**Database Management System**
Dr. S. Srinath
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Lecture No. # 7

ER Model to Relational Mapping

Hello and welcome to the next lecture in the DBMS course. Until now we have seen two main kinds of data models for representing data or managing data in several different application context like I say whether it is insurance or banking or railway reservations or company databases or so on.

(Refer Slide Time: 00:01:18)
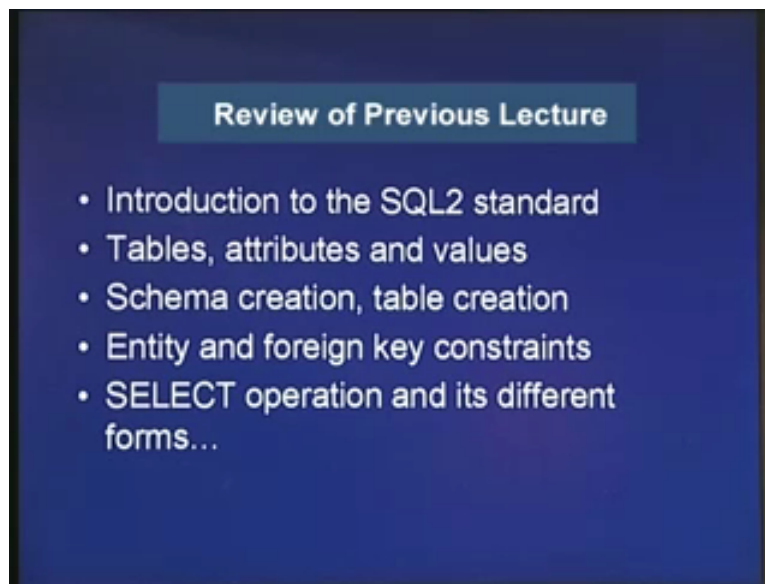
These two models where, the first was the entity relationship model or what is called as the ER model and the second was the relational data model. We also saw a typical database design process and placed these models into appropriate positions in the process. The entity relationship model or the ER model is essentially meant for human comprehension.

It basically is meant for creating a conceptual database or conceptual schema or what is termed as a logical schema of the database system and the relational data model is used for the physical schema or something that is implemented in the DBMS.

(Refer Slide Time: 00:01:46)
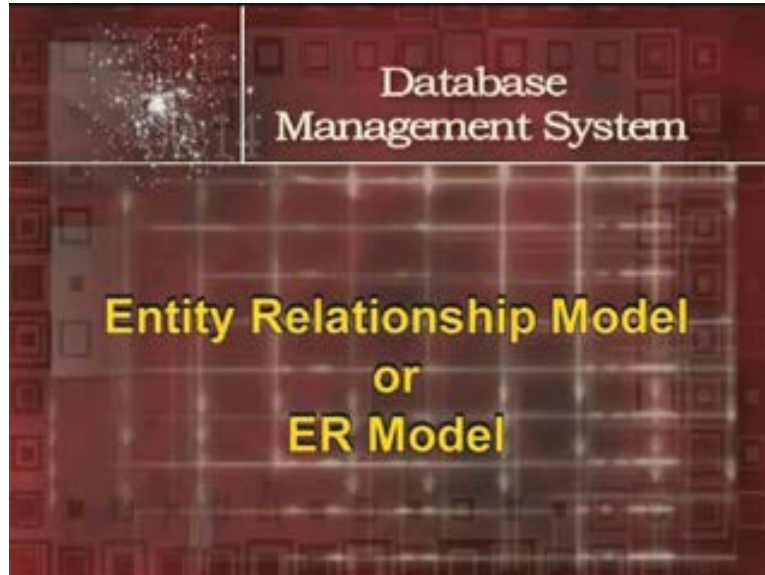
So both are DBMS independent models that is no matter which companies database that you are going to use, you can still use the same ER model for representing your data and even the same relational model for representing the schema that goes onto your DBMS. Now in this session we are going to address one important issue. Now we are going to ask a question, are these two different data models ER and relational model completely independent of each other or are they the same or is there some way I can map between the ER schema and the relation schema without having to break my head too much essentially. That means to say or to put it, to take it to its logical extreme, can I design some tools or can I design some kinds of software that takes an ER diagram of a given system and generates appropriate relational schemata for the system.

