Design & Implementation of Human-Computer Interfaces

Dr.Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati

Module No # 05
Lecture No # 22
Entity Relationship Diagram

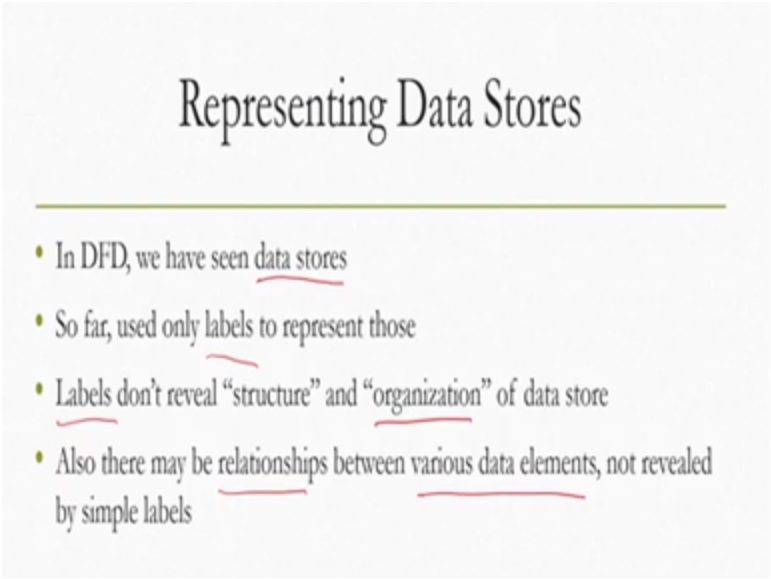
Hello and welcome to the NPTEL MOOCS course on designing and implementing human-computer interfaces. We are currently discussing a language, to express system design particularly, when we are following a function-oriented design approach. In the previous lecture, we talked about DFD or data flow diagrams which can be used as a language to express function oriented system design.

We will continue our discussion on DFD plus ERD, in lecture number 19. So, both are part of the language we can use to express function-oriented designs. We have already covered in detail the various aspects of DFD with examples. Now, we are going to discuss the other component of the language which is ERD.

(Refer Slide Time: 01:57)

ERD stands for entity relationship diagram in short ER diagram.

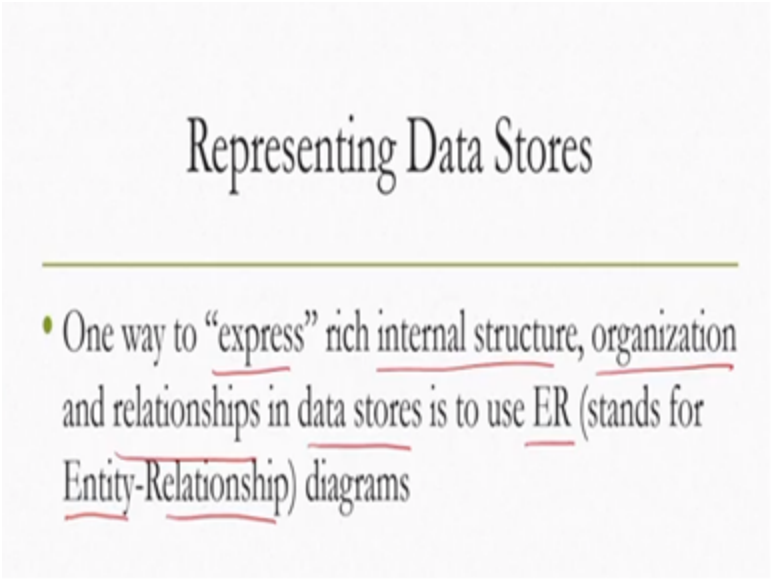
(Refer Slide Time: 02:02)



In DFD, we made use of data stores so we have seen notations to represent data stores, where we can store various types of data. But so far, in order to represent data stores, we only used a name and identifier. Only labels represent the store as a whole, but these labels that we assign to the data stores do not reveal the internal structure of the data and also how the data are organized in the data store?

So, the labels are not very informative in that sense. It only gives us an abstract idea of the presence of stored data, but it does not tell us anything more about the nature, type, and organization of that data. It is also possible that there may be relationships between various data elements, which again are not revealed by the labels that are assigned to the data stores. So, many things are hidden when we use simply a label to represent data stores in our DFD.

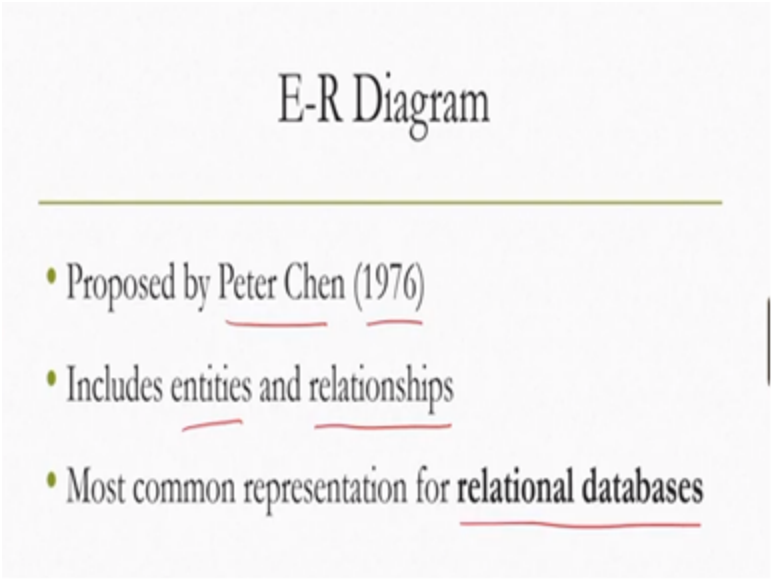
(Refer Slide Time: 03:31)



But sometimes it may be useful to learn or to represent the internal structure of the data that we are dealing with. So; that, further brainstorming can be done, and further optimization can be done before we set out to implement the system. Clearly, the notations that we have covered in DFD, do not allow us to represent the nature of the data, the organization of the data, the type of the data, or even the relationship between various data elements, that are used in a data flow diagram.

