Chapter 2
Process Modeling and Conceptual Data Modeling

Contents:

2.1 Process Modeling

Process Modeling is the process of graphically representing the function or process, that capture, manipulate, store and distribute data between a system and its environment and between components with in the system. Over the years, several different tools have been developed for process modelling. Some of the example includes data flow diagram (DFD), Unified Language Modeling (UML) diagram, Flowcharts, Grantt Charts, PERT Diagram, Decision table and so on.

2.2 Introduction to data flow diagram (DFD)

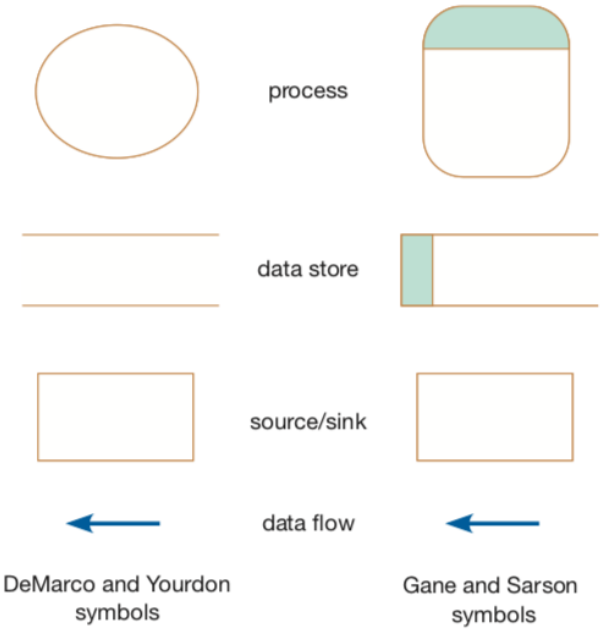
A data flow diagram (or DFD) is a graphical representation of the flow of data through an information system. It shows how information is input to and output from the system, the sources and destinations of that information, and where that information is stored.

DFD Vs Flowchart

DFDs are not as good as flowcharts for depicting the details of physical systems; on the other hand, flowcharts are not very useful for depicting purely logical information flow. In fact, flowchart has been criticized by proponents of structure analysis and design because it is too physically oriented.

2.3 Concepts used in drawings DFDs

DFDs are versatile diagramming tools. With only four symbols, we can use DFD to represent both physical and logical information system. Following are the two standard sets of symbols that represent the same things: data flows, date stores, processes and the source/sinks (or external entities). The set of symbols we follow along this course was devised by Gane and Sarson (1979). The other standard set was developed by DeMarco and Yourdon (1979). Following figure illustrare the symbol used in both standard.



A **data flow** shows data in motion, moving from one place in a system to another. A data flow could represent data on a customer order form or a payroll check; it could also represent the results of a query to a database, the con- tents of a printed report, or data on a data entry computer display form. A data flow is data that move together, so it can be composed of many individual pieces of data that are generated at the same time and that flow together to common destinations.

A **data store** is data at rest. A data store may represent one of many different physical locations for data; for example, a file folder, one or more computer-based file(s), or a notebook. To understand data movement and handling in a system, it is not impor- tant to understand the system's physical configuration. A data store might contain data about customers, students, customer orders, or supplier invoices.

A **process** is the work or actions performed on data so that they are transformed, stored, or distributed. When modeling the data processing of a system, it does not matter whether a process is performed manually or by a computer.

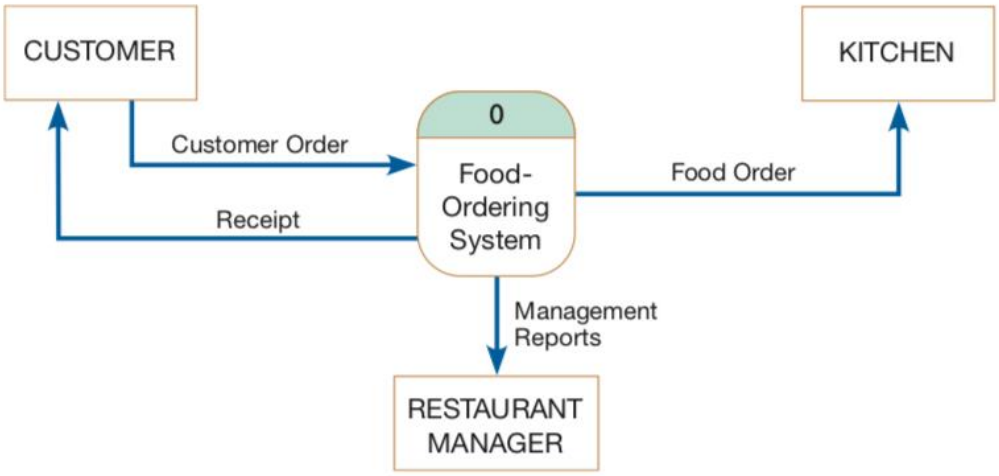
Finally, a **source/sink** is the origin and/or destination of the data. Sources/sinks are sometimes referred to as external entities because they are outside the system. Once processed, data or information leave the system and go to some other place. Because sources and sinks are outside the system we are studying, many of the characteristics of sources and sinks are of no interest to us. In particular, we do not consider the following:

