In this module, we will talk about data modeling. You will recall that when we introduced software engineering and software development methodology, we were talking about requirement analysis where we mentioned that we perform both the data modeling and the process modeling, as we go through the requirement analysis, as we understand the application environment of the user. Let us start by asking again, why we model? We build models of complex systems, because it is difficult to understand any such system in its completeness at one shot. We have to therefore build a model and try to understand it in terms of its components and how those components relate with each other.

One of the important reasons for modeling of complex systems is to improve our understanding of such systems, because we cannot understand them entirely in one single instance. We need to develop a common understanding of the problem, so that we can proceed towards the solution. This common understanding is between all the people involved and also the users for whom we are trying to propose the solution. We cannot afford a trial and error approach in fact a model will clearly establish that we are proceeding along with correct direction and that our understanding of the users environment is correct and this will be reflected in the model.

(Refer Slide Time: 2:58)
This will remove the trial and error kind of approach and it will also reduce the risk in the overall development. A model is also extremely useful to communicate the required structure and behavior of our system. We try to capture that in the model and then put it in the form which can be understood and which can be verified by others. These are the reasons why we model. Let us see how we model. We choose an appropriate modeling concept or an appropriate modeling paradigm. This should be such that our solution can be properly expressed. This choice of the right model is extremely important and it has considerable influence on shipping the solution we proposed for the problem. So we choose a model for the kind of purpose that we have at hand.

These may be modeling of data or these may be modeling of a processing defined in a given application. No single model is sufficient; in fact this is an important point that in most analysis phases, we try to build different types of models which represent the different perspectives of the same environment or the same application. It is important to approach a complex system from different points of view which might be represented by using different modeling techniques. A single module may not be sufficient in fact we have been talking about two independent models already. One is for modeling of data and one is for modeling of processing. Even in the object oriented model that we had mentioned earlier different perspectives are taken.

One could be defining the object model which is a static kind of a model which reflects the different objects which are present in the user’s environment and then there is a dynamic model which defines the interactions among these objects. So we do take different perspectives and we try to use an appropriate modeling concept or appropriate modeling paradigm to represent these perspectives.

(Refer Slide Time: 5:12)
The best models are connected to reality. In fact the purpose of the model is to abstract the important aspects of the reality and represent them very clearly. Naturally they must meet the requirements as we want to analyze for a given real world situation. These are the different issues that we must keep in mind when we define our modeling exercise. What model should we choose? In this particular module we are talking about data modeling. We will define the notion of data modeling. We are going to build these models in terms of important concepts of entity and relationship. In fact the model that we are going to discuss in detail is the entity relationship model or ER model in short. We will look at the diagramming concepts which are available in this modeling technique. We will talk about other related concepts of keys, weak entities, and then we will also talk about extensions to the ER model. These are the different topics we will cover in this particular module.

(Refer Slide Time: 6:15)

Let’s begin by seeing the purpose. The purpose of the data model is to represent the operational data in the real world. These operational data describe the various events entities and activities which take place in the business environment for which we are proposing the solution. So remember that we are trying to represent the operational data and there may be a lot of this data which describe different entities, different activities which happen in the business environment and different types of events which takes place.

All these data need to be captured in solving that application problem. The objective of data model would be to represent this operational data. The model may be described at various levels. The model may be at a logical level or a physical level. Physical level will naturally address not only what data we have, but how that data is stored and retrieved and updated and things like that. This would be the way in which the data would actually be handed.
Very often, we first try to understand the data at the logical level; the model may be also in external level or conceptual level or internal level. What we really mean here is that when we say model is at external level, it might define the model as seen by a particular user who is the user of the application. Naturally his view of the data may be a subset of the overall content in the application whereas the conceptual model represents the data in totality at a level which represents the important concepts in the application. Internal data model actually is more of the physical representation of the data and data may be stored in terms of files and so on.

Internal model would generally take into account efficient processing of the data whereas the conceptual model purely concentrates on the concepts and how those concepts are interrelated without being concern about the efficiency issues. We may model the data at various levels. In this particular module we are going to focus on the conceptual data model, in fact this is the beginning where we try to understand a given application domain in terms of all the important concepts and try to understand those and try to interrelate them.