And we have also seen in the session on functional dependences, we have seen how we can optimize a given relational schema up to a certain extent using some kinds of automated techniques. We have seen how to take a relation to a BCNF or third normal form or the fourth normal form and so on. So suppose we want to build a tool to automate this process, we need to be able to first map between an ER database schema and a relation schema and then use techniques from functional dependencies to optimize this relational schema so that we can build a database application around it.

So what we are going to study today form the underpinnings of what are called as lifecycle tools or database lifecycle tools. There are several different lifecycle tools which provide support for the entire lifecycle of a database systems, starting from the conceptualization of the problem to the actual implementation of the application and maintenance of the database and so on.

(Refer Slide Time: 00:04:25)
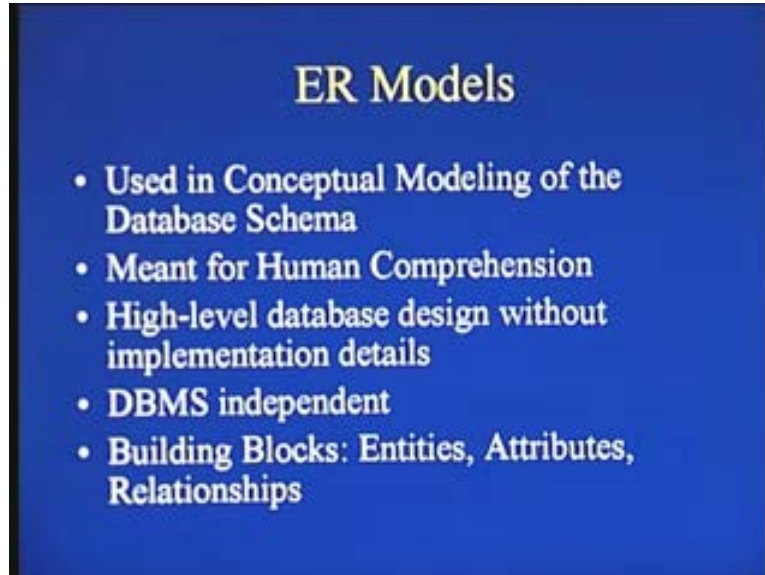
So before we begin let us briefly summarize what we have learned about ER models and relational models. The ER models as we already know is used for conceptual modeling of the database schema, conceptual modeling or to create the logical database schemata. This is meant for human comprehension, this is essentially used to show end users what you have understood about their problem domain. This is a high-level database design and there are no implementation details about that are included as part of the ER model. And of course the ER model is a DBMS independent and it is made of building blocks like entities, relationships and attributes which can be attributed to both entities and relationships.

In contrast a relational data model is the data model that is most popularly used for physical schema design. A physical schema is the schema that is actually implemented on the computer, therefore the relational data model is meant for or optimized towards machine conception. That is how do we efficiently store data in my database, how do I efficiently search for a given data element, how do I efficiently update a given data element so that it does not create anomalies, how do I efficiently delete data elements again without creating any anomalies and so on.