- Interactions that occur between sources and sinks
- What a source or sink does with information or how it operates (i.e., a source or sink is a "black box")
- How to control or redesign a source or sink because, from the perspective of the system we are studying, the data a sink receives and often what data a source provides are fixed

- How to provide sources and sinks direct access to stored data because, as external agents, they cannot directly access or manipulate data stored within the system; that is, processes within the system must receive or distribute data between the system and its environment.

2.4 DFD design (up to level 2)

To illustrate how DFDs are used to model the logic of data flows in information systems, let us consider food-ordering system of a restaurant. The restaurant uses an information system that takes customer orders, sends the orders to the kitchen, monitors goods sold and inventory, and generates reports for management.



Above diagram illustrate highest view of this system, is called a **context diagram**. **Context diagram** contains only one process, no data stores, four data flows, and three sources/sinks. The single process, labeled 0, represents the entire system; all context diagrams have only one process, labeled 0. The sources/sinks represent the environmental boundaries of the system. Because the data stores of the system are conceptually inside one process, data stores do not appear on a context diagram.

Diagram below called a level-0 diagram because it represents the primary individual processes in the system at the highest possible level. Each process has a number that ends in .0 (corresponding to the level number of the DFD). The analyst must determine which processes are represented by the single process in the context diagram. As we can see in Figure below we have identified four separate processes. The main processes represent the major functions of the system, and these major functions correspond to actions such as the following:

1. Capturing data from different sources (e.g., Process 1.0)

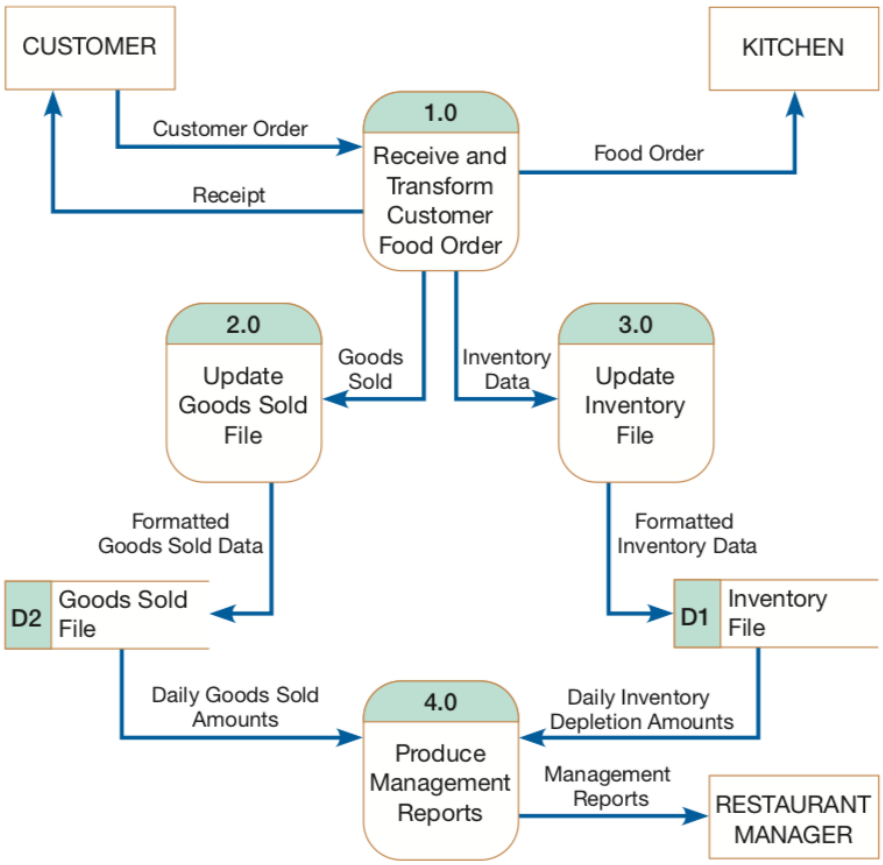
2. Maintaining data stores (e.g., Processes 2.0 and 3.0)

