Database Management System

Unit-2

Mrs. Kiran Bala Dubey

Assistant Professor

Department of Computer Science

Govt. N. P. G. College of Science,

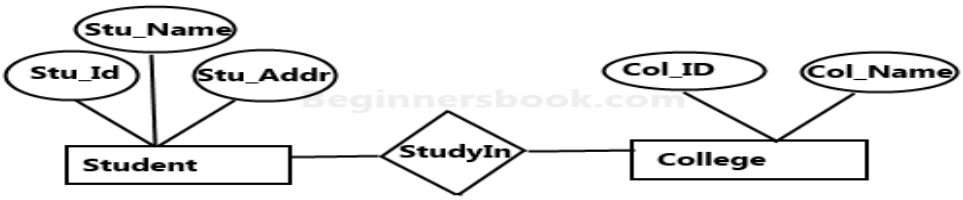
Raipur

Unit-2

- E-R Model
- Entity
- Relationship
- Concept of Keys
- Generalization
- Specialization
- Aggregation
- Converting E-R model to relation schema
- Extanded ER feature
- Introduction to UML
- Representation of UML diagram

E-R Model

- It develops a conceptual design for the database.
- This model is used to define the data elements and its relationship for a specified system.
- The main components of E-R model are: entity set and relationship set.
- An ER diagram shows the relationship among entity sets.



Sample E-R Diagram

Types of Attributes

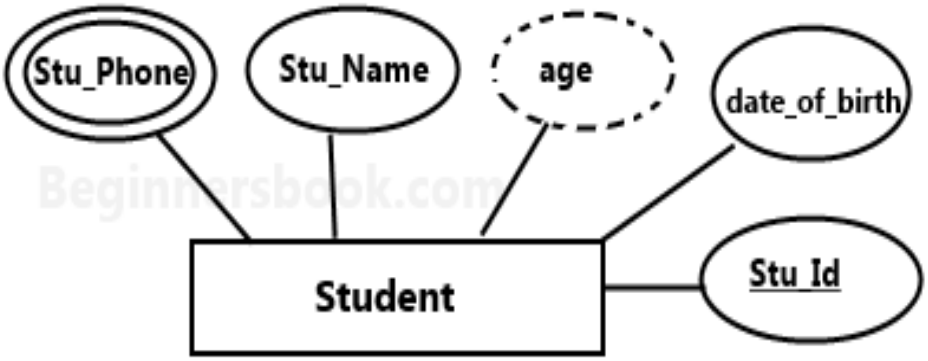
1. Simple attribute – Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

2. Composite attribute – Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.

3. Derived attribute − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from data_of_birth.

4. Single-value attribute – Single-value attributes contain single value. For example − Social_Security_Number.

5. Multi-value attribute – Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.


Stu_Phone
Stu_Name
age
date_of_birth
Student
Stu_Id

Relationship

- The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

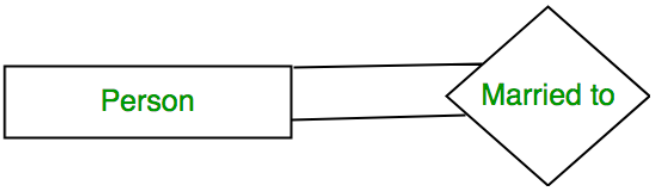
Relationship Set

- A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

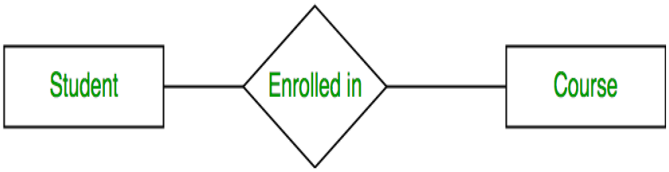
Degree of Relationship

- The number of participating entities in a relationship defines the degree of the relationship.
    - Binary = degree 2
    - Ternary = degree 3
    - n-ary = degree

