CHAPTER - 2

DATA MODELS

2.1 Logical, Physical and Conceptual Model

Data model is the collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. There are various types of data models such as:

2.1.1 Conceptual Model

The conceptual data model is the very first and the most abstract data model in the data modeling process. It is a high-level diagram that we use to define, describe, organize, and present data elements and their relationships with relatively few details. We use the conceptual data model to communicate with the different business people when defining the business requirements of the database and presenting concepts (e.g. for their feedback). We do not use these models to communicate with technical teams. This model does not have technical details, such as attributes, data types, etc.

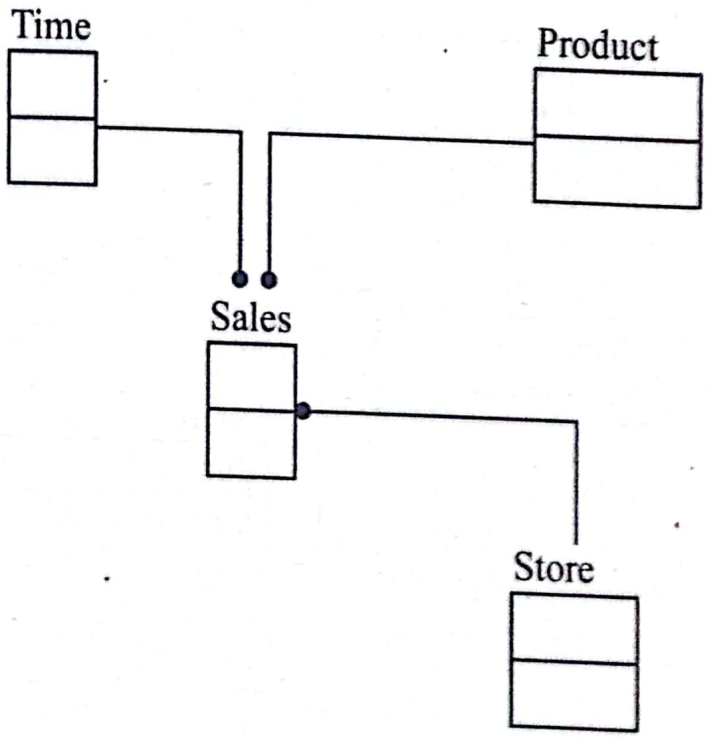


Figure: Conceptual model

In Summary:

- Identifies entities and establishes their relationships.
- Simplified and highly abstract structure.
- No attribute is specified.
- No primary key is specified.

2.1.2 Logical Model

Logical models are less abstract and provide greater detail about the concepts and relationships in the domain under consideration. They include descriptions of tables and columns, object-oriented classes, primary and foreign keys.

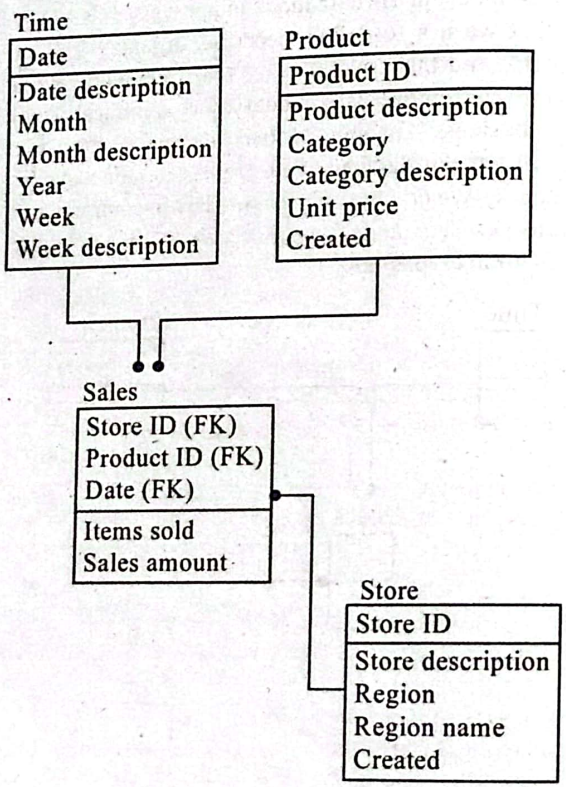


Figure: Logical model

In summary:

- Adds detail information on conceptual model.
- Adds attributes, keys, and degree of relationships.
- Requires more effort and attention to build than conceptual model.

2.1.3 Physical Model

Physical data model represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables.

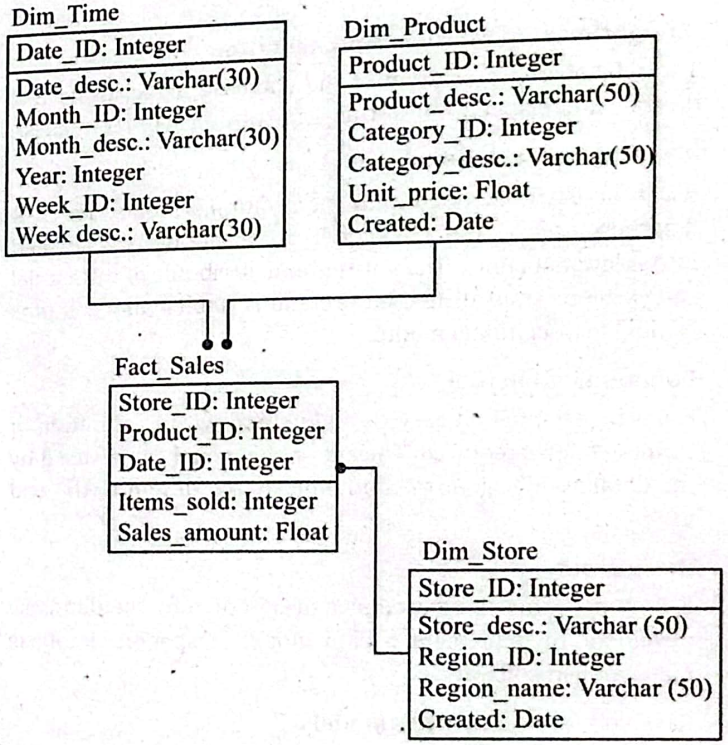


Figure: Physical model

In summary:

- Refers to converting design to relation (table).
- Converts entity to table and attributes to columns.
- Uses database compatible table name and column name.
- More difficult to understand.
- Require more efforts than logical model.

2.2 E-R (Entity-Relationship) Model

E-R model is defined as a conceptual data model that views the real world as entities and relationship. It was introduced by P.P Chen in 1976. It expresses the overall logical structure of database graphically. The major advantages of E-R model are as follows:

1. **Straightforward relation representation**
   The relation representation of the database model using E-R diagram are relatively more straightforward than other models.
2. **Mapping with relational model**
   It can be easily mapped onto the relational model. The E-R diagrams used in the E-R model can easily be transformed into relational tables. The entities and attributes of E-R model can easily be transformed into relations (tables) and columns (fields) in a relational model.
3. **Communication tool**
   It is very simple and easy to understand with a minimum of training efforts required. Therefore, the model can be used by the database designer to communicate the design to the end user.
4. **Design tool**
   E-R model can also be used as a design plan by the database developer to implement a data model in specific database management software.
5. **Easy conversion to other models**
   E-R diagrams can be easily converted to a network or hierarchical data model.
6. **Graphical representation**
   E-R model provides graphical and diagrammatical representation of various entities, their attributes and relationships between entities.
7. **Easy to modify**
   Modifications to E-R diagram at later stage is relatively easier than in other models.

Components of E-R Model

Following figure shows various components of E-R model:

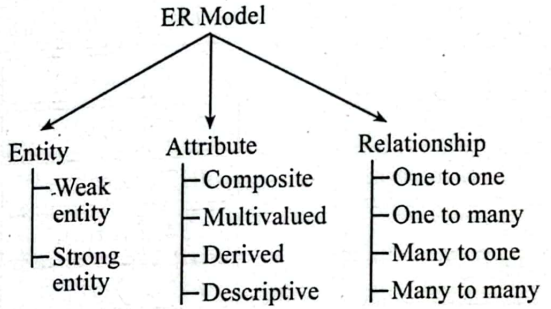


Figure: Components of ER model

2.2.1 Entity and Entity Set

Entity

An entity is a "thing" or "object" in the real world that is distinguishable from all other objects.

Example:

Each person in a university is an entity. An entity has a set of properties, and the values for some set of properties must uniquely identify an entity. For instance, a person may have a person id property whose value uniquely identifies that person. Thus, the value 677-89-9011 for person id would uniquely identify one particular person in the university. Similarly, courses can be thought of as entities, and course id uniquely identifies a course entity in the university.

An entity may be concrete, such as a person or a book, or it may be abstract, such as a course, a course offering, or a flight reservation.

Entity Set

A set of entities of same type that share the same properties are referred as entity set. All entities in an entity set have the same set of attributes.

Example:

Set of all students, persons, companies, trees.

Instructor and student is show below:

instructor (instructor_ID, instructor_name)

student (student_ID, student_name)

instructor_ID instructor_name

76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

student_ID student_name

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Figure: Entity sets instructor and student

Following are the types of Entity sets:

1. **Weak Entity Set**

Entity set which does not have sufficient attributes to form a primary key is known as weak entity set. It depends on another entity and does not contain any key attribute of its own. However, it contains a partial key called as a discriminator. Discriminator is represented by underlining with a dashed line. It is partially unique and can be combined with other strong entity set to uniquely identify the tuples.

Weak entity set is represented by double rectangles and the member of weak entity set is called as subordinate entity.

The primary key of weak entity set is combination of partial key and primary key of the strong entity set. The participation of weak entity type is always total. The relationship between weak entity type and its identifying strong entity type is called identifying relationship and it is represented by double diamond.

Example:

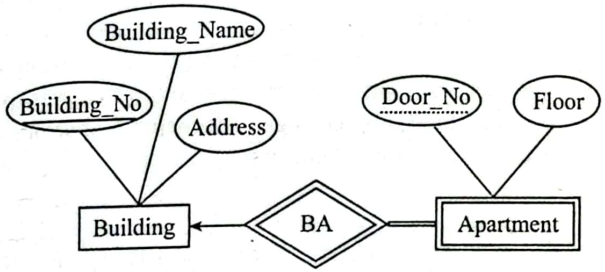


Figure: Weak entity: Apartment

In the above E-R Diagram,

- One strong entity set "Building" and one weak entity set "Apartment" are related to each other.
- Strong entity set "Building" has building number as its primary key.
- Door number is the discriminator of the weak entity set "Apartment".
- This is because door number alone cannot identify an apartment uniquely as there may be several other buildings having the same door number.
- Double line between Apartment and relationship set signifies total participation.
- It suggests that each apartment must be present in at least one building.
- Single line between Building and relationship set signifies partial participation.
- It suggests that there might exist some buildings which has no apartment.