3. Producing and distributing data to different sinks (e.g., Process 4.0)

4. High-level descriptions of data transformation operations (e.g., Process 1.0)

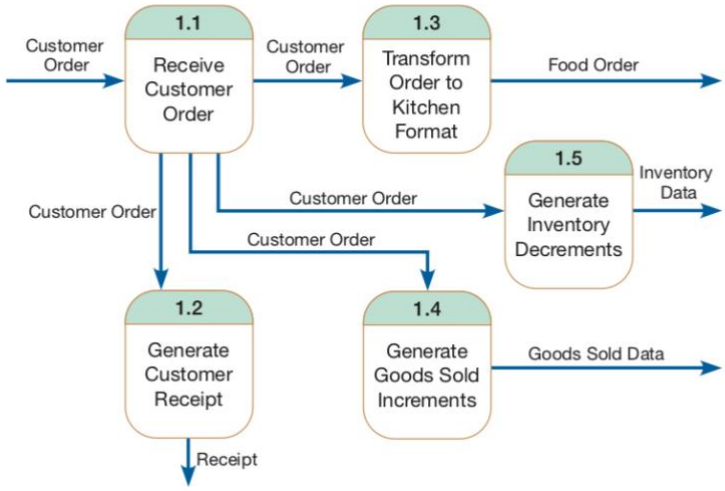
These major functions often correspond to the activities on the main system menu.

We see that the system begins with an order from a customer, as was the case with the context diagram. In the first process, labeled 1.0, we see that the customer order is processed. The result is four streams, or flows, of data: (1) the food order is transmitted to the kitchen, (2) the customer order is transformed into a list of goods sold, (3) the customer order is transformed into inventory data, and (4) the process generates a receipt for the customer.



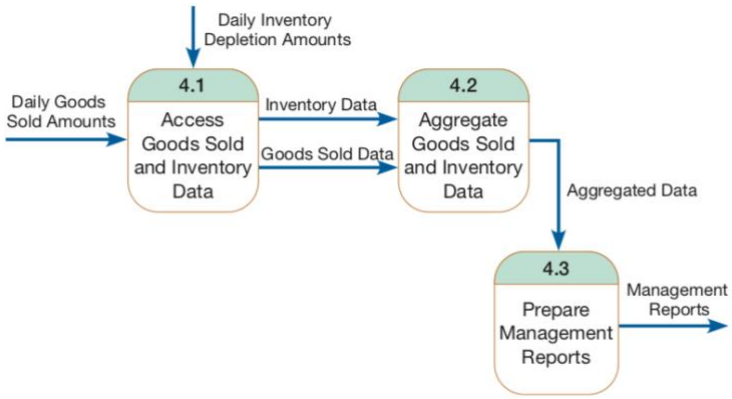
Functional decomposition is an iterative process of breaking the description or perspective of a system down into finer and finer detail. This process creates a set of hierarchically related charts in which one process on a given chart is explained in greater detail on another chart. For above system, we broke down, or decomposed, the larger system into four processes. Each resulting process (or subsystem) is also a candidate for decomposition. Each process may consist of several subprocesses. Each subprocess may also be broken down into smaller units. Decomposition continues until we have reached the point at which no subprocess can logically be broken down any further.

Lets us try decompose above level-0 DFD. The first process in above figure, called *Receive and Transform Customer Food Order*, transforms a customer's verbal food order (e.g., "Give me two cheeseburgers, one small order of fries, and one regular orange soda") into four different outputs. Process 1.0 is a good candidate process for decomposition. Think about all of the different tasks that Process 1.0 has to perform: (1) receive a customer order, (2) transform the entered order into a form meaningful for the kitchen's system, (3) transform the order into a printed receipt for the customer, (4) transform the order into goods sold data, and (5) transform the order into inventory data. At least five logically separate functions can occur in Process 1.0. We can represent the decomposition of Process 1.0 as another DFD, as shown in Figure below –



Note that each of the five processes in above figure is labeled as a subprocess of Process 1.0: Process 1.1, Process 1.2, and so on. Also note that, just as with the other DFDs we have looked at, each of the processes and data flows is named. We also notice that no sources or sinks are represented. Although we may include sources and sinks, the context and level-0 diagrams show the sources and sinks. The DFD in Figure above is called a level-1 diagram. If we should decide to decompose Processes 2.0, 3.0, or 4.0 in a similar manner, the DFDs we would create would also be level-1 diagrams. In general, a level-n diagram is a DFD that is generated from n nested decompositions from a level-0 diagram.