(Refer Slide Time: 9:09)

Subsequently when we proceed in the design, we will try to consider the physical representation of this conceptual model, the internal representation, how to make user processing more efficient, and also how to restrict the views of different users so that they access those data which is relevant to them. These external, internal or physical aspects of the model are generally worked out when we proceed into the design and implementation. Our first objective in the development of the solution is to build a conceptual data model. A good data model should meet some important requirements and these are quite straight forward. A good model should naturally be easy to understand because we will expect users to validate the kind of model we are trying to build to represent his application domain. Users can understand it if it is a simple model, if it has only a few concepts.
And it can be specified in top down fashion, so that you do not have to give all the details together but you give details in step by step fashion. In fact the top down specification is very important in any modeling because we cannot comprehend a complex system at one point. We have to see it at different levels of details and that is permitted by top down specification.

These are the characteristics of a good model. Let us also define what we mean by a model what does a model consists of? A model consists of a few important concepts. It will also give us some form by which those concepts can be represented, we call these constructs. Construct is a representation for a concept, but model primarily offers a set of concepts and it also offers a few operations on those concepts. This is what a model is made up of.

(Refer Slide Time: 11:40)

Another requirement for a model is that it should capture the meaning of the data. Naturally we are trying to understand the data which is present in that real world, and then we are going to put it in the form of a suitable conceptual model. This model should capture the real world meaning of the data. Otherwise it will be difficult to interpret the data. We have a notion of what we call data semantics. Data semantics is concerned with the meaning of the data how we convey the meaning of data. When you prepare a data model not only it should give us some techniques for representing the data but it should also give us a way of conveying the meaning of the data.

Both of these the concepts and the meanings what we generally call as syntax and semantics would be important aspects of a data model. How do we capture semantics how do we convey the meaning of the data? There may be different ways of doing that and we will see that in the ER model the different methods are provided by which we can convey the meaning of the data which we are modeling.
Some of the ways that you will encounter are given here. For example the proper naming of the data is important. The name of the data itself conveys a lot of meaning. If I tell you that an item has a price, I use the word ‘price’ for obviously indicating at what cost you can purchase. The word price has a specific meaning and by choosing the right word for the right aspect of data, I am conveying that meaning.

In fact the most important aspect of conveying the meaning of data is proper naming. This is what we must keep in mind when we are building a data model. The different types of data that we will define or the different representations that we will choose are all named very meaningfully and also named in the context of the business environment which we are modeling. Anybody who has the familiarity with that business environment will quickly understand what data we are talking about and what role it has in the real world.

Then of course there are many other constraints that we can define on the data basically constraints capture the meaning, what are the permitted values for the data, is the data unique, do they have some interdependencies and many other restrictions that we may impose. All these restrictions and other constraints and proper naming are very important part of conveying the meaning of data.

Any model will have to provide such facilities by which the meaning is captured. As we go along we will see that the models such as ER model provide facilities for these different types of expressions through which the semantics can be put down as the part of the model. Let us talk about the entity relationship model in detail. ER model has a few concepts. It is simple and easy to use. It permits to some extent a systematic top down approach, so that various details can be controlled and you need not put down everything at one point, you can convey it in a step by step fashion. And because of these reasons it is an excellent tool for communication with the users.
As we collect our requirements and we prepare the model, we can verify our model with the user. So that we know our conceptual design is correct and our understanding of data is correct. For all these reasons the ER model is known as a conceptual data model. It does not describe how the data will be stored or represented. It only talks about what kind of data concepts are there and how they are interrelated. It provides a powerful diagramming notation through which the model can be represented. This diagramming notation and the simple concepts of entity and relationship are very intuitive concepts. This is what makes the ER model very simple tool for communication with the users.

We will try to understand the different concepts that are present in the model. Let us begin by looking at the first concept the concept of an entity.

- Entity simply means an object that exists; this is an object which exists in the real world.
- The object must be distinguishable from other object.
- And finally the object may be something physical it may be a concrete entity or it may be an abstract entity. It does not have to have a physical dimension.

Here are a few examples. For example, this course on software engineering can be treated as an entity. It is an abstract entity, it exists as a course, and it is distinguishable from other courses. It satisfies all the three characteristics of an entity. Therefore we say that this course on software engineering is an object. We can think of it as an entity. Another example is Ganesh. Ganesh is a student. Ganesh is an object, it exists in the environment, let us say the university we are trying to model. We can distinguish it from other students because ganesh is only one such object and it is also physical in this case. Course may be an abstract entity, but Ganesh is obviously a physical or concrete entity. An entity is something which exists in the real world in the application environment of the real world which we are modeling.
The entity is on the other hand is a way of putting together or clubbing together a set of similar entities. These similar entities that we group together, they form a set. These entity sets need not be disjoined. For example we can have a set of suppliers or we can have a set of customers. There can be a supplier who is also a customer. We are talking about various suppliers forming one set, various customers forming one set and these two sets may be disjoint or may also be overlapping. But every entity in those sets will be either the supplier and it will play the role of supplier or it would play the role of a customer. Entities which are of similar type will form a set out of them or a collection out of them because we are always looking for similar entities and we are trying to create a model.