One way to express the internal structural organization and relationships that; may be present in data stores or across data stores is to use the ER diagram where the ER stands for entity relationship. So, entity relationship diagrams, or in short ER diagrams can be used to express the structure, organization, type, nature, and relationship of data stores.

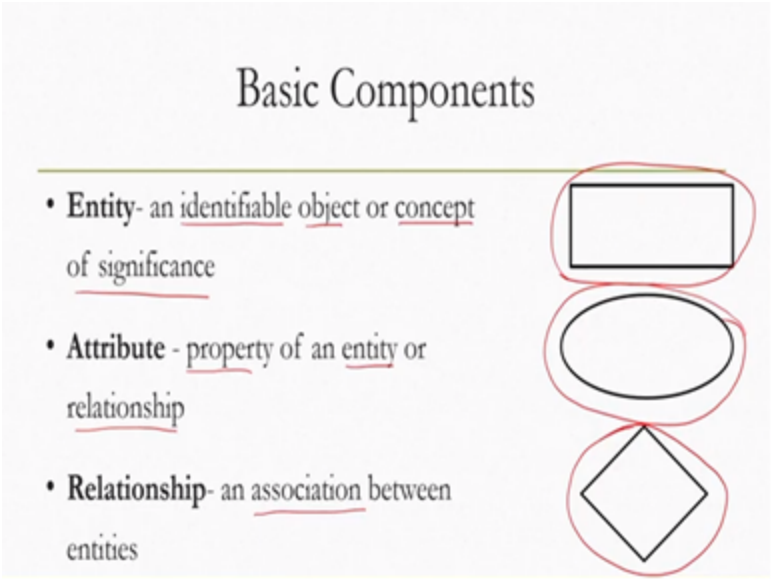
(Refer Slide Time: 05:01)



Now the ER diagram has a long history. It was first proposed by Peter Chen, way back in 1976 that was nearly 45 years ago. Now, it includes, as the name suggests, an entity relationship diagram. So, it includes entities and relationships. So, these 2 are the central concept behind the ER diagrams. ER diagrams can be used to represent relational databases, a special type of database which is used to store data, and ER diagrams are most suitable to represent relational databases.

In fact, ER diagrams can be considered to be another graphical language to represent data. In a similar way, DFD is used to represent the overall design of a system.

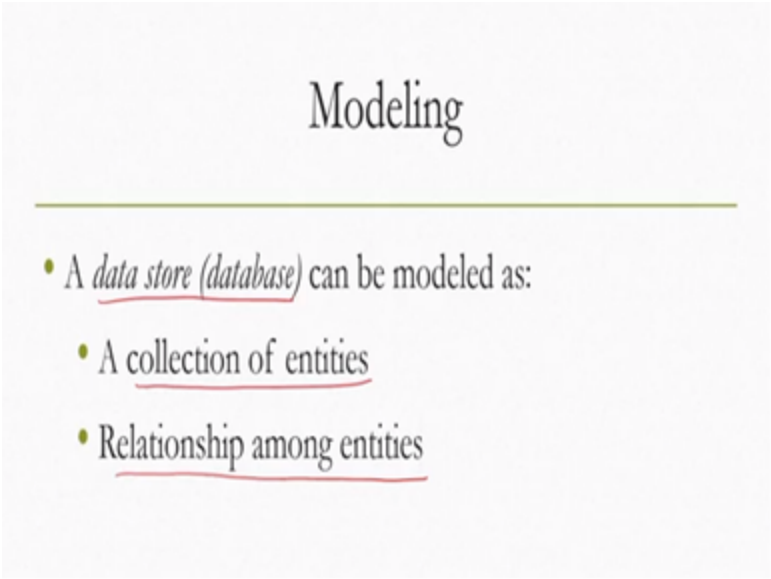
(Refer Slide Time: 06:04)



So, what are the basic components of a typical ER diagram or entity relationship diagram? There are 3 basic components. One is an entity, which is an identifiable object or concept of significance. Generally, it is represented with a rectangle as shown here. Then we have an attribute, which is a property of an entity or relationship. So, the attribute can be the property of an entity or it can be the property of even a relationship as well.

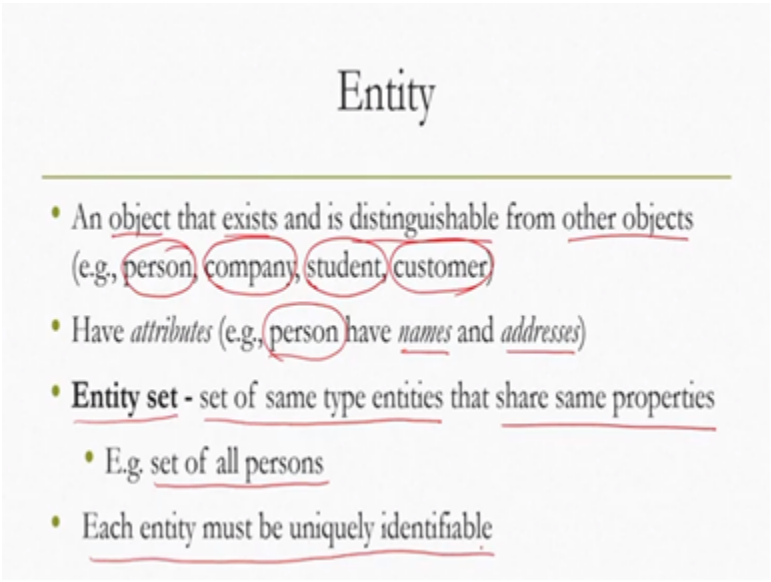
Generally, to represent an attribute, we use this elliptical shape, and finally, there is a relationship. A relationship represents an association between entities. Generally, it is represented with this particular shape as shown in the figure.

