Structuring System Data Requirements

Introduction: In previous chapter, we learned how to model and analyze data. We learned how to show data stores, or data at rest, in a data flowdiagram (**DFD**). **DFDs**, **use cases**, and **various processing logic techniques** show **how, where,** and **when** data are used or changed in an informationsystem, but these techniques do not show **the definition, structure, andrelationships within the data**. **Data modeling** develops these missing, and crucial, descriptive pieces of a system. **The most common format used for data modeling is entity-relationship (E-R) diagramming**. A similarformat used with object-oriented analysis and design methods is **class diagramming**, which is included in a special section at the end of this chapter on the object-oriented development approach to data modeling. Data models that use E-R and class diagram notations explain the characteristics and structure of data independent of how the data maybe stored in computer memory. A data model is usually developediteratively, either from scratch or from a purchased data model for theindustry or business area to be supported. Information system (IS) planners use this preliminary data model to develop an enterprise-wide data model with very broad categories of data and little details.

Conceptual Data Modeling

A **conceptual data model is a representation of organizational data. The purpose of a** conceptual data model is to show as many rules about the meaning and interrelationships among data as are possible. Conceptual data modeling is typically done in parallel with other **requirements analysis and structuring steps during systems analysis** .On larger systems development teams, **a subset of the project team concentrates** on **data modeling** while **other team members focus attention on process or logic modeling**. Analysts develop (or use from prior systems development) a conceptual data model for the current system and then build or refine a purchased conceptual data model that supports the scope and requirements for the proposed or enhanced system. The work of all team members is coordinated and shared through the project **dictionary or repository.** This repository is often maintained by a common Computer- Aided Software Engineering (CASE) or data modeling software tool.

The Conceptual Data Modeling process:

The process of conceptual data modeling begins with developing a conceptual data model for the system being replaced, if a system already exists. This is essential for planning the conversion of the **current files or database into the database of the new system**. Further, this is a good, but not a perfect, starting point for your understanding of the data requirements of the new system. Then, a new conceptual data model is built (or a standard one is purchased) that includes all of the data requirements for the new system.

Deliverables and outcomes:

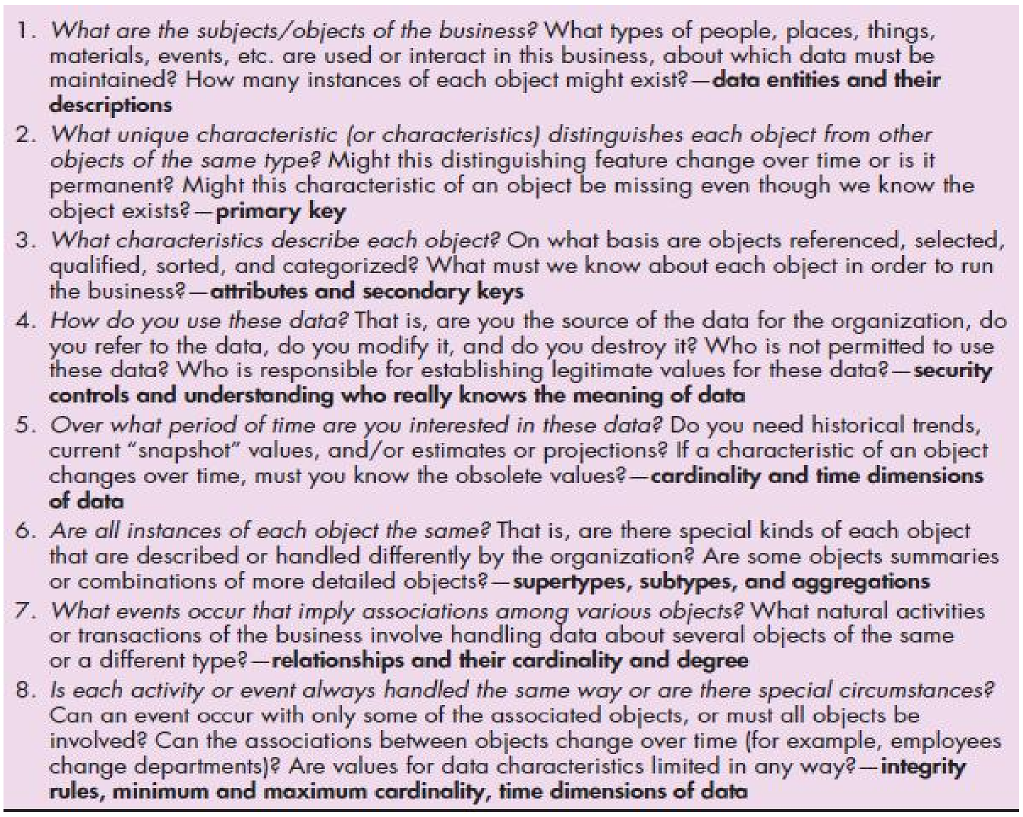
Most organizations today **do conceptual data modeling using E-R modeling**, which uses a special notation to represent as much meaning about data as possible. Because of the rapidly increasing interest in **object-oriented methods, class diagrams using unified modeling language (UML) drawing tools** such as IBM's Rational products or Microsoft Visio are also popular.

1. The primary deliverable from the conceptual data modeling step within the analysis phase is an **E-R diagram**.

2. The other deliverable from conceptual data modeling is a full set of entries about data objects that will be stored in the project **dictionary, repository, or data modeling software**. The repository is the mechanism to link data, process and logic models of an information system.

Gathering Information for Conceptual Data Modeling: Requirements determination methods must include **questions and investigations** that take a data, not only a process and logic, focus. For example, during interviews with potential system users—during Joint Application Design(JAD) sessions or through requirements interviews—you must ask specific questions in order to gain the perspective on data that you need to develop or tailor a purchased data model.

You typically do data modeling from a combination of perspectives. The **first perspective is generally called the top-down approach**. This perspective derives the business rules for a data model from an intimate understanding of the **nature of the business**, rather than from any specific information requirements in computer displays, **reports**, or business forms.

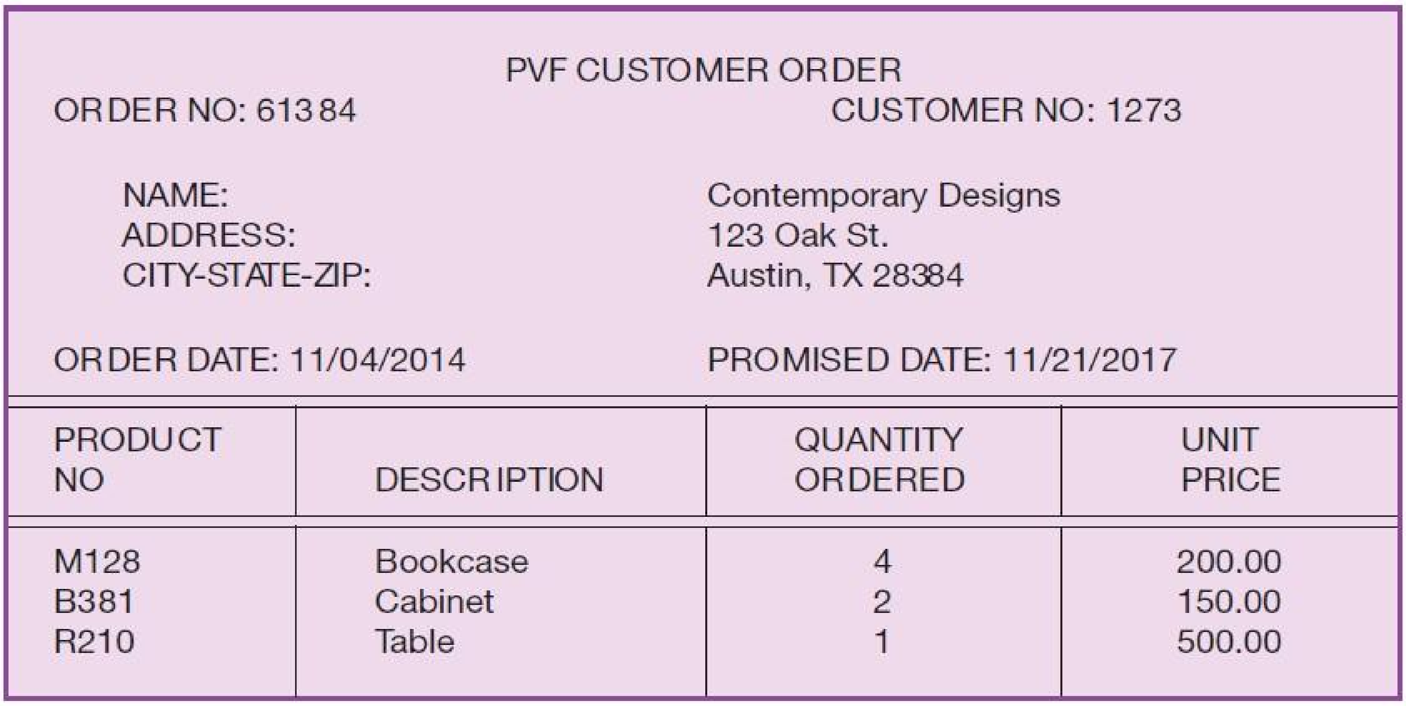


1. *What are the subjects/objects of the business?* What types of people, places, things, materials, events, etc. are used or interact in this business, about which data must be maintained? How many instances of each object might exist?—**data entities and their descriptions**
2. *What unique characteristic (or characteristics) distinguishes each object from other objects of the same type?* Might this distinguishing feature change over time or is it permanent? Might this characteristic of an object be missing even though we know the object exists?—**primary key**
3. *What characteristics describe each object?* On what basis are objects referenced, selected, qualified, sorted, and categorized? What must we know about each object in order to run the business?—**attributes and secondary keys**
4. *How do you use these data?* That is, are you the source of the data for the organization, do you refer to the data, do you modify it, and do you destroy it? Who is not permitted to use these data? Who is responsible for establishing legitimate values for these data?—**security controls and understanding who really knows the meaning of data**
5. *Over what period of time are you interested in these data?* Do you need historical trends, current "snapshot" values, and/or estimates or projections? If a characteristic of an object changes over time, must you know the obsolete values?—**cardinality and time dimensions of data**
6. *Are all instances of each object the same?* That is, are there special kinds of each object that are described or handled differently by the organization? Are some objects summaries or combinations of more detailed objects?—**supertypes, subtypes, and aggregations**
7. *What events occur that imply associations among various objects?* What natural activities or transactions of the business involve handling data about several objects of the same or a different type?—**relationships and their cardinality and degree**
8. *Is each activity or event always handled the same way or are there special circumstances?* Can an event occur with only some of the associated objects, or must all objects be involved? Can the associations between objects change over time (for example, employees change departments)? Are values for data characteristics limited in any way?—**integrity rules, minimum and maximum cardinality, time dimensions of data**

TABLE 8-1 Requirements Determination Questions for Data Modeling

You can also gather the information you need for data modeling by reviewing specific business documents—computer displays, reports, and business forms— handled within the system. This process of gaining an understanding of data is often called **a bottom-up approach**. These items will appear as data flows on DFDs and will show the data processed by the system and, hence, probably the data that must be maintained in the system's database. Consider, for example, Figure 8-4, which shows a customer order form used at Pine Valley Furniture (PVF). From this form, we determine that the following data must be kept in the database:

ORDER NO	CUSTOMER NO
ORDER DATE	NAME
PROMISED DATE	ADDRESS
PRODUCT NO	CITY-STATE-ZIP
DESCRIPTION	
QUANTITY ORDERED	
UNIT PRICE	


PVF CUSTOMER ORDER

ORDER NO: 61384 CUSTOMER NO: 1273

NAME:	Contemporary Designs
ADDRESS:	123 Oak St.
CITY-STATE-ZIP:	Austin, TX 28384

ORDER DATE: 11/04/2014 PROMISED DATE: 11/21/2017

PRODUCT NO	DESCRIPTION	QUANTITY ORDERED	UNIT PRICE
M128	Bookcase	4	200.00
B381	Cabinet	2	150.00
R210	Table	1	500.00

FIGURE 8-4
Sample customer form

Introduction to E-R Modeling

An **entity-relationship data model (E-R model) is a detailed, logical representation** of the data for an organization or for a business area. The E-R model is expressed in terms of **entities** in the business environment, the relationships or associations among those entities, and the attributes or properties of both the entities and their relationships.

An E-R model is normally expressed as an **entity-relationship diagram (E-Rdiagram), which is a graphical representation of an E-R model.** The basic E-R modeling notation uses three main constructs: data entities, relationships and their associated attributes.

An **entity** is a person, place, object, event, or concept in the user environment about which the organization wishes to maintain data. An entity has its own identity, which distinguishes it from other entities. There is an important distinction between **entity types** and **entity instances.** An entity type is a collection of entities that share common properties or characteristics. An entity instance is a single occurrence of an entity type. We use a **simple noun to name an entity type**. We **use capital letters in naming an entity**

type and, in an E-R diagram, the name is placed inside **a rectangle** representing the entity, as shownin Figure 8-6a.

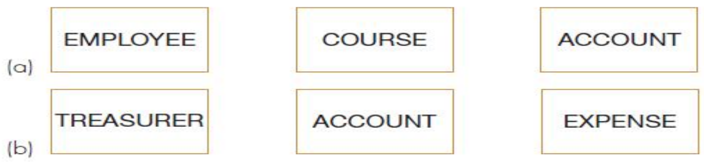


FIGURE 8-6
Representing entity types
(a) Three entity types
(b) Questionable entity types

An **entity instance (also known simply as an instance) is a single occurrence of** an entity type. An entity type is described just once in a data model, whereas many instances of that entity type may be represented by data stored in the database. For example, there is one **EMPLOYEE entity type** in most organizations, but there may be hundreds (or even thousands) of instances of this entity type stored in the database.

For example there is usually one EMPLOYEE type but there may be hundreds of instances of this entity type stored in a database. Each entity type has a set of attributes associated with it.

For example: for an entity STUDENT we have such attributes like: STUDENT NO, NAME, ADDRESS, PHONE NO. Every entity must have an attribute or set of attributes that distinguishes one instance from other instances of the same type

Naming and Defining Entity Types:

Clearly naming and defining data, such as entity types, are important tasks during requirements determination and structuring. **When naming and defining entity types**, you should use the following guidelines:

An **entity type** name is a **singular noun (**such as CUSTOMER, STUDENT, or AUTOMOBILE**).**

An entity type name should be **descriptive** and **specific** to the organization. **For example**, a PURCHASE ORDER for orders placed with suppliers is distinct from CUSTOMER ORDER for orders placed by customers. **Both of these entity types cannot be named ORDER**.

An entity type name should be concise (short and clear). For example, in a university database, use REGISTRATION for the event of a student registering for a class rather than STUDENT REGISTRATION FOR CLASS.

Event entity types should be named for the result of the event, not the activity or process of the event. For example, the event of a project manager assigning an employee to work on a project results in an **ASSIGNMENT**.

Attributes:

Each entity type has a set of attributes associatedwith it. An **attribute is a property or characteristic of an entity**. Following are some typical **entity types** and associated **attributes**:

STUDENT: Student_ID, Student_Name, Home_Address, Phone_Number,Major

AUTOMOBILE: Vehicle_ID, Color, Weight, Horsepower

EMPLOYEE: Employee_ID, Employee_Name, Payroll_Address, Skill

We use an initial **capital letter, followed by lowercase letters**, and **nouns** in naming an attribute; **underscores** may or may not be used to separatewords. In E-R diagrams, we represent an attribute by placing its name inside the rectangle for the associated entity.