(Refer Slide Time: 00:05:10)
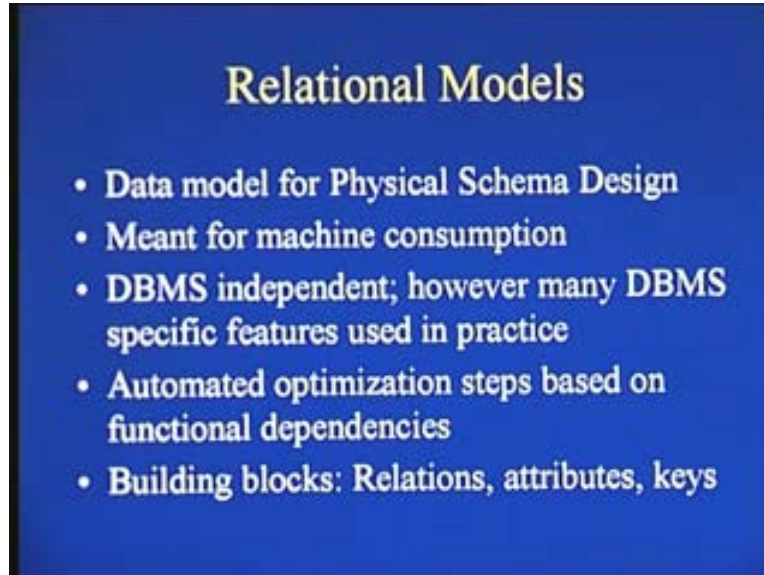
And of course the relational data model is also DBMS independent that is no matter what kind of database that you use, you can still use the same data model as far as the database that you are using is a relational database. You can use the same data model to represent your data on the DBMS. Of course reality is quite different from the concept of DBMS independent and some DBMS systems may include more features than traditionally what is supported by the relational model.

The relational model also supports some kinds of automated optimization techniques which we have seen in the session on functional dependencies where you can optimize a given relational schema, you can reason whether a given relational schema is optimal or not whether it's going to create redundant data in its DBMS or whether it's going to create some kind of anomalies during updation and deletion or so on. And, how you can systematically change the database schema without changing the correctness but increase in the overall efficiency in terms of retrieval and updates. And what are the building blocks of the relational model? We have relations which comprises of several different attributes and the notion of keys forms, a very crucial role or place a very crucial role in the relational database model.
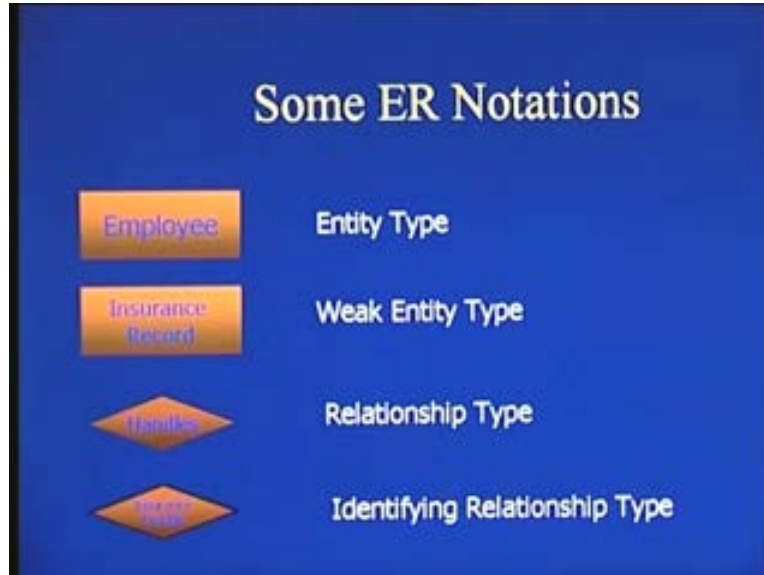
Let us come back to the ER model and look at some of the notations which we will require if we have to study translation into ER models. The entities are represented using rectangles and a strong entity type that is an entity type which has its own key attribute and which represents a physical or which represents some kind of a logical entity of the of the real life is represented by a rectangle with solid lines surrounded. For example the slide shows this entity type called employee which depicts all objects of type employee which are present in the current system.

On the other hand we also have what are called as weak entities. Weak entities are those which do not have an existence of their own or without being associated with a strong entity type. The slide shows the example of an insurance record. An insurance record doesn't mean anything unless it is associated with some person. In a company for example an employee. Therefore when we talk about insurance record we have to say, whose insurance record and so on. So that is the general idea and more specifically the insurance record entity type does not have any key attribute, it has to be associated with a strong entity type called employee which in turn has a key attribute.