(Refer Slide Time: 07:20)



So using these 3 basic concepts entity, relationship, and attributes we can represent or model data items that are used in a system design and represented using graphical notions such as DFD. So, a data store or a database which is used in DFD can be modeled as a collection of entities with relationships among those entities. So, both are used to model the data store or databases that are used as part of the system design, namely a set of entities and the relationship between them. So, let us try to go a little deeper and learn in a little more detail the idea of the entity.

(Refer Slide Time: 08:23)

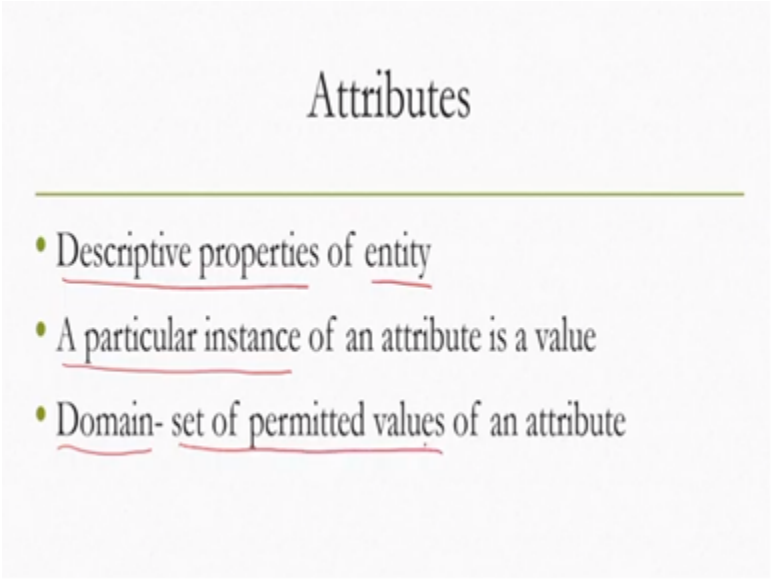


The entity is an object that exists and is distinguishable from other objects. For example, a person can be considered to be an entity, a company can be considered to be an entity, a student

can be considered to be an entity, and a customer can be considered to be an entity. So these are some of the examples of what can be considered to be an entity. Generally, entities have attributes or properties.

For example, if we consider a person to be an entity. Then the person can have attributes such as names, addresses or date of birth, or age. So, these are attributes that are assigned to the entity. Then we have the notion of the entity set. So, this is a set of same-type entities that share the same properties. For example, a set of all persons constitute an entity set. Whenever we are defining something as an entity, in order to model a data store, we have to keep in mind that each entity must be uniquely identifiable. So, we have to ensure that in order to be able to suitably model the data store.

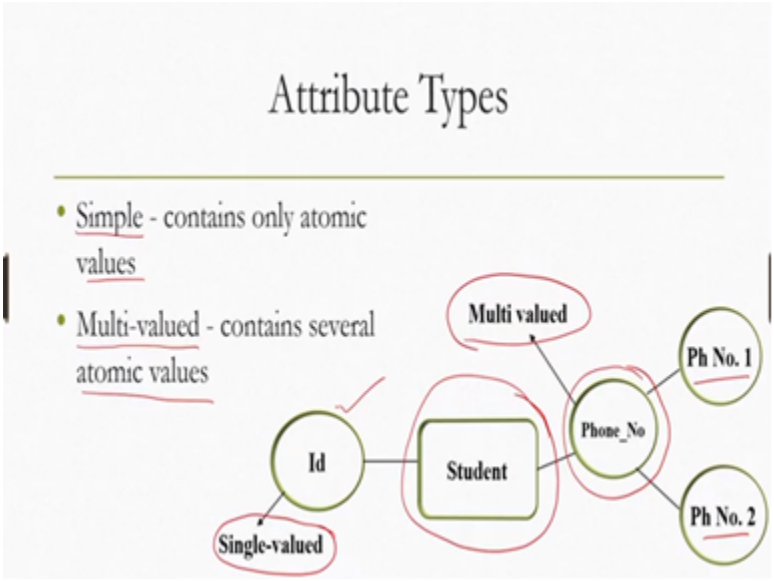
(Refer Slide Time: 09:58)



Next is the idea of attributes. So, attributes are essentially descriptive properties of the entity. If we are defining something as an entity and it has some properties. We call those properties attributes. Now, attributes are broad generic terms, we can assign values to an attribute. So, that is a particular instance of an attribute. Whenever we are assigning a value to the attribute that means that is an instance of the attribute.

Then we can define a domain of values for an attribute, which is a set of permitted values for that particular attribute. So, whenever we are defining an attribute for an entity, we can also define the required domain of values that are permitted for that particular attribute.

