CHAPTER

Two

DATA MODELING USING THE ENTITY-RELATIONSHIP(ER) MODEL

CHAPTER OUTLINE

2.1 HIGH-LEVEL CONCEPTUAL DATA MODELS FOR DATABASE DESIGN

High-level conceptual data models provide concepts for presenting data in ways that are close to the way people perceive data. A typical example is the entity relationship model, which uses main concepts like entities, attributes and relationships. An entity represents a real-world object such as an employee or a project. The entity has attributes that represent properties such as an employee's name, address and birthdate. A relationship represents an association among entities; for example, an employee works on many projects. A relationship exists between the employee and each project.

For designing a good database application this conceptual modeling is playing an important role. This conceptual data modeling is playing higher level this will present the idea about the Entity-Relationship Model.

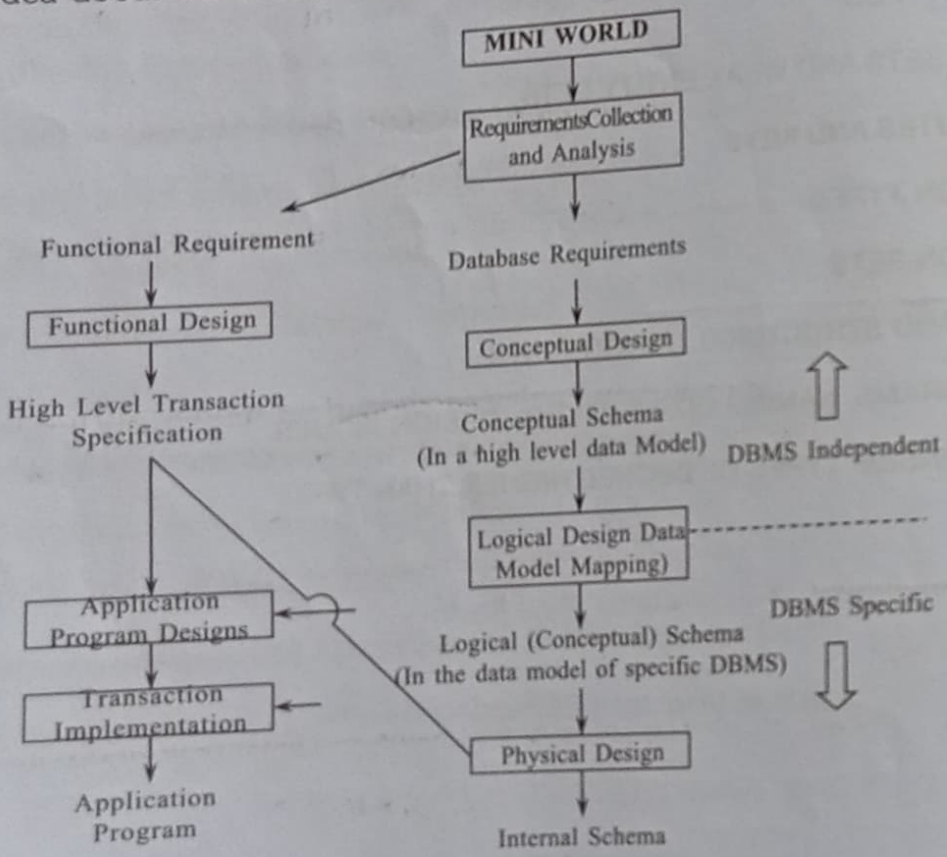


FIG : High Level Conceptual Data Models for Database Design

From the above diagram each step can be illustrated below.

Requirements Collection and Analysis :

- Prospective users are interviewed to understand and document data requirements

- This step results in a concise set of user requirements, which should be detailed and complete.
- The functional requirements should be specified, as well as the data requirements. Functional requirements consist of user operations that will be applied to the database, including retrievals and updates.
- Functional requirements can be documented using diagrams such as sequence diagrams, data flow diagrams, scenarios, etc.

Conceptual Design :

- Once the requirements are collected and analyzed, the designers go about creating the conceptual schema.
- **Conceptual schema :** concise description of data requirements of the users, and includes a detailed description of the entity types, relationships and constraints.
- The concepts do not include implementation details; therefore the end users easily understand them, and they can be used as a communication tool.
- The conceptual schema is used to ensure all user requirements are met, and they do not conflict.

Logical Design :

- Many DBMS systems use an implementation data model, so the conceptual schema is transformed from the high-level data model into the implementation data model.
- This step is called logical design or data model mapping, which results in the implementation data model of the DBMS.

Physical Design :

- Internal storage structures, indexes, access paths and file organizations are specified.
- Application programs are designed and implemented.

2.2 UNDERSTAND A DATABASE APPLICATION

A database application is a computer program whose primary purpose is entering and retrieving information from a computerized database. Early examples of database applications were accounting systems and airline reservations systems, such as SABRE, developed starting in 1957.

A characteristic of modern database applications is that they facilitate simultaneous updates and queries from multiple users. Systems in the 1970s might have accomplished this by having each user in front of a 3270 terminal to a mainframe computer. By the mid-1980s it was becoming more common to give each user a personal computer and have a program running on that PC that is connected to a database server. Information would be pulled from the database, transmitted over a network, and then arranged, graphed, or otherwise formatted by the program running on the PC. Starting in the mid-1990s it became more common to build database applications with a web interface rather than developing custom software to run on a user's PC, the user would use the same Web browser program for every application.

A database application with a Web interface had the advantage that it could be used on devices of different sizes, with different hardware, and with different operating systems. Examples of early database applications with Web interfaces include amazon.com, which used the Oracle relational database management system, the photo.net online community.

Electronic medical records are referred to on emrexperts.com, in December 2010, as "a software database application".

Some of the most complex database applications remain accounting systems, such as SAP, which may contain thousands of tables in only a single module. Many of today's most widely used computer systems are database applications, for example, Facebook, which was built on top of ~~MySQL.~~

The term ***"database application"*** comes from the practice of dividing computer software into systems programs, such as operating system, compilers, the file system, and tools such as the database management system, and application programs, such as a payroll check processor. On a standard PC running Microsoft Windows, for example, the Windows operating system contains all of the system programs like games, word processors, spreadsheet programs, photo editing programs, etc. would be application programs. As "application" is short for ***"application program"***, ***"database application"*** is short for ***"database application program"***.

List of database applications

- Amazon
- CNN(Canadian Neonatal Network)
- eBay
- Facebook

- Fandango
- Filemaker(Mac OS)
- Microsoft Access
- Oracle relational database
- SAP (Systems, Applications & Products in Data Processing)
- Ticketmaster
- Wikipedia
- Yelp
- YouTube

2.3 ENTITY TYPES

Entity Relationship (ER) Model :

The most popular high-level conceptual data model is the ER model. It is frequently used for the conceptual design of database applications.