Therefore such weak entity types are depicted using dashed lines or dotted lines for the rectangle. we then have the relationship type for example a relationship called handles, so employee handles project or something like that which is represented by a diamond and a normal relationship type is represented by a diamond using solid lines whereas, what are called as identifying relationship types. That is the relationship types that identify a weak entity or provide an identity for weak entities by associating them with strong entities, they are shown with double lines in the diamond.
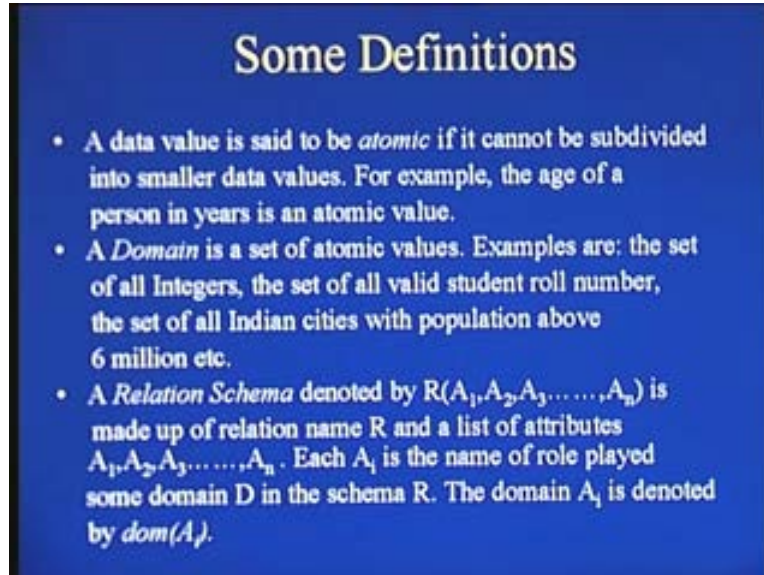
(Refer Slide Time: 00:09:36)
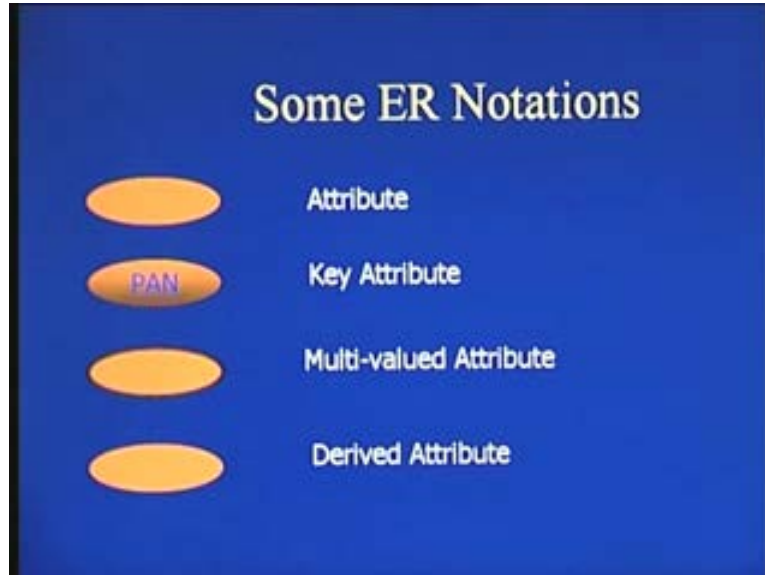
Entities and attributes are associated or entities and relationships are associated with attributes which are some values in a given domain. Attributes are depicted using ovals and normal attribute or a simple attribute is depicted by an oval with a solid line and key attributes in this example an attribute called pan or pan number which uniquely identifies each income tax payer is shown as a key attribute and it is shown underlined saying that this attribute is a key attribute for this entity type.

And then there are multi-valued attributes which can have several values for the same attribute. We took an example of the color of a peacock. Now the color of a peacock is actually given by several different colors and all of which in combination form the color of the bird. Such kinds of attributes are depicted using double lines as shown in the slide here. And then there are derived attributes that is attributes whose values can be derived from other attributes and these are shown using dotted lines. We took an example of the age of an employee that is if you know the date of birth of an employee and the current date, we can derive the age of an employee.