(Refer Slide Time: 10:59)

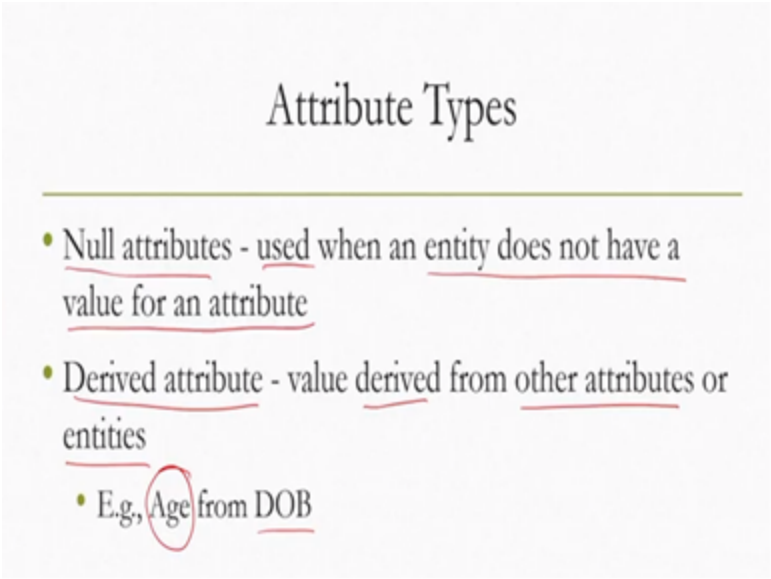


So, there are broadly 2 types of attributes. One is a simple attribute, which contains only atomic values and one can be a multi-valued attribute, which can contain several atomic values. Let us consider one example; suppose we have defined an entity to be a student. So, we have defined the student to be an entity. Now, the student has several attributes. One of those attributes is a student id or identifier.

So here, we can store only one atomic value. So, this is a single-valued attribute. Now suppose, along with the student we are also storing phone numbers. So, that can be another attribute. However, a student can possess multiple phone numbers. So, for each student, we might have more than one phone number. So, we can store phone number 1, for number 2 in that way up to the nth phone number.

So, the phone number attribute can contain more than one atomic value where each phone number can be considered to be an atomic value. Such an attribute is a multi-valued attribute so, we have 2 types of attributes one is a single value, and one is multi-valued.

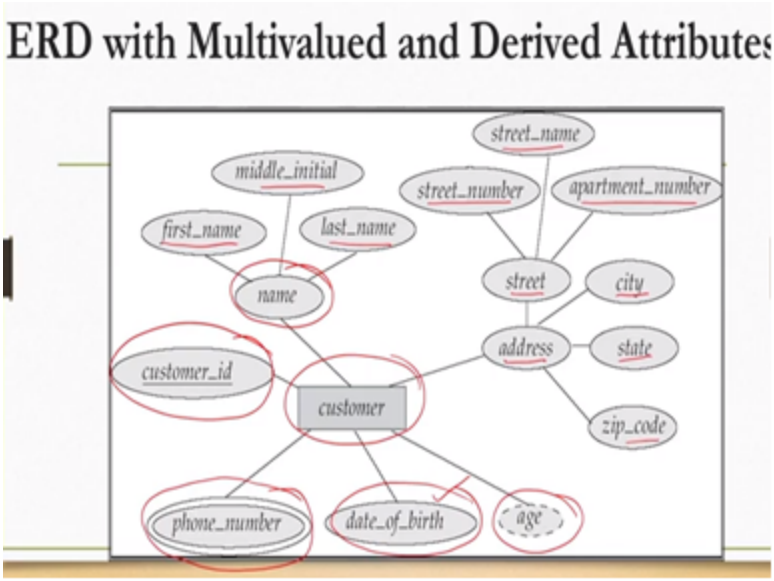
(Refer Slide Time: 12:33)



There can be some null attributes as well. Generally, null attributes are used when an entity does not have a value for an attribute. We can also have derived attributes where the value that is assigned to the attribute is generally derived from other attributes or entities. So, if we are assigning values to some attributes which can be derived from other attributes, then those attribute values are derived values and those attributes are called derived attributes.

For example, if we have a date of birth to be an attribute as well as age to be another attribute, then age can be derived from the date of birth. So, age is a derived attribute. Let us see one more example of these different types of attribute values in an ER diagram.

(Refer Slide Time: 13:32)

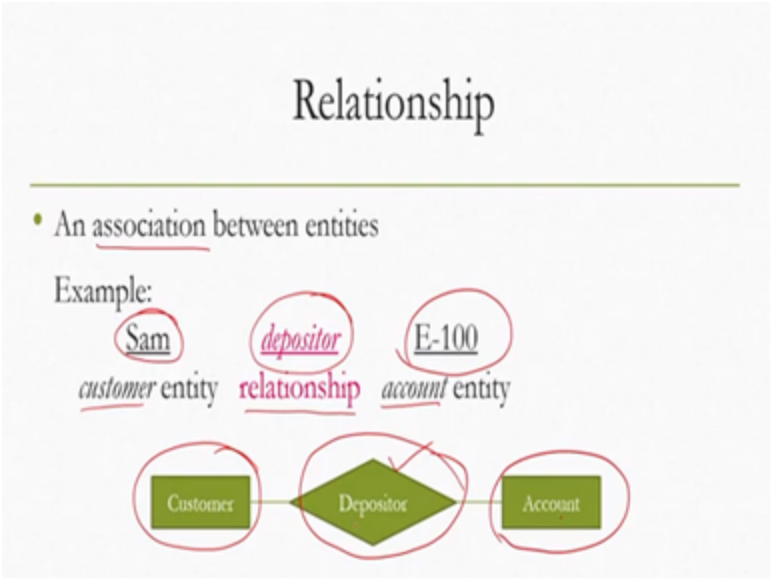


So, this example shows a typical ER diagram as you can see, we have an entity and several attributes. Now, as you can see some of the attributes are single-valued like date of birth and customer id. Some are multi-valued like the name which can have 3 atomic values; first name, middle name, and last name. Similarly, addresses can be multivalued which can have street, city, state, and zip code as atomic values.

Whereas street can itself be a multi-valued attribute having street number, street name, apartment number, and so on. We can also have derived attributes like age. Now, this attribute value can be derived from the date of birth attribute. We can also have null attributes. Suppose, with a customer we have this attribute of the phone number, but for a particular customer, we do not have the phone number value with us.