Here to uniquely identify any apartment,

- First, building number is required to identify the particular building.
- Secondly, door number of the apartment is required to uniquely identify the apartment.

Thus, Primary key of Apartment = Primary key of Building + Its own discriminator.

2. Strong Entity Set

Entity set that has a primary key is known as strong entity set. It has its own primary key and attributes and is represented by single rectangle. The member of strong entity set is called as dominant entity set.

Example:

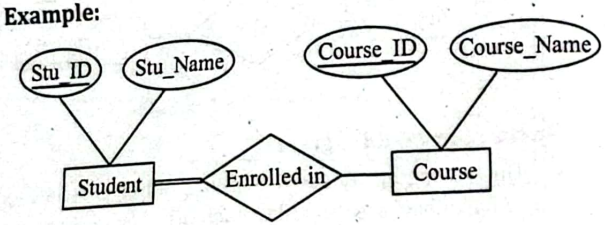


Figure: Strong entity

In the above ER diagram,

- Two strong entity sets "Student" and "Course" are related to each other.
- Student ID and Student name are the attributes of entity set "Student".
- Student ID is the primary key using which any student can be identified uniquely.
- Course ID and Course name are the attributes of entity set "Course".
- Course ID is the primary key using which any course can be identified uniquely.
- Double line between Student and relationship set signifies total participation.
- It suggests that each student must be enrolled in at least one course.
- Single line between Course and relationship set signifies partial participation.
- It suggests that there might exist some courses for which no enrollments are made.

Comparison between weak entity set and strong entity set.

Strong Entity Set	Weak Entity Set
• Strong entity set always has primary key;	• Weak entity set has partial discriminator key.
• Strong entity set is not dependent of any other entity.	• Weak entity set is dependent on strong entity.
• Strong entity set is represented by single rectangle.	• Weak entity set is represented by double rectangle.
• Relationship between two strong entity set is represented by single diamond.	• While the relation between one strong entity set and one weak entity set is represented by double diamond.
• Strong entity set has either total participation or not.	• Weak entity set always has total participation.

2.2.2 Attributes and Keys

Attributes are the descriptive properties of an entity set. Ellipse is used to represent an attribute.

Example:

Name, BirthDate, Roll No, ID, Age, Contact Number, etc. can be attributes of a student entity set.

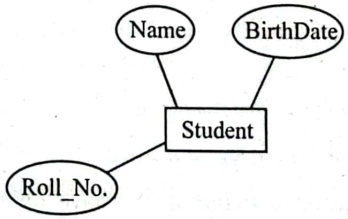


Figure: Attributes of entity student

More examples of entity and attribute:

customer (customer_id, customer_name, customer_street, customer_city)

loan (loan_number, amount)

Here customer and loan are the entities whereas other are the attributes of particular entity.

Simple and Composite Attributes

- **Simple or atomic attributes** are those attributes which cannot be divided into subparts.

  Example:

  student_id, phone_no are simple attributes

- **Composite attributes** are those attributes which can be divided into subparts. They are composed of several basic attributes.

  Example:

  Name can be divided into first name and last name. So Name is composite attribute.

In the following figure Roll_No and BirthDate are simple attributes whereas Name is composite attribute.

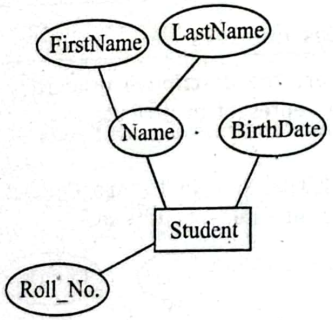


Figure: Simple and composite attributes

Single-Valued and Multi-Valued Attributes

- ***Singled* valued attributes** can have only one value. It is represented by single ellipse.

Example:

id, roll, citizenship_no, etc. Most attributes are single valued for particular entity.

- **Multi-valued attributes** can have more than one values. It is represented by double ellipse.

Example:

phone number, college degree, address etc.

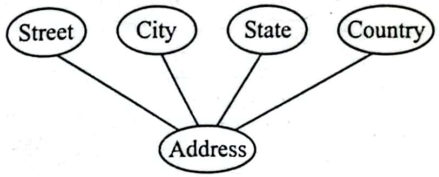


Figure: Multi-valued attribute: Address

Address is multi-valued attribute in above figure.

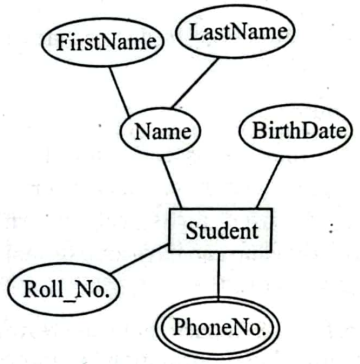


Figure: Composite, single-valued and multi-valued attribute

In above figure :

- Name is composite attribute.
- BirthDate, Roll_No. is single-valued attribute.
- PhoneNo. is multi-valued attribute.

Derived Attributes

Attributes whose values are calculated from the values of other related attributes are called derived attributes. Derived attributes are represented by dashed ellipse.

Example:

Age can be derived from DOB attribute i.e., Age = SystemDate – DateOfBirth.

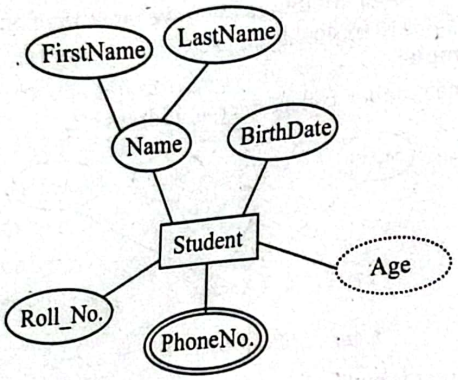


Figure: Derived attribute

In the above figure, Age is derived attribute as it can be derived from BirthDate.

Descriptive Attributes

The attributes that are used for describing the relationship are called descriptive attributes, also referred as relationship attributes. They are used for storing information about the relationship. A relationship can have zero or more attributes.

Example:

Sagar works for MIS Department from November 2021. In the following figure, date information is captured by the since in Works_In.

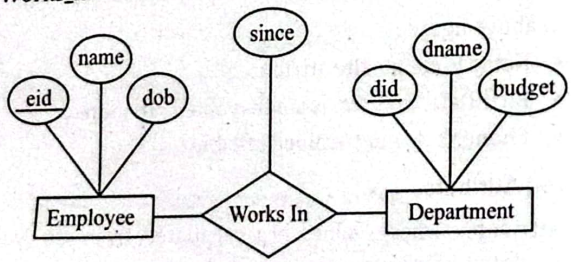


Figure: Descriptive attributes

Keys

Keys are the group of one or more attributes that uniquely identify an entity in the entity set.

Example:

Roll_No of a student makes him/her identifiable among students.

Types of keys

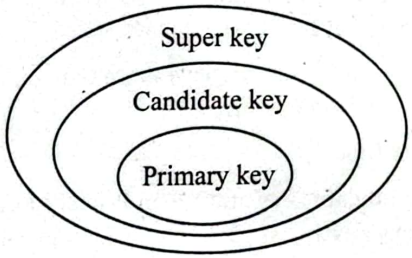


Figure: Keys in relational data model

1. **Super Key**

   Super key is an attribute or set of attributes that uniquely identify each record of relation.

   Consider the following schema :

   Employee (Emp_SSN, Emp_No, Emp_Name)

   Above schema has following super keys:

   {Emp_SSN}
   {Emp_No}
   {Emp_Name}
   {Emp_SSN, Emp_No}
   {Emp_SSN, Emp_Name}
   {Emp_SSN, Emp_No, Emp_Name}
   { Emp_No, Emp_Name}

2. **Candidate Key**

   It is also a minimal super key that uniquely identifies either an entity or a relationship. Candidate keys are selected from set of super keys.

Example:

Emp_SSN, Emp_No

3. **Primary Key**

Primary key is a candidate key that the database designer selects while designing the database.

Example:

We can say that either {Emp_SSN} or {Emp_no} can be chosen as a primary key for above schema.

NOTE: Any entity set can have more than one candidate key but only one primary key.

4. **Alternate Key**

All the candidate keys other than Primary Key are known as Alternative Keys.

5. **Foreign Key**

It is an attribute in any entity set which is also a primary key in any other entity set.

2.2.3 Relationship and Relationship Set

A relationship is the association among several entities. It connects different entities through a meaningful relation. A relationship is represented by diamond shape in ER diagram. Relationship is classified in terms of degree, connectivity, cardinality, and existence.

Example:

44553 (Peltier) — *advisor* — 22222 (Einstein)

student entity — relationship — instructor entity

Relationship Set

A relationship set is a mathematical relation among $n \geq 2$ entities, each taken from entity sets. If $E_1, E_2, \ldots, E_n$ are entity sets, then a relationship set R is a subset of:

$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

where $(e_1, e_2, \ldots, e_n)$ is a relationship

Example:

(44553,22222) ∈ advisor is the relationship.

i.e. Relationship Set *advisor*

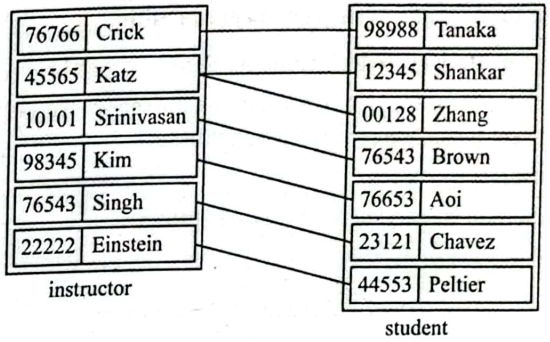


Figure: *Relationship set advisor*

The function that an entity plays in a relationship is called that entity's role. Since entity sets participating in a relationship set are generally distinct, roles are implicit and are not usually specified. However, they are useful when the meaning of a relationship needs clarification. Such is the case when the entity sets of a relationship set are not distinct; that is, the same entity set participates in a relationship set more than once, in different roles. In this type of relationship set, sometimes called a recursive relationship set, explicit role names are necessary to specify how an entity participates in a relationship instance. For example, 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.

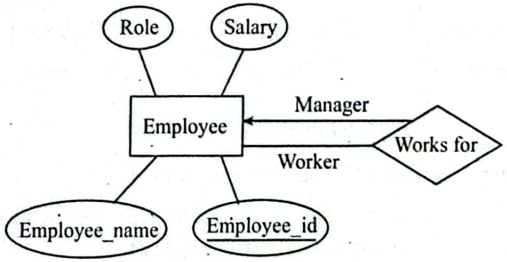


Figure: *E-R diagram with role indicators.*

Degree of Relationship Set

The degree of relationship set is defined as the number of entity sets associated with the relationship sets.