There maybe thousands of students in the university. We are not going to model each one of them because they are all similar from application point of view, they do similar things, and they execute similar transactions, their data has to be stored. Naturally we are more concerned not with individuals, but collection of similar entities, what we call an entity set. Another example set of books in a library. There may be thousands of books, but we treat book as a set. And each book in the library is an entity. Now we know the difference. The book as an entity set is a collection of all books and these are available for students to read and borrow. But when the student goes to library, he issues a particular book that means he actually issues a particular entity from the collection of all books. We should keep these important concepts in mind that entity is a specific entity or specific book or a specific student or a specific course. Whereas entity set is a set of similar entities like all books or all students or all customers.

Entity set is also called entity type or entity class. These are alternative terms for the same concept called the entity set. And entity therefore is an occurrence or an instance of some entity type Student is entity set or it is an entity type whereas Ganesh is an occurrence of student or Ganesh is an instance of student.
We must clearly understand, instance is a real world object like Ganesh, whereas student is a concept, it is a conceptual entity it represents all students together in the application domain.

(Refer Slide Time: 21:29)

We often use the word entity to mean the entity set. This is because when we are modeling as I said earlier, we are not distinguishing different entities because their behavior is similar in a given entity set. We talk of student in general rather than talking about Ganesh in particular. Therefore we use the word entity just to mean actually the entity set. Our modeling will be concerned more with entity sets rather than with entities. Entity sets are named using singular common nouns. This is a very important point. Proper naming of entity sets is very important to convey the meaning.
Here are some of the examples: book, student, and course. These could be useful entities in a university environment. The next concept is the concept of attribute. An entity has a set of attributes. Attribute defines some property of interest. For example, the entity ‘book’ has an attribute ‘price’. We know that books are having a well defined price at which they can be bought or sold. Therefore the book has price as an attribute. Every attribute is given a suitable name. Again the name should be meaningful. ‘Price’ is the name of the attribute associated with the book entity.

This is how we put the concepts together and try to give a very precise meaning. We say that price is an attribute, it belongs to the entity set called book. All books now will have some price value. Attributes have value for each entity and these values would be naturally different from entity to entity. A particular book may have a price of two hundred rupees and another book may have a price of three hundred rupees. Every attribute has a value and every entity which has that attribute will be associated with the value. Value may of course change with time. For example, salary may be an attribute of an employee, naturally the salary changes.

Same set of attributes are defined for entities in an entity set and that is how we say an entity is a set of similar entities. They are similar because same attributes are applicable to all of them. If I have thousand students in a university, all of them share same attributes because they are of interest to me from same point of view.
Naturally in a given entity set, all of the entities will have same type of attributes. Let us take an example of entity called book and this entity has the attributes which are listed here. Every book has a title, has an accession number, has a publisher, has a price, and has an ISBN: which is a unique book number, has an author and also year of publication.

We have listed these attributes because they may be of interest to us in the context of application. Of course the book has many other attributes, for example number of pages, the language in which the book is written and so on.
But we are not listed them as attributes here, possibly because they are of not much interest to us in the application that we are developing, but they may be in some other application. When we talk of an entity we must always keep in mind the context of the application we are developing and in that context we should list the attributes which are of interest to us, for which the data will be obtained and will be stored.

What we are saying here is that in the application we are building, we need the following data about the book. Therefore we have defined attributes for all the books in our library. So these attributes will apply to all the books. Naturally as we said before, every book in the library would have a value for each of these attributes. An attribute may be multi valued, what it means is that an attribute may have more than one value for a given entity. And a good example of that is the author attribute. Some books may have more than one author. What it means is that when under author for a given book we may list more than one name. All of these names represent values for the author attribute. Therefore we say that author attribute is a multi valued attribute.