Then it can be considered to be a null attribute. Because we are unable to supply a value for this particular attribute for a customer. So, we can define an entity and assign different types of attributes to that entity.

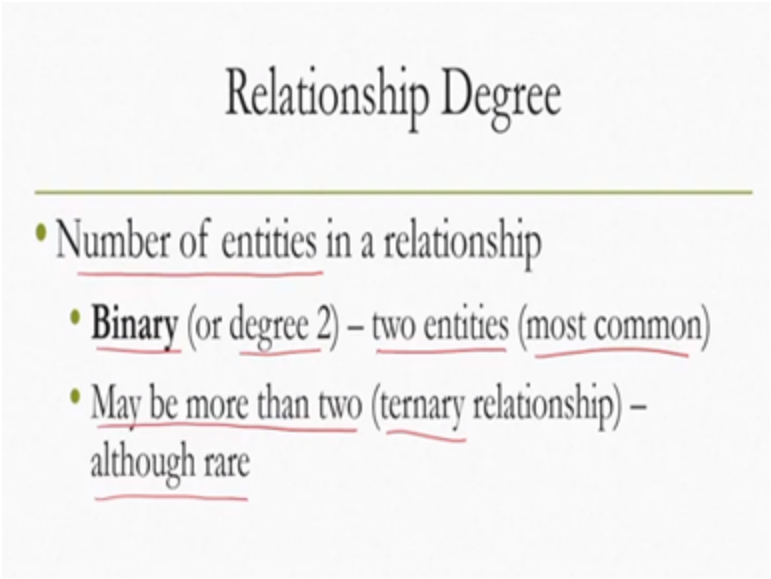
(Refer Slide Time: 15:18)



The third crucial component is the relationship between entities. A relationship essentially defines an association between entities. For example, suppose Sam is an entity. E-100 is an entity. So, Sam is a customer entity E-100 is an account entity. So, these 2 are 2 entities. Now, they are associated with each other in a relationship called depositors. So, this depositor is a relationship.

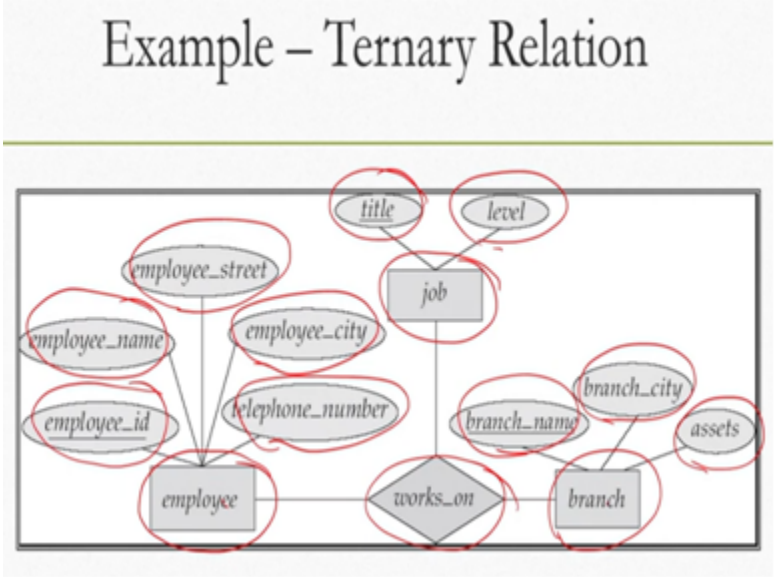
So, Sam is a depositor holding an E-100 account. So, we can represent it graphically in this way. So, the customer entity is there, the account entity is there and there is this relationship between them, shown with this particular symbol. Now here, Sam is actually an instantiation of the customer entity. E-100 are an instantiation of the account entity. But in the diagram, we are representing generic forms of entities and relationships. So, we have a customer; we have an account and we have a relationship with them as depositors.

(Refer Slide Time: 16:47)



Now there is this concept of degree of relationship. This is essentially the number of entities that are associated with a particular relationship. This number defines the degree of the relationship. We have a binary relationship or degree 2, which means 2 entities are associated with that particular relationship. This is the most common form of relationship. But we can also have more than 2 entities to share a relationship, such as a ternary relationship. But this is generally rare.

(Refer Slide Time: 17:25)

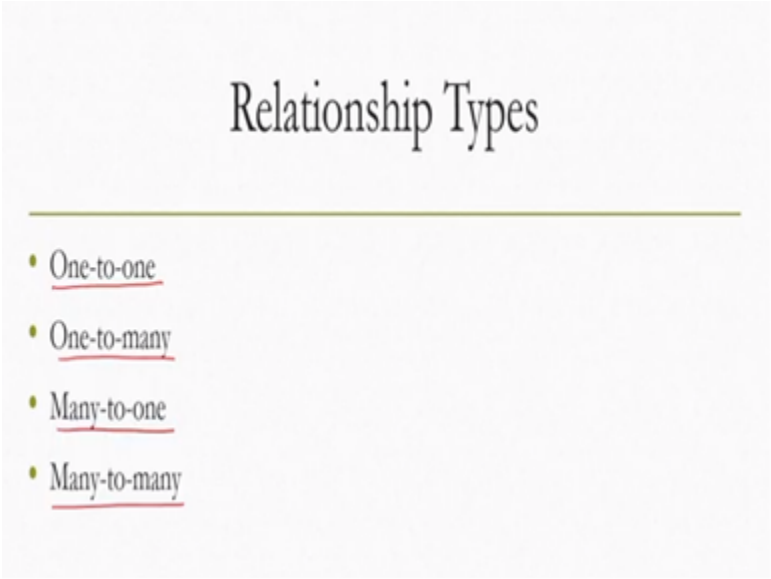


Let us see one example of a ternary relationship. So, we have an employee entity, a branch entity, and a job entity. Now, the employee entity has some attributes like employee id, name, street address, city address, and telephone number. These are some of the attributes defined for the

particular entity. The branch entity has attributes like the name of the branch, the city in which the branch is located, and the total assets in the branch.