There are three types of degree of relationship:

1. **Unary or Recursive Relationship Set (Degree=1)**

   Unary relationship set is a relationship set where only one entity set participates in a relationship set.

   Example:

   One person is married to only one person.

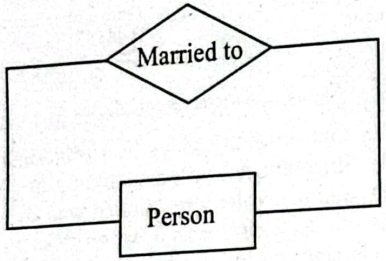


Figure: Unary relationship set

2. **Binary Relationship Set (Degree=2)**

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

   Example:

   Student enrolled in Course.

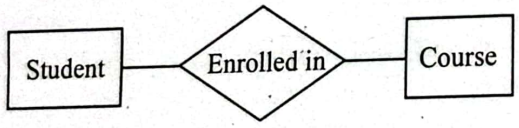


Figure: Binary relationship set

3. **Ternary Relationship Set (Degree=3)**

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

Example:

Employee works in Department. Employee Works in Location.

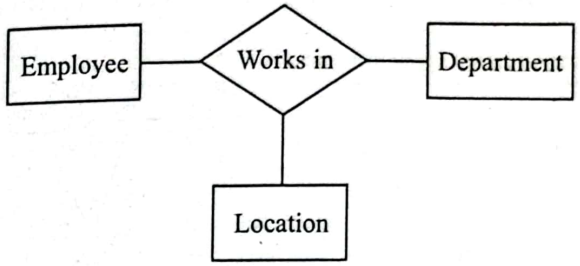


Figure: Ternary relationship set

4. **n-ary Relationship Set (Degree=n)**

   n-ary relationship is a relationship where 'n' entities participate in a relationship.

Mapping Cardinalities

The number of times an entity of an entity set participates in a relationship is known as mapping cardinalities or cardinality ratios. It is most useful in describing binary relationships. Cardinality can be of different types:

- One-to-one (1:1)
- One-to-many (1:N)
- Many-to-one (N:1)
- Many-to-many (N:N)

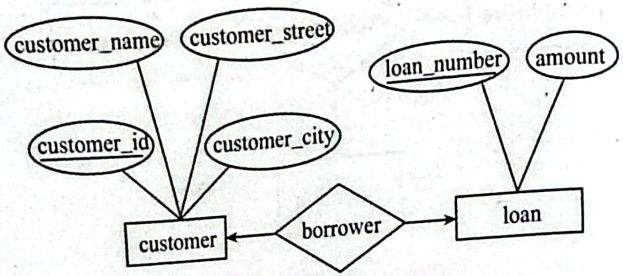
One-to-One

A *one-to-one* relationship is when at most one instant of an entity (say A) is associated with one instant of another entity (say B).

Example:

A customer is associated with at most one loan via the relationship borrower.

A loan is associated with at most one customer via borrower

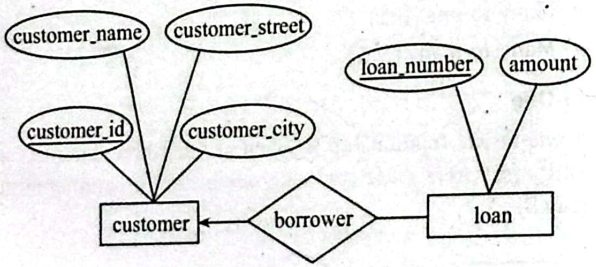


One-to- Many

A *one-to-many* relationship is when one instant of entity A is associated with many instances of entity B.

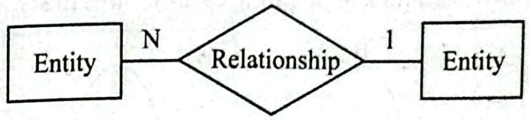
Example:

In the one-to-many relationship, a loan is associated with at most one customer via borrower, a customer is associated with several (including 0) loans via borrower.



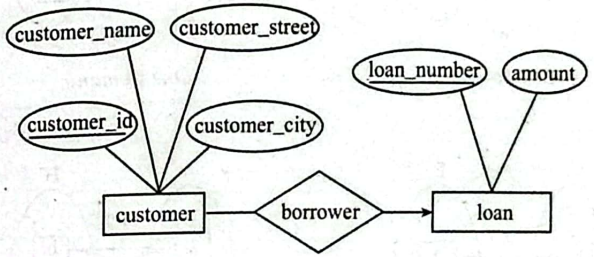
Many-to-One

A many-to-one relationship is when many instants of entity A *is associated with one instant of entity B.*



Example:

In a many-to-one relationship, a loan is associated with several (including 0) customers via borrower, a customer is associated with at most one loan via borrower.



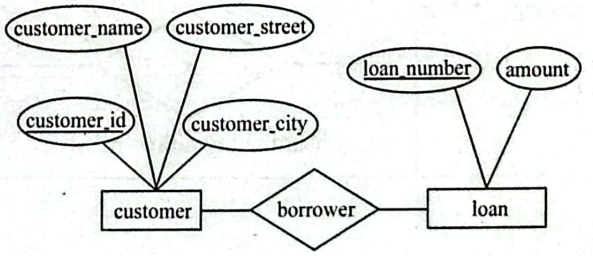
Many-to-Many

A *many-to-many* relationship is when one or many instants of entity A is associated with one or many instants of entity B.

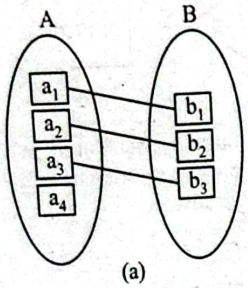
Example:

A customer is associated with several (possibly 0) loans via borrower.

A loan is associated with several (possibly 0) customers via borrower.

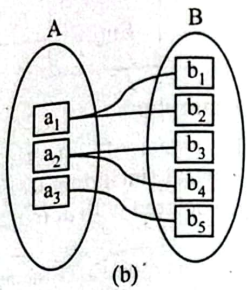


More examples of mapping cardinalities in set format:



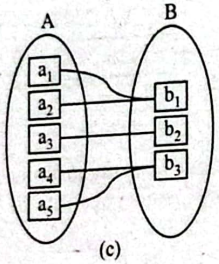
(a)

One to one



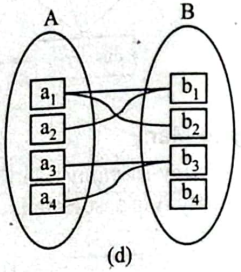
(b)

One to many



(c)

Many to one

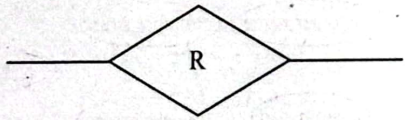


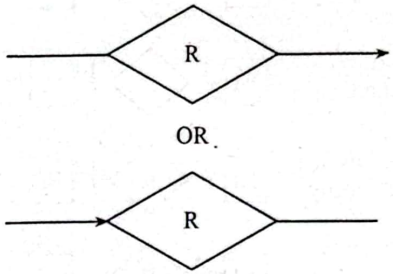
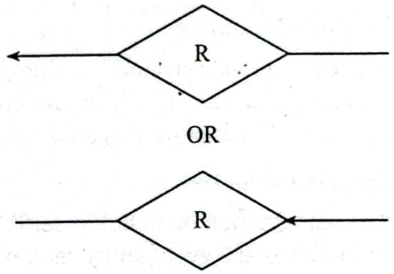
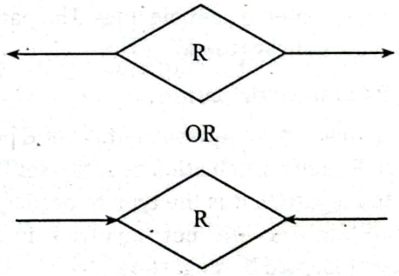
(d)

Many to many

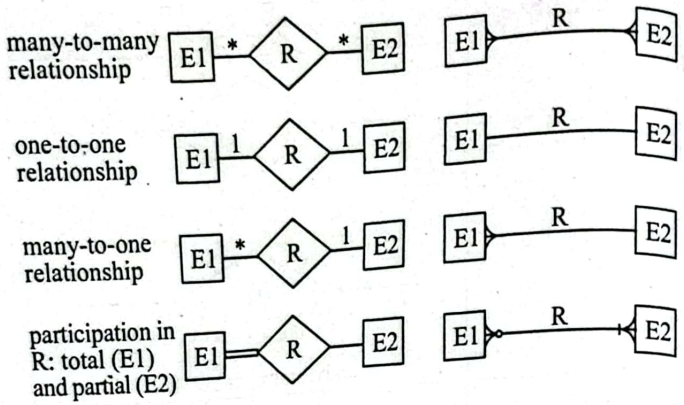
Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Furthermore, we can express these cardinality as following symbols:

Cardinality	Symbols
Many-to-many (*. * or m:n)	 Cardinality ratio = m : n

Cardinality	Symbols
Many-to-one (*:1 or n:1)	 Cardinality ratio =n:1
One-to-many (1: * or 1:n)	 Cardinality ratio =1:n
One-to-one (1:1)	 Cardinality ratio =1:1

Further according to Crow's feet notation, we can also express these cardinality as following:



Participation Constraints

Participation constraints define the least number of relationship instances in which an entity must compulsorily participate. The types of participation constraints are:

a. Total Participation

The participation of an entity set E in a relationship set R is sed to be total if every entity set E participates in at least one relationship in R. It is the type of participation constraints where each entity is involved in the relationship. It is represented by double lines. The participation of weak entity type is always total.

b. Partial Participation

If only some entities in entity set E participate in relationship in R then participation of entity set E in relationship R is said to be partial. It is the type of participation constraints where all entities are not involved in the relationship. It is represented by single line.

Example:

Participation of loan in borrower is total participation. i.e., every loan must have a customer associated to it via borrower

Participation of customer in borrower is partial participation. i.e., every customer may not have loan.

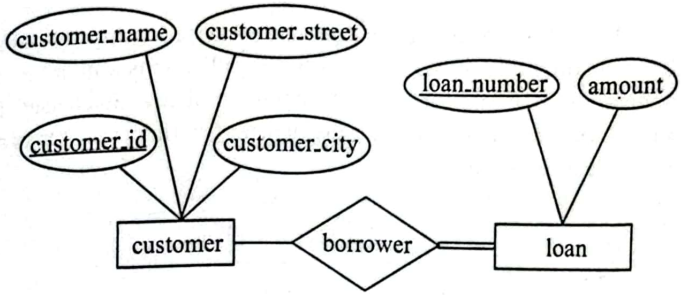


Figure: Participation constraint

2.2.4 Extended ER (EER) Features

As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.

Hence, 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

Before discussing the concepts of specialization, generalization, and aggregation two another entity types of super classes (supertype) and subclass (subtype) are described:

Superclass Entity Type (Supertype)

A superclass entity type is a generic entity type that includes one or more distinct subclasses that require to be represented in a data model. It means members belong to subclass are same as the entity in the superclass. The relationship between a superclass and subclass is a one-to-one (1 : 1) relationship. In some cases, a superclass can have overlapping subclasses.

Subclass Entity Type (Subtype)

A subclass entity type is a more specialized entity type tha has a distinct role in the organization. A subclass is a member o superclass. It is one of the data-modeling abstractions used i EER. A subclass may be further divided, and, in that case, it acts a superclass for its subclasses.

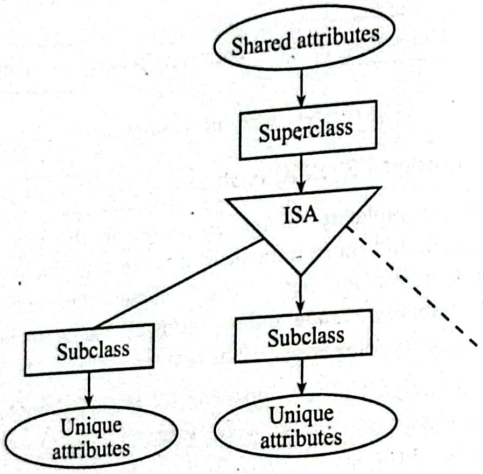


Figure: Superclass/Subclass relationship

Specialization

Specialization and generalization are opposite to each othe Specialization is a top-down approach in which one higher leve entity can be broken down into two lower-level entity. The ide behind Specialization is to find the subsets of entities that hav few distinguish attributes. It is represented by a triangl component labeled ISA same as generalization. The IS relationship is also referred to as superclass-subclass relationshi The circle is another symbol for ISA. While using symbol-circle w use subset symbol to denote inheritance.

Example:

Consider an entity employee which can be further classifie as sub-entities Technician, Engineer & Accountant becaus these sub entities have some distinguish attributes.

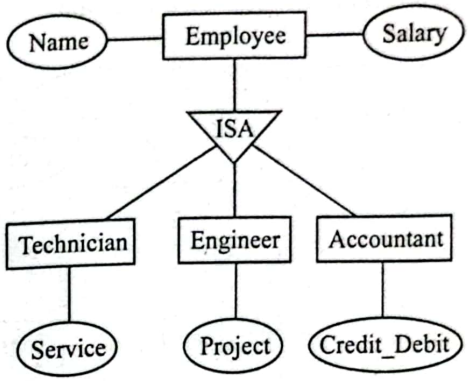


Figure: Specialization

Schema

Technician (Name, Salary, Service)

Engineer (Name, Salary, Project)

Accountant (Name, Salary, Credit_Debit)

Here Employee is superclass entity type and Technician, Engineer and Accountant are subclass entity type.

More example of specialization

1. Entity type Employee before specialization

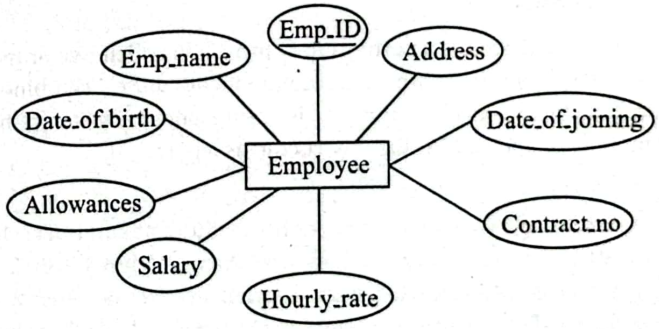


Figure: Employee before specialization

2. **Entity type Employee after specialization**

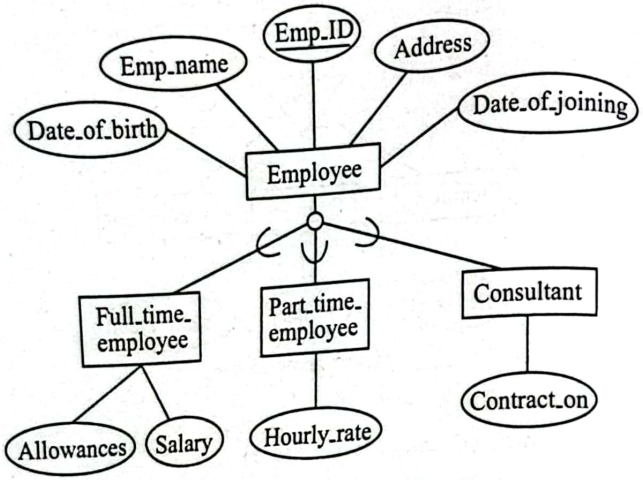


Figure: Employee after specialization

Creation of three subtypes for the EMPLOYEE supertype in above figure, is an example of specialization. The three subclasses have many attributes in common but there are also attributes that are unique to each subtype, for example, SALARY and ALLOWANCES for the FULL-TIME-EMPLOYEE.

Generalization

Generalization is a bottom-up approach in which two or more lower-level entities with some common attributes combine to form a higher-level entity. It is represented by a triangle component labeled ISA (E.g., instructor "is a person").

Example:

Let's say we have two entities Student and Teacher. Attributes of entity Student are: Name, Address & Grade whereas attributes of entity Teacher are: Name, Address & Salary. The ER diagram before generalization looks like this:

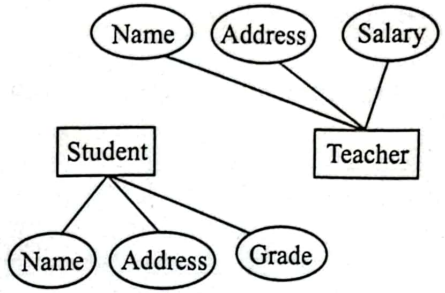


Figure: Before generalization

These two entities have two common attributes: Name and Address, we can make a generalized entity with these common attributes. Let's have a look at the ER model after generalization.

The ER diagram after generalization

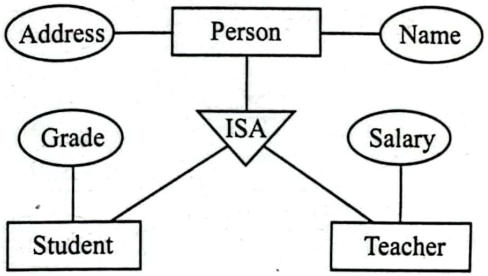


Figure: Generalization

We have created a new generalized entity Person and this entity has the common attributes of both the entities. As we can see in the above ER diagram that after the generalization process the entities Student and Teacher only has the specialized attributes Grade and Salary respectively and their common attributes (Name & Address) are now associated with a new entity Person which is in the relationship with both the entities (Student & Teacher).

More examples of generalization

Example:

(a) Three entity types of namely Car, Truck and Two Wheeler:

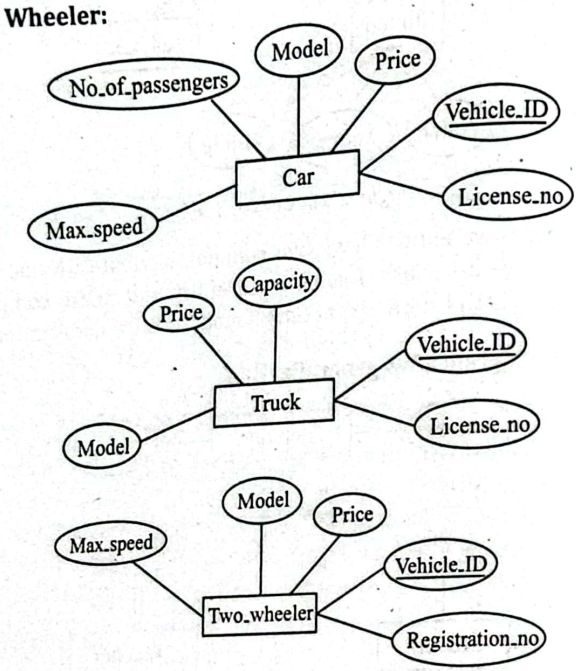


Figure (a): Generalization

(b) Generalization to Vehicle type:

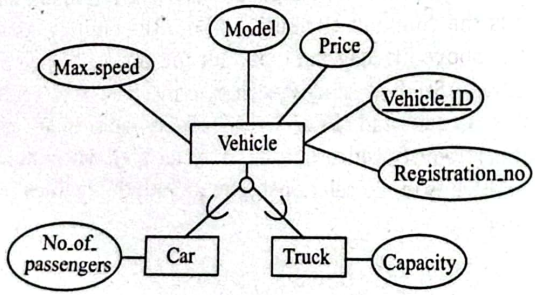


Figure (b): Generalization

Another example of generalization is shown in above figure (b). As shown in above figure (a), three entity types are defined as Car, Truck and Two_wheeler. After analysis it is observed that these entities have a number of common attributes such as REGISTRATION-NO, VEHICLE-ID, MODEL, PRICE and MAX-SPEED. This fact basically indicates that each of these three entity types is a version of a general vehicle type. Above figure(b) illustrates the generalized model of entity type VEHICLE together with the resulting supertype/subtype relationships. The entity CAR has the specific attribute as NO-OF-PASSENGERS, while the entity type TRUCK has specific attribute as CAPACITY. Thus, generalization has allowed to group entity types along with their common attributes, while at the same type preserving specific attributes that are specific to each subtype.

Comparison between Generalization and Specialization

Generalization	Specialization
• Generalization extracts the common features of multiple entities to form a new entity.	• Specialization splits an entity to form multiple new entities that inherit some feature of the splitting entity.
• It is bottom-up approach.	• It is top-down approach.
• The higher-level entity must have lower-level entities.	• The higher-level entity may not have lower-level entities.
• Generalization reduces the size of a schema.	• Specialization increases the size of a schema.
• Generalization is applied on group of entities.	• Specialization is applied on a single entity.
• Generalization results in forming a single entity from multiple entities.	• Specialization results in forming the multiple entity from a single entity.

Aggregation

Aggregation is an abstraction process in which a relationship set is considered as higher-level entity set. In other words, aggregation is a process when relation between the entities is treated as a single entity.

Example:

Consider an example of ternary relationship having three entity sets Employee, Job and Branch with relationship set works-on as shown in Figure a. The information about Managers on employee, managers of particular jobs and of different branches can be taken easily.