An attribute which uniquely identifies an entity is called a candidate key for that entity set. Remember that this is important and this has to be present because every entity must be distinguishable from another entity. So every book must be distinguishable from another book. Therefore book must have an attribute which uniquely identifies every book in the library. Such attributes are called candidate key or key in short. Key attribute is an attribute which uniquely identifies an entity.

(Refer Slide Time: 28:32)

Some attributes can be composite attributes, which means they contain multiple values of different type. For example the date may contain month, year, or the day and address may contain city, pin code and so on. These are called composite attributes because they can be decomposed further. They contain other parts and these parts can also be named if necessary.
Or we may not be interested in decomposing them further, but they do have a composite value. Then the next concept is that of a domain. Domain allows us to define a set of permitted values for an attribute. What kind of values can attribute take. These may be defined by listing the values or by defining the type.

For example, the date of joining for a given employee can be of type date. Marks obtained by a student can be of type integer number. Whereas an attribute like grid may be listed in terms of values it can take. Grades may take values A B C D E F, may be these are the only six values permitted in that case and we list those values. Domain is again an important concept in defining our application. We list the set of permitted values for a given attribute. This is again part of defining a constraint what is called an integrity constraint, validation constraint, this is important for validating the data values in the database.

(Refer Slide Time: 30:04)

If we have a domain for an attribute such as say the ‘marks’ attribute, ‘marks’ attribute must have an integer value. Any other value which is not integer cannot be accepted. So domain when we define like this, is actually defining the integrity constraint or validation requirement for the data. Let us come to the concept of primary keys. We have just now defined the concept of candidate key or key in general. Basically this is a related concept. It is just a slight extension of that concept, purpose is again the same. We want to distinguish the occurrences of entities in a given entity set.

We may have an entity set called student. There are four thousand students in this institute, how do I distinguish one student from another student? This would be done using the concept of a key. Distinction therefore, is always made using some value of attribute or attributes. A set of one or more attributes can uniquely identify the entity. In that case, this set of one or more attribute is called its candidate key.
Here are some examples. Roll number for a student. We can call a roll number as a candidate key attribute for the student entity. Similarly accession number for a book in a library is its candidate key. Note that title of the book is not a candidate key because in a library we may have many copies of the book with the same title. Therefore the libraries give an accession number to every book and every book in the library has a different accession number. Similarly, the roll number of a student, every student has a unique roll number using which we can clearly identify the student we are talking about.

No subset of candidate key is a key itself. In case a key contain more than one attribute then all those attributes must be required and we cannot do away with a subset of that. We cannot drop any attribute from that composite key. A candidate key may be a single attribute or a multiple attribute. But when it is containing more than one attribute it should not have any redundant attribute. All of them must be required in order to identify an entity. An entity may have multiple candidate keys which mean that I may identify people or books or whatever entity we are talking about in more than one way. Consider the example of say employees of a small organization. In that we may have employee numbers to identify them. But names may also be adequate, names may also be unique. In that case we say that the employee has two candidate keys. One is the employee number and the other is the employee name.

How do we define primary key? Basically primary key is a candidate key chosen by the designer as the principal means of identifying an entity. You say although both employee number and employee number are candidate keys, I will prefer employee number as a key and therefore employee number will be designated as a primary key. In fact all of them can also be loosely called primary key. But we are just naming a candidate key as a primary key as a main way or the primary way of identifying the entities. So very often we call all candidate keys also as a primary key. But the way we had defined it now, it should be unique and there should be only one primary key for a given entity set.
Let us look at an example which is taken from a university or a college situation. We will identify few entities and their attributes. Many of these are obvious and you will be able to understand them quite easily. Here is a student entity with the attributes; roll number, name, hostel number and date of birth. Here is a course entity which has attributes; course number, name and credits for the course. Here is the teacher entity which has employee number, name, rank, room number and telephone as its attributes. Finally we have the department as an entity and the department has only two attributes name and telephone. As a small exercise you can try to identify primary keys for these entities.