(Refer Slide Time: 00:10:58)

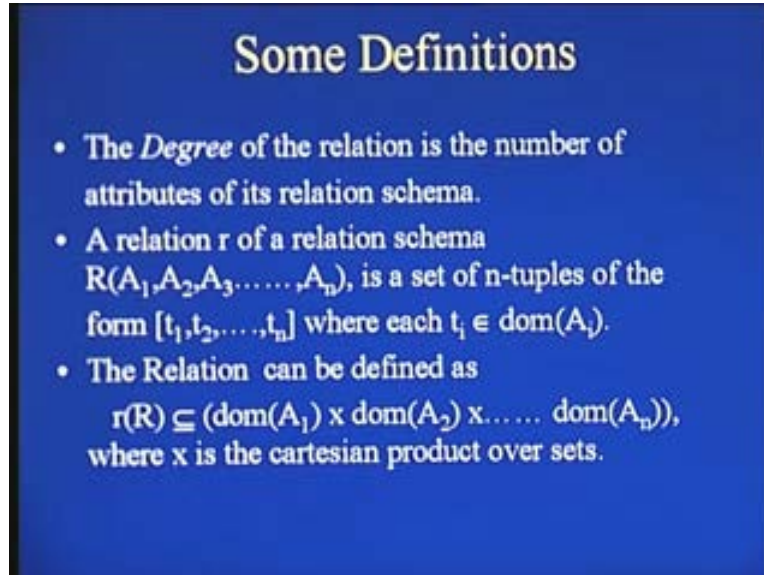Let us look at some definitions from the relational model. The relational model is based around the notion of a mathematical relation. Now a mathematical relation is set to comprise of atomic values or atomic data values. And what is atomic data value? A data value is called atomic if it cannot be sub divided into smaller values for example the age of a person.

Similarly each data value is set to reside in a domain, in the ER model a domain is also called a value set which is term that generally used by several people and in the relational model usually the term domain is used which is going to specify the range of values that a particular attribute can take.

Similarly a relation schema or a relational schema is denoted by a schema name that is in this example is shown by the name R and a set of attributes in this example shown by A_1 A_2 until A_n and each attribute has a specific value that lies within the domain specified as domain of A_i.

(Refer Slide Time: 00:12:13)

We have also defined what is known as the degree of a particular relationship. The degree of the relationship is simply the number of attributes in its relation schema. If you remember the same definition of the degree of a relationship also applied to the ER model. That is a relationship diamond can be a binary relationship or a ternary relationship, unary relationship or a n-ary relationship that is it can be associated with 1, 2, 3 or any number of entity types and the slide shows that the relation is actually a subset of the Cartesian product of all of the domains that form the attributes.

In the relational model the notion of keys play a very crucial role especially we saw in the notion in the process of decomposing relational schema in order to make them normalized or conform into let us say BCNF or third normal form or fourth normal form and so on.

So let us revisit the notion of keys in a little more detail and keys are again very important when we translate from an ER model to a relational model. We have to be aware which attributes are the key and which attributes are the foreign key and so on. So a key constraint in the relational modal essentially defines the notion of a superkey which is a set of attributes of a relation which can uniquely identify each tuple in the relation that is each instance of the relation and a key or a minimal superkey is something which is minimal in a sense that if you remove any element of the minimal superkey, it seizes to be a superkey anymore. And there is also the well-known entity integrity constraint in the relational model which says that the primary key of a given tuple may never be null.

The primary key is the minimal superkey that is going to be used to uniquely to identify a given tuple in the relation. And we also saw the notion of referential integrity which is again an important issue in the relational data model and the referential integrity constraint says that if a tuple of one relation refers to another tuple of another relation, it should refer to an existing tuple. That means foreign keys that is primary keys of another relation embedded into the tuple of yet another relation should refer to tuples that already exist in the first relation.

(Refer Slide Time: 15:09)

So the foreign key constraints are shown in the slide here that is first of all the attribute of the foreign key or the domain of the foreign should be the same as the domain of the primary key of the other relation and they have to refer to existing tuples in the other relation.

