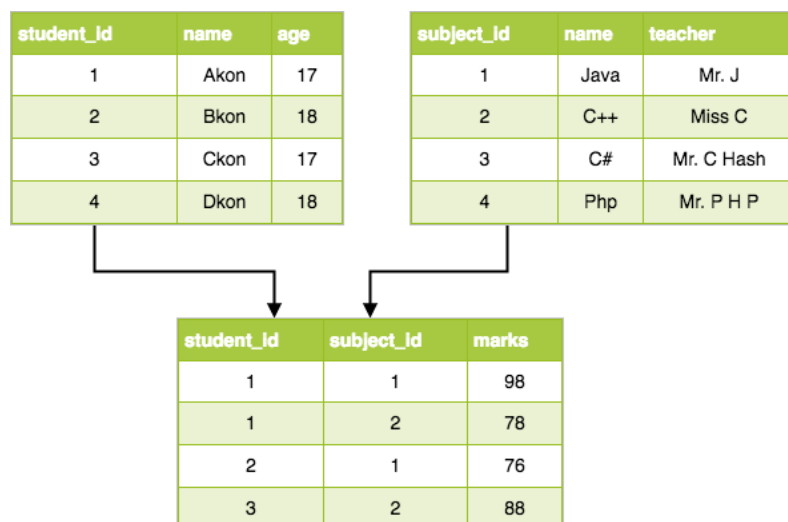
The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records.



1.2.3 Relational Data Model

The most common model, the relational model sorts data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity in question, such as price, zip code, or birth date. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, when it's called a foreign key.

Each row, also called a tuple and the model also accounts for the types of relationships between those tables, including one-to-one, one-to-many, and many-to-many relationships.



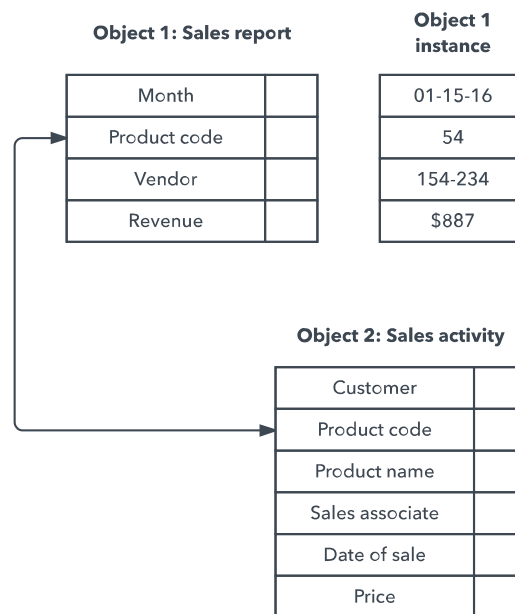
1.3 Conceptual Data Model

These models are used for describing data and data relationship. So, it uses the concept such as entities, attributes, relationships etc.

The most widely used conceptual model is entity relationship (E-R) model. The conceptual model represents the global view of data.

2.1 Object Oriented Model

This model defines a database as a collection of objects, or reusable software elements, with associated features and methods.



2.2 Entity - Relationship Model

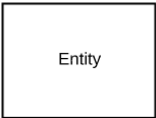
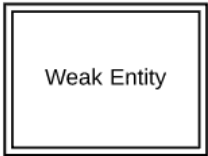
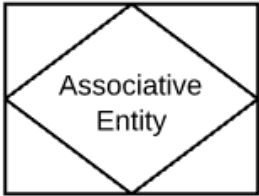
An Entity Relationship (ER) Model or Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.

ER Models use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes.

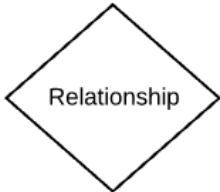

➤ ERD entity symbols

Entities are objects or concepts that represent important data. Entities are typically nouns such as product, customer, location, or promotion.

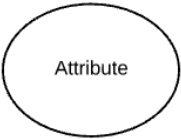

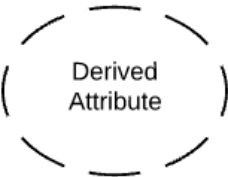
Entity Name	Symbol	Meaning
Strong entity		Strong entities are independent from other entities, and are often called parent entities.
Weak Entity		Weak entities depend on some other entity type. They don't have primary keys, and have no meaning in the diagram without their parent entity.
Associative Entity		Associative entities relate the instances of several entity types.

➤ ERD relationship symbols

Within entity-relationship diagrams, relationships are used to document the interaction between two entities. Relationships are usually verbs and provide useful information that could not be discerned with just the entity types.

Name	Symbol	Description
Relationship		Relationships are associations between or among entities.
Weak relationship		Weak Relationships are connections between a weak entity and its owner.

➤ ERD attribute symbols

Name	Symbol	Description
Attribute		Attributes are characteristics of an entity
Multivalued attribute		Multivalued attributes are those that can take on more than one value
Derived attribute		Derived attributes are attributes whose value can be calculated from related attribute values

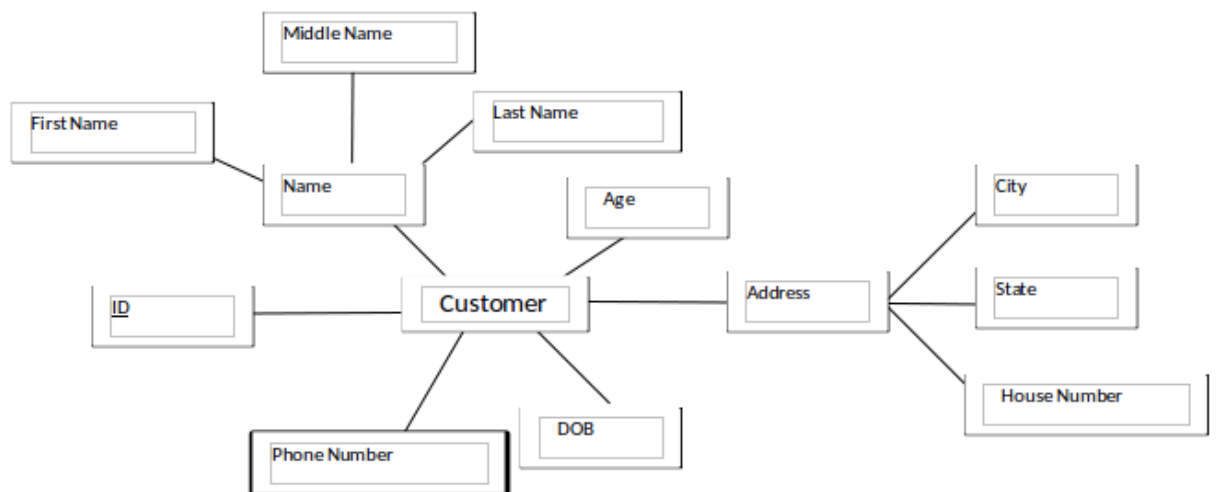


Fig: ERD showing composite, multi-valued and derived attributes

Relationship

A relationship is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table.

Courses		
Course_ID	Course_Name	Teacher_ID
Course_001	Biology	Teacher_001
Course_002	Math	Teacher_001
Course_003	English	Teacher_003

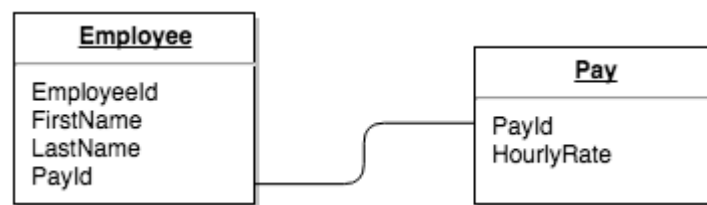
Teachers	
Teacher_ID	Teacher_Name
Teacher_001	Carmen
Teacher_002	Veronica
Teacher_003	Jorge

We can say that the Teacher_ID foreign key has helped to establish a **relationship** between the Courses and the Teachers tables.

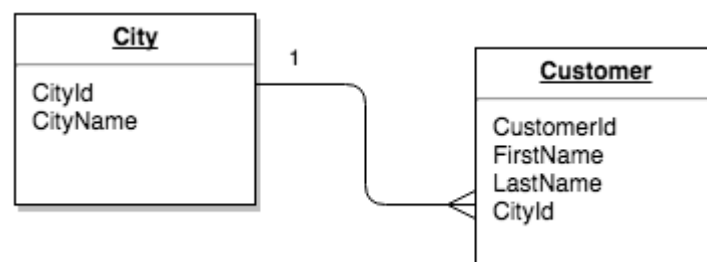
Degree of Relationship(Cardinality)

Cardinality defines the **numerical attributes of the relationship between two entities or entity sets**. The three main cardinal relationships are **one-to-one, one-to-many, and many-many**.

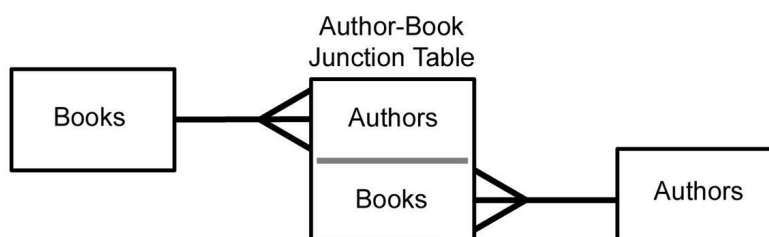
- ✗ A **one-to-one example** would be one student associated with one mailing address.



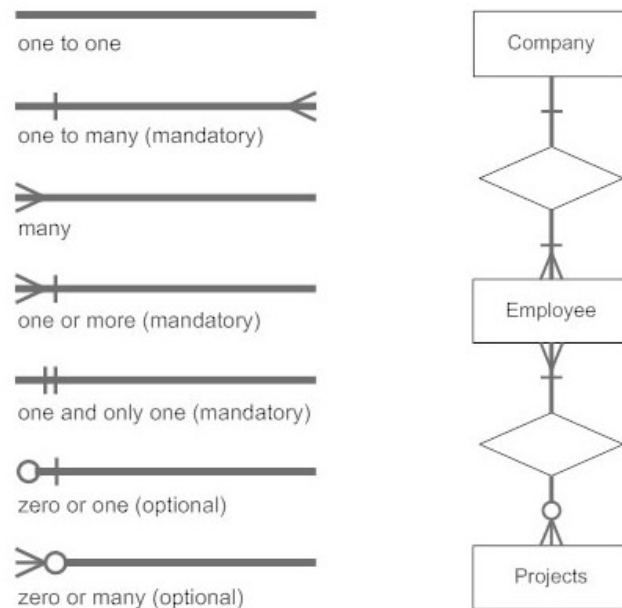
- ✗ A **one-to-many example** (or many-to-one, depending on the relationship direction): One student registers for multiple courses, but all those courses have a single line back to that one student.



- ✗ **Many-to-Many Example:** Students as a group are associated with multiple faculty members, and faculty members in turn are associated with multiple students.



Relationship Symbols and Example



Advantage of ER model

1. Easy and simple to understand with minimal training.
2. It is possible to find connection from one node to all other nodes.
3. It has explicit(dedicated) linkages of entities.

Disadvantages of ER model

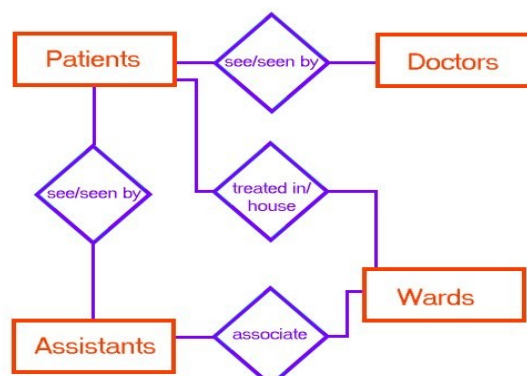
1. Limited relationship representation.
2. No representation of data manipulation.