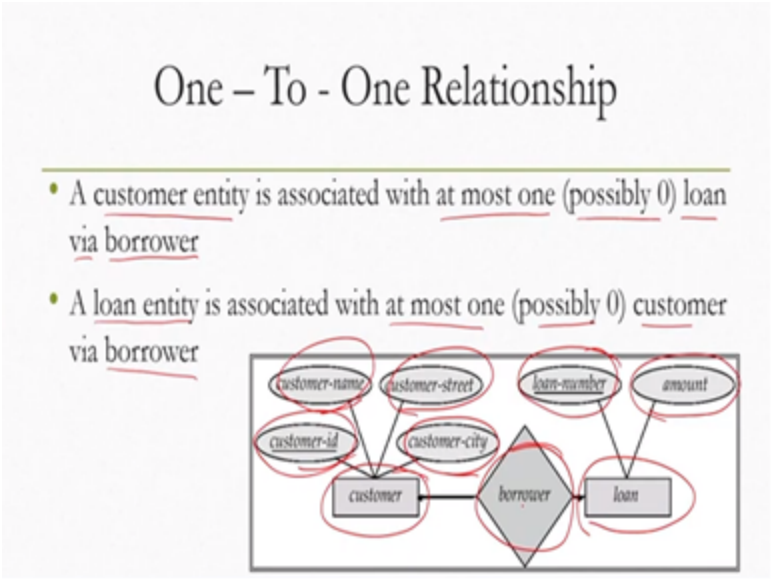
The job entities have attributes like the title of the job and the level of the job. Now, between these 3 entities, we can define a relationship works on. So, the employee works in the branch and the employee has this particular job. So, the employee's job and branch are related or associated through this working relationship. Here, 3 entities are associated with the relationship. So, we can say that this relationship has degree 3 or it is a ternary relationship. But generally, such types of relationships are rare. Relationships of degree 2 are more common than those are rare relationships.

(Refer Slide Time: 19:06)



Now, relationships can be one to one, one to many, many to one, and many to many. Let us see with examples what these types mean.

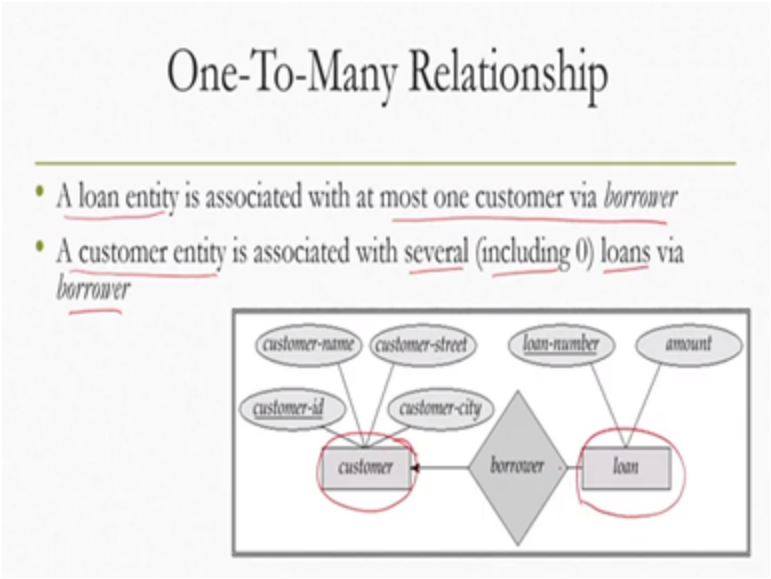
(Refer Slide Time: 19:22)



So, first is one-to-one. An example is a customer entity that is associated with at most one. It can be zero also loan entity via the borrower relationship. See if we can form such a relationship then that is one to one relationship. A loan entity is associated with at most one can be zero also, the customer entity via the borrower. So, it is graphically shown here. So, we have a customer entity, loan entity, and borrower relationship defined between them.

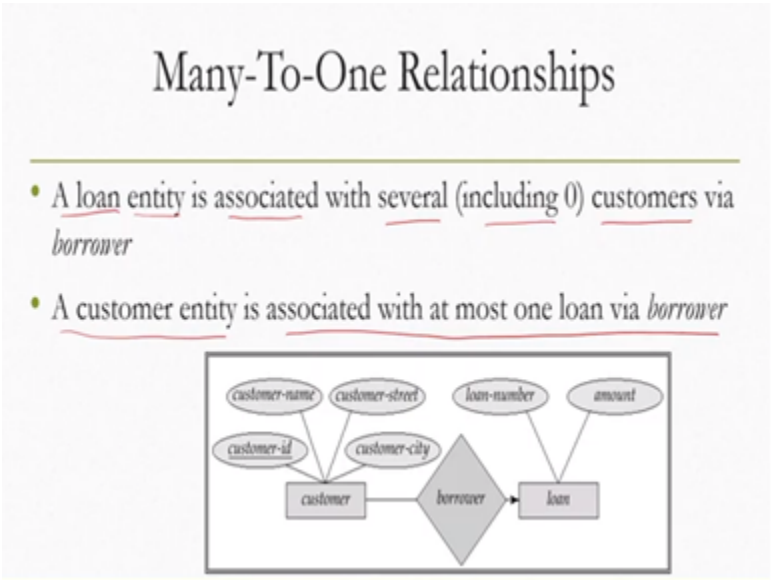
Now, the customer entity has attributes like id, name, street address, and city. A loan entity has attributes like loan number and amount. So, when we are defining this relationship as a maximum one, that is a customer can avail of a maximum of one loan or a loan can be associated with a maximum of one customer. Then this particular relationship is one-to-one. So, we can say that in this case, the borrower relationship is a one-to-one relationship.