(Refer Slide Time: 00:15:27)

And we also saw that relations can be or popularly viewed as tables and which is what is a notion used in SQL that is a relation of the form student with three attributes roll number, name and lab can be re specified in the form of a table with the name student and three kinds of columns called roll number, name and lab.
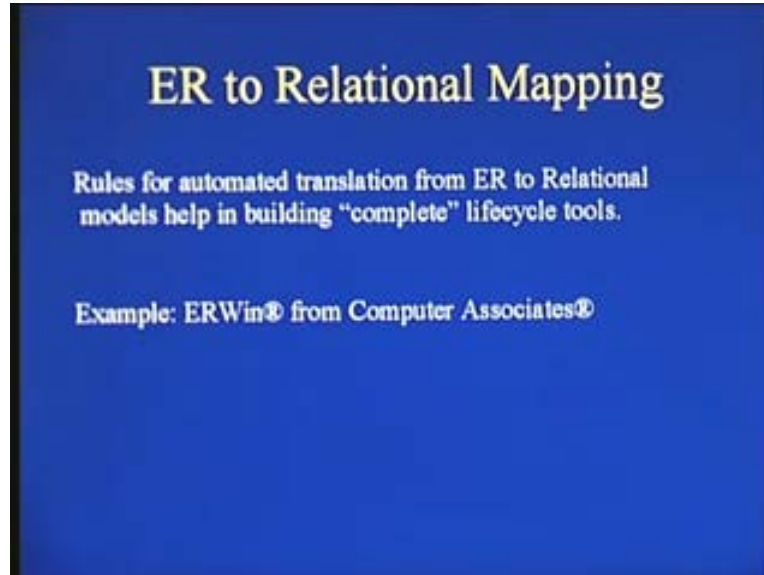
So let us now come to the issue of mapping between a given ER model under a relational database model. Now I had said in the beginning of this session, why such a mapping is important. There are several different commercial tools that are available which are called as lifecycle tools of DBMS design. A lifecycle tool provides support in several or in most of the phases of a typical database lifecycle.

That means the tool should be able to or using the tool you should be able to create a logical schema, talk to your end users saying this is what I have understood by your requirements of your system, these are the different data elements, these are the different functionality requirements that form your system and so on and then using the same tool you should be able to create physical schemata from the logical schema by automatically translating them to whatever extent possible.

In practice it's not possible to completely automate this process that is automatically generate a relational model and optimize it. Sometimes some kind of human intervention is necessary, in order when the human knows some domain knowledge cannot be captured into the ER model but there are several such tools, an example is the tool called ERWin from computer associates which provide such a support for automatically translating between ER and the relational model.

(Refer Slide Time: 00:17:34)
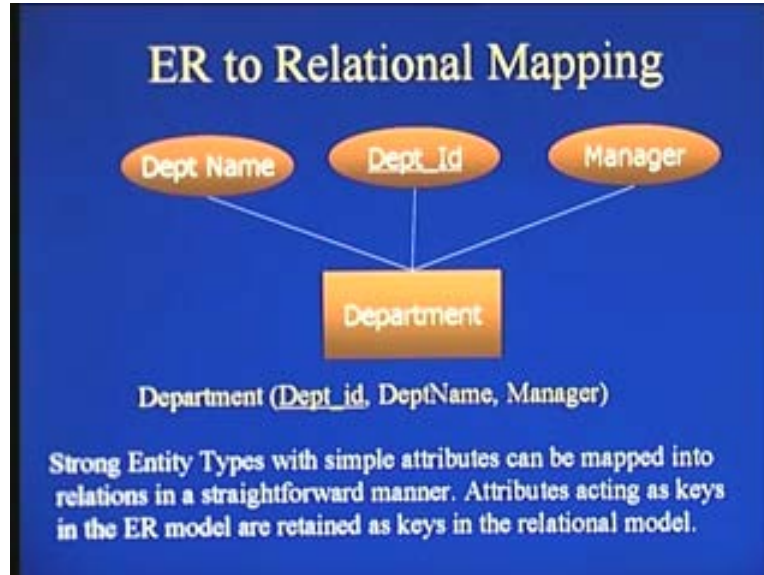
So let us see how we can go about such a translation. The first case that we are going to take is the case of a simple relation or a simple entity. So the slide here shows a simple entity type called department and it has three different attributes department name, department id and manager and the department id obviously is the key or the key attribute of this department. Now given such a relation, it is fairly obvious to, given such an entity type it is fairly obvious to see that it can be translated into a relation of the type department that is also shown in the slide here just below the figure.