Process four named *Produced Management Report* can also decomposed as level 1 diagram as follows-



Process 4.3, names Prepare Management Reports can further be decomposed to get level-2 DFD as :-

2.5 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. E-R diagram is one popular tools for conceptual data modeling which is discussed below.

2.6 Entity relationship diagrams

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities. And that's why it's called "Entity" "Relationship" diagram (ERD)!

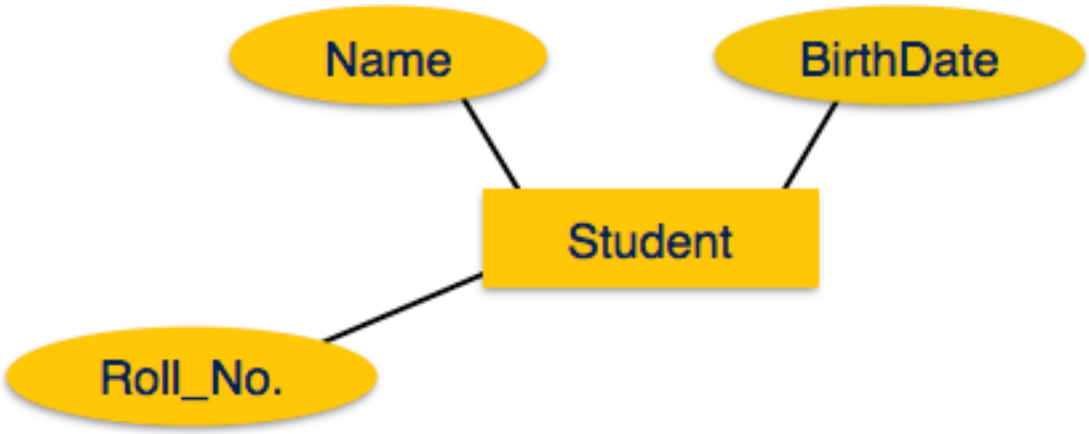
An ER Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. They use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

Entity

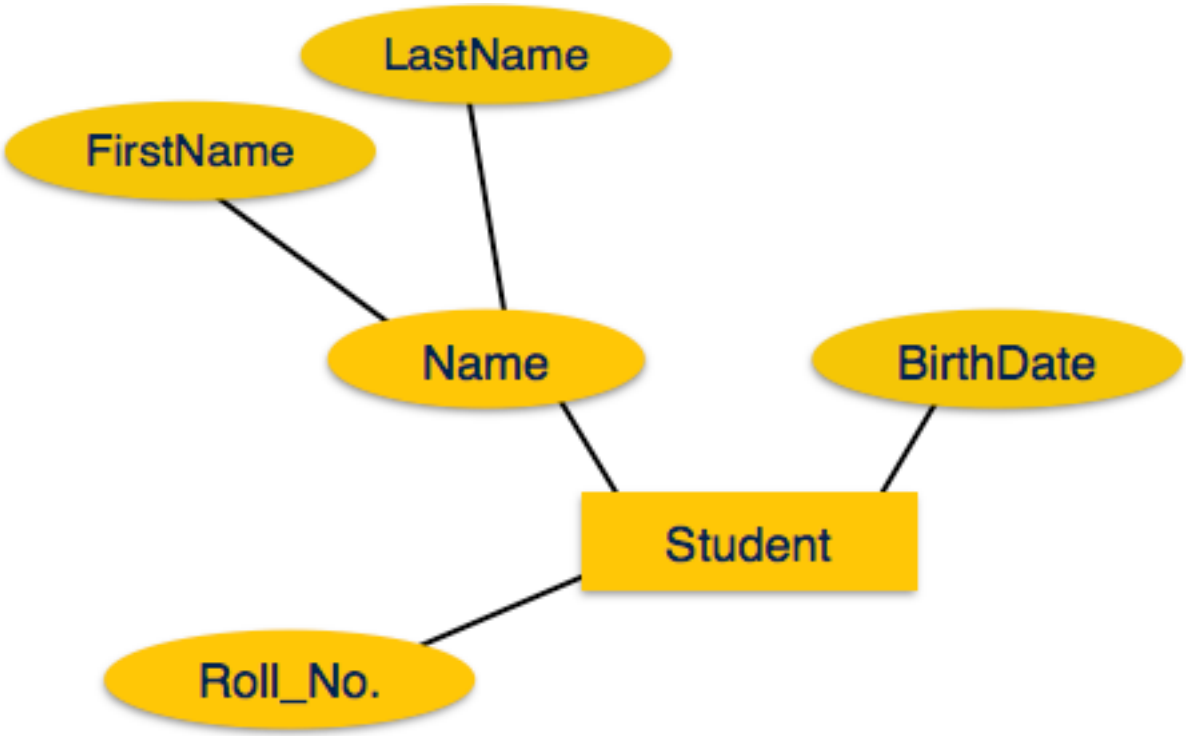
Entities are real world objects like student, teacher, car, personal_info and so on. Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Attributes