1. Unary relationship -



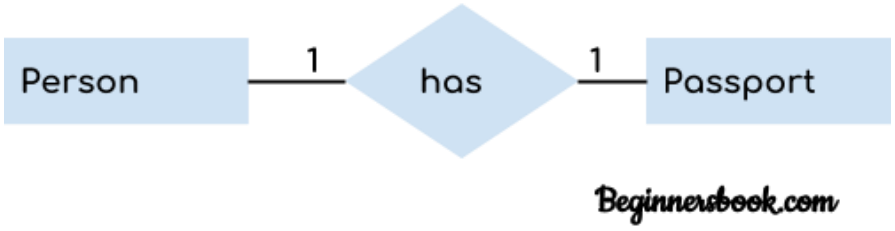
2. Binary relationship -



Mapping Cardinalities

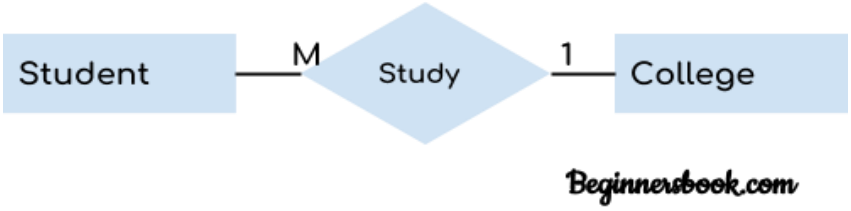
Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

•**One-to-one –** One entity from entity set A can be associated with at most one entity of entity set B and vice versa.

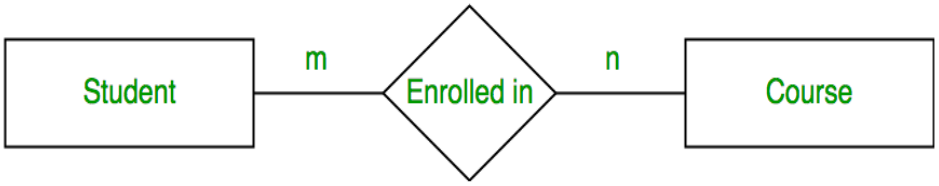


- **One-to-many –**One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.

- **Many-to-one –** More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



- **Many-to-many –** One entity from A can be associated with more than one entity from B and vice versa.



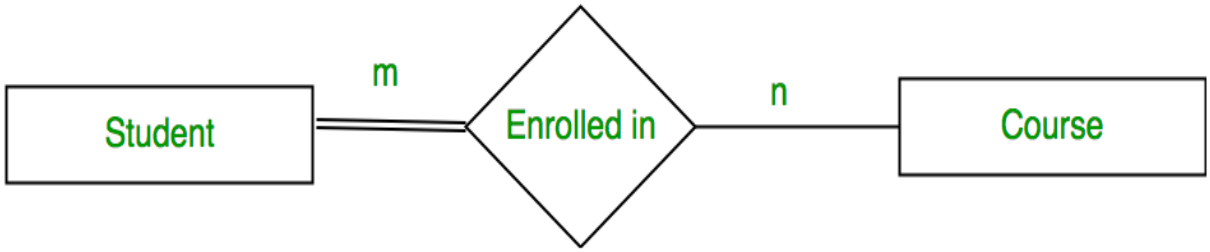
Participation Constraint:

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

Total Participation – Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown by double line in ER diagram.

Partial Participation – The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.

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


Student
m
Enrolled in
n
Course

Concept of Keys

- KEYS in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table).
- **A superkey** is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification. For eg. rollno + enrollno
- **A Primary key** is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. For eg. Emp_code
- **A Candidate Key** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.For eg. Stud ID, Roll No, and email are candidate key

- **An Alternet key** is a column or group of columns in a table that uniquely identify every row in that table. All the keys which are not primary key are called an Alternate Key. For eg. Person_name & Mobile.
- **A Foreign key** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It is a field/column in the table that is the primary key of another table.For eg. It_code is a primary key for Stock table which is a foreign key for Sales table.
- A key which has multiple attributes to uniquely identify rows in a table is called a **composite key.** For eg. Person_name & Mobile.