(Refer Slide Time: 35:48)

Let us note some points about this example. We will further refine as we proceed and we will also take many such simple examples from the university environment, because it is all familiar to all of us. We can always prefer to take examples from university environment, but we will refine these four entities that we just now introduced. There are very important points to be noted, the first point is that our focus could have indicated more entities. We must remember that in the college environment there can be additional entities. Hostel could be an entity; Semester could be an abstract entity. Or instead of naming teacher as entity, we could have named teacher as an attribute of ‘course’. So that we say that course have a teacher. How do we decide in a given modeling situation whether something is an entity or not? This would depend on the focus of our design or the focus of our application. How do we perceive the reality? Hostel in the real world is obviously an entity, it’s a physical entity. It has rooms, it has address, and it has a warden and so on. It has many interesting attributes. But in the application we are developing, the purpose of the hostel may be only to act as an attribute for a student to tell us in which room number or in which hostel number the student is staying. In that case we are not interested in hostel directly, but we are interested in the hostel through student entity.
So we will have to decide what is the focus of my application? Do I need to know who is the warden of the hostel? Do i need to know some other details of the hostel? In that case, hostel will be an entity. But if the role of the hostel is only to tell me where a student stays then, I can treat it as an attribute of student. This important point must be kept in mind that what is the perception we have, what is the focus of our application development and based on that you will identify the entities and attributes. We cannot loose this perspective throughout the modeling exercise.

(Refer Slide Time: 38:35)

Here is an exercise for you. Given a hospital environment, most of us are quite familiar with this environment. You identify a few entities and attributes. Let us go to the next concept, the concept of relationship. Relationship represents some associations among entities. In the real world entities exist and they get interact with each other, they get associated with each other. We want to capture that association through the concept of a relationship.

Let us take a few examples. A particular book can be prescribed as a text for a particular course. Remember book is an entity, similarly course is an entity. But now we are relating a book and a course. We are saying the book called the database systems by author C.J Date is a textbook for the course identified as CS644. We are talking about a book here called database systems. We are talking about a course here called CS644. These two are related through the concept of a textbook for a course. Textbook is actually defining a relationship between book on one hand and the course on the other hand.
Similarly another example would be student Ganesh who has enrolled for the course CS644. Ganesh is an entity and course is an entity. The fact that this student, has enrolled for this course is captured through the concept of a relationship. As you can see here that the purpose of relationship is quite distinct from the purpose of the entity. Entity identifies independent objects in the real world. These objects interact with each other. A student enrolls for a course: Ganesh enrolls for course CS644. This will be captured through the concept of relationship. Again we do not talk of specific relationship instances. We want to capture relationships of similar type and this we will do with the concept of relationship set.

This is same as before, when we talked about entity and entity set; we are talking about relationship and relationship set. Whenever we talk of relationship, we basically imply the relationship set, because in modeling we are interested in set of similar relationships and not specific relationships. Therefore these two words relationship and relationship set are often used interchangeably. Relationships exist between entity sets. A binary relationship exists between two entity sets. A ternary relationship exist among three entity sets and “n”nary relationship in general exists among n entity sets. When we define relationships we will have to identify which entities are involved in that relationship.
A Binary relationship set as we said just now exists between two entities. Here is an example: A relationship which we are naming it as STUDY exists between student entity and course entity. Of course we may view it differently depending on the context or the application and we may find that the relationship STUDY actually is a ternary relationship among student course and teacher. What is the difference? The two statements we just made, are they same or is there any important difference between them? Even in the first case the teacher entity may be there in domain that we are modeling.

Assume that the domain that we are modeling contains student, courses and teachers. Is this STUDY relationship between student and course as a binary relationship or it is a ternary relationship between student, course and teacher. Is there a difference or it does not matter? In fact there should be a difference otherwise we have a modeling situation where people may arbitrarily model it as a binary or as ternary relationship. The advantage of ER model is that it precisely defines the difference between these two situations and which one is correct will depend on the application environment. It will depend on the university you are modeling.

Let us see the difference. If you consider the first one, it means that as far as the student and the course is concerned, I do not want to find out which teacher is teaching that course. Given that a student is learning a course I can always find the teacher who is teaching that course independently. Whereas in the second case, we are saying that it is not enough to say that student Ganesh is studying course CS101. You have to also say under which teacher he is studying that course. This is important from what is the basic indivisible fact. What it means is that in the second case, we may have a university where the same course is taught by many teachers concurrently and it is not enough for Ganesh to say that i am studying CS101. He has to say that I am studying course CS101 under teacher Deepak or whatever. This is the important difference.
In the first case, it is not necessary to mention the teacher we can find it out given the course. In the second case it is not adequate. Given the course I cannot find out which teacher the student Ganesh is studying with. This is the strength of the ER model that it is able to convey such finer aspects of the semantics and it can distinguish between these two situations.