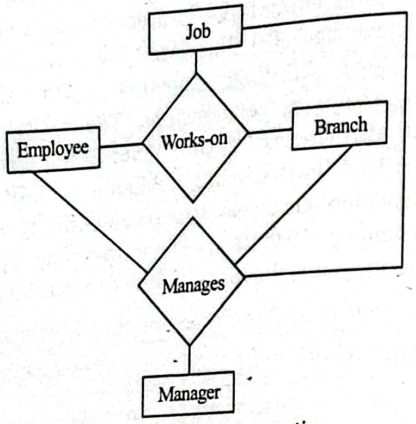


Figure (A): Before aggregation

We cannot combine Works-on and managers relationship sets because some workers are not managers. Using aggregation, Works-on relationship set acts as higher entity set and solve this drawback of E-R Model. E-R Model with aggregation is shown in Figure:

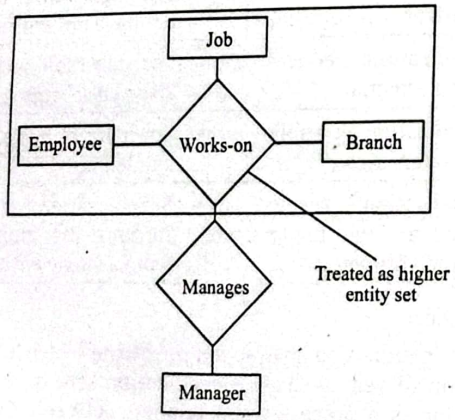


Figure (B): After aggregation

More Examples of Generalization and Specialization

Consider the example of a bank as shown in figure below in which Person entity is superclass entity type which is further divided into Employee and Customer entities. Here Employee and Customer entities are subclass entity type.

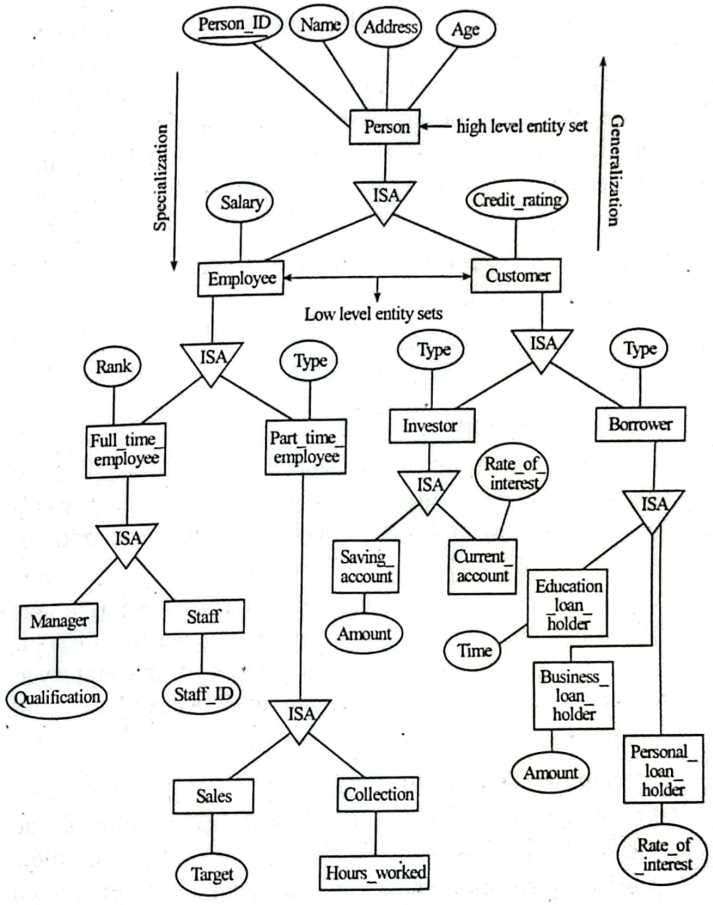


Figure: Specialization and generalization

Basic concept is that by E-R model, a person belongs to a bank is known, and by EER how it belongs to bank is known

means, what is the exact relationship between bank and that person? A person may be employee or customer which can be further categorized into Manager, Staff or Investor, Borrower and so on.

Attribute Inheritance

Specialization and generalization lead to attribute inheritance between higher level entity set and lower level entity set. Inheritance is a process by which lower-level entity set inherits (or taken) some properties of its higher-level entity set. Consider the above Figure. Here entity sets Employee and Customer inherits attributes Person_ID, Name, Address, Age from Person entity set.

Specialization and generalization constraints

a. Disjoint Constraints
b. Overlapping Constraints
c. Completeness constraint

a. Disjoint Constraints

According to Disjoint constraint if the subclasses of a specialization/ generalization are disjoint then an entity can be a member of only one subclass of that specialization/ generalization.

Consider above figure, subclasses Full Time Employee and Part Time Employee of superclass Employee are disjoint. Suppose any employee 'Martin' works as part time employee for Bank then it can only belong to subclass 'Part Time Employee'.

b. Overlapping Constraints

According to overlapping constraints, the same entity may be a member of more than one subclass of the specialization. Overlap constraint is shown by placing an o in specialization circle.

Example:

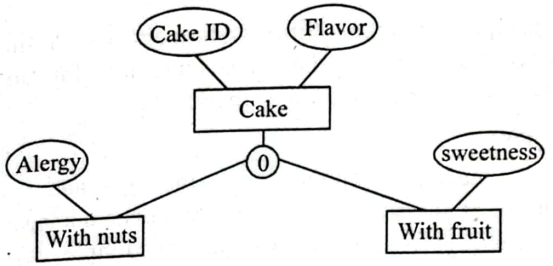


Figure: Overlapping Constraints

In a bakery, cake can be made with nuts or with fruits. However, a cake can also be made with nuts and fruits at the same time

c. Completeness Constraints

The completeness constraint may be either total or partial.

A total specialization constraint specifies that every entity that belongs to a superclass must belong to at least one subclass of the specialization. In EER, a total specialization is shown by using a double line that is going out of the superclass to connect the super class to the circle (this notation is similar to the notation for total participation in a relationship.).

A partial specialization constraint specifies that some entities might not belong to any of the subclasses of the specialization. In EER, a single line going out of the superclass is used to display a partial specialization, meaning that an entity need not belong to any of the subclasses.

Disjointness and Completeness constraints are independent, so we might have the following combinations of specializations:

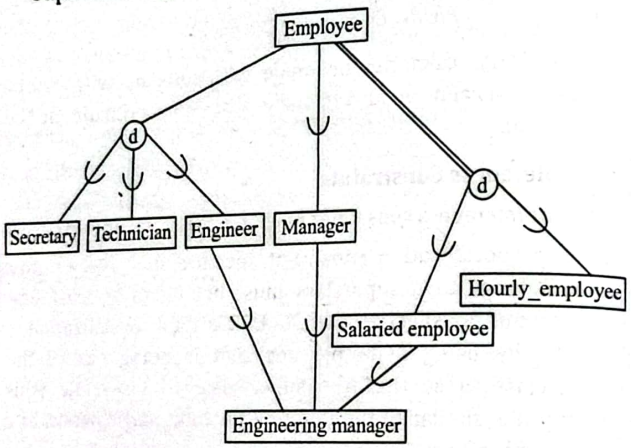
- Disjoint, total
- Disjoint, partial
- Overlapping, total
- Overlapping, partial

If we have a subclass that has multiple super classes, then the hierarchy is called a specialization lattice.

i.e., Multiple superclass/subclass connections for the same subclass. Having more than one superclass for the same subclass is also called multiple inheritance.

Example:

Engineer manager is an engineer, a manager, and a salaried employee at the same time. Here, Engineer manager has 3 superclasses (Manager, Engineer, Salaried_Employee).



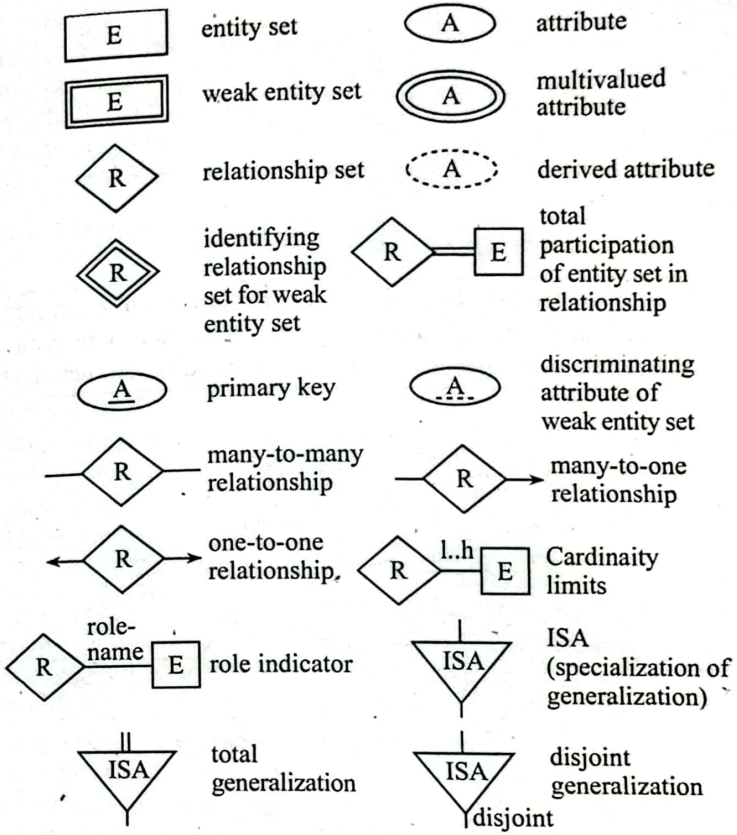
Here, we have 2 levels of superclass/subclass connections

2.2.5 Entity–Relationship Diagram

An entity relationship diagram (ERD) shows the relationship of entity sets of a database. E-R diagrams represents the logical ructure of a database. Following are the steps to create an ER agram:

a. Entity identification (Usually nouns)
b. Relationship identification (usually verbs)
c. Cardinality identification (1 to 1 or 1 to M)
d. Identity attribute (Mention all types of attributes)
e. Create ERD

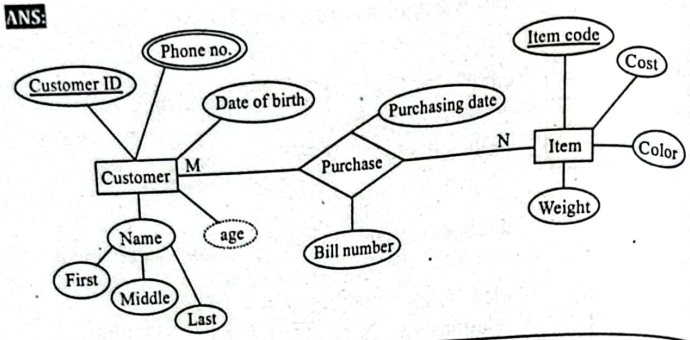
Symbols used in E-R diagrams are shown below:



Example:

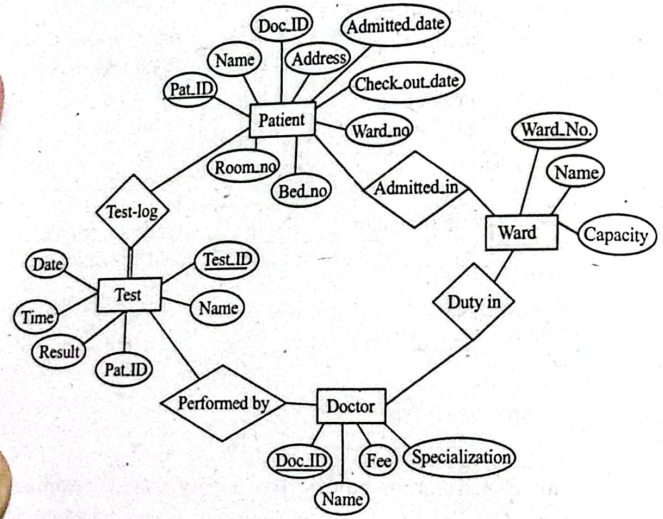
1. Make an E-R diagram having two entity sets, Customer and Item. Cardinality Ratio is many to many because a customer can buy any number of items and same item can be purchased by more than one customer.

ANS:



2. Construct an ER diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of various tests and examinations conducted.

ANS:



2.3 Relation with UML Class Diagrams

Class diagrams and ER diagrams both model the structure of a system. A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes,

operations (or methods), and the relationships among objects. Entity-relationship diagrams (ER diagrams) are graphical representations of entities and the relationships between them. ER diagrams are commonly used in database design to visualize the data requirements of a system or application However, both types of diagrams can be useful for understanding complex systems and can be used together in some cases, such as when designing an application that interacts with a database.

2.4 Alternate Data Models

2.4.1 Relational Data Model

This model was first proposed by Dr. E.F. Codd of IBM in 1970. The relational data model is based on the mathematical concept of a relation, which is physically represented as a table. In relational data structure, a relation is a table with columns and rows. An attribute is a named column of a relation. A domain is the set of allowable values for one or more attributes. A tuple is a row of a relation. The degree of a relation is the number of attributes it contains. The cardinality of a relation is the number of tuples it contains.

To understand the true meaning of the term *relation,* we have to review some concepts from mathematics. Suppose that we have two sets, M_1 and M_2, where $M_1 = \{2,4\}$ and $M_2 = \{1,3,5\}$. The Cartesian product of these two sets, written as $M_1 \times M_2$, is the set of all ordered pairs such that the first element is a member of M_1, and the second element is a member of M_2. An alternative way of expressing this is to find all combinations of elements with the first from M_1 and the second from M_2. In our case, we have:

$M_1 \times M_2 = \{(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)\}$

Any subset of this Cartesian product is a relation.

Example:

We could produce a relation R such that:

$R = \{(x, y) \mid x \in M_1, y \in M_2, \text{and } y = 1\}$

$R1 = \{(2, 1), (4, 1)\}$ is the relation.

Relation Schema

The name of the relation defined by a set of attribute and domain name pairs is called relation schema.

Let $A_1, A_2, \ldots\ldots, A_n$ be attributes with domains $D_1, D_2, \ldots\ldots, D_n$. Then the set $\{A_1:D_1, A_2:D_2, \ldots\ldots, A_n:D_n\}$ is a relation schema.

In general, we can define relation schema as name of the relation with its attributes.

Example:

Branch (branchNo, street, city, postcode) is a relation schema.

A relation R defined by a relation schema S is a set of mappings from the attribute names to their corresponding domains.

Thus, relation R is a set of n-tuples:

$(A_1:d_1, A_2:d_2, \ldots\ldots, A_n:d_n)$ such that $d_1 \in D_1, d_2 \in D_2, \ldots\ldots, d_n \in D_n$

Each element in the n-tuple consists of an attribute and a value for that attribute. Normally, when we write out a relation as a table, we list the attribute names as column headings and write out the tuples as rows having the form $(d_1, d_2, \ldots\ldots, d_n)$ where each value is taken from the appropriate domain. In this way, we can think of a relation in the relational model as any subset of the Cartesian product of the domains of the attributes. A table is simply a physical representation of such relation.

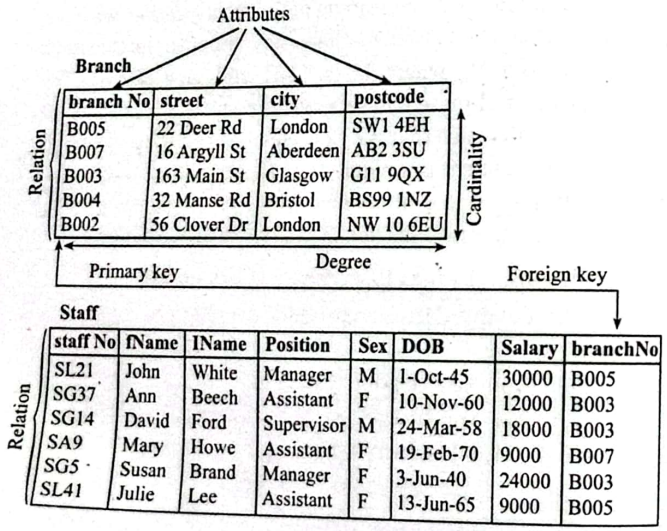


Figure: Relational data model (showing basic data structures)

In our example, the Branch relation shown in above figure has attributes branch No, street, city, and postcode, each with its corresponding domain. The Branch relation is any subset of the cartesian product of the domains, or any set of four-tuples in which the first element is from the domain branch No, the second is from the domain street, and so on.

One of the four-tuples is:

{(B005, 22 Deer Rd, London, SW1 4EH)}

or more correctly: {(branchNo: B005, street: 22 Deer Rd, city: London, postcode: SW1 4EH)}

We refer to this as a relation instance. A set of relation schemas, each with a distinct name is called Relational database schema.

If $R_1, R_2, \ldots\ldots, R_n$ are a set of relation schemas, then we can write the relational database schema, or simply relational schema, R, as:

$R = \{R_1, R_2, \ldots\ldots, R_n\}$

2.4.2 Hierarchical Data Model

Hierarchical data model is based on tree structure. A hierarchical database consists of collection of records, that are connected to each other by links.

Record

A record is a collection of attributes; each contains only one data value.

Link

A link is an association between two records. The tree structure used in hierarchical model is known as rooted tree. The root node of that tree is dummy node or an empty node. So, hierarchical model is a collection of rooted trees or database tree.

2.4.3 Network Data Model

As a result of limitations in the hierarchical model, designers developed the Network Data Model. The ability of this model to handle *many to many (N: N)* relations between its records is the

main distinguishing feature from the hierarchical model. Thus this model permits a child record to have more than one parent. In this model, directed graphs are used instead of tree structure in which a node can have more than one parent.

Network model is based on graph structure. A network database consists of collection of records, which are connected to each other by links.

Record

A record is a collection of attributes; each contain only one data value.

Link

A link is as association between two records.

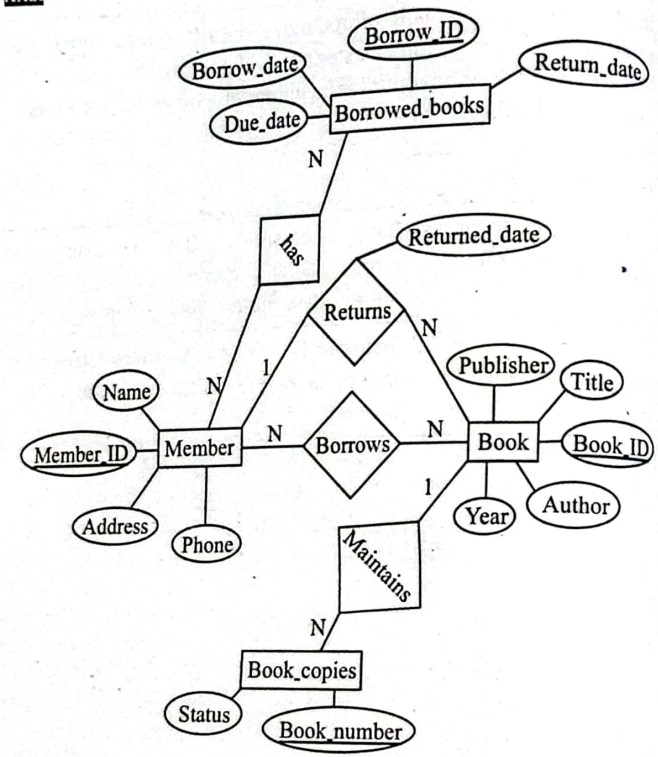
Comparison of three models

Hierarchical Model	Network Model	Relational Model
It is based on tree structure.	It is based on graph structure.	It is based on mathematical concept of relation.
Hierarchical database consists of collection of records, connected to each other by links.	Network database consists of collection of records, connected to each other by links.	Relational database consists of tables and data is stored in tabular form.
It is easy to understand and more efficient than Network data model.	It offers more flexibility than hierarchical model.	It offers simplicity in representing data than network and hierarchical model.
The hierarchical structure is asymmetric and is a major drawback that leads to unnecessary complications for the user.	The network structure is more symmetric than hierarchical. model.	The relational structure is more symmetric than network and hierarchical structure.
Queries are easy to write than in network model but more complicated than relational model.	Queries are more complicated to write than hierarchical and relational model.	Queries are easy to write than other models.
Information is replicated in hierarchical model which leads to inconsistency and wastage of space.	It offers more data consistency than hierarchical model.	It offers more data consistency than other two models.
It is difficult to access values at lower level.	Data accessing is more easier than hierarchical model.	Data accessing and navigation is easy than other models.
Structural independence is missing.	Structural independence is missing.	It offers structural independence.