(Refer Slide Time: 20:41)



Let us see one example of a one-to-many relationship. We will use the same set of customer loans and borrowers. So, if we now define a loan entity as associated with at most one customer via the borrower. But a customer entity is associated with several including zero loans via borrower, then that is one to many relationships. So, we can have one customer who can have many loans, if we redefine this relationship in this way then we can say that borrower is one to many.

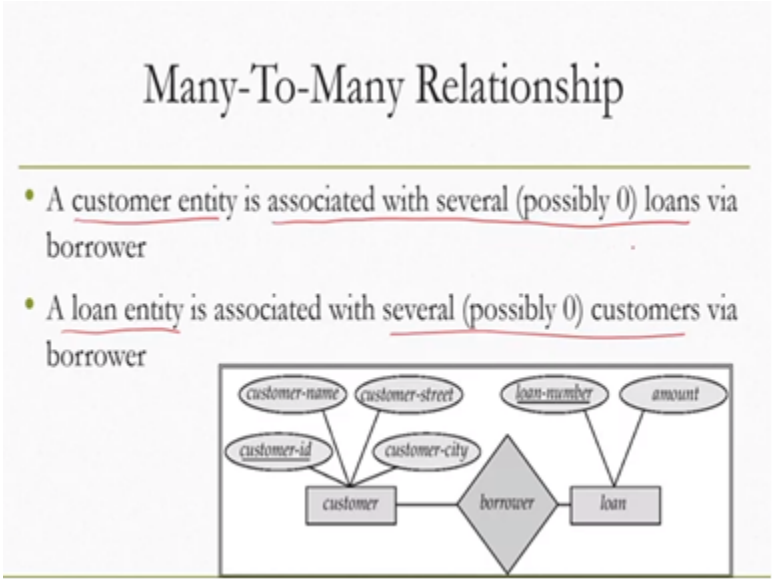
(Refer Slide Time: 20:24)



Then we can have many-to-one relationships. Now define a loan entity to be associated with several including zero customers via borrower, then that is many to one. So here, we should keep

in mind that these are hypothetical examples. So, it need not be practical. But still just to give you some idea of what these relations mean. We are redefining the settings a customer entity is associated with at most one loan by a borrower. However, a loan can be associated with several customers, in such a case it is many to one relationship.

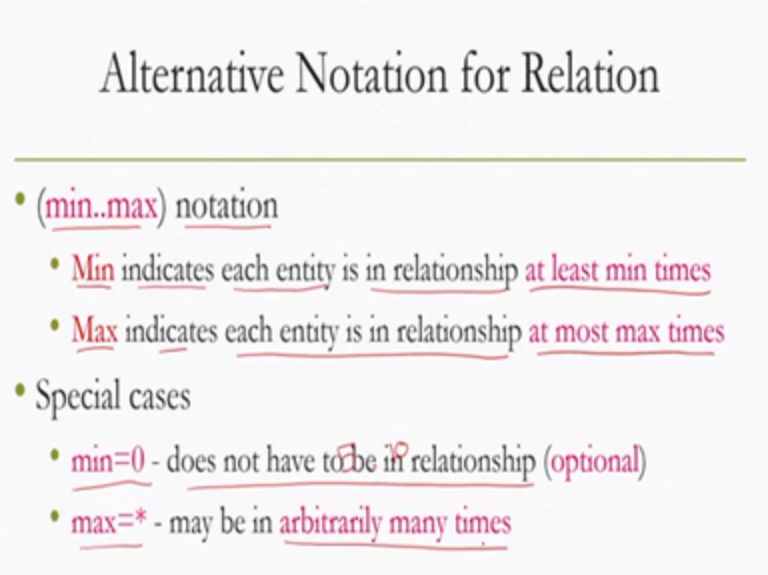
(Refer Slide Time: 22:08)



Finally many too many, a customer entity is associated with several loans. As well as, loan entities associated with several customers. So earlier, what we are seeing in the case of one-to-one, customer entities associated with one loan entity and vice versa. In one to many, a customer entity is associated with many loans, but one loan is associated with one customer only. Many to one customer are associated with one loan.

But one loan is associated with many customers and then finally we have many too many, where several customers can be associated with several loans. One customer can be associated with several loans and a loan entity can be associated with several customers. So, both are possible in that case we can say that the borrower relationship is many to many.

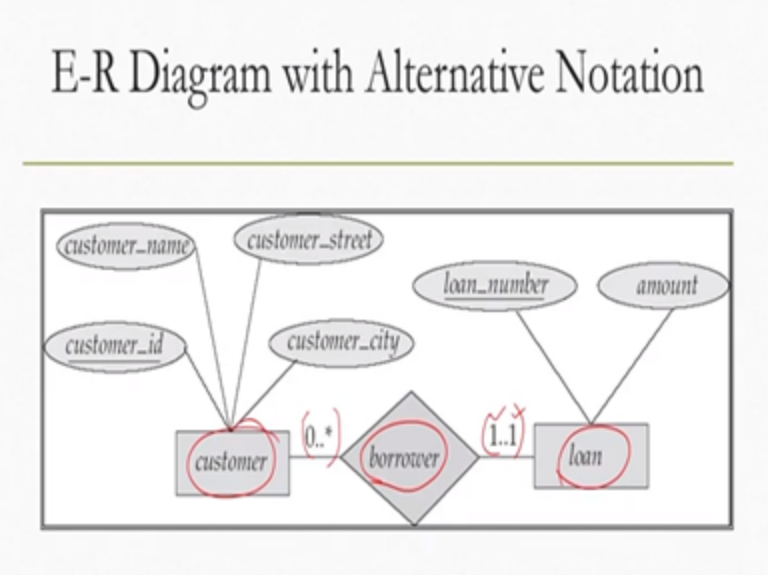
(Refer Slide Time: 23:11)



Now, we can use some notations to indicate the type of relationship. So, we can use min-max notation, where min indicates each entity is in a relationship at least min times. The max value indicates each entity is in the relationship at least max times. So, this is one notation to indicate the type of relationship where we can have one value for mean say 5, then dots, then the value for max say 10.