(Refer Slide Time: 46:18)

Depending on the situation we are modeling, we have to define binary relationships or ternary relationships as appropriate, so that the association among entities is clearly captured. Another important thing is that a relationship itself may have attributes. The STUDY relationship has attributes, the grade and the semester. The grade will tell us what grade the student has received in that course. Suppose student Ganesh is studying course CS101 and he gets A grade. This grade A must be treated as a attribute of STUDY, it is not a attribute of student Ganesh because just saying that student Ganesh has got A grade is not enough, you have to say in which course he has got that grade. Therefore it is neither the property of student nor the property of course but it is the property associated with the relationship between the student and the course. This is an important point because grade cannot be defined as attribute of student nor it can be defined as an attribute of course but it must be defined as a attribute of STUDY relationship. Similarly the semester in which the student has registered in that course will be an attribute. We always define these attributes for the relationships as we see them in the application domain. Relationships are named using verbs or nouns. Here are a few examples ‘Study’ is a relationship as we have seen. Enroll or Order, a customer may place an order for a part. So ‘order’ will be a relationship because customer is an already existing and defined entity, part is an entity. Order relates customer and part. So these relationships should be named using either verbs or nouns and they should be chosen very carefully so that the meaning of the relationship is conveyed to the user.
Continuing the exercise which was mentioned earlier, you should now identify the different relationships which may exist in the hospital example for which you had identified entities earlier. This will clearly explain to you whether the concepts that we have discussed, you understand them properly or not. So you should try to work out this exercise and identify entities and relationship in a hospital example.

Let us next see how we can depict the relationship, how do we visualize it. It is very important concept and often mistakes are made in understanding and modeling a relationship. First we will try to see how we can show a relationship in some simple diagramming form. We will show entity set as a collection, we will show entity instances by some small circles within those collections and we will show the relationship by connecting the entities which are involved in the relationship. We will use a simple diagramming notion to understand the important concept of a relationship. Here is the example which shows the ‘STUDY’ relationship.
You will see an entity set here. This entity set is the student entity set and here are four students shown. These are the instances. This is the entity called ‘Ram’, an entity called Sita etc. This is a course entity set. We have listed a few courses here, like a course on database management system, a course on data structures. These are the instances of course, these are the instances of student and now we want to capture the STUDY relationship between these. These STUDY relationships are shown by small rectangles and by connecting the two entities which are related.

This small rectangle here indicates that the student ‘Ram’ is studying the course COBOL, he is also studying the course DBMS. In this pictorial representation we are showing the relationships as something which connects entity from student entity and entity from course. In fact we can think of relationship as navigation paths to allow us to go from one entity to another entity. I want to find out which courses Vinoth is studying. I start at Vinoth, follow this path and I know that the student Vinoth has registered for one course. Sita has registered for three courses and so on.

This is a very useful way of visualizing entities and relationships. Such diagrams are called instance diagrams. They show entity sets, they show entities as example entities like sita or ram or courses like data structures and it tries to visualize relationship among them. One important thing here is that one rectangle can only connect one student and one course. The same rectangle cannot connect the same student Ram with another course DBMS. That must be shown by another rectangle because this is a set and every element here is uniquely connecting one course and one student.
This is how we can represent entities and relationships in a simple diagramming fashion to convey our understanding of the entities and how they are interacting in the real world. What is the primary key for relationship? Just as entities have key attributes, can we define the concept of key for a relationship? This is defined in a very simple way. A primary key of a relationship is made up of primary key of the participating entities. A relationship does not have a key of its own directly, but it is made up of primary keys of the participating entities.

If you look at the study relationship, what is the key of study? The key of STUDY relationship consists of a composite attribute set. It consists of two attributes roll number and course number taken together. Remember that roll number is the key of student and course number is the key of course. These two attributes define the key for the relationship STUDY besides this key of course, it can have other attributes as we mentioned earlier like grade and semester. We will next consider the important concept of relationship cardinality which is identifying some constraint over the relationship. This constraint is captured by indicating how the relationship connects entities between the entities among which the relationship is defined. It characterizes the relationship further and it is given by indicating how many entities of one entity set, participate in a relationship.
We will take many examples of this. It is a very important concept. It is related to the relationship concept. Cardinality is a way of characterizing the relationship further and it is indicated in terms of number of entities which participate in a given relationship. Just to mention an example, if you see the study relationship, if you take one student, then how many courses he may be studying? He may be studying more than one course. We will say that as far as courses are concerned there may be many courses related with one student entity. This is how we try to indicate occurrences of different entities in a given relationship and this is captured through the concept of cardinality.