The diagrammatic notation associated with the ER model, is referred to as the ER diagram. ER diagrams show the basic data structures and constraints.

The basic object of an ER diagram is the entity. An entity is a "thing" or object in the real world that is distinguishable from other objects. Examples of entities might be a physical entity, such as a student, a house, a product etc, or conceptual entities such as a company, a job position, a course, etc.

Entities have attributes, which basically are the properties/characteristics of a particular entity.

Examples of entities and attributes :

Entity	Attributes	Values
Car	Color	Red
	Make	Volkswagen
	Model	Bora
	Year	2000

Entity in DBMS can be a real-world object with an existence, For example, in a College database, the entities can be Professor, Students, Courses, etc.

Entities has attributes, which can be considered as properties describing it, for example, for Professor entity, the attributes are Professor_Name, Professor_Address, Professor_Salary, etc. The attribute value gets stored in the database.

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An Entity Type defines a collection or set of entities that have the same attributes. Each entity type in the database is described by its name and attributes. The entity share the same attributes, but each entity has its own value for each attribute. Entities are the instances of people, places, things, or events that are of interest. Individual entities are grouped into sets that are similar to one another.

Entity Type Example :

- **Entity Type :**
  - Student
- **Entity Attributes :**
  - Student ID,
  - Name,
  - Surname,
  - Date of Birth,
  - Department

Entity types are named after the entities that belong to the set of interest. It is a common convention that the names of Entity Types are singular and that, at least, the first letter is capitalized. For example,

1. An entity type named Student defines a collection of student entities.
2. An entity type named Invoice defines a collection of invoice entities.
3. An entity type named InvoiceLine defines a collection of invoice line entities.
4. An entity type named Product defines a collection of product entities.

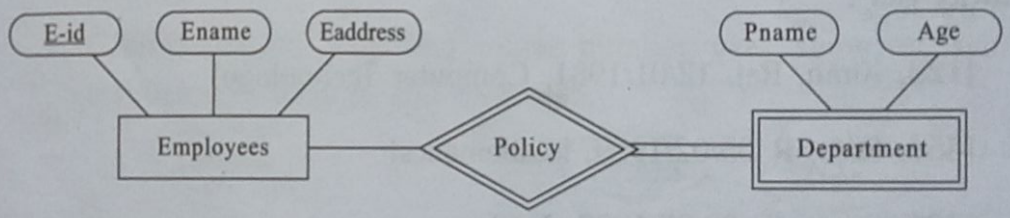
When we want to include a certain set of entities in our Entity Relationship Model, we define an Entity Type to represent that set.

Entity Types :

- Weak Entities
- Strong Entities
- Entity Sets
- Attributes
- Relationship Types
- Entity Relationship Diagrams

If the existence of an entity x depends on the existence of another entity y, then x is said to be existence dependent on y.

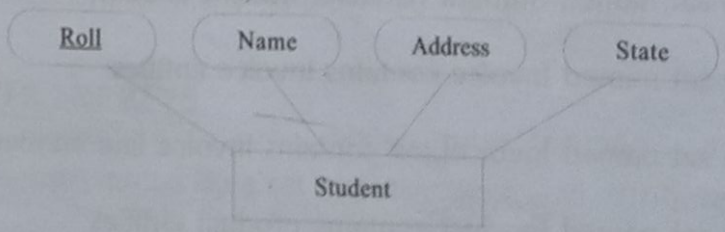
- Operationally y is deleted then entity y is said to be dominated entity and x is said to be subordinate entity.



Weak Entity type : The entity types that do not have key attributes of their own are called weak entity.

- A weak entity type always has a total participation constraint with respect to its identifying relationship because a weak entity cannot be identified without an owner entity
- We can represent a weak entity set by double rectangle
- We underline the discriminator of a weak entity set with a dashed line.

Strong entity type : The entity type that do have a key attributes are called strong entity type.



2.4 ENTITY SETS AND WEAK ENTITY SETS

Entity Sets : The collection of all entities of a particular entity type in the database at any point in time is called an entity set. The entity type (Student) and the entity set (Student) can be referred using the same name.

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

Entity Set Example :

- **Entity Type :** Student
- **Entity Set :**

  [123, Kiran, Raj, 12/01/1981, Computer Technology]

  [456, Raju, P, 05/02/1989, Mathematics]

  [789, Srinu, S, 02/08/1987, Arts]

The entity type describes the intension, or schema for a set of entities that share the same structure. The collection of entities of a particular entity type is grouped into the entity set, called the extension.

An Entity Set is a collection of related entities all of the same type. An Entity Set is also known as the Extension of an entity type.

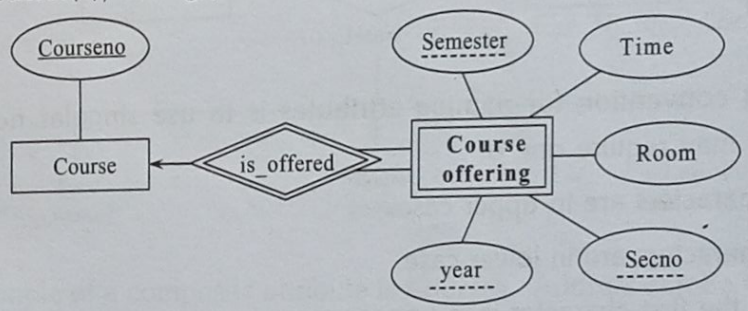
Entity sets and entity types are named after the entities that belong to the set. It is a common convention that entity set names are singular and that, at least, the first letter is capitalized. For example,

1. An entity set named Student contains student entities.
2. An entity set named Invoice contains invoice entities.
3. An entity set named InvoiceLine contains invoice line entities.
4. An entity set named Product contains product entities.

Weak Entity :

Sometimes we know certain entities only exist in relationship to others. For example, at a university or college we are likely to have two entity sets : Course and CourseOffering. Suppose an instance of Course is ***"Advanced Database"*** and an instance of CourseOffering is ***"Advanced Database - Section 3"***. A reasonable business rule is that the Course Offering can only exist if the corresponding Course exists. Another business rule could be that to identify an instance of CourseOffering we also need to know the identifier of the Course. In our example, suppose the identifier of the Course is its name ***"Advanced Database"*** and the identifier for the Course Offering includes that name : ***"Advanced Database - Section 3"***. In these circumstances we say the entity set CourseOffering is a weak entity set and the entity set Course is a strong (or regular) entity set. In the diagram following, we show the convention for indicating a weak entity set : a double-lined rectangle.

- Course, including number, title, credits, syllabus, and prerequisites;
- CourseOffering, including course number, year, semester, section number, instructor(s), timings, and classroom;



Weak entities will always participate in one or more identifying relationships.

Strong Entity :

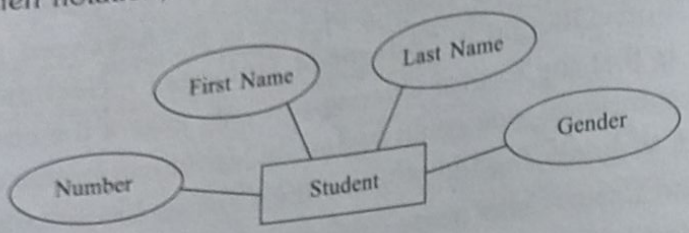
If an entity type has a key attribute specified then it is a strong entity type. For example, at a university or college we have student entities and we would define a student entity type. Universities and Colleges assign unique student numbers, one per student. Student has a key attribute and would be considered a strong entity type.

2.5 ATTRIBUTES AND KEYS

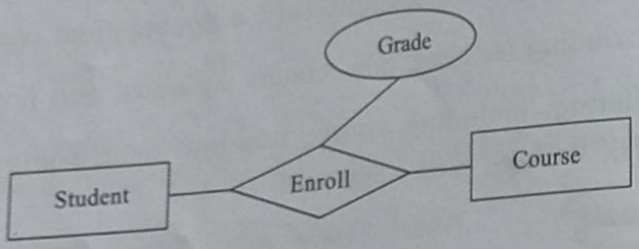
Each entity is represented by a set of properties called **attributes.**

Attributes are the characteristics that describe entities and relationships. For example, a Student entity may be described by attributes including: student number, name,

address, date of birth, degree. An Invoice entity may be described by attributes including: invoice number, invoice date, invoice total. In an Entity Relationship Diagram using the Chen notation, we illustrate attributes using ovals as shown below.



Suppose we have entity types Student and Course that are associated via an enroll relationship. An attribute that helps to describe a student enrolled in a course is grade :



A common convention for naming attributes is to use singular nouns. A naming convention may require one of :

- All characters are in upper case.
- All characters are in lower case.
- Only the first character is in upper case.
- Each part of a multipart name has the first character capitalized.

A typical convention is for attribute names to have a prefix that indicates the entity the attribute describes. Subsequent characters are sufficiently descriptive to identify the attribute. Some examples of attribute names :

- empLname = employee last name
- stuGpa = student grade point average
- prodCode = product code
- invNum = invoice number

In practice a naming convention is important, and you should expect the organization you are working for to have a standard approach for naming things appearing in a model. A substantial data model will have tens, if not into the hundreds, of entity sets,

many more attributes and relationships. It becomes important to easily understand the concept underlying a specific name; a naming convention can be helpful.

Types of Attributes :

Key Attributes : An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely.

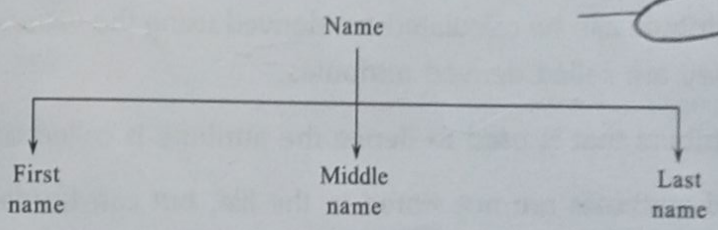
Simple Attributes : Attributes that cannot be divided into sub parts are called simple attributes or atomic attributes.

Eg : Roll number is example of simple attribute

Composite Attributes : Attributes that can be divided into sub parts are called composite attribute.

Eg : Date of Birth is example of composite attribute, because it may be divided into ***"Birth day"***, ***"Birth month"*** and ***"Birth year"***.

OR Name is also an example of composite attribute.



- Other example of a composite attribute is Address. Address can be broken down into a number of subparts, such as Street Address, City, and Postal Code. Street Address may be further broken down by Number, Street Name and Apartment/Unit number.
- Composite attributes can be used if the attribute is referred to as the whole, and the atomic attributes are not referred to. For example, if you wish to store the Company Location, unless you will use the atomic information such as Postal Code, or City separately from the other Location information (Street Address etc) then there is no need to subdivide it into its component attributes, and the whole Location can be designated as a simple attribute.

Single-Valued Attributes : An attribute which can assume one and only one value is called ***"Single-valued attribute".***

For example : Age of a person is an example of single valued Attribution.

Multi-valued Attribute : An attribute which may assume a set of values is called "Multi-valued Attribute".

e.g., An Employee entity set with the attribute phone-number is an example of multivalue attributes because an employee may have zero, one or many phone numbers.

- Most attributes have a single value for each entity, such as a car only has one model, a student has only one ID number, an employee has only one data of birth. These attributes are called single-valued attributes.
- Sometimes an attribute can have multiple values for a single entity, for example, a doctor may have more than one specialty (or may have only one specialty), a customer may have more than one mobile phone number, or they may not have one at all. These attributes are called multi-valued attributes.
- Multi-valued attributes may have a lower and upper bounds to constrain the number of values allowed. For example, a doctor must have at least one specialty, but no more than 3 specialties.

Stored vs. Derived Attributes :

- If an attribute can be calculated are derived using the value of another attribute, then they are called derived attributes.
- The attribute that is used to derive the attribute is called a stored attribute.
- Derived attributes are not stored in the file, but can be derived when needed from the stored attributes.

Eg : In a payroll system the HRA and PE are derived from salary attributes.

Null Valued Attributes :

- There are cases where an attribute does not have an applicable value for an attribute. For these situations, the value null is created.
- A person who does not have a mobile phone would have null stored at the value for the Mobile Phone Number attribute.
- Null can also be used in situations where the attribute value is unknown. There are two cases where this can occur, one where it is known that the attribute is valued, but the value is missing, for example hair color. Every person has a hair color, but the information may be missing. Another situation is if mobile phone number is null, it is not known if the person does not have a mobile phone or if that information is just missing.

Complex Attributes :

- Complex attributes are attributes that are nested in an arbitrary way.
- For example a person can have more than one residence, and each residence can have more than one phone, therefore it is a complex attribute that can be represented as:
- {Multi-valued attributes are displayed between braces}
- (Complex Attributes are represented using parentheses)

E.g. {AddressPhone({Phone(AreaCode, PhoneNumber)}, Address(StreetAddress(Number, Street, ApartmentNumber), City, State, Zip))}

Keys : Key is a field that uniquely identifies the records, tables or data.

In the relational data model there are many keys. Some of these keys are :

- Primary key
- Foreign key
- Unique key
- Super key
- Candidate key

Primary Key : A primary key is one or more column(s) in a table used to uniquely identify each row in the table.

Ex : Student

Roll No	Name	Address
1	Vijay	Noida
2	Sandeep	G.Noida
3	Ajay	Noida

Roll No. is a primary key

Note : Primary key cannot contain Null value.

Unique Key : Unique key is one or more column(s) in a table used to uniquely identify each row in the table. It can contain NULL value.

Roll No	Name	Address
1	Vijay	Noida
-	Sandeep	G.Noida
3	Ajay	Noida

Roll - No. is unique key.

Difference between primary key and Unique key

- Unique key may contain NULL value but primary key can never contain NULL value.

Super Key : A super key is a set of one or more attributes that, taken collectively; allow us to identify uniquely a tuple in the relation.

Eg : {Roll-No}, {Roll-No, Name}, {Roll-No, Address}, {Roll-No, Name, Address} all these sets are super key.

Candidate Key : If a relational schema has more than one key, each is called a candidate key.

- All the keys which satisfy the condition of primary key can be candidate key.

Roll No	Name	Phone No	City
112	Vijay	578910	Noida
113	Gopal	558012	Noida
114	Sanjay	656383	G.Noida
115	Santosh	656373	G.Noida

{Roll-No} and {Phone-No} are two candidate key. Because we can consider anyone of these as a primary key.

Foreign Key : Foreign keys represent relationships between tables.

- A foreign key is a column whose values are derived from the primary key of some other table.

OR

- A foreign key is a key which is the primary key in the one table and used to relate with some attributes in other table with same data entry.

Ex : Student and Marks table

Roll_No	Name	Subject
1	Vijay	Computer
2	Gopal	Math
3	Sanjay	Physics

Roll_No	Sem	Marks
1	III	80
2	V	75
3	VII	70

Here Roll_No of marks table is foreign key.

2.6 RELATION TYPES

Whenever an attribute of one entity type refers to another entity type, some relationship exists.

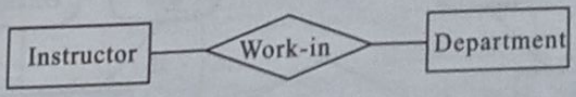
For example, Manager of a department refers to an employee who manages the department, Controlling Department of the project, refers to the department that controls the project. Supervisor of an employee refers to the employee who supervises that employee.

As an example, one can imagine a STUDENT entity being associated to an ACADEMIC_COURSE entity via, say, an ENROLLED_IN relationship. Whenever an attribute of one entity type refers to an entity (of the same or different entity type), we say that a relationship exists between the two entity types.

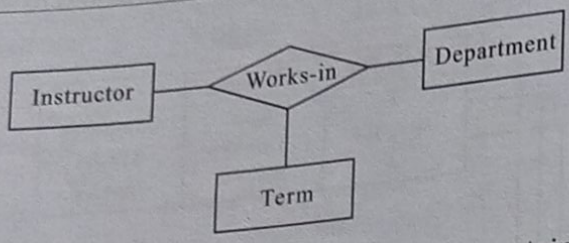
In an ER diagram, these references are not represented as attributes, but as relationships.

A Relationship Type defines a relationship set among entities of certain entity types

A relationship type is illustrated in an ER Diagram using a diamond symbol. Consider the following diagram that shows two entity types (Department and Instructor) and the concept (a relationship) that instructors work in departments. This relationship involves two entity types and is referred to as a binary relationship.

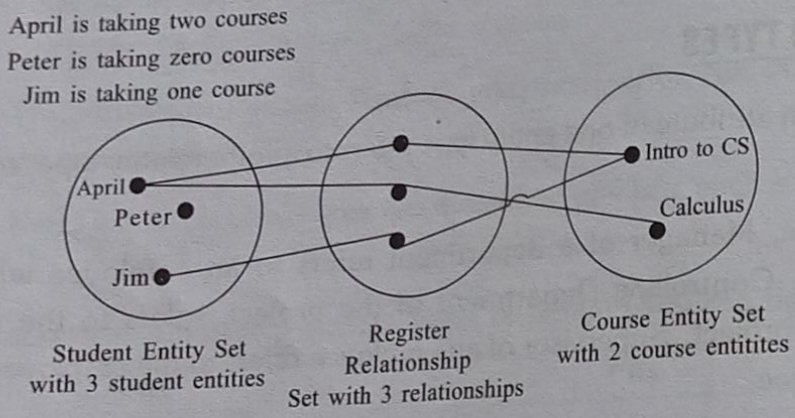


Below is a ternary relationship (a relationship type involving three entity types).



Relationship types are useful for capturing/expressing certain business rules.

The next diagram shows two entity sets and one relationship set, and following diagram shows an ERD. Each set comprises a number of instances. In the ERD we define a relationship type for ***"register"*** and two entity types for ***"student"*** and ***"course"***.



2.7 RELATION SETS

A relationship set is an association among two or more entities.

- A relationship set is a set of relationship of the same type of the same value.
- A mathematically relation on n >= 2 entity set. If E_1, E_2,..... En are entity sets then a relationship set are a subset of {(e_1, e_2,....., e_n)/$e_1e_1 \in E_1$, $e_2 \in E_2$,....., $e_n \in E_n$}. Where (e_1, e_2, e_3, ... , e_n) is a set of relationship.

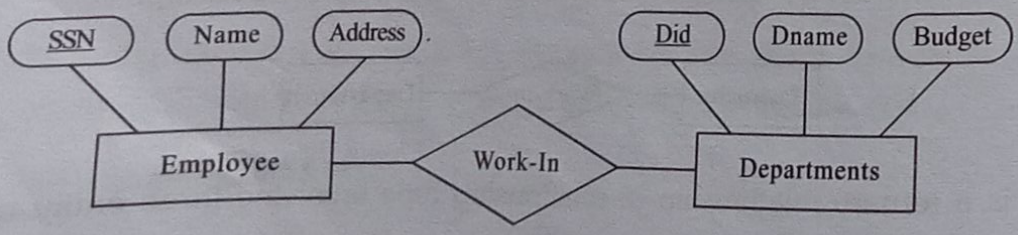


FIG : Diagram on Relation Sets

2.8 ROLES AND STRUCTURAL CONSTRAINTS

Role Names : Each entity type that participates in a relationship type plays a particular role in the relationship. The role name shows the role that a particular entity from the entity type plays in each relationship.

Example : In the Company diagram, in the WorksFor relationship type, the employee plays the role of employee or worker, and the department entity plays the role of department or employer.