Strong & Weak Entity set

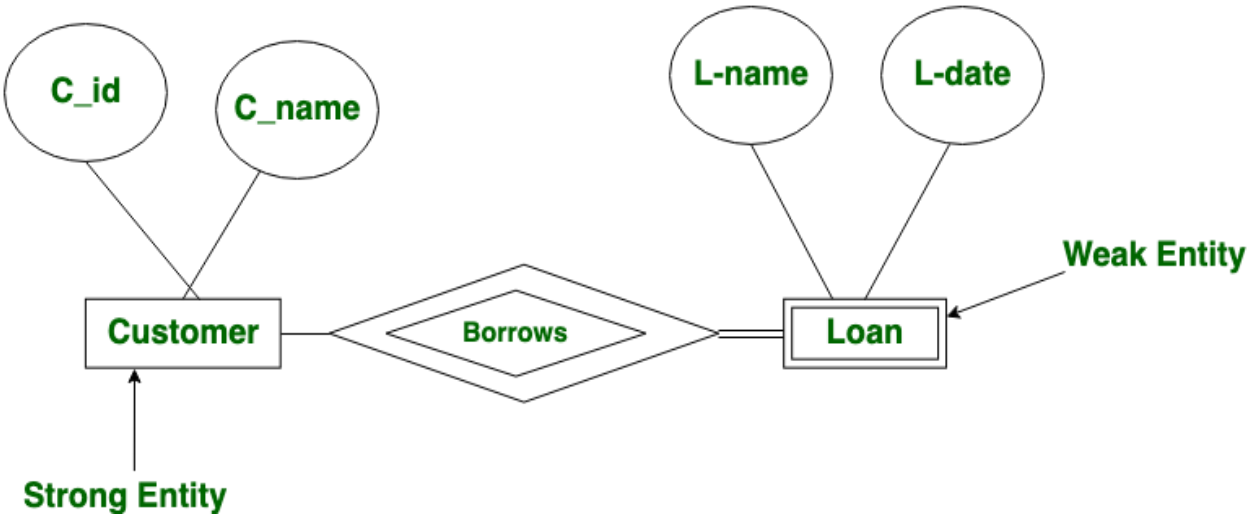
Strong Entity:

A strong entity is not dependent of any other entity in the relation. A strong entity will always have a primary key. Strong entities are represented by a single rectangle. The relationship of two strong entities is represented by a single diamond.

Weak Entity:

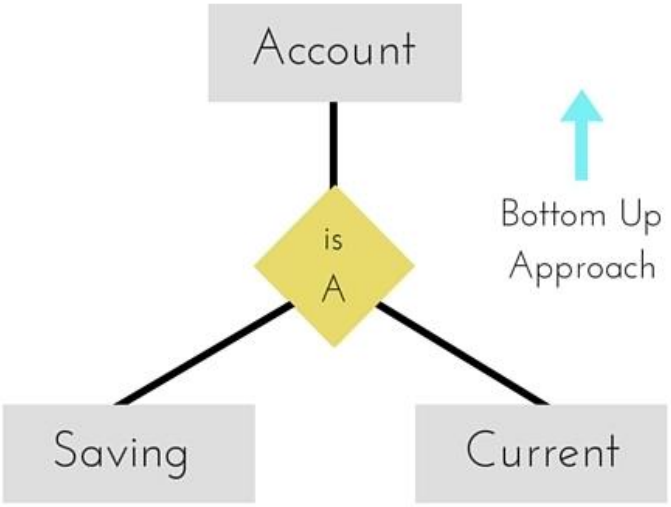
A weak entity is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have any primary key. It instead has a partial discriminator key. A weak entity is represented by a double rectangle.

The relation between one strong and one weak entity is represented by a double diamond.