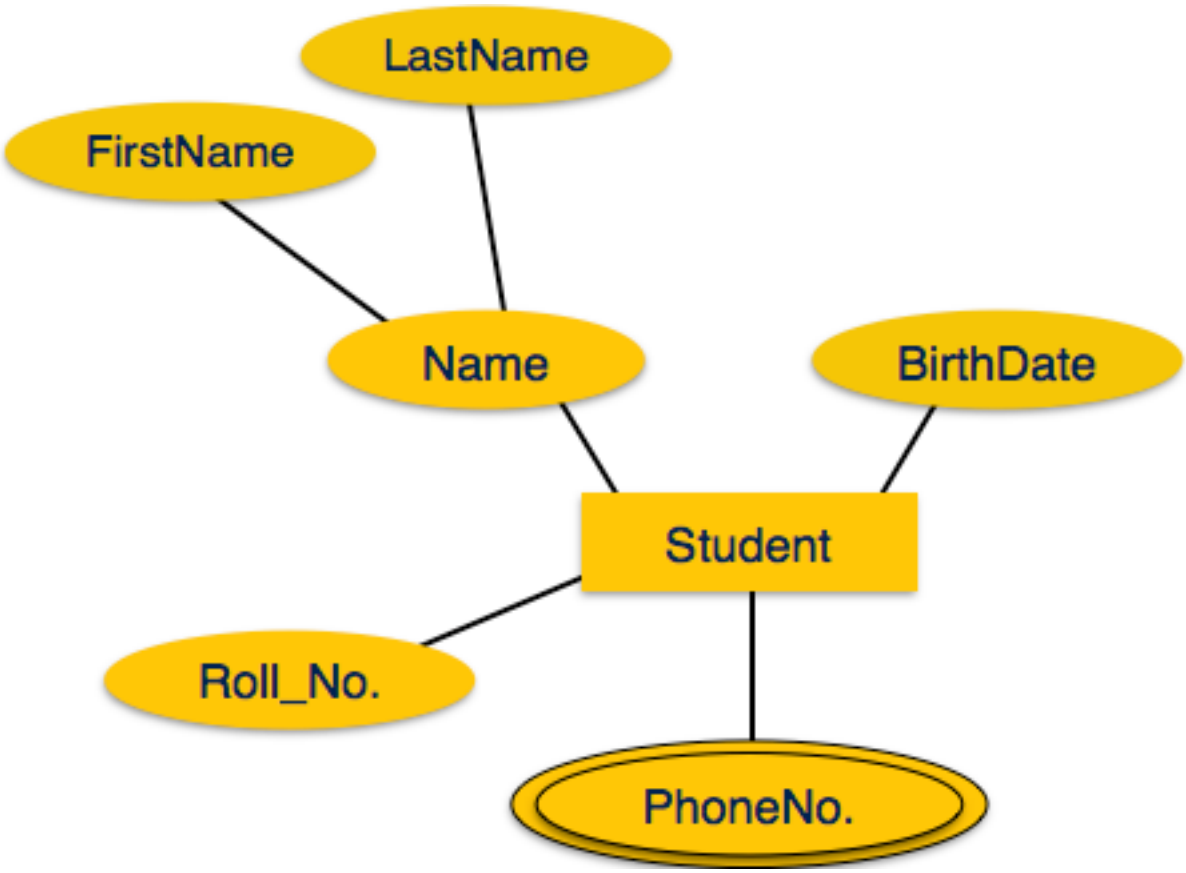
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