In some cases the same entity participates more than once in a relationship type in different roles.

For example, the Supervision relationship type relates an employee to a supervisor, where both the employee and supervisor are of the same employee entity type, therefore the employee entity participates twice in the relationship, once in the role of supervisor, and once in the role of supervise.

Structural constraints :

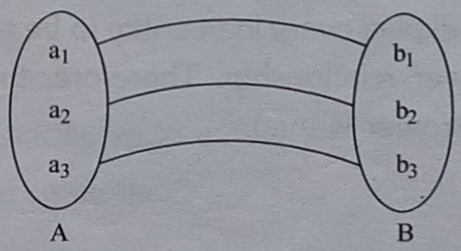
An E-R enterprise scheme may define certain constraints to which the contents of a database must confirm.

Constraints
- Mapping cardinalities
- Participation constraints

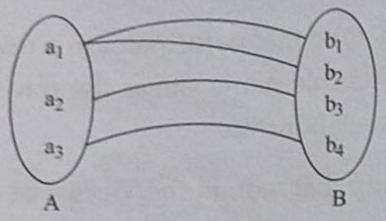
Mapping Cardinalities : Mapping cardinalities or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set mapping cardinalities are most useful in describing binary relationship sets.

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

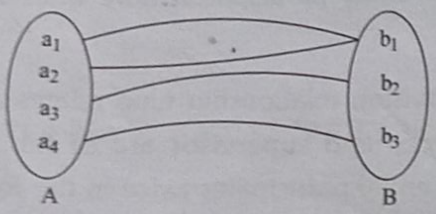
1. **One to One :** An entity in A is associated with exactly one entity in B. And one entity in B is associated with exactly one entity in A.



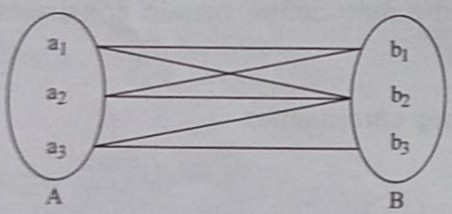
2. **One to Many :** An entity in A is associated with any number of entities in B. An entity in B can be associated with at most one entity in A.



3. **Many to One :** An entity in A is associated with at most one entity in B. An entity in B, can be associated with any number of entity in A.



4. **Many to Many :** An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

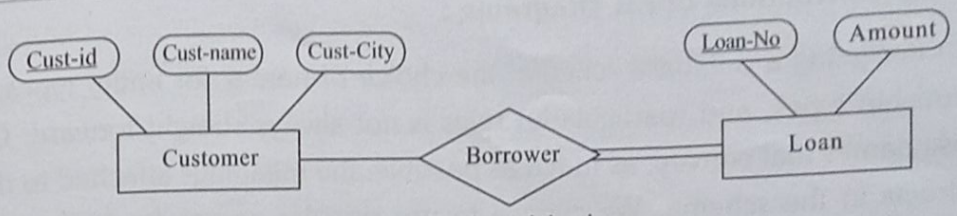


Participation Constraints : Participation constraints specifies whether the existence of an entity set depends on its being related to another entity.

Participation constraints
- Total
- Partial

(i) Total Participate : The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.

For Example : We expect every loan entity to be related to at least one customer through the borrower relationship. Therefore, the participation of loan in the relationship set borrower is total.



(ii) **Partial Participate :** If only some entities in E participate in relationships in R. The participation of entity set E in relationship R is said to be partial participation.

For Example : An individual can be a bank customer whether or not he has a loan with the bank.

Therefore the participation of customer in the borrower relationship set in partial.

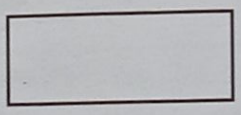
2.9 ER DIAGRAMS, NAMING CONVENTIONS, DESIGN ISSUES

Entity relationship diagram (ER diagram)

- The overall logical structure of a data base can be expressed graphically by an ER diagram.
- It consists of following components.

1. **Rectangle :** It represents entity, entity sets.

   Shape of rectangle :



2. **Elipse :** It represents attributes.

3. **Diamond (or) Rhombus :**

   It represents relationship among entity sets.

4. **Lines :** It is used to link attributes to entitys sets and entity sets to relationship.

   Ex : Draw ER diagram of students.

Naming Conventions of ER Diagrams :

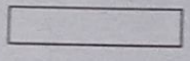
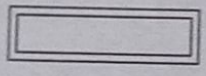
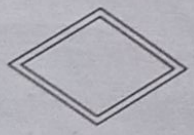
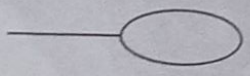
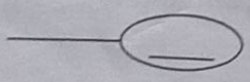
When designing a database schema, the choice of names for entity types, attributes, relationship types, and (particularly) roles is not always straight forward. One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema. We choose to use singular names for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type. In our ER diagrams, we will use the convention that entity type and relationship type names are uppercase letters, attribute names have their initial letter capitalized, and role names are lowercase letters.

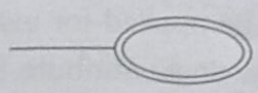
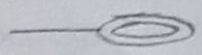
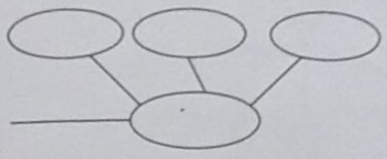
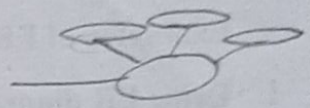
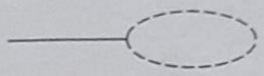
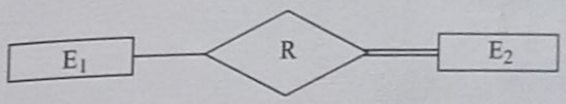
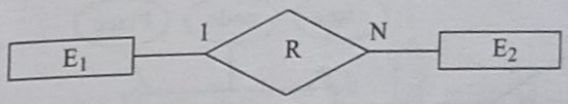
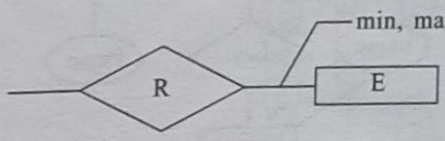
As a general practice, given a narrative description of the database requirements, the nouns appearing in the narrative tend to give rise to entity type names, and the verbs tend to indicate names of relationship types. Attribute names generally arise from additional nouns that describe the nouns corresponding to entity types.

Another naming consideration involves choosing binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom.

Design Choices for ER Conceptual Design :

It is occasionally difficult to decide whether a particular concept in the real world should be modeled as an entity type, an attribute, or a relationship type.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute

	Multivalued Attribute 
	Composite Attribute 
	Deriver Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1 : N for E_1 : E_2 in R
	Structural Constraint (min, max) on Participation of E in R

In general, the schema design process should be considered an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.