Examples

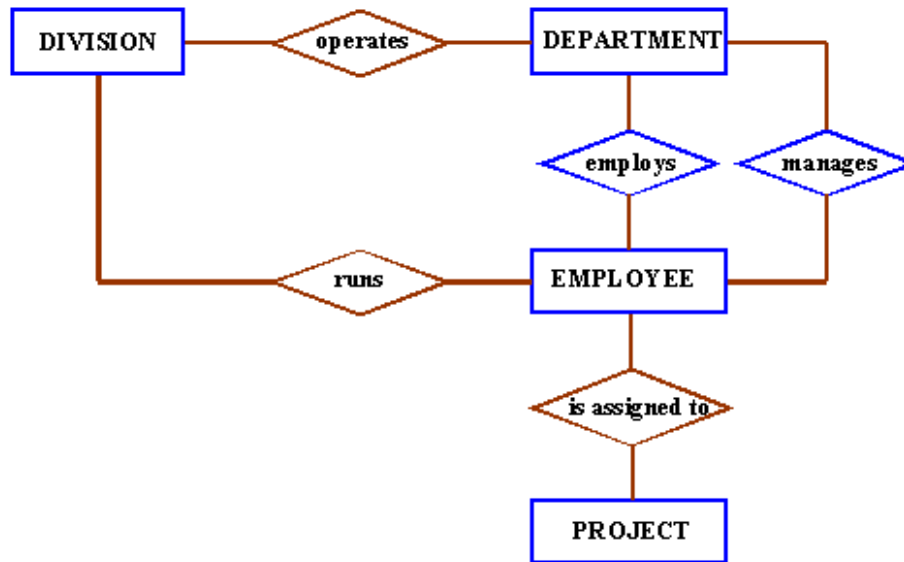
General Idea

- Step 1: Identify the entities
- Step 2: Find the relationship between the entities
- Step 3: Identify the attributes for each entity.
- Step 4: Draw ERD with the information

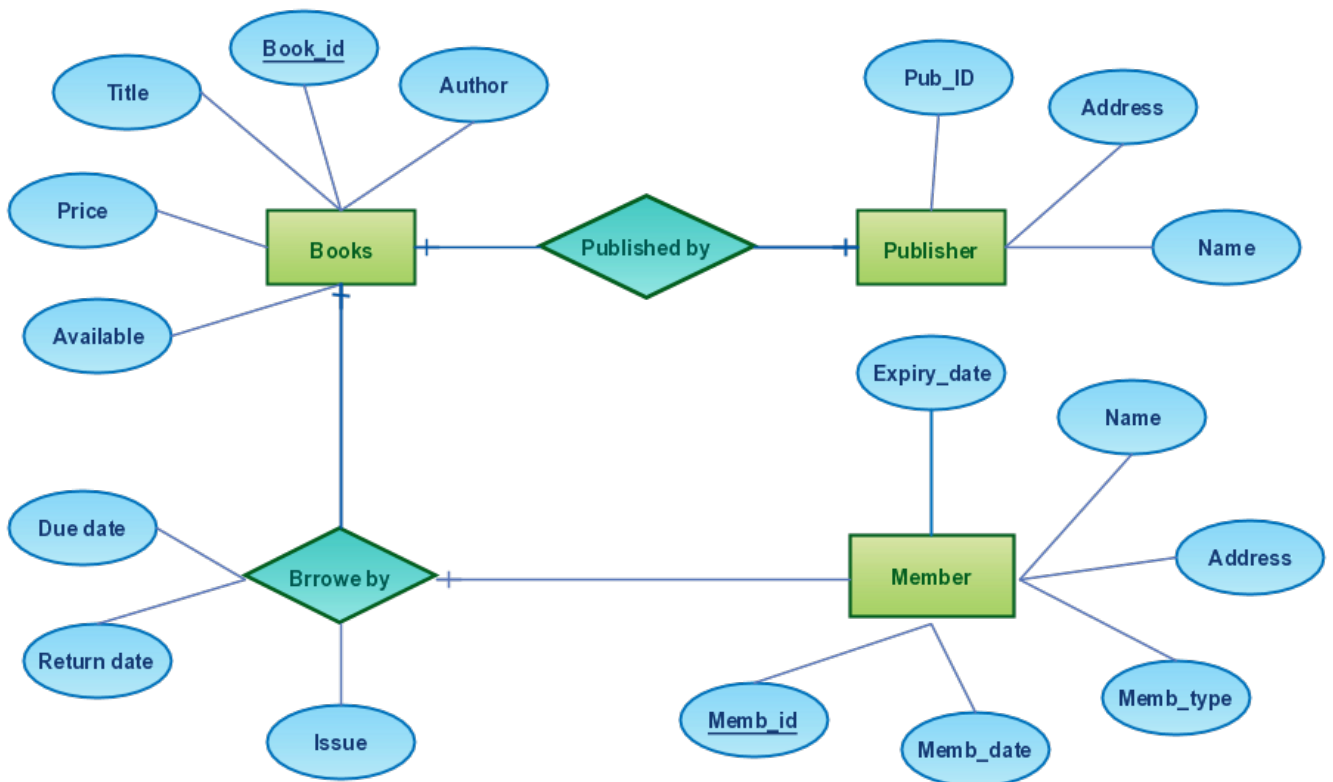
1. Simple ER Diagram for Clinic



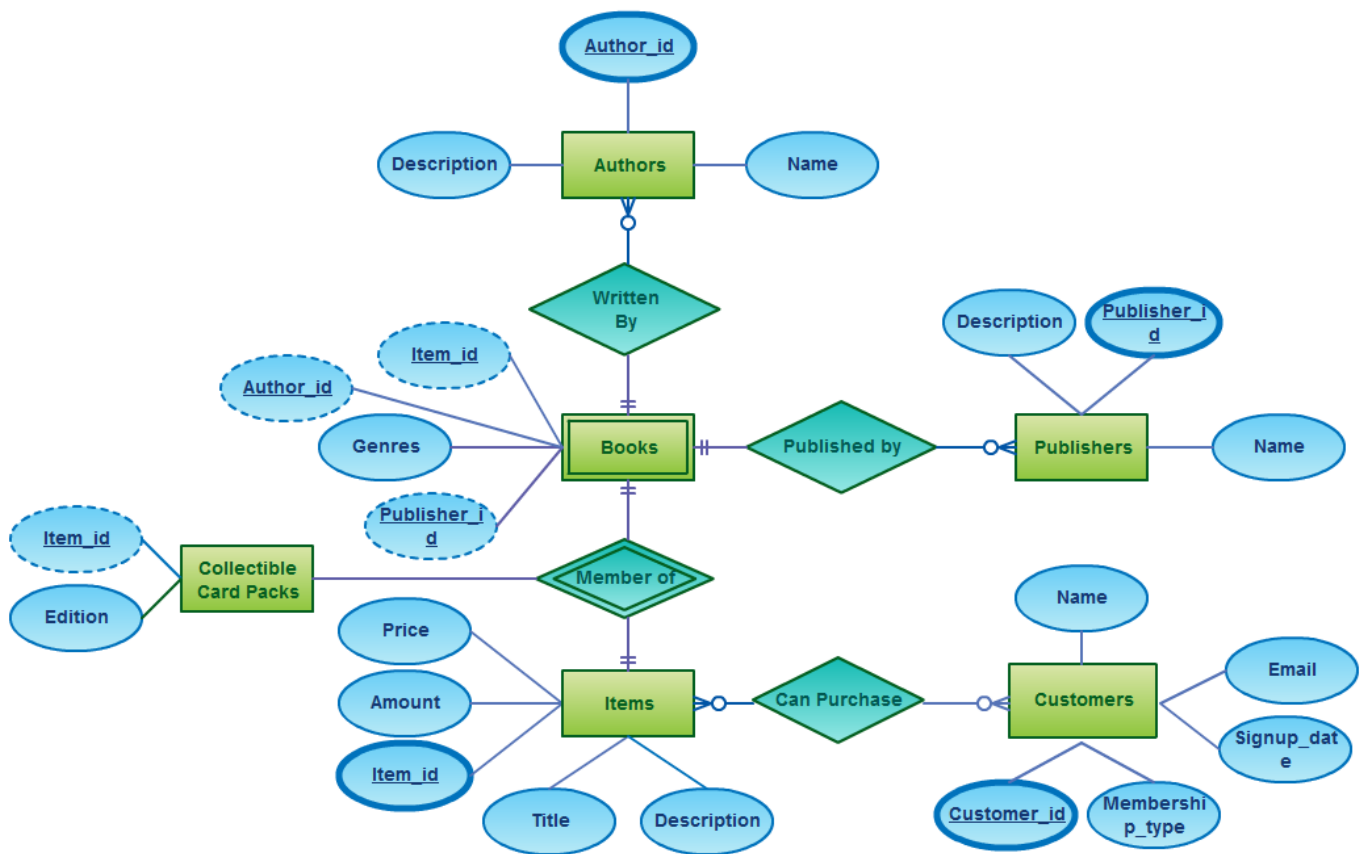
2. Simple ER Diagrams for Department



3. Draw ER Diagram For Library Management System



4. Draw ER Diagram For Book Store



Specialization (Categorization), Generalization, and Aggregation

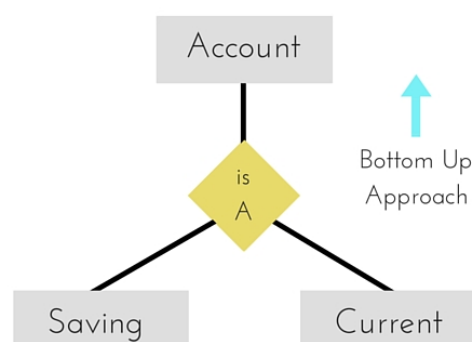
As part of the Enhanced ER Model, along with other improvements, three new concepts were added to the existing ER Model, they were:

1. Generalization
2. Specialization
3. Aggregation

- Generalization:

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity.

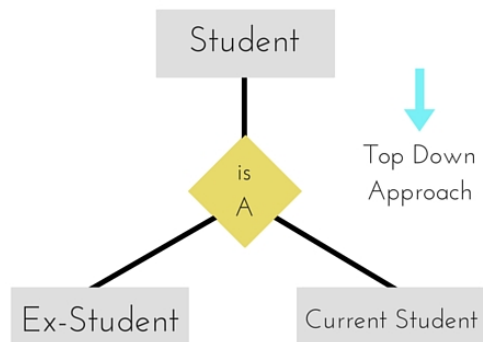
Hence, entities are combined to form a more generalized entity, in other words, sub-classes are combined to form a super-class.



For example, **Saving** and **Current** account types entities can be generalized and an entity with name **Account** can be created, which covers both

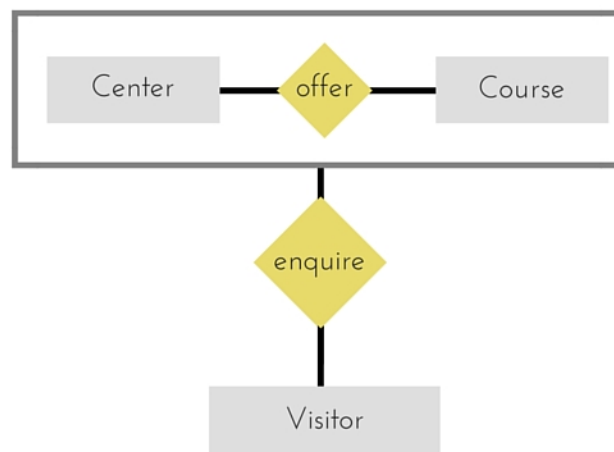
- Specialization:

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity.



- Aggregation:

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



In the diagram above, the relationship between Center and Course together, is acting as an Entity, which is in relationship with another entity Visitor.