So, if we use this type of notation then the mean value indicates that each entity is in the relationship at least mean times and the max value indicates that each entity is in the relationship at most max times. There can be special cases for this notation when the mean can be set to zero. So, this indicates that there need not be any relationship between the 2 entities and max is represented with a star. That indicates that there can be arbitrarily many instances of the relationship.

(Refer Slide Time: 24:43)

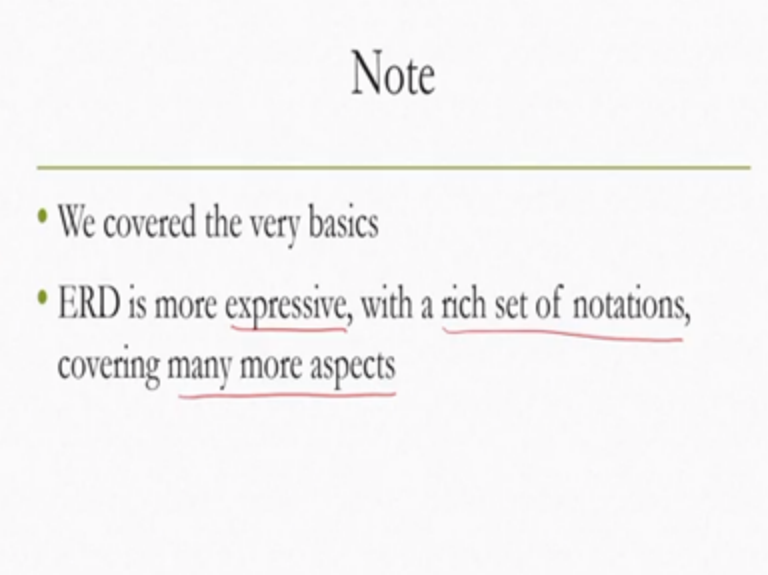


Let us see an example. Suppose, we are defining the relationship using the particular notation in the example setting that is customer entity, loan entity, and borrower relationship. Now, the lines between the entities and relationships we are now labeling with the min-max notation so as shown here. So, there are 2 notations on the 2 lines between customer and borrower.

The association is indicated by the min-max notation 0 and star, where the mean is 0 and the max is the star. That means there need not be any relationship there need not be any customer, borrower relationship and there can be arbitrarily many customer-borrower relationships at the most. Between loan and borrower, it is 1, 1. So, mean is 1, max is 1, using the min-max notation. That means a loan can be borrowed at least once or and a loan can be borrowed at most once.

So in both cases, the value is the same. Now, that is up to us, how do we define the relationship? If we want to have more flexibility we can have separate values for min and max so this example illustrates the idea of min max notation.

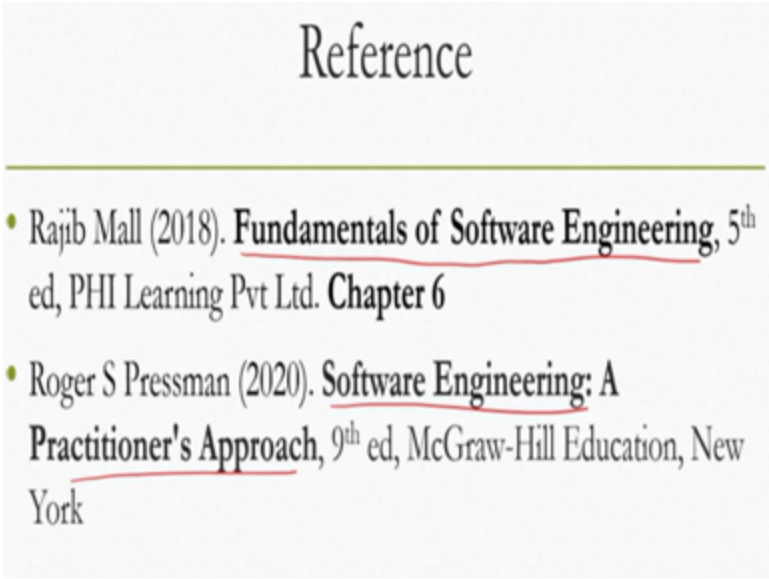
(Refer Slide Time: 26:25)



So, with that, I would like to conclude this topic on ERD. So, one thing to be noted here is that ERD is a very expressive language and there are many more notations and many more conventions followed to represent complex data organization data stores. Here, we covered ERD in a very basic way at a very basic level, but you should always keep in mind that ERD is more expressive, having a very rich set of notations, covering many more aspects of the representation of data.

Since a full-length discussion on ERD will be out of scope for this course, we will refrain from doing so, however, if you are more interested then you can refer to the references.

(Refer Slide Time: 27:28)



You can find the material in these books fundamentals of software engineering and software engineering practitioner's approach so you may refer to these books for more details on both the topics DFD as well as ERD and how they are used to represent system design. So with that we have come to the end of the lecture. In this lecture we have covered how we can use DFD and ER diagrams to express a system design that we have arrived at following a function-oriented approach.

So, we learned about several notations for DFD and we have seen examples to better understand the DFD. We have also learned about several notations and major components of ER diagrams and we have gone through several examples to understand the basic concepts of ER diagrams in more detail. I hope you have enjoyed the learning and you understood the concepts that are covered in these lectures. I am looking forward to meeting you all in the next lecture. Thank you and good bye.