SOLUTION TO EXAMS' AND OTHER IMPORTANT QUESTIONS

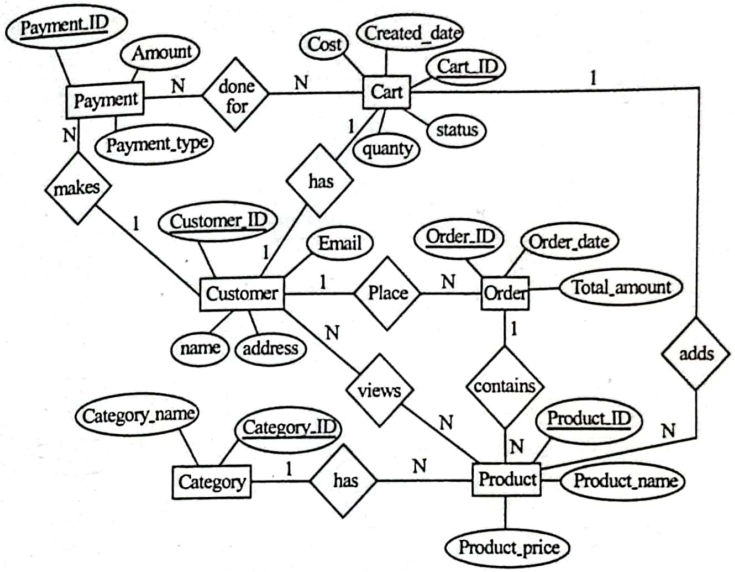
1. Construct an E-R diagram for keeping records for Library Management Systems. [2019 Spring, 2023 Spring]

ANS:



2. Draw an E-R diagram for online shop management system. Assume relevant entities and attributes for the given system. [2022 Fall]

ANS:

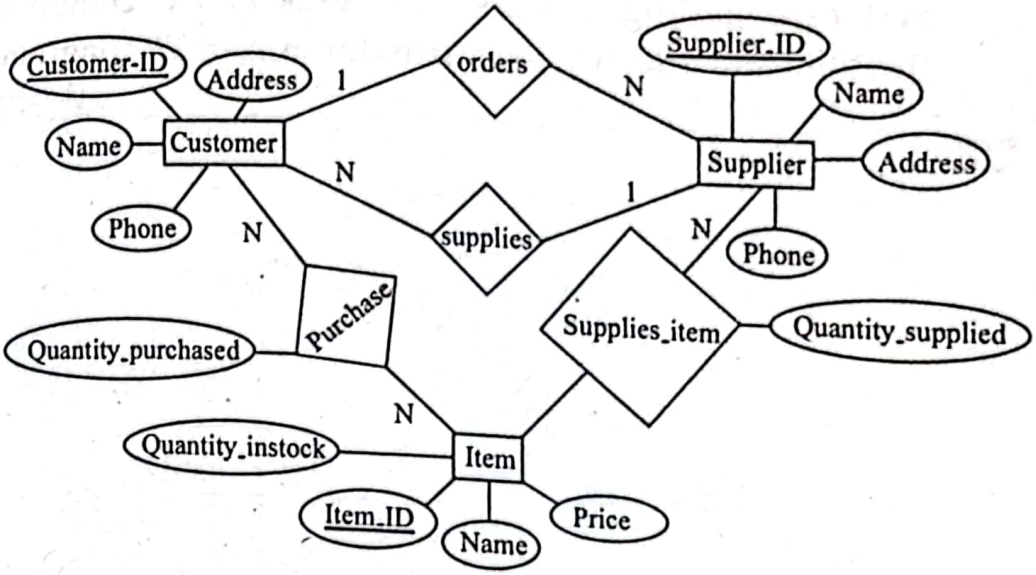


3. Differentiate between Data Model and ER model. Draw an E-R diagram for a Library Management System including primary key, weak entity, composite attribute, derived attribute and multivalued attribute in your ER diagram. [2021 Fall]

ANS:

The ER model is a particular kind of data model that is concentrated on defining entities, attributes, and relationships in relational database design. A data model is a general word that incorporates numerous sorts of models used to describe data.

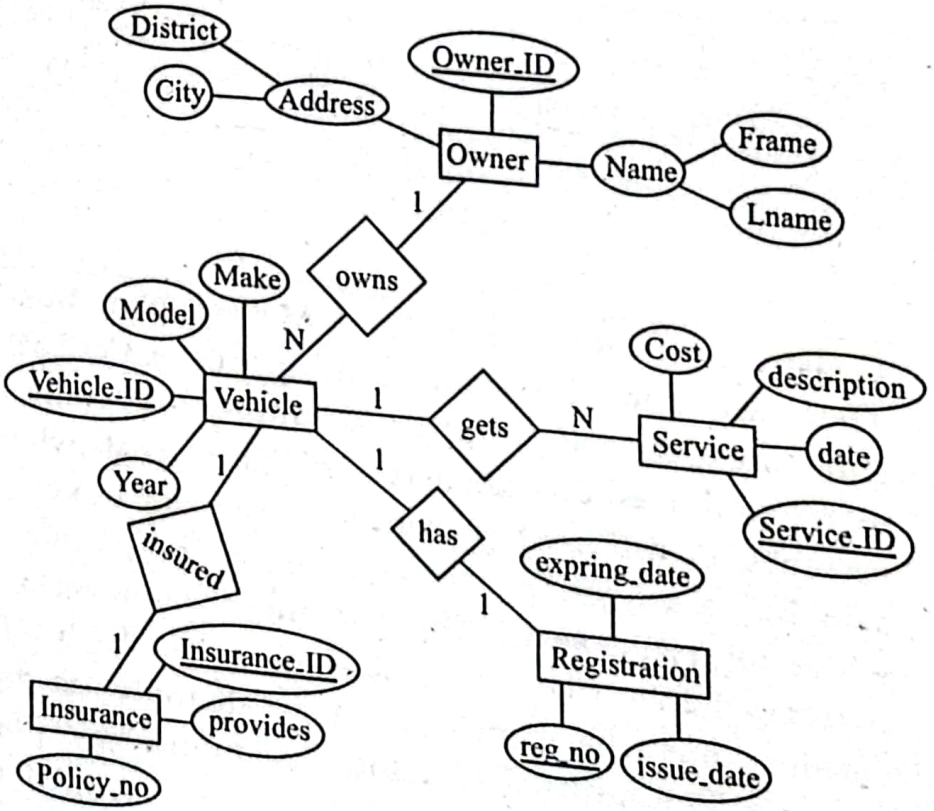
ANS:



6. Draw an E-R Diagram for a Vehicle Management System including primary key, weak entity, composite attribute, derived attribute and mulivalued attributes in your ER Diagram.

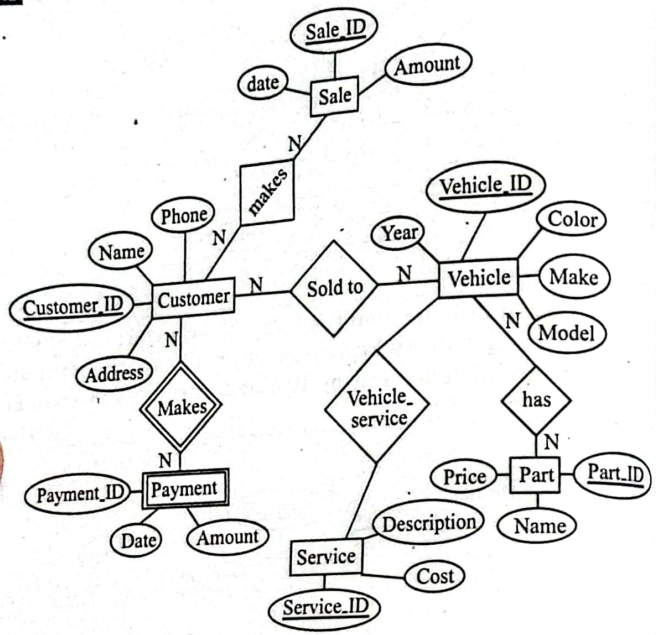
[2020 Fall]

ANS:



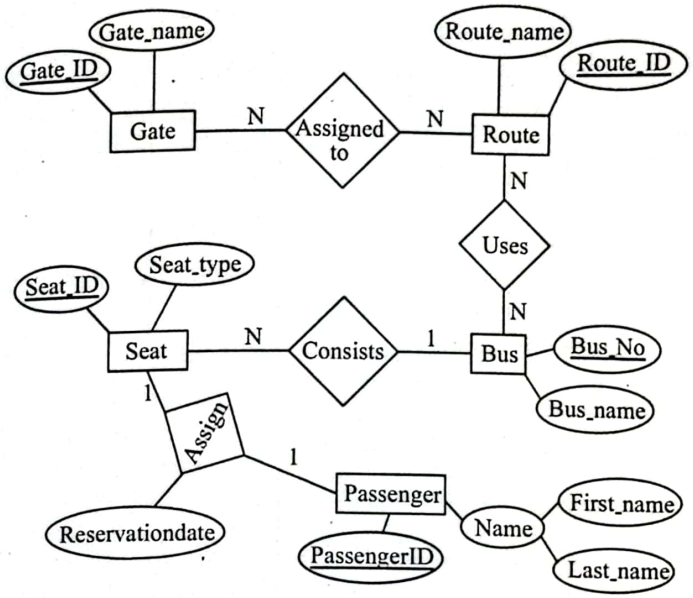
7. Draw an E-R diagram for a Gandaki Auto Vehicle Shop System including primary key, weak entity, composite attribute and multivalued attributes in your ER diagram. [2019 Fall]

ANS:



8. Construct an ER diagram for a Metropolitan Bus Park. There are many gates for entering bus park. Different gates are assigned to different routes. A route uses different buses. Bus consists of different seats which are assigned to different passengers. Frequent travelers are also in passenger. Associate a log of reservation date while reserving seats. The passenger name must have two attributes first_name & last_name. Each of the entities must have primary key attribute as far as possible. The cardinality mappings should be explained properly. [2018 Spring]

ANS:

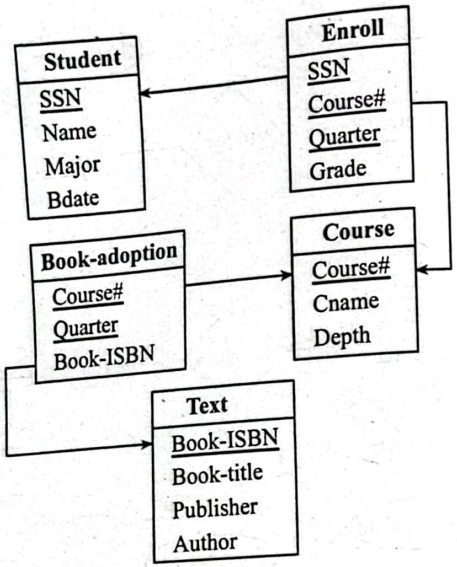


9. Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)
COURSE(Course#, Cname, Dept)
ENROLL(SSN, Course#, Quarter, Grade)
BOOK ADOPTION(Course#, Quarter, Book_ISBN)
TEXT(Book_ISBN, Book Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this schema. [2017 Spring]

ANS:

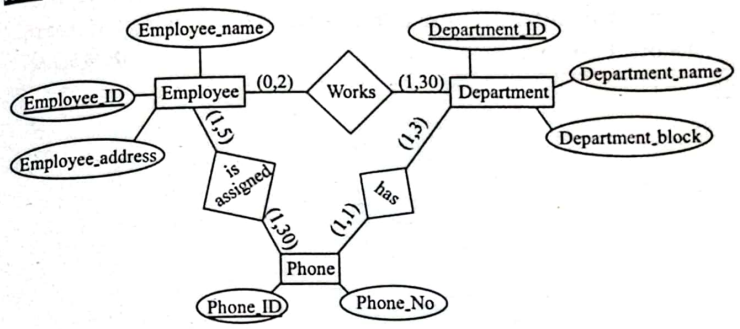


10. Suppose you are given the following requirements for a simple database for the Employee Management System:

i. An employee may work in up to two departments or may not be assigned to any department.
ii. Each department must have one and may have up to three phone numbers.
iii. Each department can have anywhere between 1 and 30 employees.
iv. Each phone is used by one, and only one, department.
v. Each phone is assigned to at least one, and may be assigned to up to 30 employees.
vi. Each employee is assigned at least one, but no more than 5 phones.

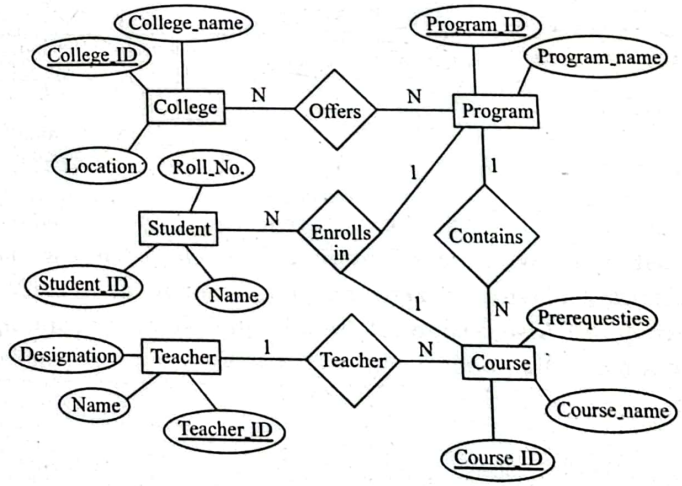
Construct a clean and concise ER diagram for the database. Clearly indicate the cardinality mappings. [2018 Fall]

ANS:



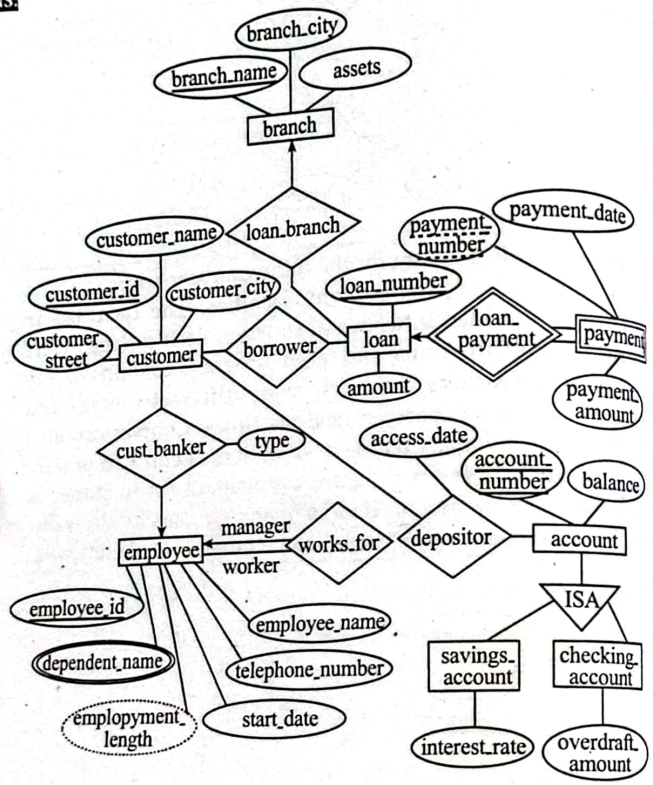
11. Draw an ER diagram for the following scenario.
A university contains many faculties. The faculties in turn are divided into several colleges. Each college offers numerous programs and each program contains many courses. Teachers can teach many different courses and even the same course numerous times. Courses can also be taught by many teachers. A student is enrolled in only one program but a program can contain many students. Students can be enrolled in many courses at the same time and the courses have many students enrolled. [2017 Fall]

ANS:



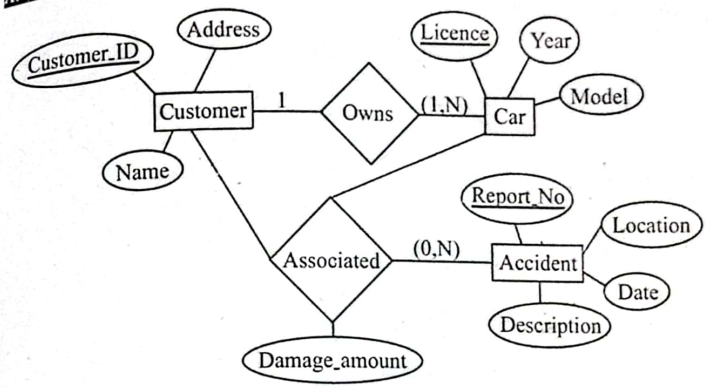
12. Construct an ER diagram for a banking enterprise that keeps the information about employee, customer, loan, account and payment. [2016 Spring]

ANS:



13. Construct E-R model for a car insurance company whose customer own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Also design relational database corresponding to the E-R diagram. [2016 Fall]

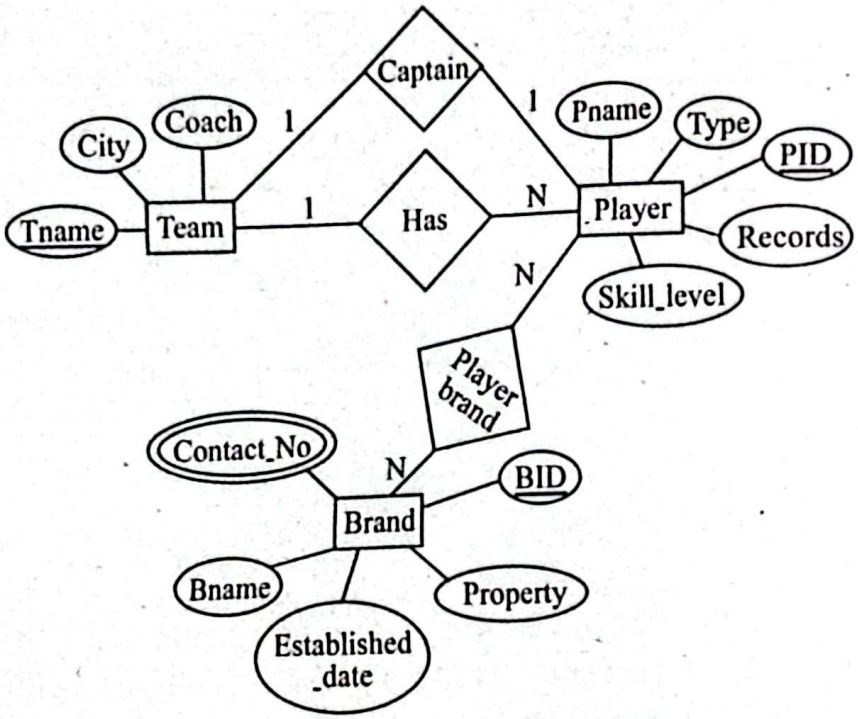
ANS:



14. Suppose you are given the following requirements for a simple database for the National Cricket League (NCL):
   - the NCL has many teams
   - each team has a name, a city, a coach, a captain, and a set of players
   - each player belongs to only one team
   - each player has a name, a type (such as batsman or bowler), a skill level, and a set of records
   - a team captain is also a player,
   - each player is sponsored by at least one brand
   - a brand has its name, established date, property, multiple contact_no.

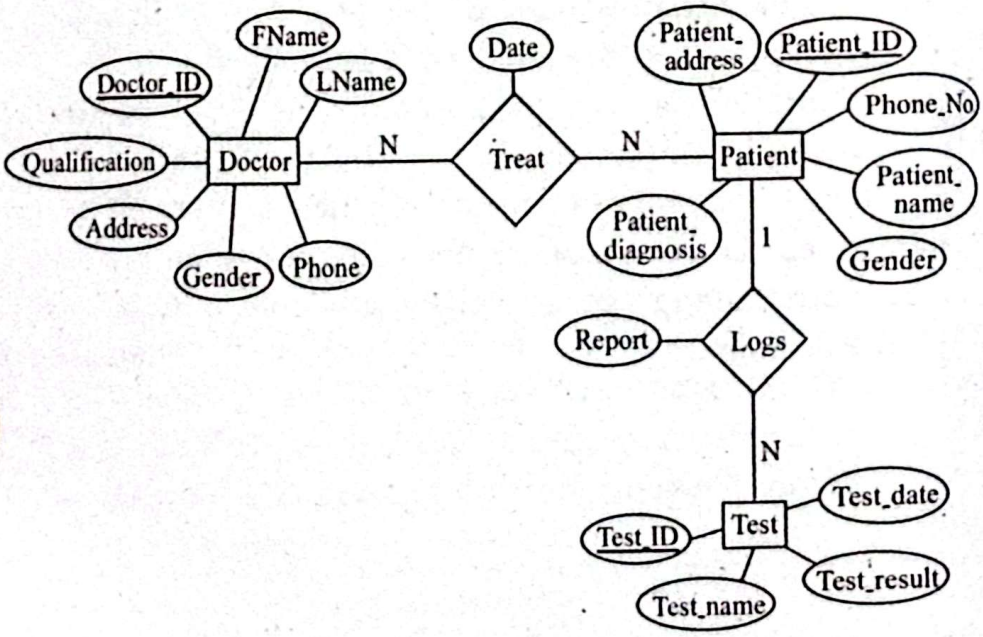
   Construct a clean and concise ER diagram for the NCL database. List your assumptions and clearly indicate the cardinality mappings as well as any role indicators in your ER diagram. [2015 Spring]

ANS:



15. Draw an ER diagram for the database of a hospital with a set of patients and a set of medical doctors. With each patient a log of the various tests conducted is also associated. Make your assumption if necessary. [2014 Fall]

ANS:



◆◆◆