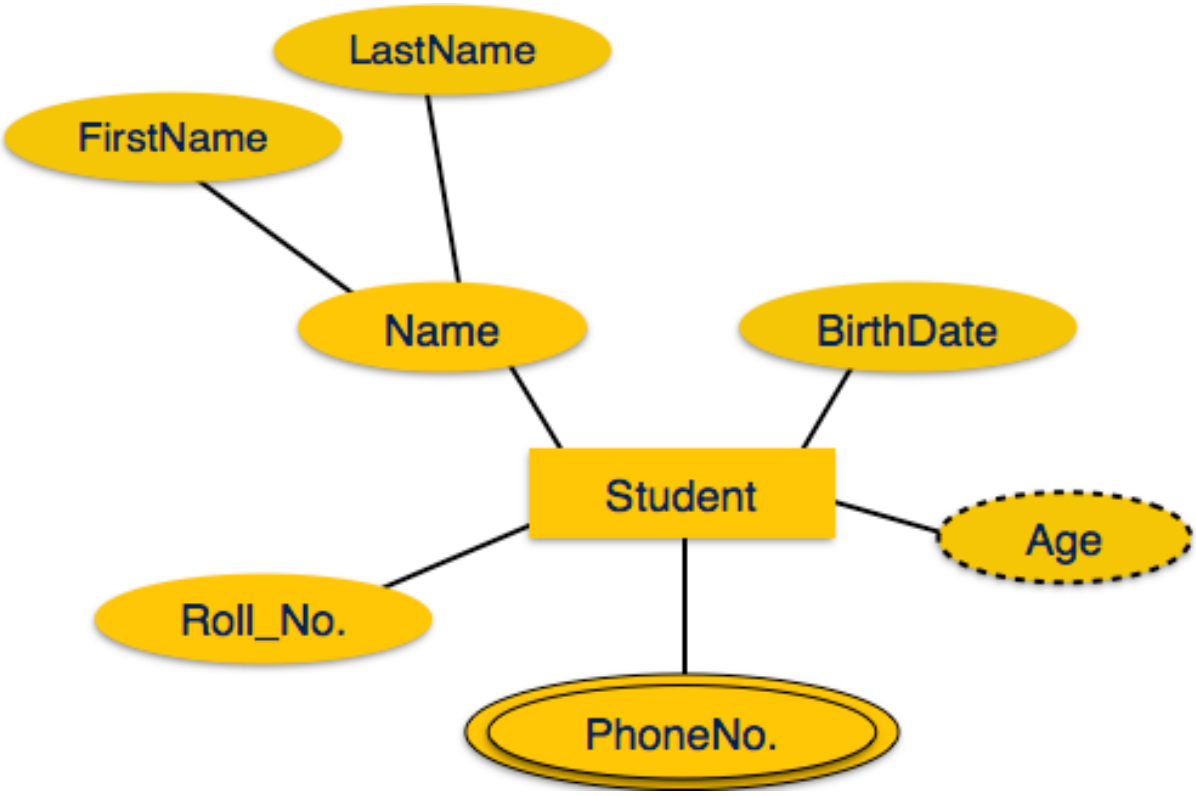
If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multivalued attributes are depicted by double ellipse.



Derived attributes are depicted by dashed ellipse.



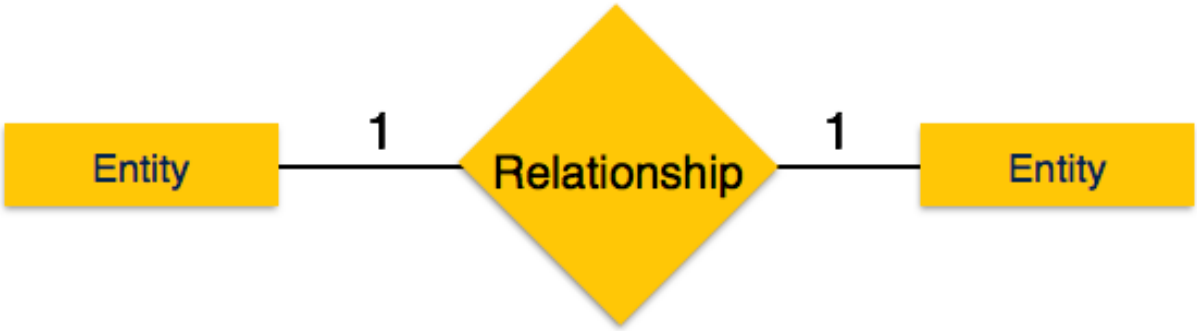
Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