Keys

A key is an important constraint which is a group of one or more attributes that uniquely identify an entity in the entity set.

A key is a column value in a table that is used to uniquely identify a row of data in a table or establish a relationship with another table.

There are six types of keys.

1. **Primary Key** – A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.

<u>Stu_Id</u>	Stu_Name	Stu_Age
101	Steve	23
102	John	24
103	Robert	28
104	Carl	22

In the above Student table, the Stu_Id column uniquely identifies each row of the table.

Note:

- We denote the primary key by underlining the column name.
- The value of primary key should be unique for each row of the table. Primary key column cannot contain duplicate values.
- Primary key column should not contain nulls.
- Primary keys are not necessarily to be a single column; more than one column can also be a primary key for a table.

For e.g. {Stu_Id, Stu_Name} collectively can play a role of primary key in the above table

2. **Super Key** – A super key is a set of one or more columns (attributes) to uniquely identify rows in a table.

E.g. for entity set “Student”={roll no, class, name, address, dob, year}

Some of super keys are {roll no, class, name, address}, {roll no, class, Address, dob}, {roll no, class, name, dob, year} etc.

3. **Candidate Key** – A super key with no redundant attribute is known as candidate key.

Q. How candidate key is different from super key?

Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is: It should not have any redundant attribute. That’s the reason they are also termed as minimal super key

4. **Alternate Key** – Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys.
5. **Composite Key** – A key that consists of more than one attribute to uniquely identify rows (also known as records & tuples) in a table is called composite key.
6. **Foreign Key** – Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

Converting ER diagrams into Schema/Relational Models

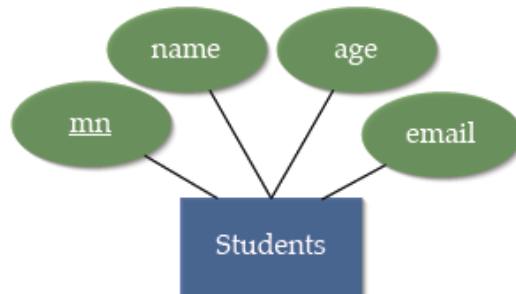
Since ER diagram gives us the good knowledge about the requirement and the mapping of the entities in it, we can easily convert them as tables and columns. i.e using ER diagrams one can easily created relational data model, which nothing but the logical view of the database.

Basic Rules:

1. Representation of Entity Sets Tables

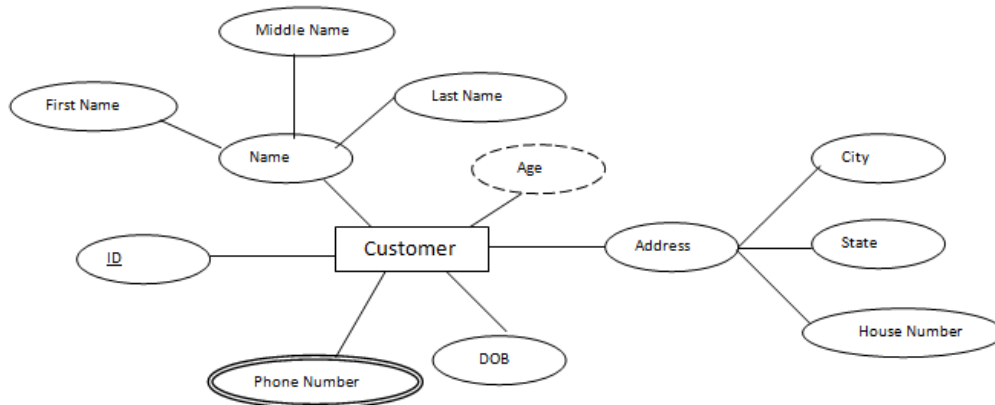
All the entities represented in the rectangular box in the ER diagram become independent tables in the database

- Representation of Strong Entity Sets with Simple Attributes



Students(MN, name, age email)

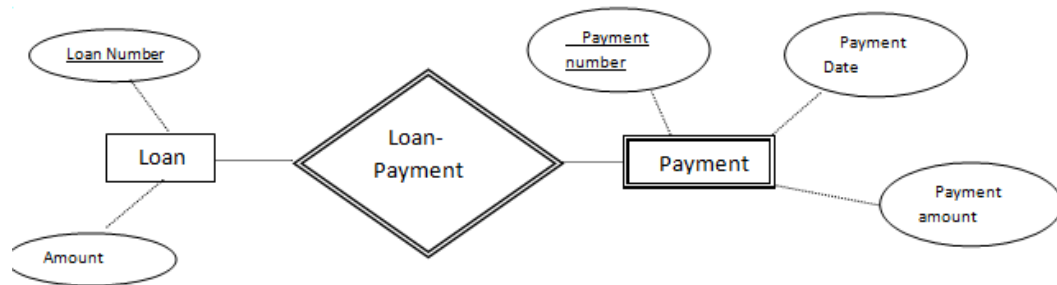
- Representation of Strong Entity Sets with Complex Attributes



Customer(ID, First Name, Middle Name, Last Name, City, State, House Number, DOB)
Customer_Phone(ID, Phone Number)

- Representation of Weak Entity

Weak entity types are converted into a table of their own, with the primary key of the strong entity acting as a foreign key in the table .



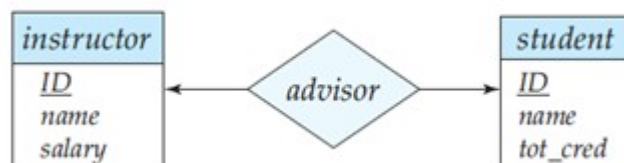
Loan(Loan Number, Amount)

Payment(Loan Number, Payment Number, Payment Date, Payment Amount)

2. Representation of Relationship Set

- One to one Relationship

The primary key of either of the participants can become a foreign key in the other.