Types of Attributes

Atomic vs composite,

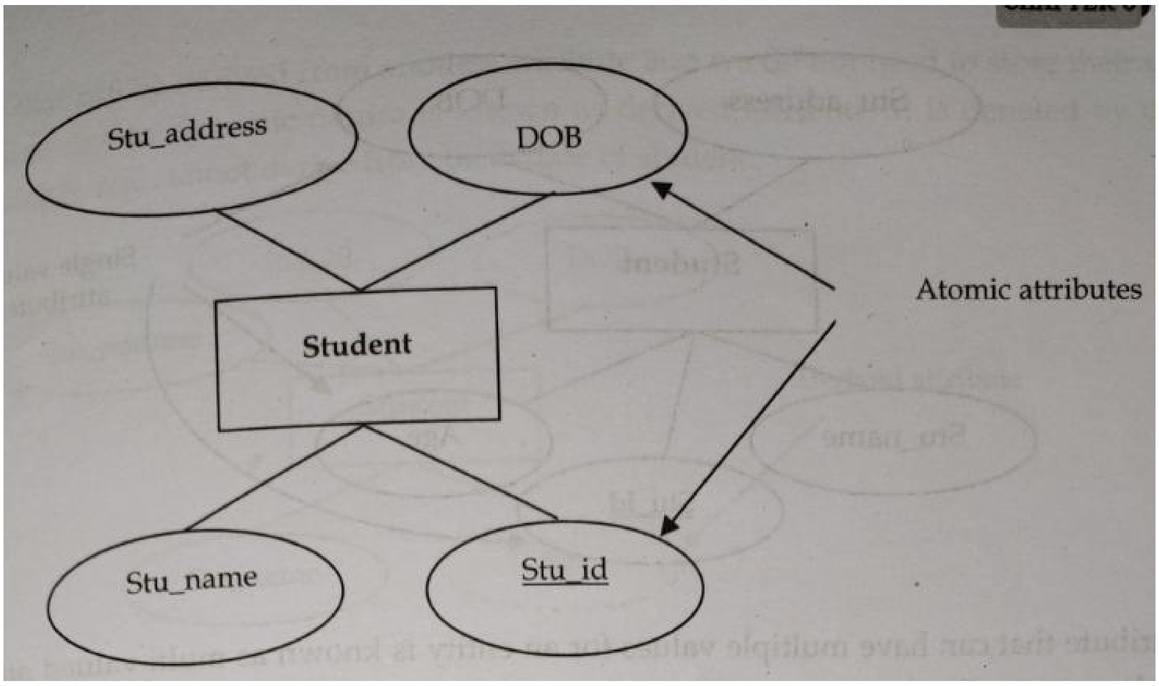
Single valued vs multi valued,

stored vs derived,

NULL value and

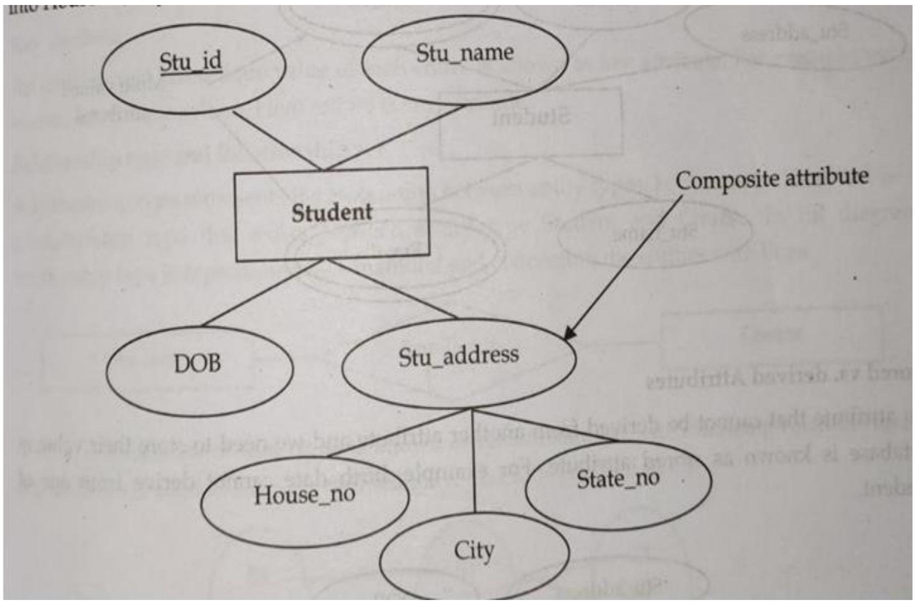
Key attributes

1. **Atomic vs composite**: An attribute that cannot be divided into smallerindependent attributes is known as atomic attribute. For example, assume **Student** is an entity and its attributes are **Name, Age, DOB, Address** and **Phone no**. Here the Stu_id, DOB attributes of Student (entity) cannot further divide. So Stu_Id, and DOB are atomic attribute.

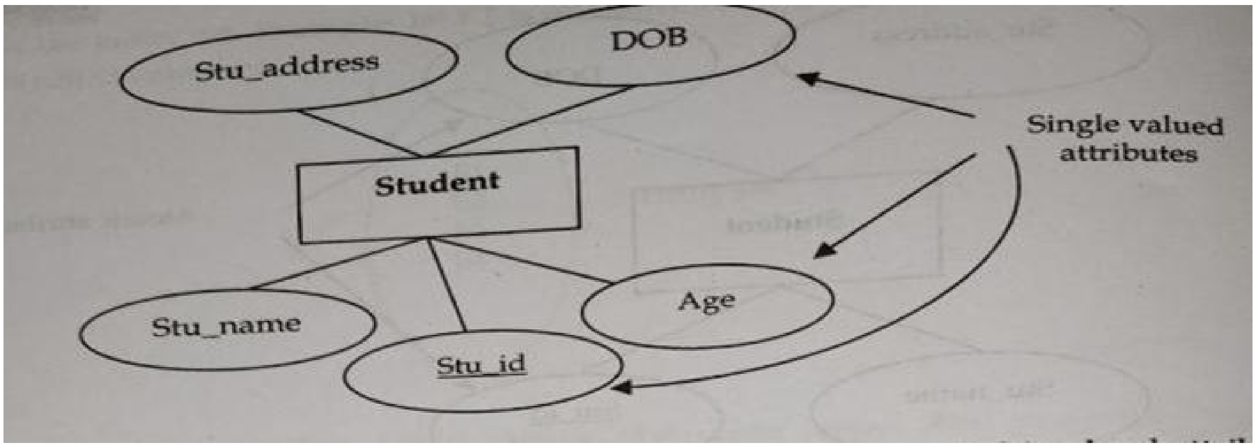


Composite attribute: An attribute that can be divided intosmaller independent attribute is known as composite attribute.eg Assume student is an entityand its attributes are

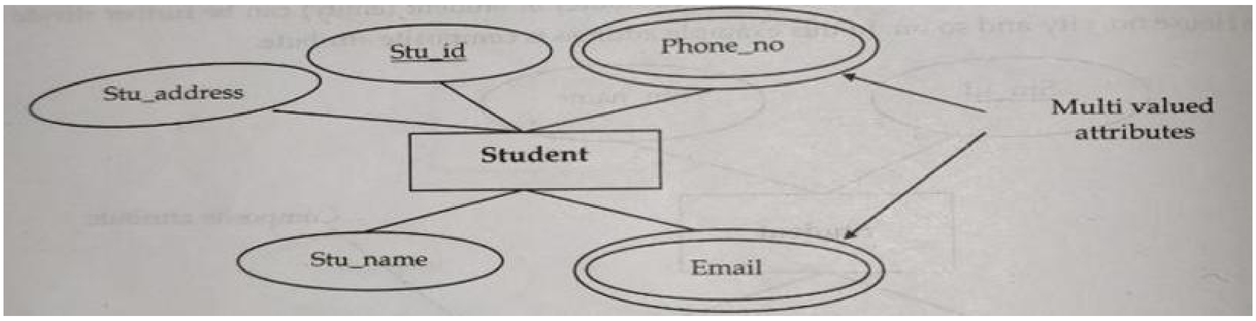
Stu_id,name,DOB,address and phone no. Here the address(attribute) of student(entity) can be further divided into house no.,city and so on. In this example address is composite attribute.