C_id
C_name
L-name
L-date
Weak Entity
Customer
Borrows
Loan
Strong Entity

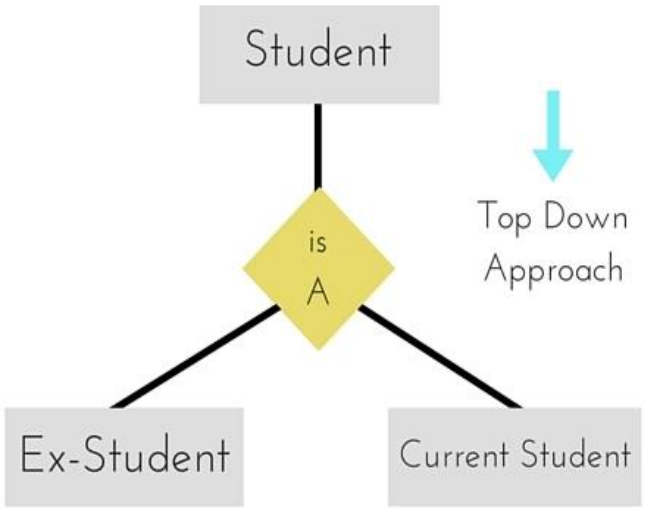
Generalization

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



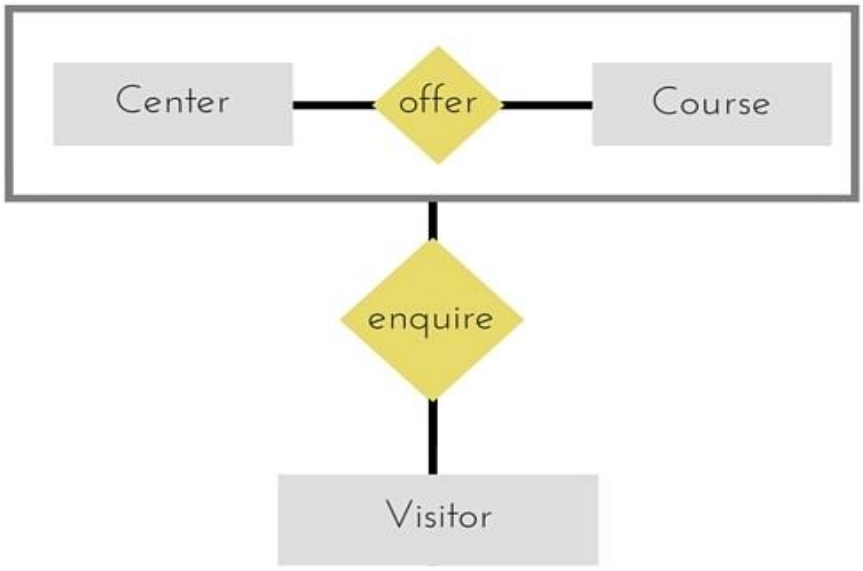
Specialization

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

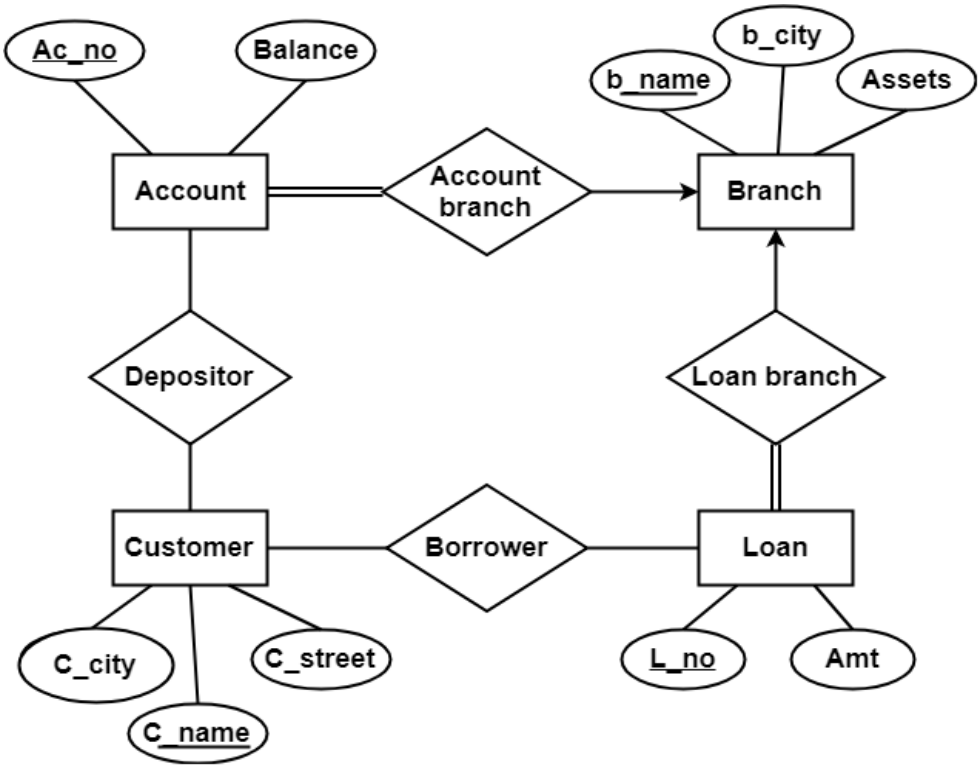


Aggregation

- Aggregration is a process when relation between two entities is treated as a single entity

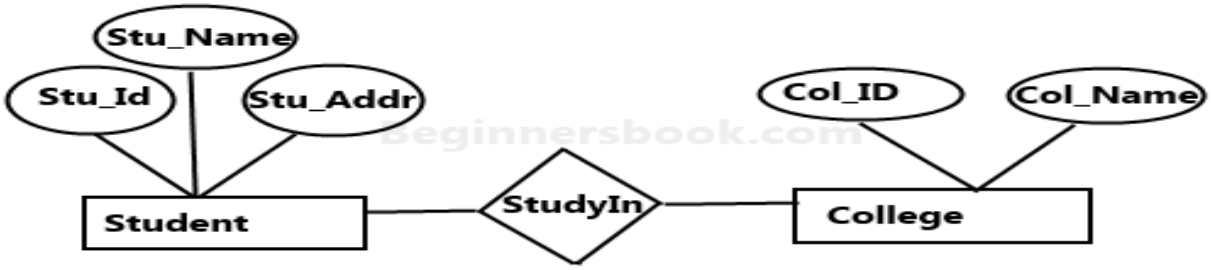


Converting E-R model to relation schema



Converting E-R model to relation schema

- Account (Ac_no , Balance , b_name)
- Branch (b_name , b_city , Assets)
- Loan (L_no , Amt , b_name)
- Borrower (C_name , L_no)
- Customer (C_name , C_street , C_city)
- Depositor (C_name , Ac_no)



Sample E-R Diagram

- Student(Stu_id, Stu_Name,Stu_Addr)
- College(Col_ID, Col_Name)

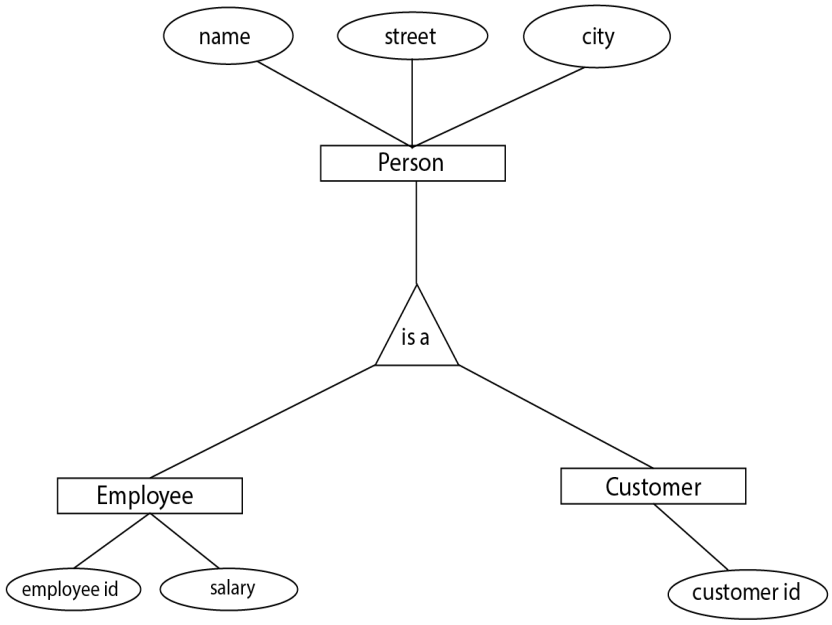


Figure: Entity - Relation Diagram

Extanded ER feature

- **Specialization –** The process of designating to sub grouping within an entity set is called specialization. In above figure, the “person” is distinguish in to whether they are “employee” or “customer”.
- **Generalization –** generalization is relationship that exist between higher level entity set and one or more lower level entity sets. Generalization synthesizes these entity sets into single entity set.
- **Aggregation:** there is a one limitation with E-R model that it cannot express relationships among relationships. So aggregation is an abstraction through which relationship is treated as higher level entities.

Introduction to UML

- Unified Modeling Language (UML) is a general purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.
- Unified Modeling language (UML) is a standardized graphical modeling language enabling developers to specify, visualize, construct, and document the artifacts (major elements) of the software system. Thus, UML makes these artifacts scalable, secure and robust in execution.
- UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.
- UML is an important aspect involved in object-oriented software development.
- It uses graphic notation to create visual models of software systems.

Diagrams in UML can be broadly classified as:

- **Structural Diagrams –** Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.
- **Behavior Diagrams –** Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

Following are some object-oriented concepts that are needed to begin with UML:

Object: An object is a real world entity. There are many objects present within a single system. It is a fundamental building block of UML.

Class: A class is a software blueprint for objects, which means that it defines the variables and methods common to all the objects of a particular type.

Abstraction: Abstraction is the process of depicting the essential characteristics of an object to the users while hiding the irrelevant information. Basically, it is used to imagine the functioning of an object.

Inheritance: Inheritance is the process of deriving a new class from the existing ones.

Polymorphism: It is a mechanism of representing objects having multiple forms used for different purposes.

Encapsulation: It binds the data and the object together as a single unit, enabling tight coupling between them.

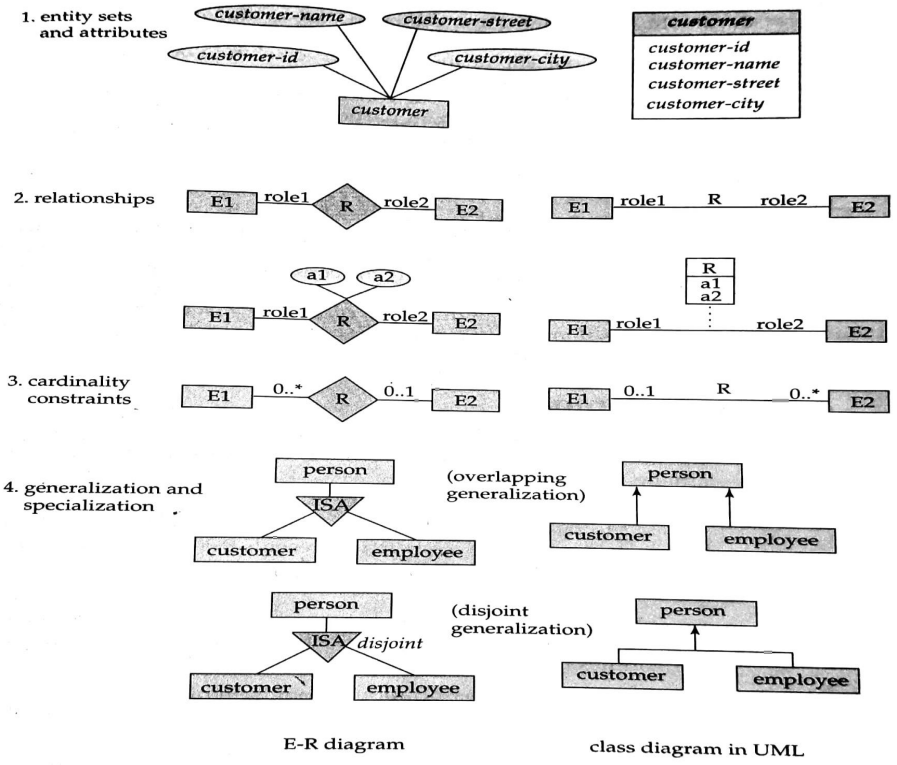


Figure 2.28 Symbols used in the UML class diagram notation.