A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type. It is often the case that a pair of such attributes that are inverses of one another are refined into a binary relationship. It is important to note that in our notation, once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy.

Similarly, an attribute that exists in several entity types may be elevated or promoted to an independent entity type. For example, suppose that several entity types in a University database, such as Student, Instructor, and Course, each has an attribute Department in the initial design; the designer may then choose to create an entity type Department with a single attribute Dept_name and relate it to the three entity types (Student, Instrutor, and Course) via appropriate relationships. Other attributes/ relationships of Department may be discovered later.

An inverse refinement to the previous case may be applied-for example, if an entity type Department exists in the initial design with a single attribute Dept_name and is related to only one other entity type, Student. In this case, Department may be reduced or demoted to an attribute of Student.

Examples of ER Diagrams :

I. Draw ER diagram of Students :

Entities :		**Attributes :**
1. Students	→	Pin, Name, Marks, Age
2. College	→	Name, Code, Place, Ph.No
3. Branch	→	Name, Code

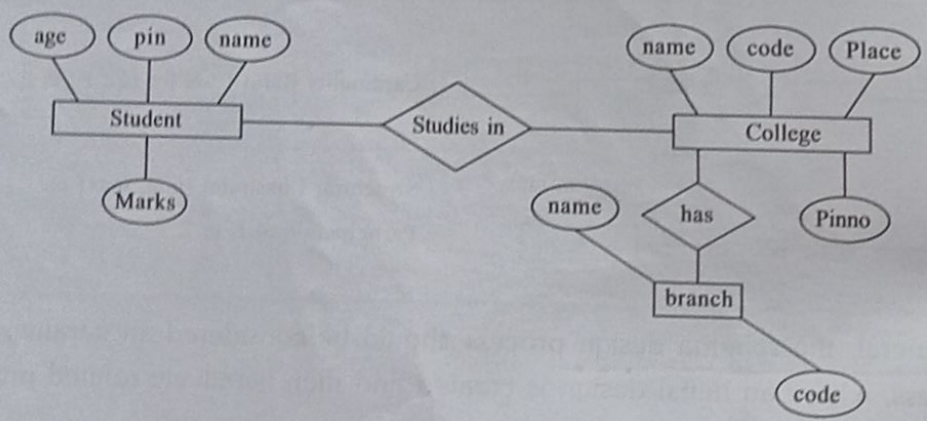


FIG : ER Diagram of Student

II. Draw ER Diagram for University

Entities :		**Attributes**
1. University	→	Name, place, phno.code
2. College	→	Name, code, place, ph,no
3. Principal	→	Name, id, phno, qualities
4. Department	→	ID, location, HOD
5. Student	→	Pin, name, marks, age
6. Faculty	→	Name, Id, phno
7. Course	→	ID, Name
8. Section	→	ID, Name

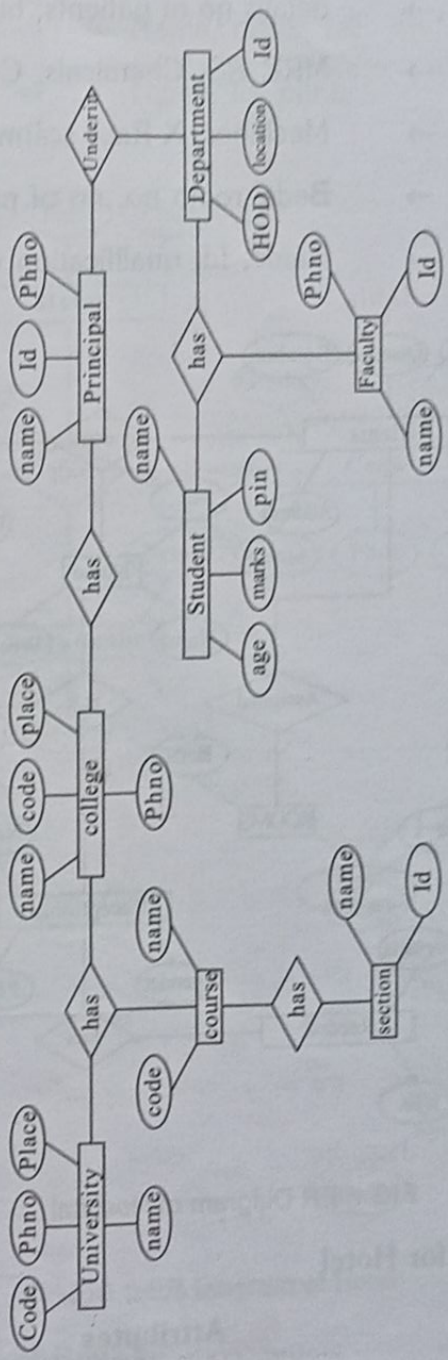


FIG : ER Diagram of University

Draw ER Diagram for Hospital

Entities		Attributes
→ Doctor	→	name, id, phno, qualificaiton
→ Patients	→	name, disease, room no, addres
→ Employees	→	name, work, phno, clerk
→ Receptionist	→	name, phno, gender

→ Record → details no of patients, bills
→ Medicine → MRP, RS, Chemicals, Company
→ Equipment → Mechines, X-Rays scanners
→ Room → Beds, room no, no of patients
→ Nurse → Name, Id, qualification wore