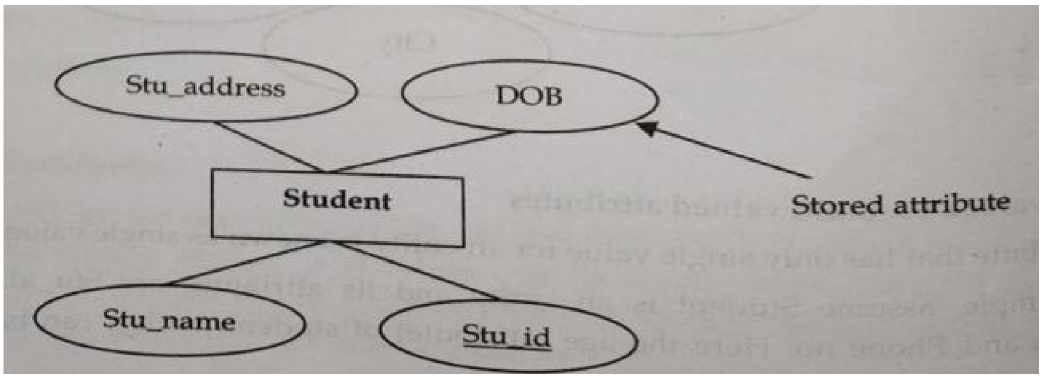
Single value vs Multivalued attributes:
An attribute that has only single value for an entity is known as single valued attribute.eg student is an entity and its attributes are stu_id, name ,DOB, age, address and Phone no. Here the age (attribute) of student can have only one value.So age is single valued attribute.



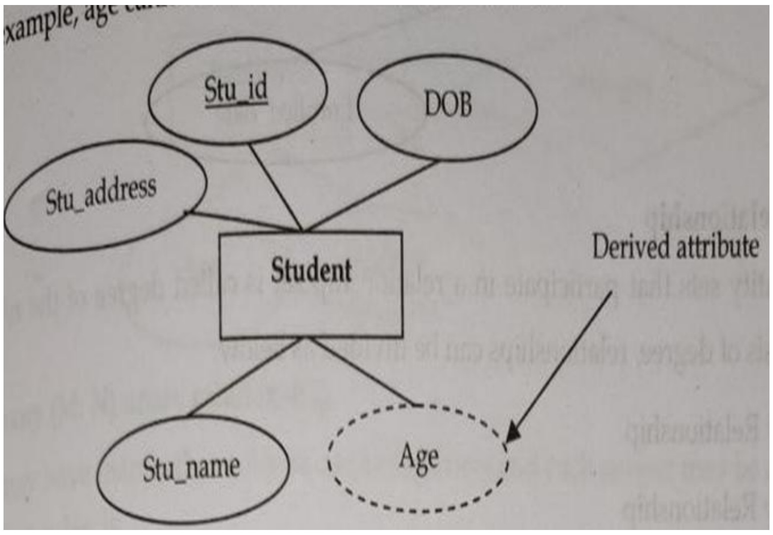
An attribute that can have multiple values for an entity is known as multivalued attribute. Eg Assume student is an entity and its attributes are stu_id,name,age,address and phone no. Here the phone no. and e-mail (attribute) of student(entity) can have multiple values because a student may have many phone numbers. Here phone no. and e-mail are multivalued attribute.



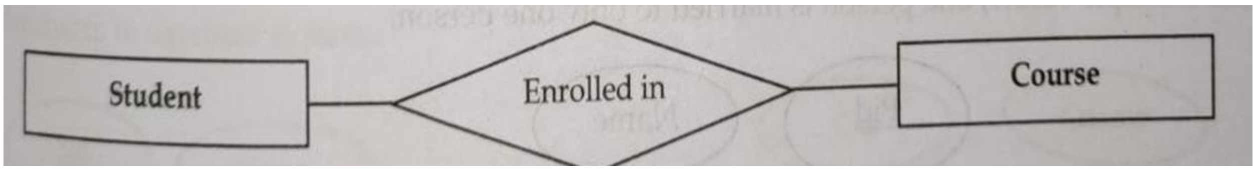
Stored vs Drived attributes:
An attribute that can’t be derived from another attribute and we need to store their value in database is known as stored attribute. Eg birthdate can’t be derive from any other attribute.



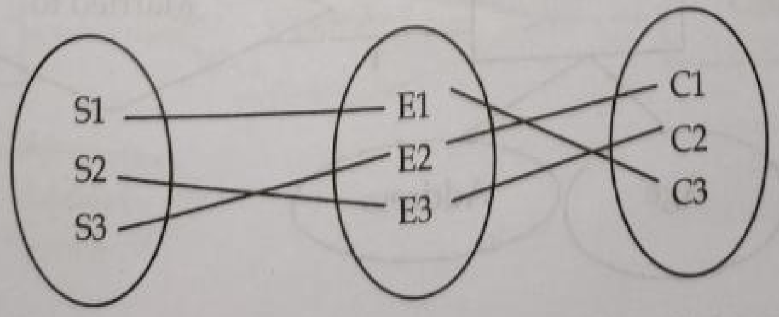
An attribute that can be derived from another attribute and we do not need to store their value in the database due to dynamic nature is known as derived attribute. It is denoted by dotted oval. Eg.age can be derived from birth date of student.



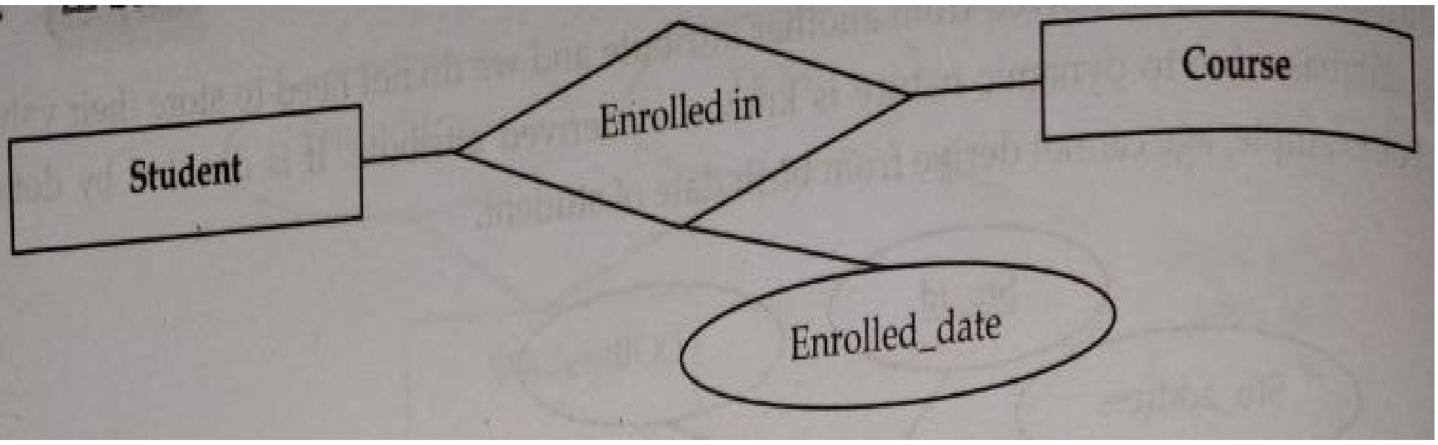
Null attribute: An attribute which has not any value for an entity is known as null valued attribute. Eg. Assume student is an entity and its attributes are name,age,address and phone no. There may be a chance when a student has no phone no. In that case phone no is called null valued attribute.
Key attribute: An attribute that has unique value of each entity is known as key attribute. E. every student has unique roll no.. Here roll no. is the key attribute.
Relationship type and relationship set:
A relation type represents the association between entity types. Eg. Enrolled in is a relationship type that exists between entiey type student and course. In ER diagram relationship type is represented by a diamond and connecting the entities with lines.



A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.



In another way, we can say that association between two entity sets is called relationship set. A relationship set may also have attributes called descriptive attributes. For example, the Enrolled_in relationship set between entity sets student and course may have the attribute enrolled_date.



Conceptual Data Modeling And The E-R Model

Degree of a Relationship

Number of entity sets that participate in a relationship set is called degree of the relationship set. On the basis of degree, relationships can be divided as below:

- Unary Relationship
- Binary Relationship
- N-ary Relationship

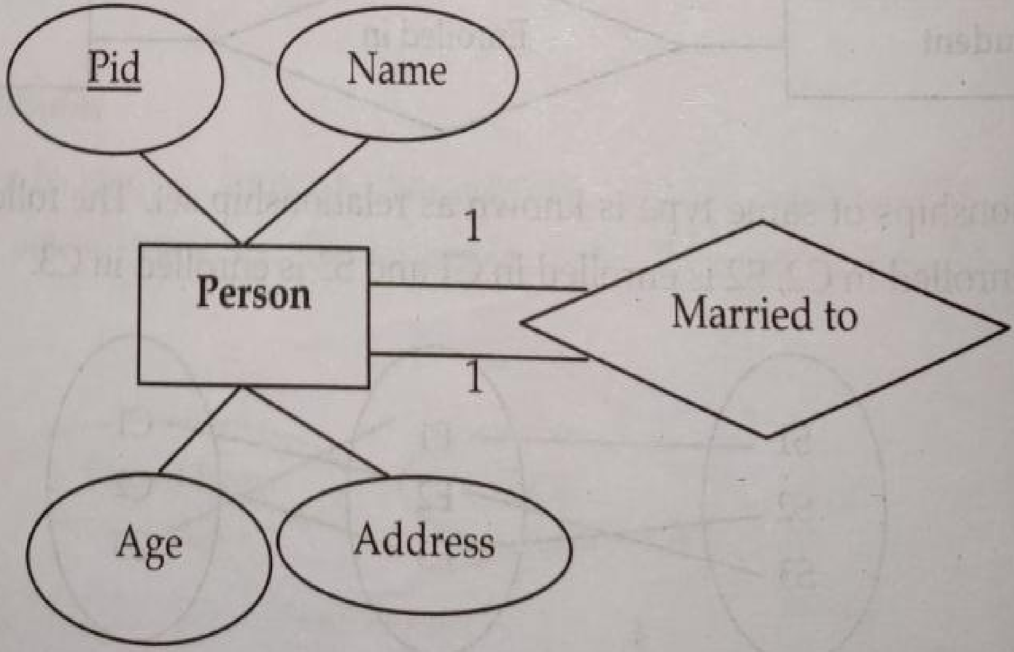
Unary Relationship

If only one entity set participates in a relation, the relationship is called as unary relationship. Here same entity set participates in relationship twice with different roles. Role names are specified above the link joining entity set and relationship set. This type of relationship set is sometimes called a recursive relationship set. There are three types of unary relationships:

- 1:1 unary relationship
- 1:M unary relationship
- M:N unary relationship

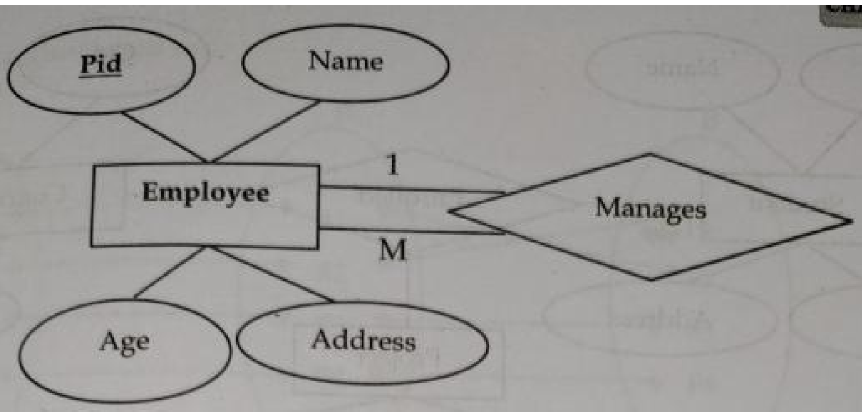
One to one (1:1) unary relationship

In the example below, one person is married to only one person.



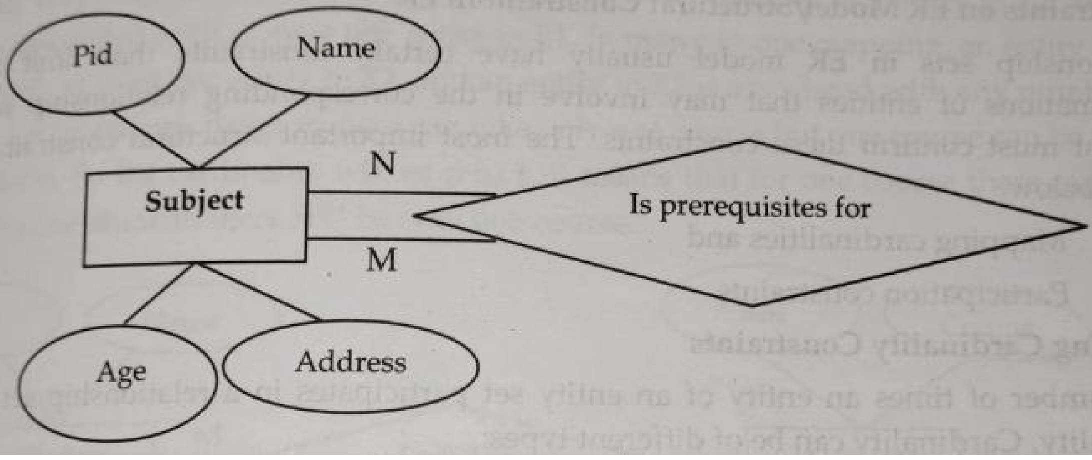
One to many (1: M) unary relationship

An employee may manage many employees but an employee is managed by only one employee. This types of relationship with employee relationship set itself is called 1:M unary relationship as shown in below;



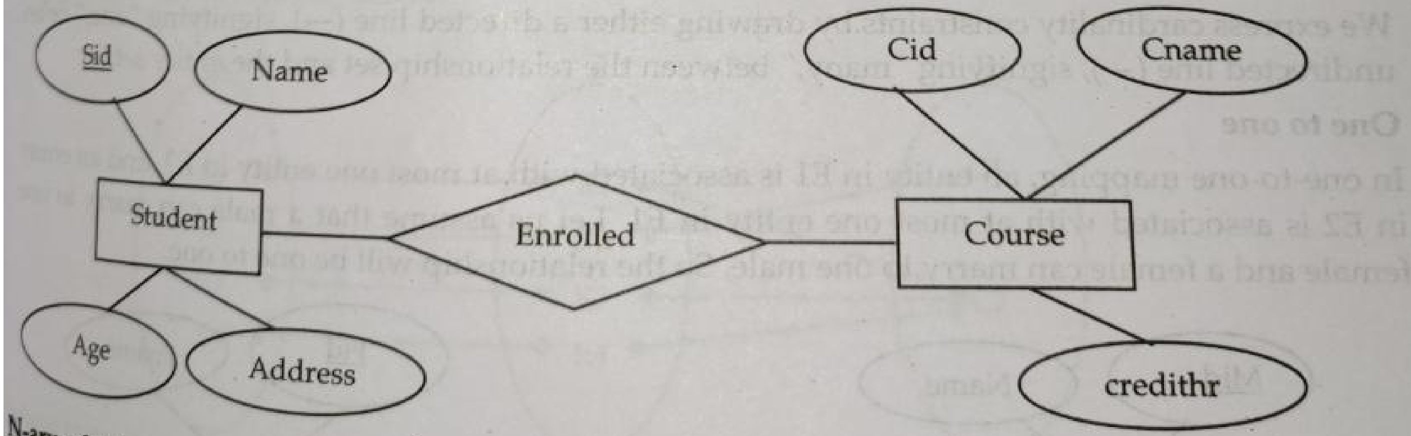
Many to many (M: N) unary relationship

A subject may have many other subjects as prerequisites and each subject may be a prerequisite to many other subjects.



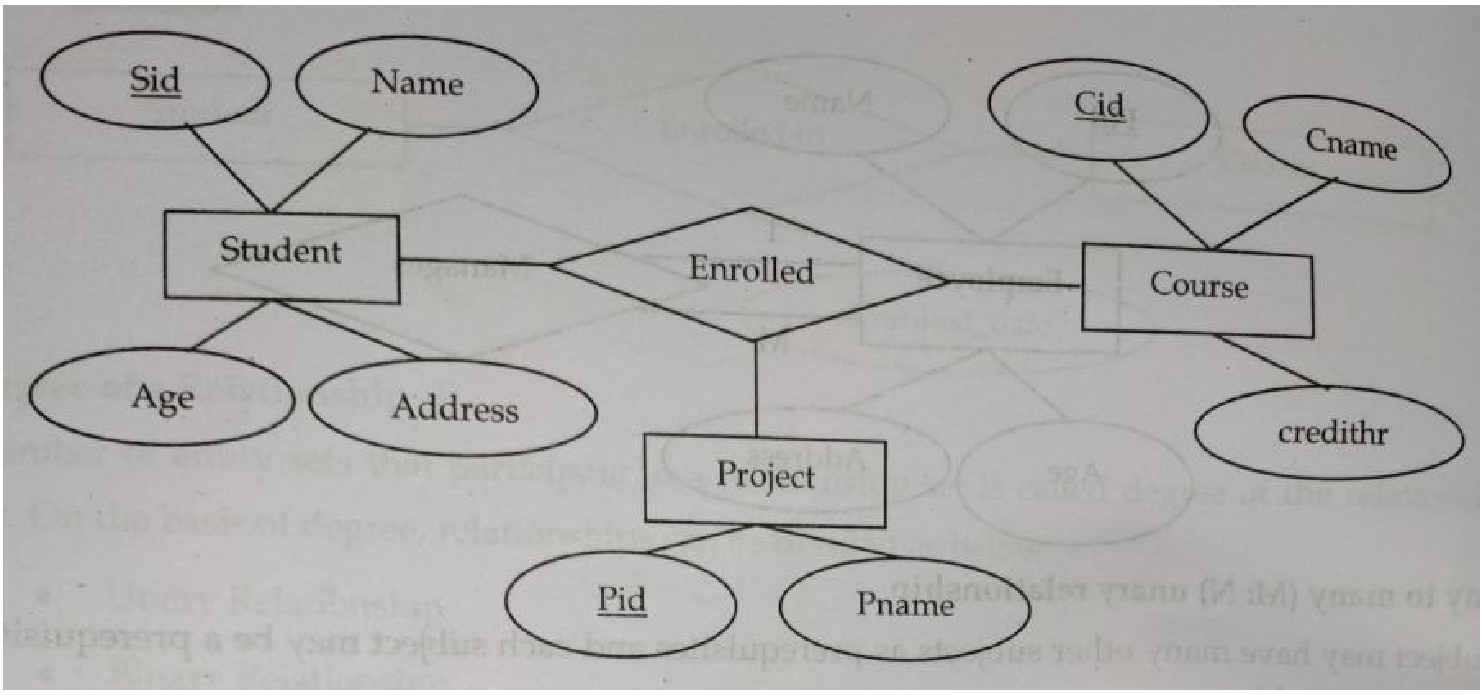
Binary relationship

When there are two entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course. This is the most common type of relationship in database systems.



N-ary relationship

When there are n entities set participating in a relation, the relationship is called as n-ary relationship. If n=1 then it is called unary relationship, if n=2 then it is called binary relationship. Generally in N-ary relationship there are more than two entities participating with a single relationship i.e. n > 2.



Constraints on ER Model/Structural Constraint in ER

Relationship sets in ER model usually have certain constraints that limit the possible combinations of entities that may involve in the corresponding relationship set. Database content must confirm these constraints. The most important structural constraints in ER are listed below:

- Mapping cardinalities and
- Participation constraints

Mapping Cardinality Constraints

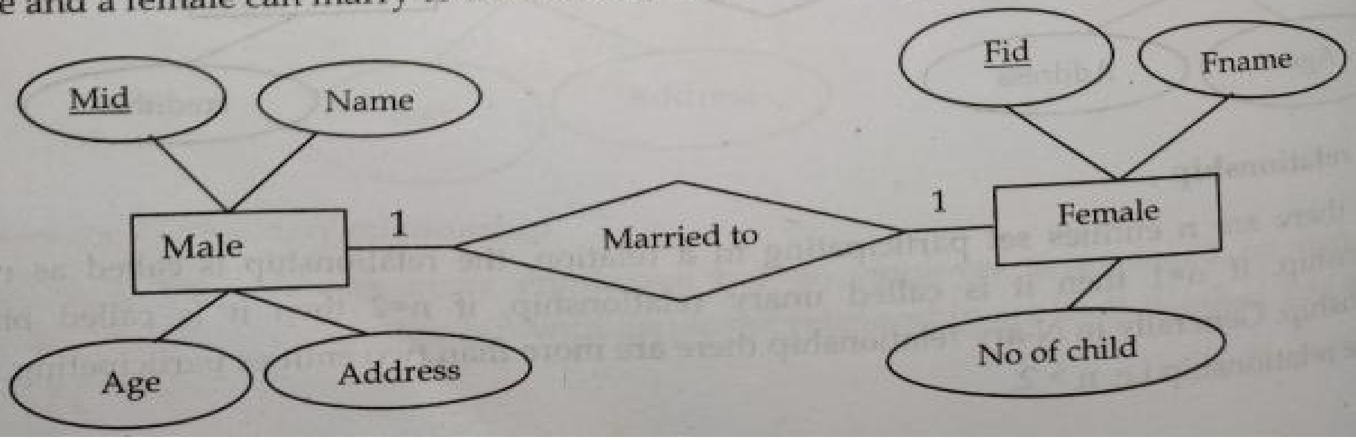
The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

- One-to- One
- One-to- Many
- Many-to-One
- Many-to-Many

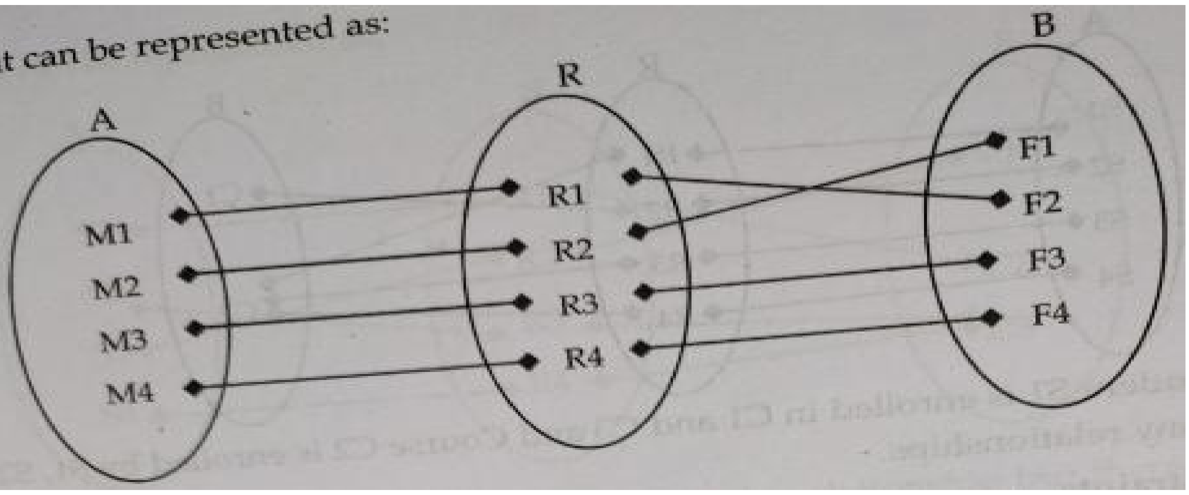
We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

One to one

In one-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.

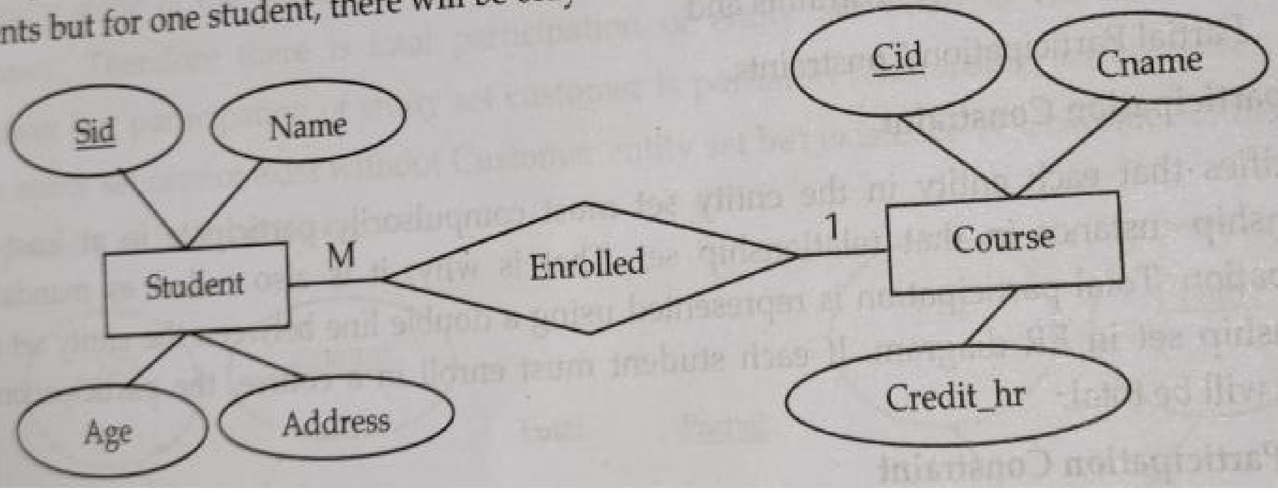


Using Sets, it can be represented as:

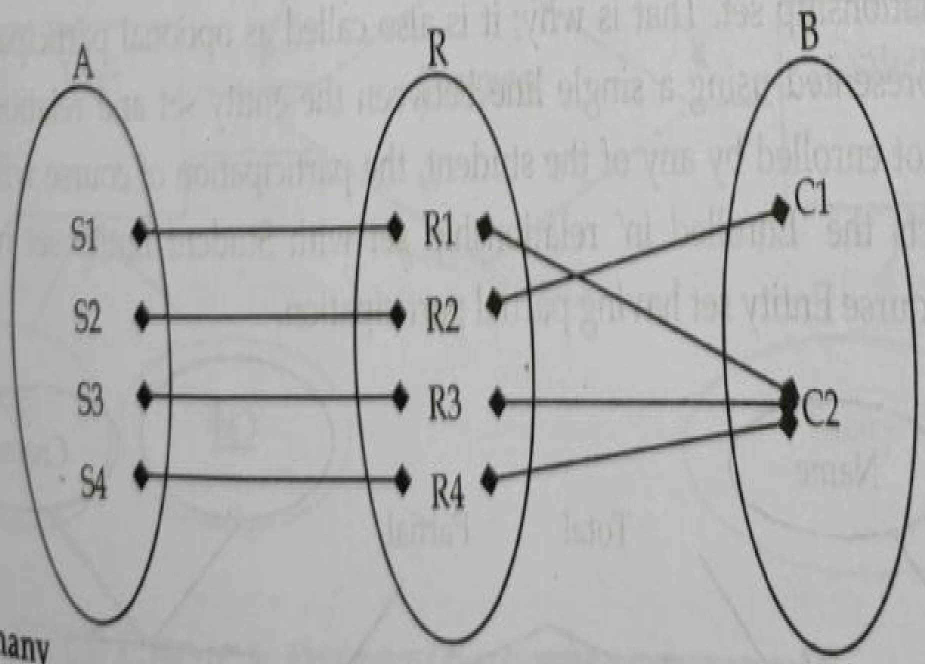


One to many or many to one

In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1. In many-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.



Using Sets, it can be represented as:



Many to many

In many-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

Participation Constraints

Participation Constraint is applied on the entity participating in the relationship set. Constraint on ER model that determines whether all or only some entity occurrences participate in a relationship is called participation constraint. It specifies whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints:

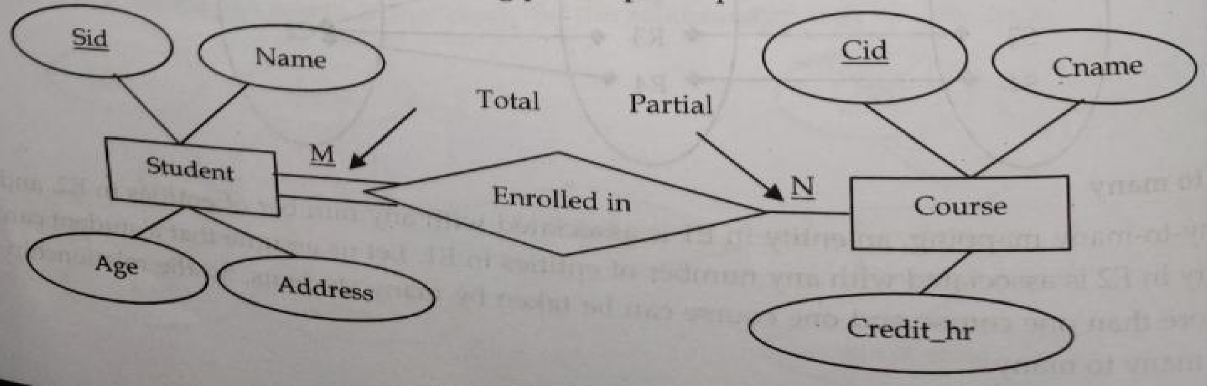
- Total Participation Constraints and
- Partial Participation Constraints.

Total participation Constraint

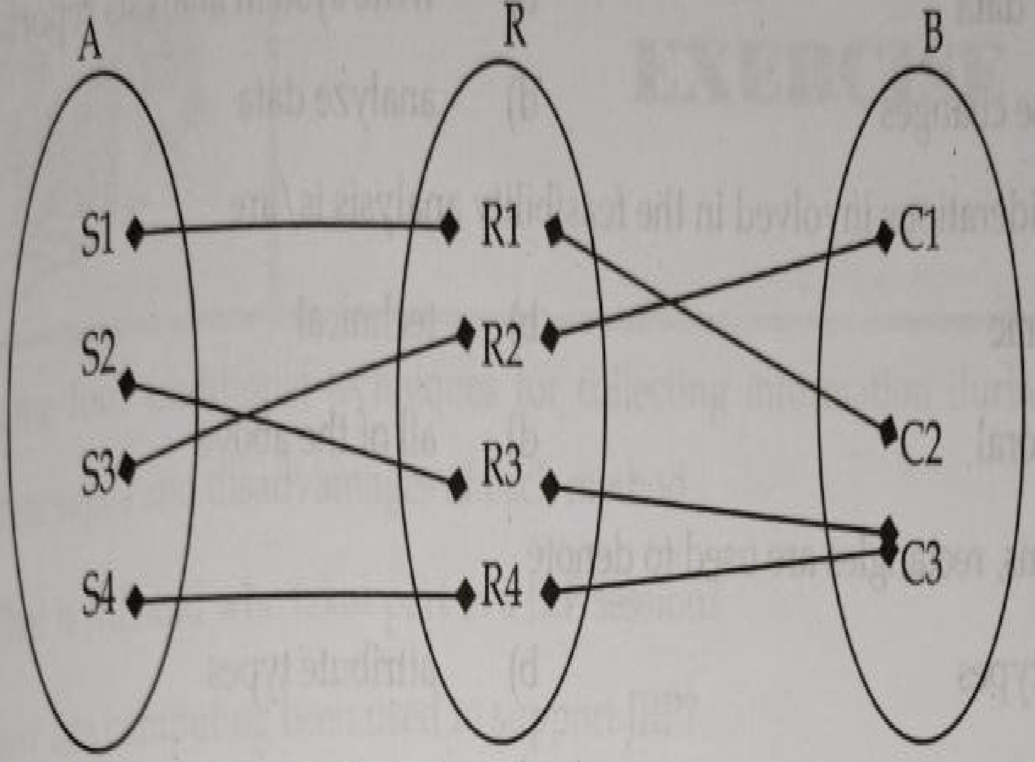
It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set. That is why; it is also called as mandatory participation. Total participation is represented using a double line between the entity set and relationship set in ER diagram. If each student must enroll in a course, the participation of student will be total.

Partial Participation Constraint

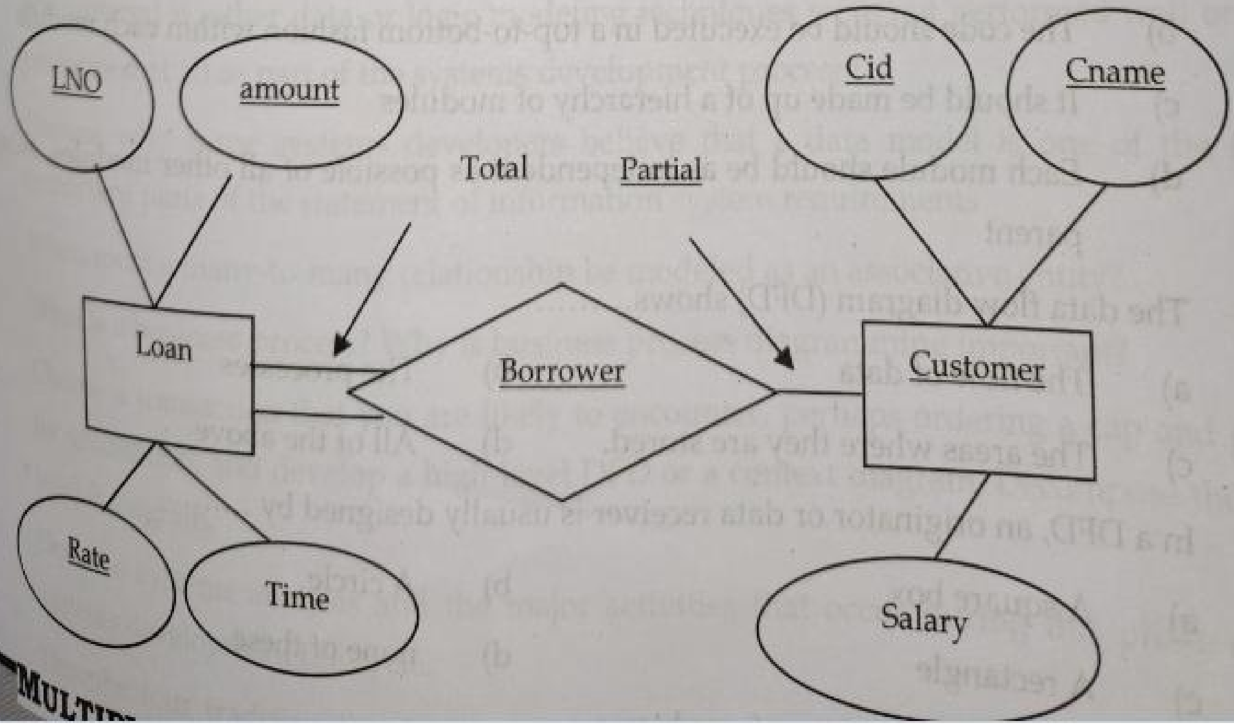
It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set. That is why; it is also called as optional participation. Partial participation is represented using a single line between the entity set and relationship set. If some courses 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.



Using set, it can be represented as,



Every student in Student Entity set is participating in relationship but there exists a course C3 which is not taking part in the relationship. Thus participation of student relation with relationship 'Enrolled in' is called total and participation of course relation with given relationship is called partial. Similarly, let's take an another example, consider Customer and Loan entity sets in a banking system, and a relationship set borrower between them indicates that only some of the customers have Loan but every Loan should be associated with some customer. Therefore there is total participation of entity set Loan in the relationship set borrower but participation of entity set customer is partial in relationship set borrower. Here, Loan entity set cannot exist without Customer entity set but existence of Customer entity set is independent of Loan entity set.



- **Representing Supertypes And Subtypes (274)**
- Often two or more entity types seem very similar (maybe they have almost the same name), but there are a few differences. That is, these entity types share common properties but also have one or more distinct attributes or relationships. To address this situation, the E-R model has been extended to include supertype/subtype relationships.
- A **subtype is a** sub grouping of the entities in an entity type that is meaningful to the organization. **For example, STUDENT** is an entity type in a university. Two subtypes of STUDENT are GRADUATE STUDENT and UNDERGRADUATE STUDENT. A **supertype is a generic entity type that has a relationship with one or** more subtypes.
- **Business Rules:** Conceptual data modeling is a step-by-step process for documenting information requirements, and it is concerned with both the structure of data and with rules about the integrity of those data. **Business rules are** specifications that **preserve the integrity of the logical data model**. Four basic types of business rules are as follows:

1. Entity integrity: Each instance of an entity type must have a uniqueidentifier that is not null.

2. Referential integrity constraints: Rules concerning the relationshipsbetween entity types.

3. Domains: Constraints on valid values for attributes.

4. Triggering operations: Other business rules that protect the validity ofattribute values.

- **Domains:** A domain is the set of all data types and ranges of values that attributes may assume.
- **triggering operations:** A **triggering operation (also called a trigger) is an assertion or rule that governs** the validity of data manipulation operations such as insert, update, and delete. The scope of triggering operations may be limited to attributes within one entity or it may extend to attributes in two or more entities.

- **Role of Packaged Conceptual Data Models:** There are two principal types of packaged data models: **universal data models** applicable to nearly any business or organization and **industry-specific data models**.
- **Universal Data Models**
- Numerous core subject areas are common to many (or even most) organizations, such as customers, products, accounts, documents, and projects. Although they differ in detail, the underlying data structures are often quite similar for these subjects. Further, there are core business functions such as purchasing, accounting, receiving, and project management that follow common patterns. **Universal data models** are templates for one or more of these subject areas and/or functions. All of the expected components of data models are generally included: **entities**, **relationships**, **attributes**, **primary** and **foreign keys**, and even sample data.

- **Industry-Specific Data Models**

Industry-specific data models are generic data models that are designedto be used by organizations within specific industries. Data models are available for nearly every major industry group, including health care, telecommunications, discrete manufacturing, process manufacturing, banking, insurance, and higher education. These models are based on the premise that data model patterns for organizations are very similar within a particular industry (“a bank is a bank”). However, the datamodels for one industry (such as banking) are quite different from those for another (such as hospitals).