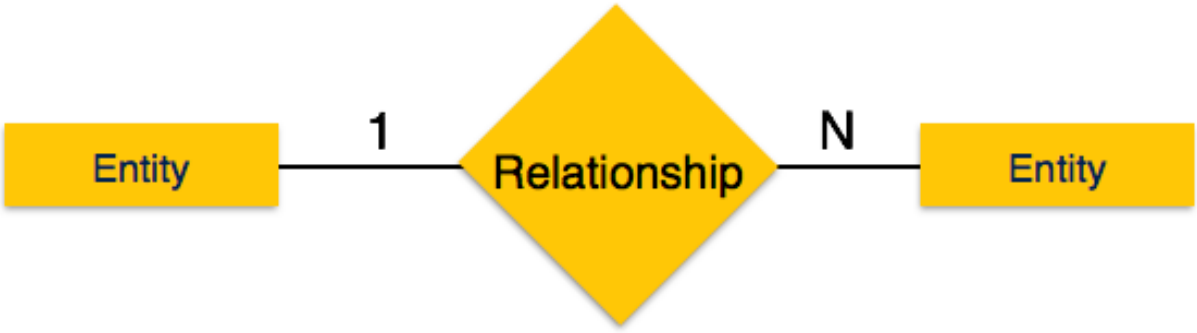
Binary Relationship and Cardinality

A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

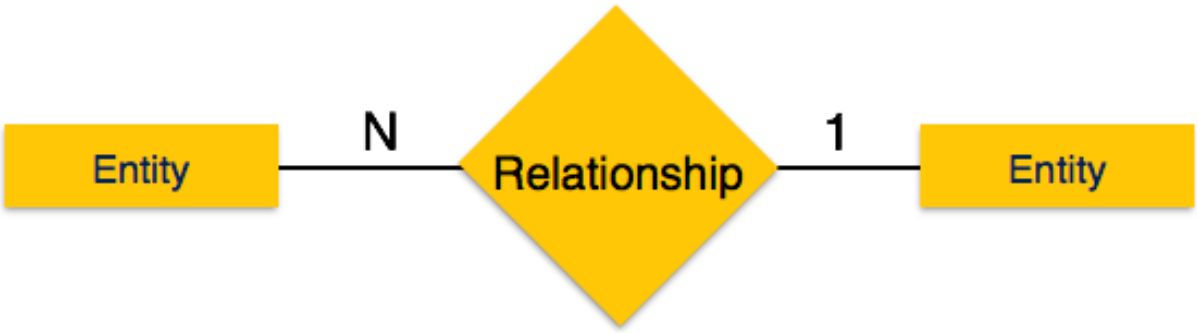
- **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



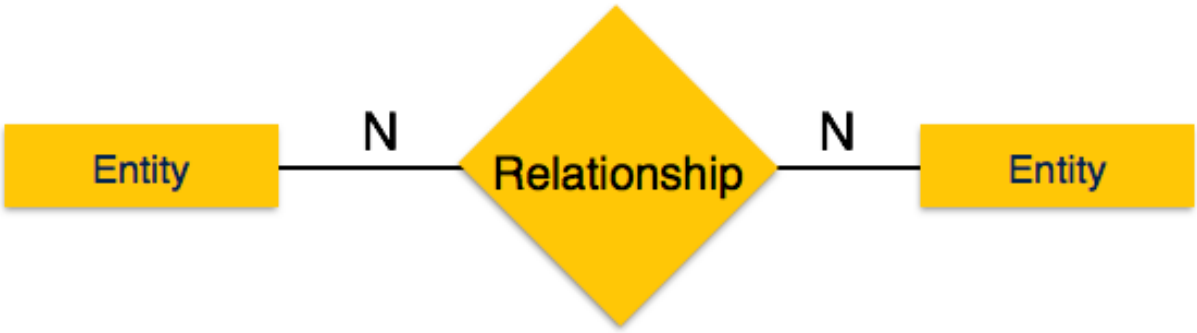
- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.



- **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.



- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



E-R diagram example: