Modeling Tools for Systems Analyst

Context Diagram:

- Include a title that states what system is being modeled and whether it is an existing system or a proposed system.
- One single process /bubble in shape of circle-labeled with system name or verb-object phrase
- Show only System, Environmental Entities(EE), and Data Flows
- Matches with Level 0 DFD

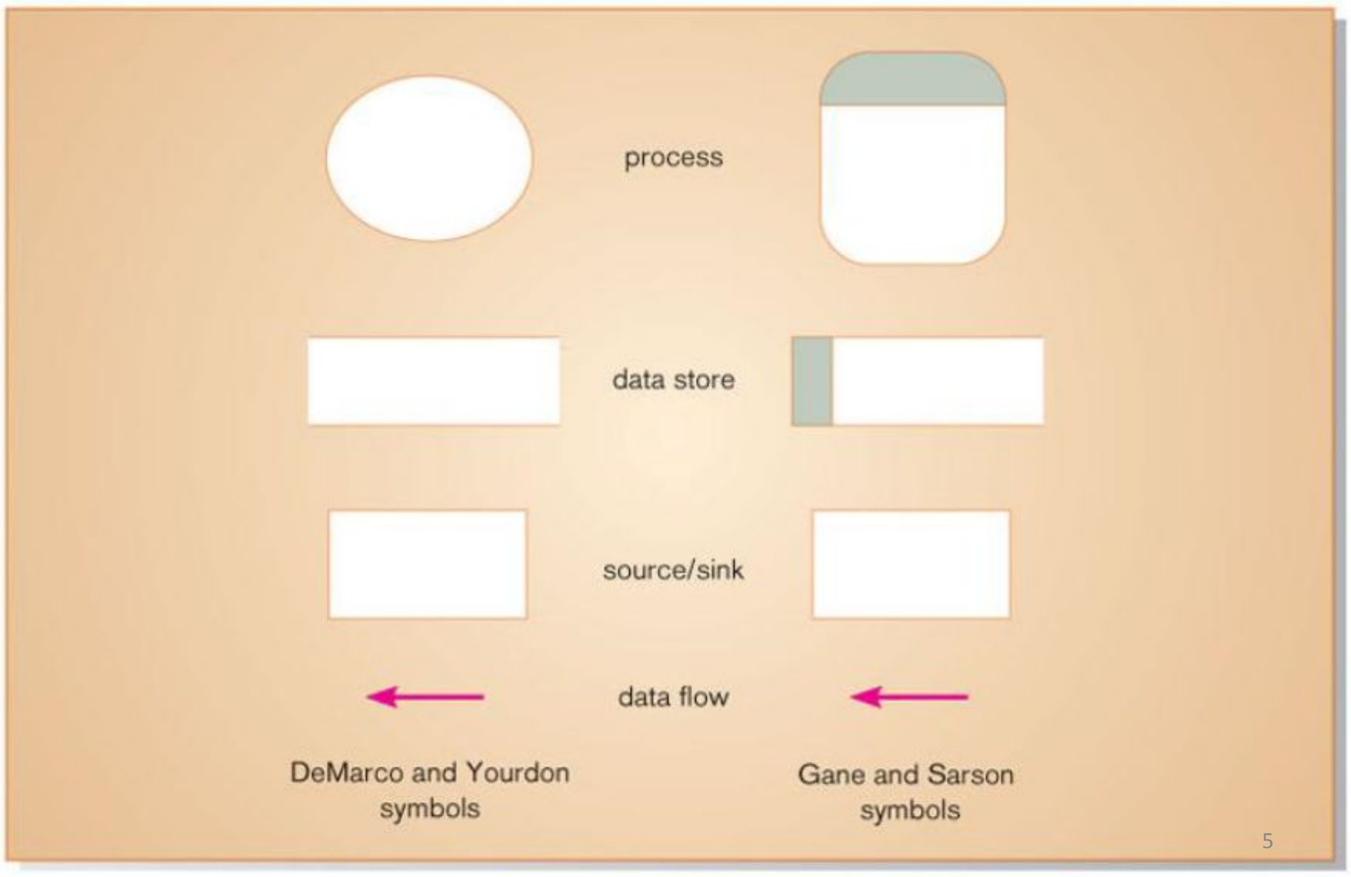
Data Flow Diagram (DFD)

- Process modeling involves graphically representing the functions or processes that capture, manipulate, store, and distribute data between a system and its environment and between components within a system.
- A common form of a process model is a data flow diagram (DFD).

Data Flow Diagram (DFD) (cont..)

- A data flow diagram (DFD) is a tool that depicts the flow of data through a system and the work or processing performed by that system.
- It is also called bubble chart, transformation graph, or process model.
- There are two different sets of data flow diagram symbols, but each set consists of four symbols that represent the same things: ***data flows***, **data stores**, **processes**, and **sources/sinks (or external entities).**

The figure below shows two different sets of symbols developed by DeMacro and Yourdon, Gane and Sarson



- According to ***Gane and Sarson, rounded rectangles*** *represent* ***process or work*** *to be done,* ***squares*** *represent* ***external agents***, ***open-ended boxes*** *represent* ***data store*** *(sometimes called files or databases), and* ***arrows*** *represent* ***data flows or inputs and outputs*** *to and from the processes.*

- According to ***DeMacro and Yourdon*** *,* ***circle*** *represent* ***process or work*** *to be done,* ***rectangle*** *represent* ***external agents****,* ***open-ended boxes*** *represent (sometimes called files or databases),* ***data store*** *and* ***arrows*** *represent* ***data flows or inputs and outputs*** *to and from the processes.*

DFDs Components:

- ***Process*** *is the work or actions performed on data so that they are transformed, stored or distributed.*
  - Processes must transform data
  - Must be numbered
  - Numbering must be consistent across all levels of DFDs
  - Must have verb-object format
  - Actors should be included, but not necessary for Figure 0 DFD
  - No more than 7 processes per DFD

DFDs Components(Cont…):

- ***Data store*** *is the data at rest (inside the system) that may take the form of many different physical representations.*
  - Open rectangle
  - Must be labeled with S number or D number
  - Consistent across all diagrams and levels
  - Do not communicate with each other or EEs directly

DFDs Components(Cont…):

- ***External entity (source/sink)*** *is the origin and/or destination of data.*
  - No data flows between EEs
- ***Data flow*** *represents data in motion, moving from one place in a system to another.*
  - Arrow pointing in direction of data flow
  - Data flows must match exactly between higher-level and lower-level DFDs
  - No two data flows should have the same name

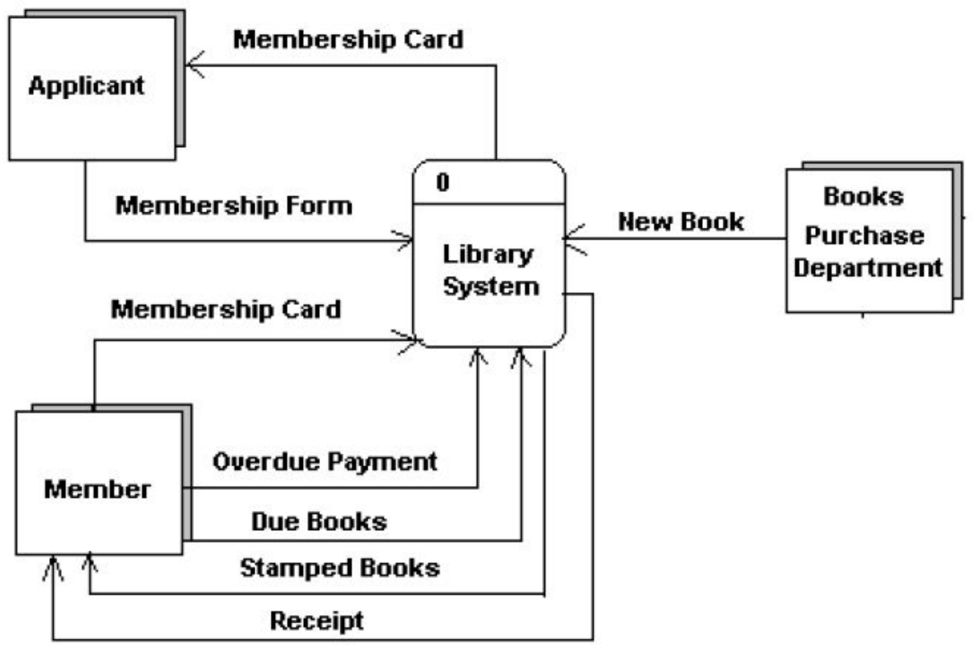
Developing DFDs

- The highest-level view of a system is called **context diagram or context DFD**.
- A context DFD is used to document the scope of the project of development.
- It is also called **environmental model**.

Developing DFDs (Cont…)

- A project's scope defines what aspect of the business, an information system or application is supposed to support and how the system being modeled must interact with other systems and the business as a whole.
- Because the scope of any project is always subject to change, the context diagram is also subject to constant change.
- The context DFD contains one and only one process and it has **no data storage**.
- Sometimes this process is identified by the number "0".

The figure below shows the context diagram of library system:



Developing DFDs (Cont…)

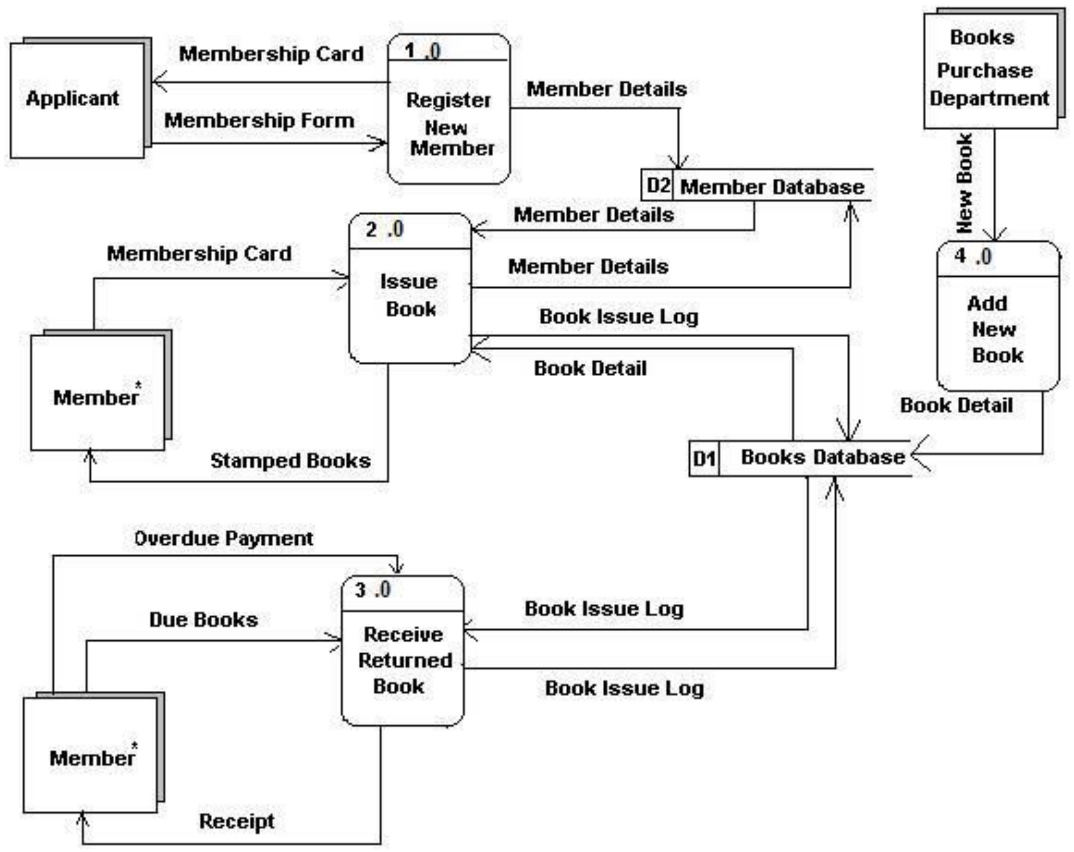
- The next step is to draw a DFD with major processes that are represented by the single process in the context diagram.
- These major processes represent the major functions of the system.
- This diagram also includes data stores.

Developing DFDs (Cont…)

- A level-0 diagram is a DFD that represents a system’s major processes and data flows.
- Level-0 DFD is also known as context diagram.
- In this diagram, sources/sinks should be same as in the context diagram.
- Each process has a number that ends in .0

- We always started with a high-level context diagram.
- **Functional decomposition** is an iterative process of breaking the description or perspective of a system down into finer and finer detail.
- When we repeat the decomposition until we reached the point where no subprocesses can logically be broken down any further, we get the lowest level of DFD called primitive DFD.

- When we decompose level-0 DFD, we get level-1 DFD.
- In level-1 DFD, we label subprocesses of process 0 to 0.1, 0.2, OR subprocesses of process 1.0 to 1.1, 1.2, and so on.
- Similarly, subprocesses of process 0.2 to 0.2.1, 0.2.2, and so on.
- In general, a level-n diagram is a DFD that is generated from n nested decompositions from a level-0 diagram**.**



Level 1 DFD

Entity Relationship Diagram (E-R Diagram)

Entity Relationship Diagram (E-R Diagram)

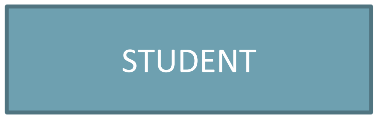
- During *analysis phase, a systems analyst uses entity relationship data model (E-R model) as a conceptual data model.*
- *A conceptual data model is a detailed model that captures the overall structure of organizational data while being independent of any database management system or other implementation consideration.*
- *And an 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-R diagram), which is a graphical representation of an E-R model. It has three basic concepts: entities, attributes, and relationships.*

Entities

- An entity is a person, place, object, event, or concept in the user environment about which the organization wishes to capture and store data.
- An **entity type (sometimes called an entity class or entity set) is a collection of entities that share common properties or characteristics.**

Each entity type in E-R diagram is given a name. When **naming entity types**, we should use the following **guidelines**:

- An entity type name is a ***singular noun*** *like CUSTOMER, STUDENT, or AUTOMOBILE.*
- An entity type name should be ***descriptive and specific*** *like PURCHASE ORDER for orders placed with suppliers to distinguish it from CUSTOMER ORDER for orders placed by customers.*
- An entity type name should be ***concise*** *like* ***REGISTRATION*** *for the event of a student registering for a class rather than* ***STUDENT REGISTRATION FOR CLASS****.*
- An entity type in E-R diagram is drawn using **rectangle.**
- **For example, the figure below shows STUDENT entity type.**



Attributes

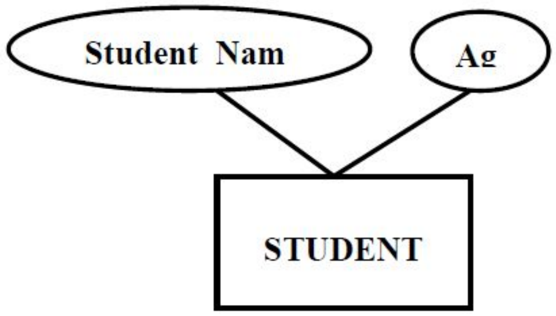
- Each entity type has a set of attributes associated with it.
- An attribute is a property or characteristic of an entity that is of interest to the organization.
- For example, STUDENT entity type may have Student_ID, Student_Name, Home_Address, Phone_Number, and Major as its attributes. Similarly, EMPLOYEE entity type may have Employee_ID, Employee_name, and Skill, Address as its attributes.

Each attribute in E-R diagram is given a name.

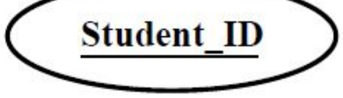
When naming **attributes**, we should use the following **guidelines**:

- An attribute is a ***noun*** *like Customer_ID, Age, or Skill.*
- An attribute name should be ***unique***. *No two attributes of the same entity type may have the same name, and it is desirable, for clarity purposes, that no two attributes across all entity types have the same name.*
- To make an attribute unique and **clear**, *each attribute name should follow a standard form. For example, Student_GPA as opposed to GPA_of_Students.*
- Similar attributes of different entity types should use the **similar but distinguishing names**. For example, Student_Residence_City_Name for STUDENT entity type and Faculty_Residence_City_Name for FACULTY entity type.

- An attribute in E-R diagram is drawn using an **ellipse. We place attribute name inside the ellipse with a line connecting it to the associated entity type. For example, the figure** below shows Student_Name and Age attributes of STUDENT entity type.

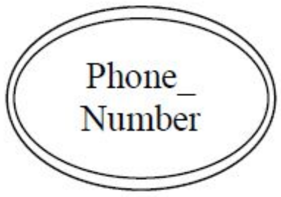


- Every entity type must have an attribute or set of attributes that distinguishes one instance from other instances of the same type.
- A **candidate key is an attribute or combination of attributes that uniquely identifies each instance of an entity type. For example, a candidate key for a STUDENT entity type might be Student_ID.**
- Some entity type may have more than one candidate key. In such a case, we must choose one of the candidate keys as the identifier.
- An **identifier (or primary key) is a candidate key that has been selected to be used as the unique characteristic for an entity type. We can use the following selection rules to select identifiers:**
  - Choose a candidate key that will **not change** its value over the life of each instance of the entity type.
  - Choose a candidate key that will **never be null**.
- The name of the identifier is underlined on an E-R diagram. For example, the figure below shows Student_Id as an identifier for a STUDENT entity type.



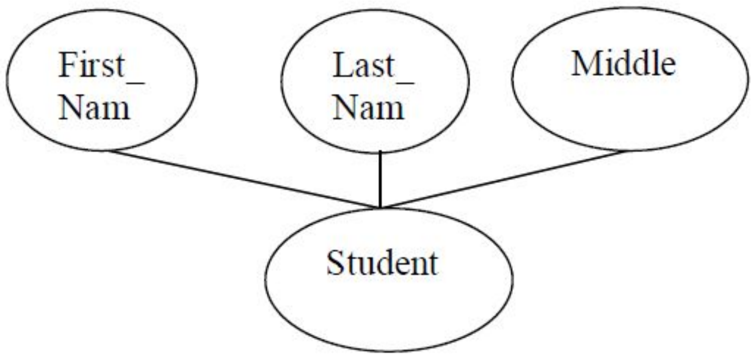
A **multivalued attribute** may take more than one value for each entity instance.

For example Phone_Number attribute of STUDENT entity type. We use a double-lined ellipse to represent multivalued attribute.



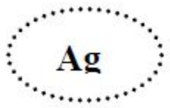
An attribute that has meaningful component parts is called **composite attribute.**

For example, Student_Name attribute of STUDENT entity type has First_Name, Middle_Name, and Last_Nmae as its component parts.



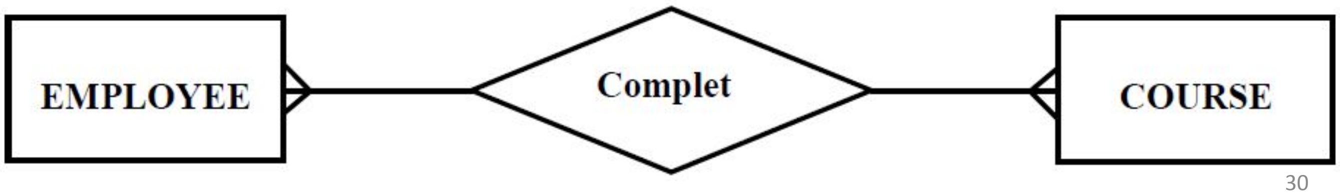
An attribute whose value can be computed from related attribute values is called **derived attribute.**

For example, value of Age attribute is computed from Date_of_Birth attribute. We use dashed ellipse to denote derived attribute.



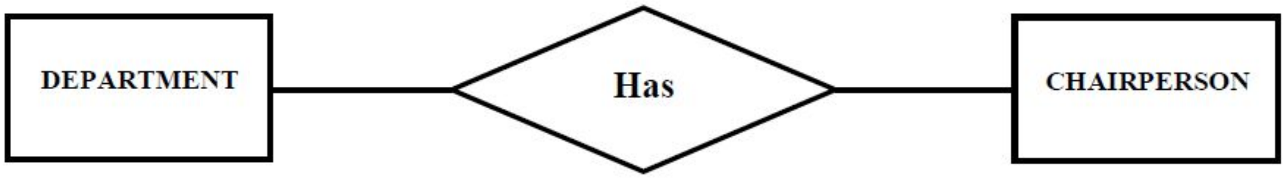
Relationships

- A relationship is an association between the instances of one or more entity types that is of interest to the organization. We use diamond to denote relationships. Relationships are labeled with *verb phrases. For example, if a training department in a company is interested in tracking with training courses each of its employees has completed, this leads to a relationship (called Completes) between the EMPLOYEE and COURSE entity types as shown in the figure below.*

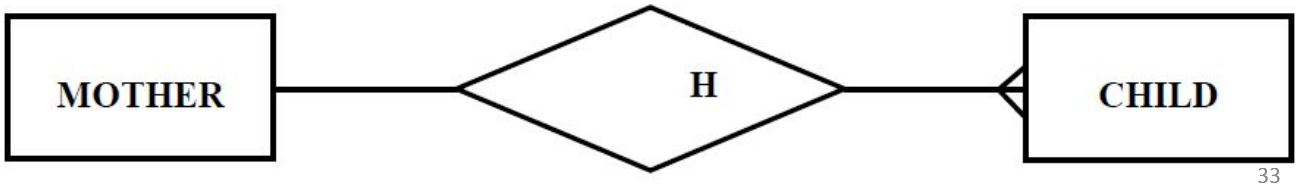


- As indicated by arrows, the *cardinality of this relationship is many-to-many since each employee may complete more than one course, and each course may be completed by more than one employee.*
- The **cardinality of a relationship** is the number of instances of one entity type that can (or must) be associated with each instance of another entity type. The cardinality of a relationship can be in one of the following **four** forms**:**
  - ***one-to-one,***
  - ***one-to-many,***
  - ***many-to-one, and***
  - ***many-to-many.***

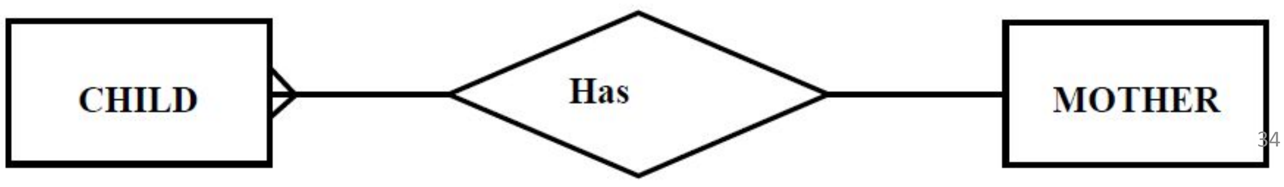
One-to-one: An instance in entity type A is associated with at most one instance in entity type B, and an instance in entity type B is associated with at most one instance in entity type A. For example, cardinality between DEPARTMENT and CHAIRPERSON.



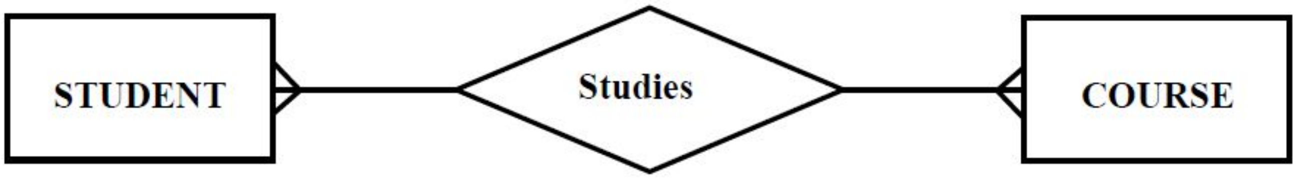
- **One-to-many:** An instance in entity type A is associated with any number (zero or more) of instances in entity type B, and an instance in entity type B, however, can be associated with at most one instance in entity type A. For example, cardinality between MOTHER and CHILD.



- **Many-to-one:** An instance in entity type A is associated with at most one instance in entity type B, and an instance in entity type B, however, can be associated with any number (zero or more) of instances in entity type A. For example, cardinality between CHILD and MOTHER.