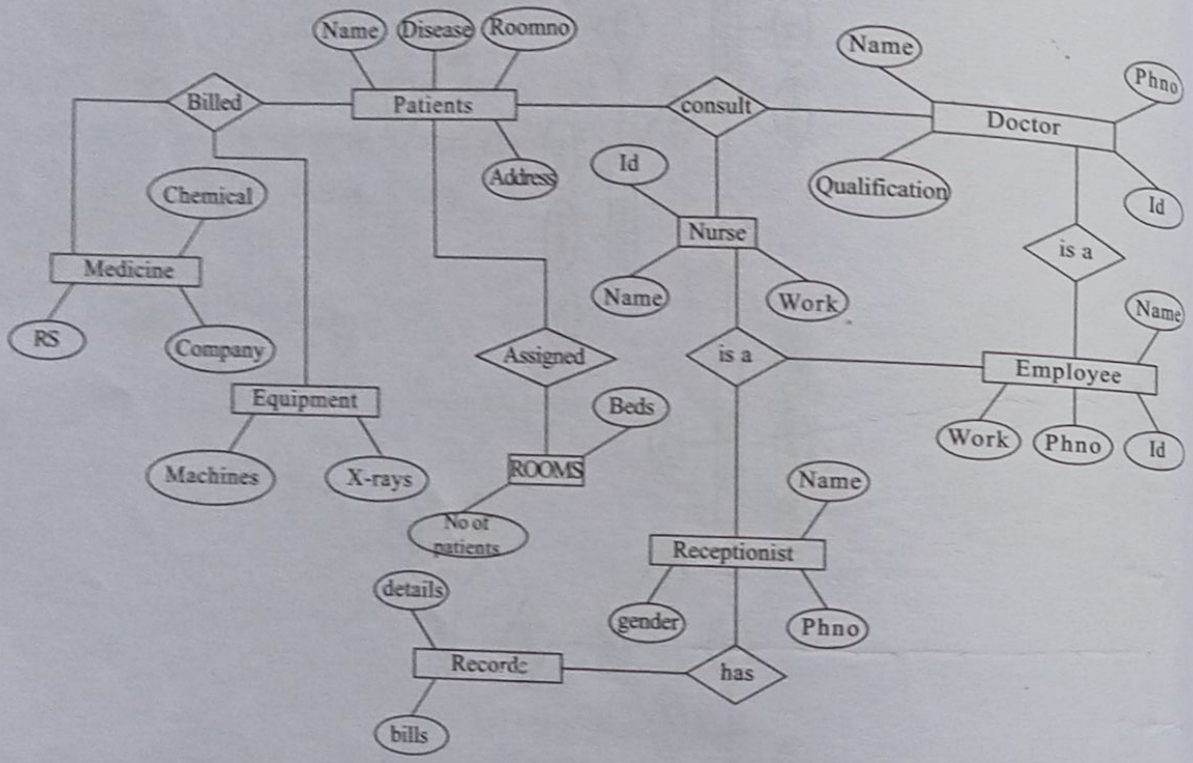


FIG : ER Diagram of Hospital

IV. Draw an ER Diagram for Hotel

Entities		Attributes
→ Hotel	→	Name, Room, place, phno
→ Cook	→	Name, Id, phno, salary, gender
→ Room	→	AC, Non A/C, roomno
→ Receptionist	→	Name, id, salary, gender
→ Food	→	Veg, Non-veg
→ Server/Baser	→	Name, id, salary

→ Customer → Name, order, bill

→ Clerk → Name, id, phno

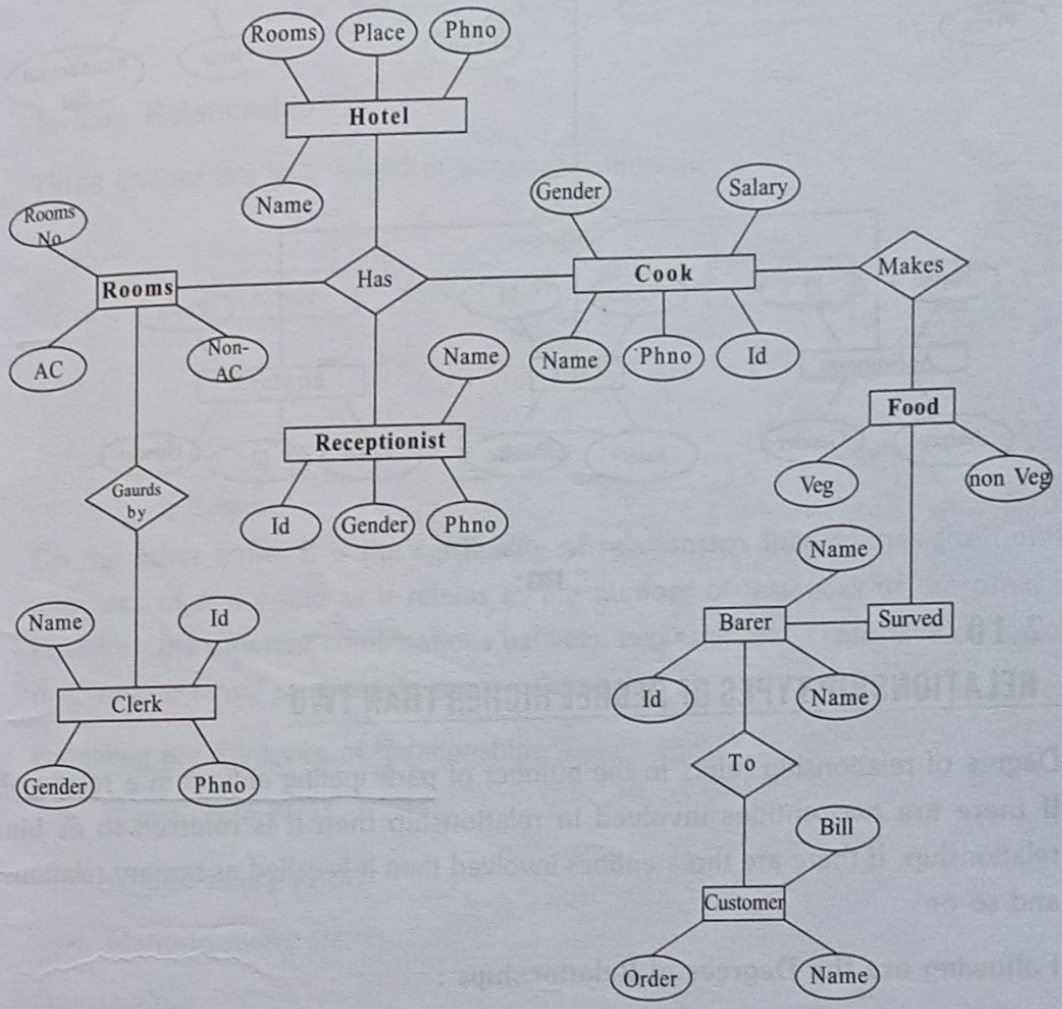


FIG : ER Diagram of Hotel

V. Draw an ER diagram of Bank :

Entities		Attributes
→ Bank	→	name, place, ph.no
→ Accountant	→	name, phno, Id, salary, gender
→ Customer	→	name, account, no, ph.no
→ Employee	→	name, Id, ph.no, gender
→ Clerk	→	name, Id, ph.no
→ Branch	→	name, place, ph.no