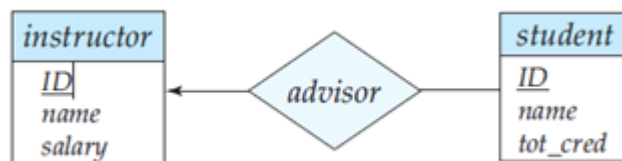


Student(Student.ID, name, tot_cred, instructor.ID)

Instructor(Instructor.ID, name, salary)

- One to Many or Many to One Relationship

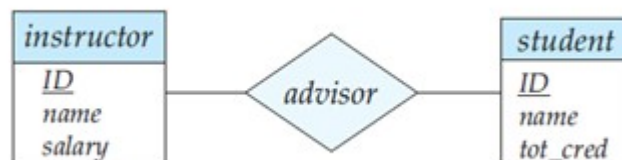
The primary key of the relation on the “one” side of the relationship becomes a foreign key in the relation on the “many” side



Instructor(ID, name, salary)

Student(Student.ID, name, tot_cred, instructor.ID)

- Many to Many Relationship

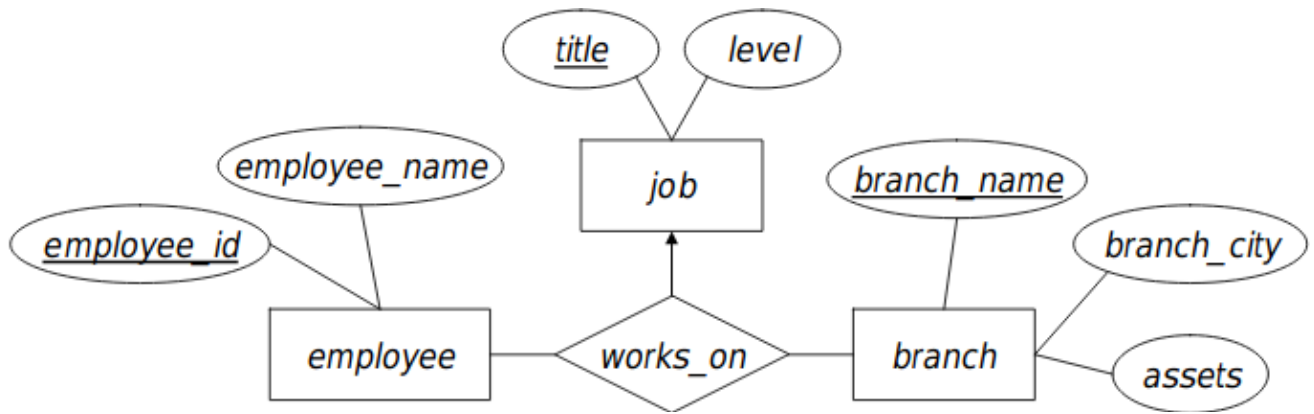


Instructor(ID, Name, salary)

Student(ID, name, tot_cred)

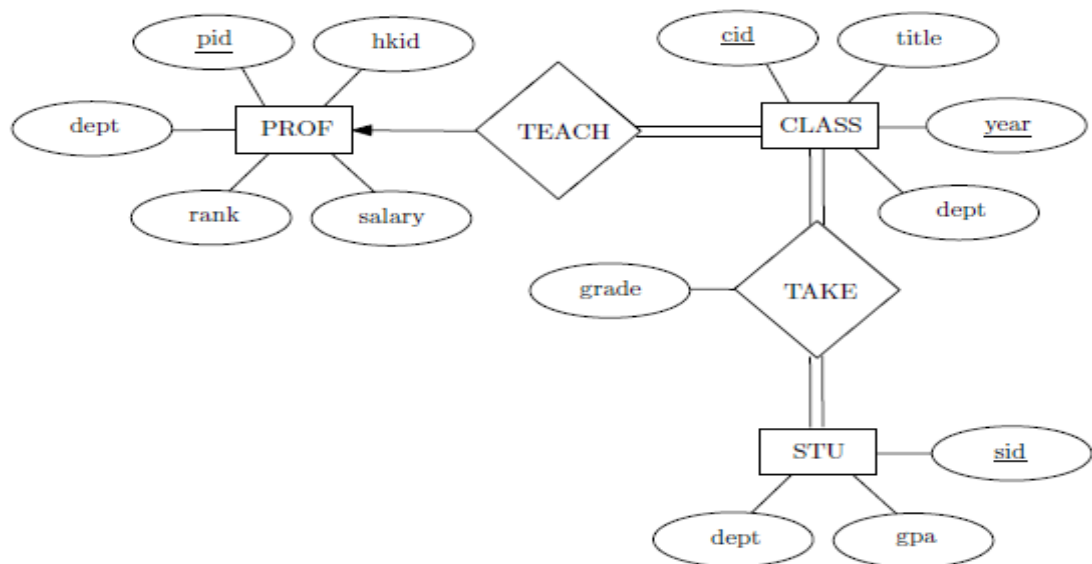
Advisor(instructor.ID, student.ID)

Q. Convert the following ER diagram into its equivalent schema



job(title, level)
 employee(employee_id, employee_name)
 branch(branch_name, branch_city, assets)
 works_on(employee_id, branch_name, title)

Q. Convert the following ER diagram into its equivalent schema



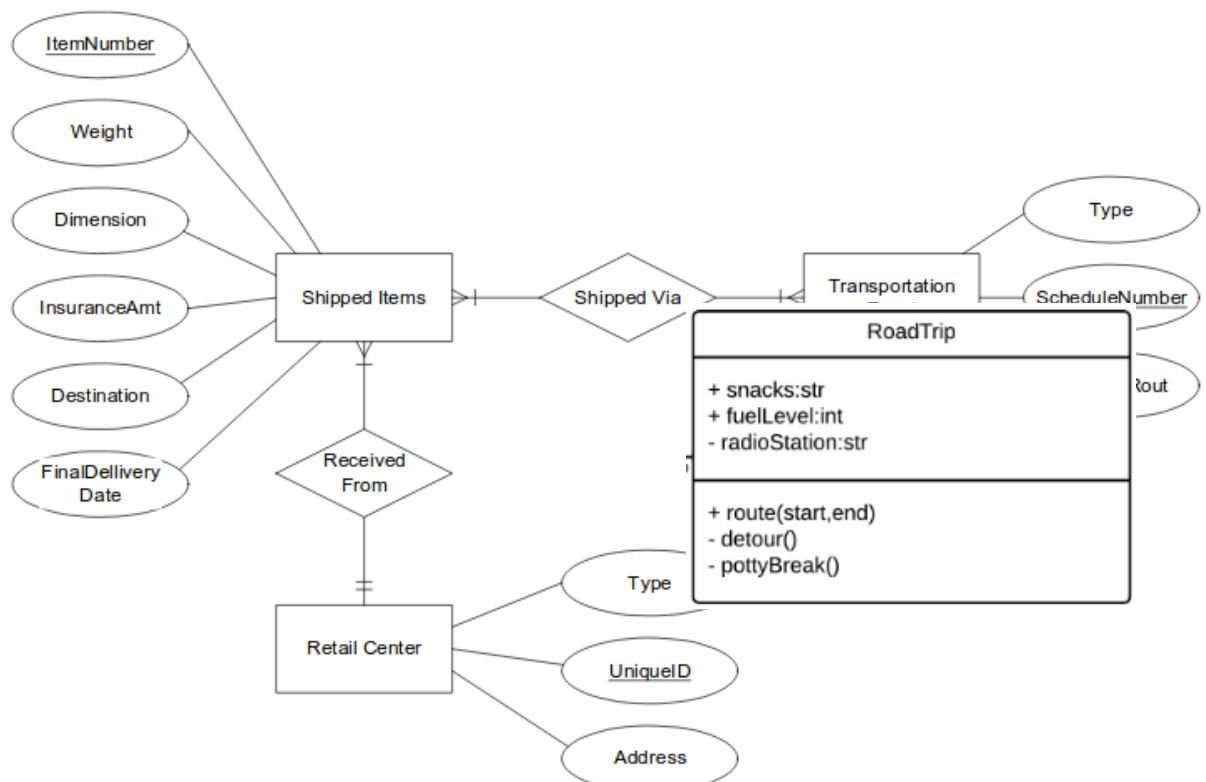
PROF(pid, hkid , dept , rank , salary)
 //note primary key is pid
 CLASS(cid, year, title, dept, pid)
 //note foreign key pid references primary key pid in prof, primary key is (cid+year)
 STU(sid, dept, gpa)
 //note : primary key is sid
 TAKE(cid, year, sid, grade)
 //note primary key is (cid+year +sid)

Q. Create an ER diagram and then to schema.

UPS prides itself on having up-to-date information on the processing and current location of each shipped item. To do this, UPS relies on a company-wide information system. Shipped items are the heart of the UPS product tracking information system. Shipped items can be characterized by item number (unique), weight, dimensions, insurance amount, destination, and final delivery date. Shipped items are received into the UPS system at a single retail center. Retail centers are characterized by their type, uniqueID, and address. Shipped items make their way to their destination via one or more standard UPS transportation events (i.e., flights, truck deliveries). These transportation events are characterized by a unique scheduleNumber, a type (e.g, flight, truck), and a deliveryRoute.

Please create an Entity Relationship diagram that captures this information about the UPS system.

Soln:



Shipped Items(ItemNumber, weight, Dimension, InsuranceAmt, Destination,FinalDelivery)
TransportationEvent(Type, ScheduleNumber, DeliveryRoute)
RetailCenter(Type,UniqueID,Address)
ReceivedFrom(ShippedItems.ItemNumber , RetailCenter.UniqueID)
ShippedVia(ShippedItems.ItemNumber, TransportationEvent.ScheduleNumber)

UML Class Diagram

UML stands for unified modeling language. It is a graphical language for modeling a system structure and its behavior.

UML is not any programming language, it is a set of diagrams that can be used to specify, construct, visualize and document the system design.

Class diagram is one of the UML diagrams which are used to show the structure of the system. These diagrams are composed of classes and their relationships.

A UML class describes a set of objects that share the same attributes, operations, and relationship. Class diagrams may specify both the conceptual **[what]** and implementation **[how]** details of the system. Class diagrams represent structural and **not** behavioral relationships that exist among system entities

The following points should be **remembered** while drawing a class diagram:

- The name of the class diagram should be meaningful.
- Each element and their relationships should be identified in advance
- Responsibility (attributes and methods) of each class should be clearly identified.

Basic components of a class diagram

The standard class diagram is composed of three sections:

- **Upper section:** Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- **Middle section:** Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom section:** Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

Member access modifiers

All classes have different access levels depending on the access modifier (visibility). Here are the access levels with their corresponding symbols:

- Public (+)
- Private (-)
- Protected (#)
- Derived (/)
- Static

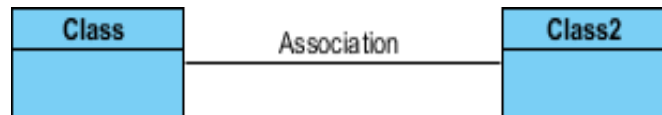
Association, aggregation and composition in class diagram

x Association

(owners feed pets, pets please owners (association))

If two classes in a model need to communicate with each other, there must be link between them, and that can be represented by an association (connector).

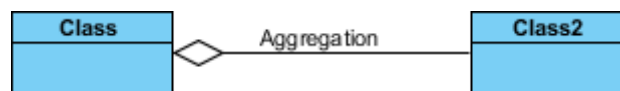
Association can be represented by a line between these classes with an arrow indicating the navigation direction. In case arrow is on the both sides, association has bidirectional association.



X Aggregation

Aggregation implies a relationship where the child can exist independently of the parent.

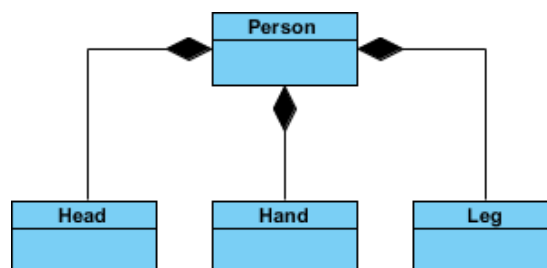
Example: Class (parent) and Student (child). Delete the Class and the Students still exist



X Composition

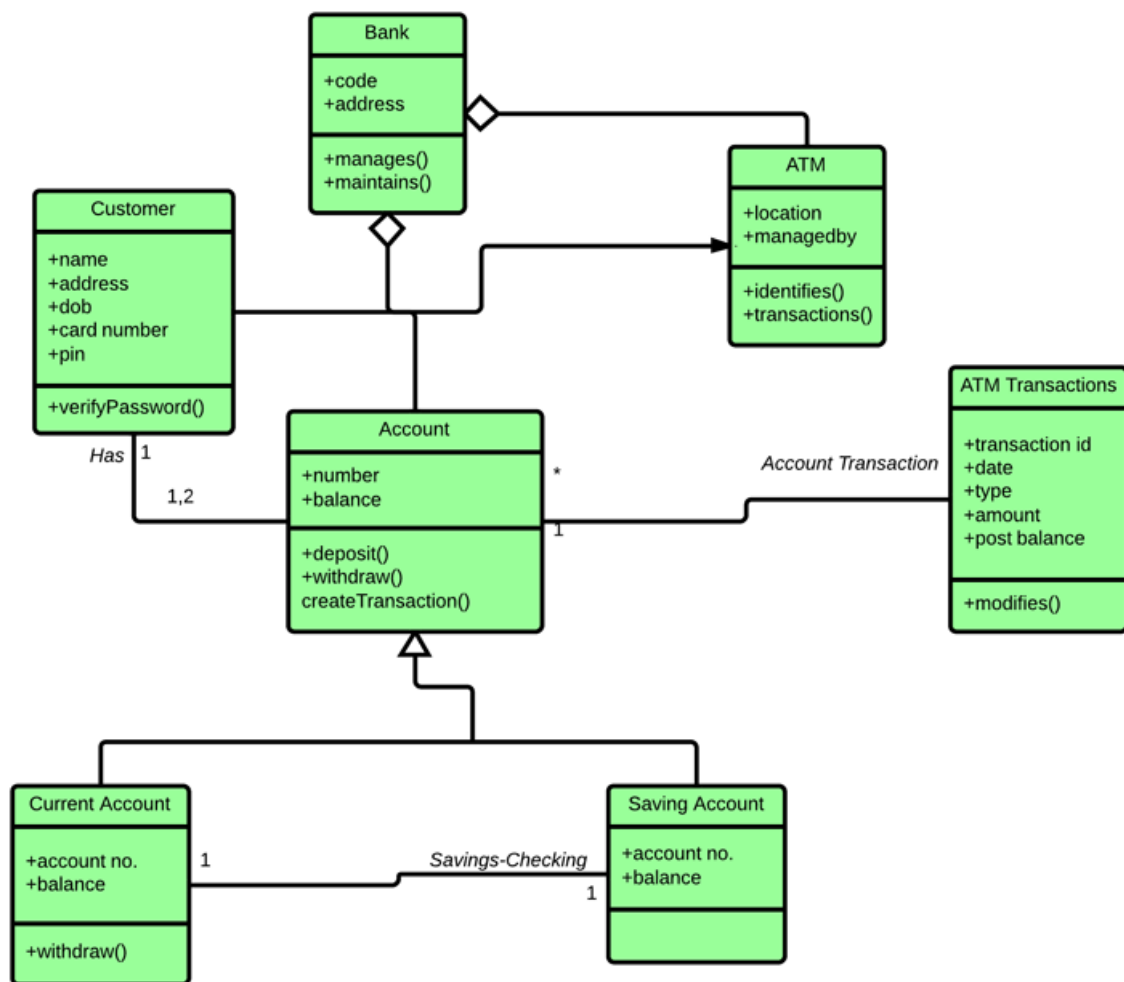
Composition implies a relationship where the child cannot exist independent of the parent.

Example: House (parent) and Room (child). Rooms don't exist separate to a House



(Note: You can further Study Inheritance, Realization, dependency)

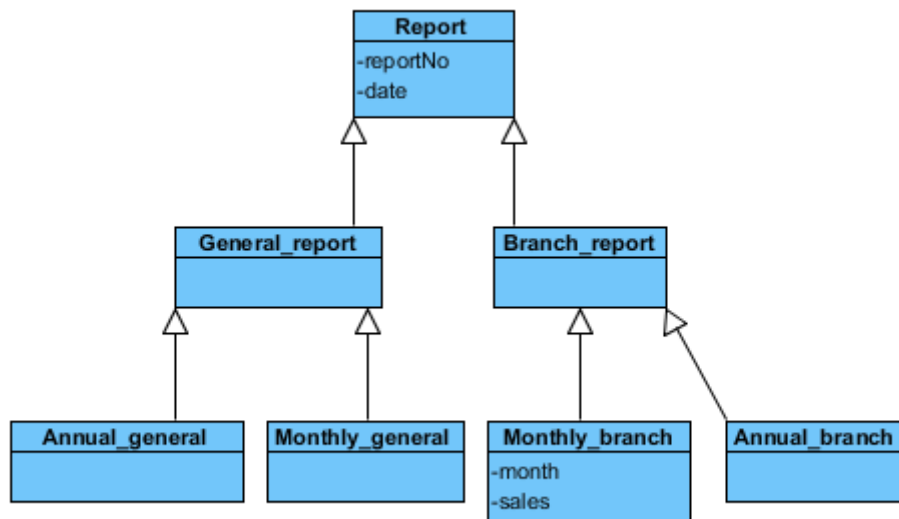
Example: Class diagram for an ATM System



Q. Consider a store with multiple branches and a system generates reports for each branch monthly and annually and an overall report as well monthly and annually. So in total there are 4 types of reports

1. Monthly branch-wise report
2. Monthly overall report
3. Annual branch-wise report
4. Annual overall report

How are they to be depicted in a class diagram?



Assignment 2:

All the PU board Questions From this Chapter