- **Many-to-many**: An instance in entity type A is associated with any number (zero or more) of instances in entity type B, and an instance in entity type B is associated with any number (zero or more) of instances in entity type A. For example, cardinality between STUDENT and COURSE.



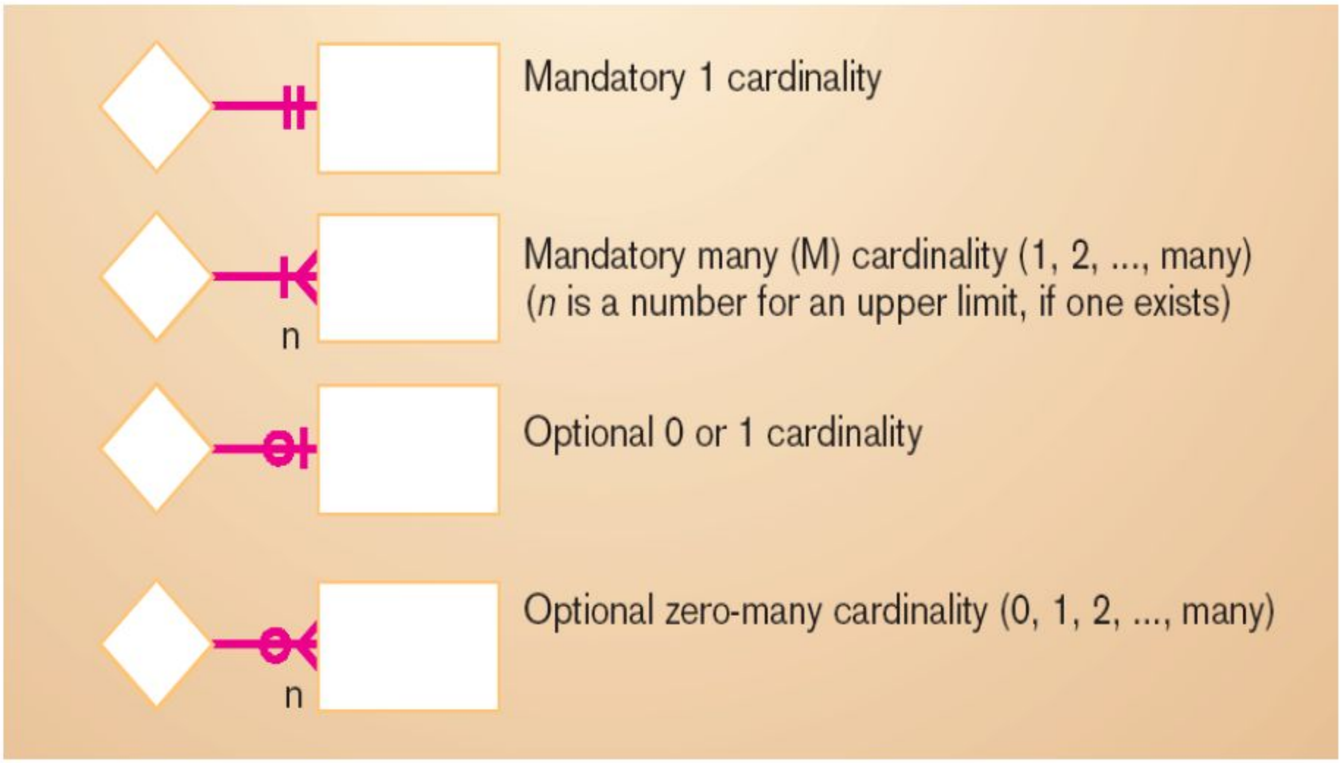

Mandatory 1 cardinality
Mandatory many (M) cardinality (1, 2, ..., many)
(n is a number for an upper limit, if one exists)
n
Optional 0 or 1 cardinality
Optional zero-many cardinality (0, 1, 2, ..., many)
n

Fig: Cardinality symbols

Some Exercises

1. Construct an ER diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.
2. Construct an ER diagram for a hospital with a set of patients and a set of doctors. Associate with each patient a log of the various tests and examinations conducted.
3. Construct an ER diagram of the library system in your college.
4. Construct an ER diagram to maintain data about students, instructors, semester, and courses in your University.
5. Construct an ERD to record the marks that students get in different exams of different course offerings.