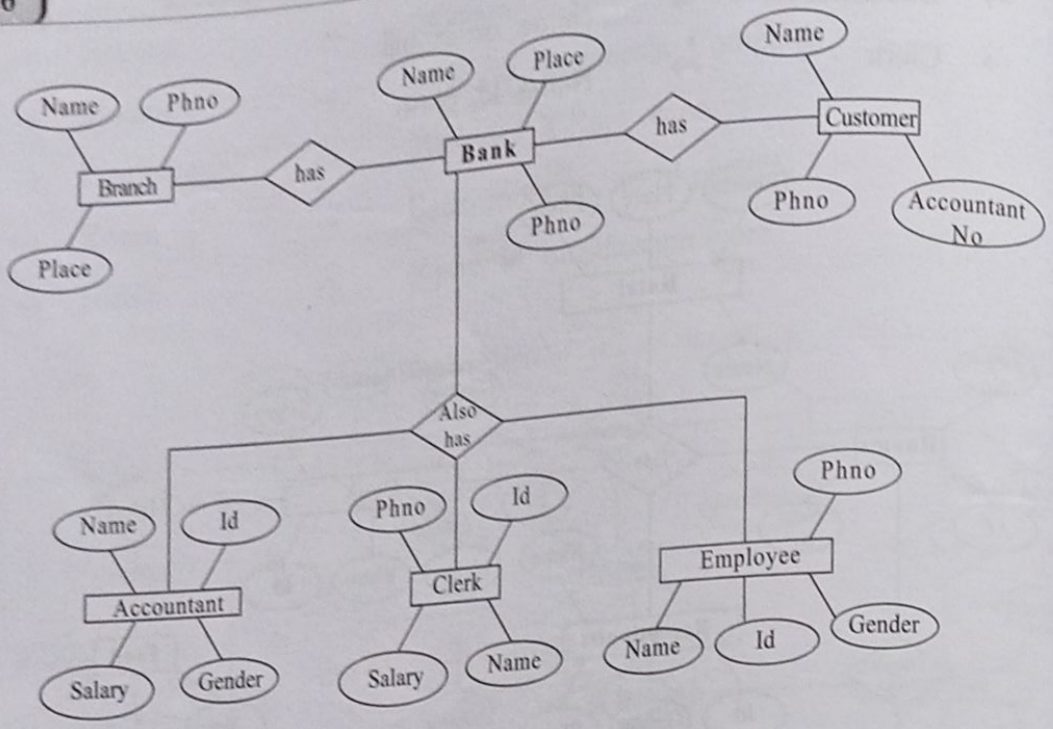


FIG:

2.10 RELATIONSHIP TYPES OF DEGREE HIGHER THAN TWO

Degree of relationship refers to the number of participating entities in a relationship. If there are two entities involved in relationship then it is referred to as binary relationship. If there are three entities involved then it is called as ternary relationship and so on.

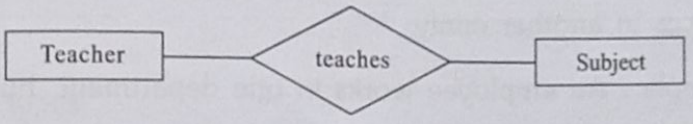
Following are the Degrees of Relationships :

1. Single Entity - Unary
2. Double Entities - Binary
3. Triple Entities - Ternary
4. N Entities - N- ary

Relationship types of degree 2 are called **binary**, Relationship types of degree 3 are called ternary and of degree n are called n-ary. In general, an n-ary relationship is not equivalent to n binary relationships. Constraints are harder to specify for higher-degree relationships ($n > 2$) than for binary relationships. If needed, the binary and n-ary relationships can all be included in the schema design. In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types).

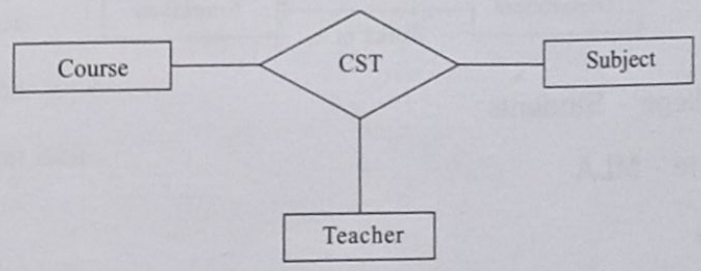
Binary Relationship

Two entities are involved in binary relationship.



Ternary Relationship

Three entities are in involved in ternary relationship



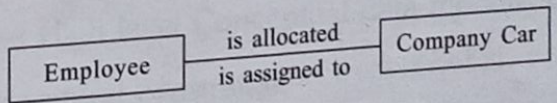
On the other hand, it is the cardinality of relationship that defines the number of instances of one entity as it relates to the number of instances of the other entity. Based on the different combinations between two entities we can have either one-to-one, one-to-many or many-to-many relationship.

Following are the types of Relationships.

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

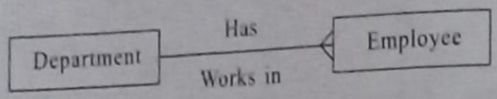
1. One-to-One

- In a one-to-one relationship, one occurrence of an entity relates to only one occurrence in another entity.
- A one-to-one relationship rarely exists in practice.
- **For example :** if an employee is allocated a company car then that car can only be driven by that employee.
- Therefore, employee and company car have a one-to-one relationship.



2. One-to-Many

- In a one-to-many relationship, one occurrence in an entity relates to many occurrences in another entity.
- For example : An employee works in one department, but a department has many employees.
- Therefore, department and employee have a one-to-many relationship.

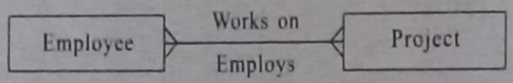


Example : College - Students

State - MLA

3. Many-to-Many

- In a many-to-many relationship, many occurrences in an entity relate to many occurrences in another entity.
- Same as a one-to-one relationship, the many-to-many relationship rarely exists in practice.
- For example : At the same time, an employee can work on several projects, and a project has a team of many employees.
- Therefore, employee and project have a many-to-many relationship.



Example : Bank - Customers

College - Students

Countries - People

REVIEW QUESTIONS

One Mark Questions :

1. Define entity.
2. Define entity types.
3. Define attribute.
4. Define key.
5. Define relation.
6. Define relation type.
7. Define relation set.
8. Define role.
9. What is ER diagram ?

Three Mark Questions :

1. Describe a database application.
2. List some database applications.
3. Differentiate between strong entity types and week entity types.
4. Demonstrate entity sets and week entity sets.
5. Describe different types of attributes.
6. Describe different types of keys.
7. Write about relation types.
8. Write about ER diagrams.
9. Write the naming conventions used in ER diagrams.
10. Write the design choices used for ER conceptual design.
11. Draw the ER diagram for Student relation.

Five Mark Questions :

1. Describe how to use High level Conceptual data models for database design.
2. Explain roles and structural constraints.
3. Explain relationship types of degree higher than two.

4. Design the ER diagram for university.
5. Design ER diagram for Hospital relation.
6. Design ER diagram for Bank relation.
7. Design ER diagram for Hotel relation.