So this ER model can be translated into such a relation where the name is called department and which has three different attributes department id, department name and manager. And also note that the key attribute is retained that is the department id which is the key attribute of this entity here becomes the primary key of the relation, that's found.

So this is straight forward. That is as long as we have a simple entity type with simple attributes, note that the attributes are also simple, there are no multi-valued attributes or composite attributes and so on. And it can be translated in a straight forward fashion to the relational model.

(Refer Slide Time: 00:18:57)
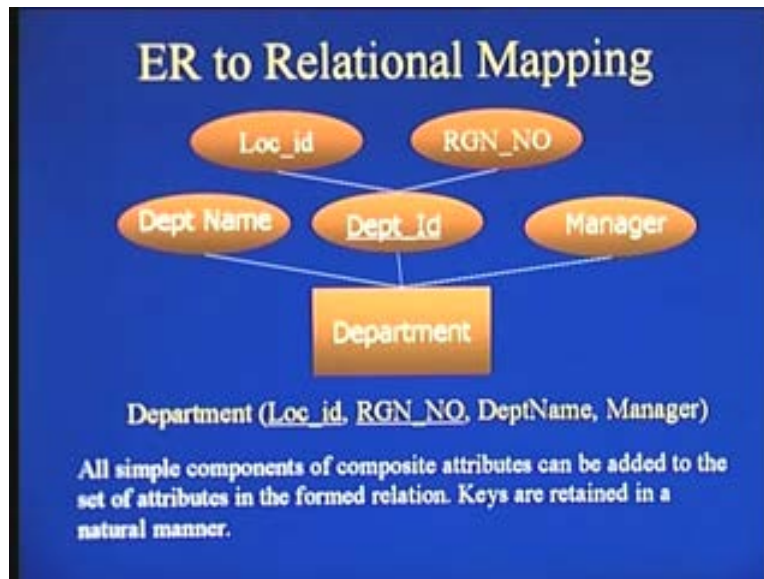
What happens if we have a composite attribute? Remember that a composite attribute is something that is made up of sub attributes. A composite attributes is different from a multi-valued attribute that is a multi-valued attribute is something which can have many values for the same attribute, the color of a bird can have several different colors. On the other hand a composite attribute is made up of two or more other attributes each having its own domain. For example the slide shows a composite attribute called department id for the same example of a department entity type, so the department id is a attribute here which has which is a composite attribute which in turn is made up of two other attributes called location id and region number so on.

Now a location id could have a different domain, let us say location ids are given alphabets like a b c d and so on and region number are given numbers 1, 2, 3, 4 and so on. So both of them may have different domains and they combine to form the attribute called department id which in turn is also the primary key of this department. Therefore we are considering two different aspects here, one is how to deal with composite attributes and the second is what happens if the composite attribute is the key attribute of the entity type.

So the slide here shows the example of and shows how we can translate this into relational model. Firstly, the name department of the entity type becomes the name of the relation called department. And all the other simple attributes are retained, department name is retained as department name, manager is retained as manager and only here for the composite attribute, the department id never appears here. It's just that all the simple components of the composite attribute are straight away loaded into this relation. That is location id comes here and region number comes here and in fact if either of these two let us say location id or region number is again a composite attribute and it has some more attributes, just take all the simple components of the composite attributes.

So don't take region number and just take whatever is the simple component of this region number and add it to the relation here. And all of these simple components which form the composite attribute which is the key becomes the primary key. That is the primary key here is a composite key made up of two or more different attributes which combinedly identify or help in identifying a tuple of the relation.

The next example that we are going to see is the example of how to map relationships. First of all let us look at the following relationship that is shown in this slide here. What characteristics can we ascribe to the relationship that is shown here. Firstly, we notice that the relationship here is a one is to one relationship that is one employee is associated with one insurance record or rather the other way around in this case, that is one insurance record is associated with one employee and have a look at the association as well.

(Refer Slide Time: 00:21:51)
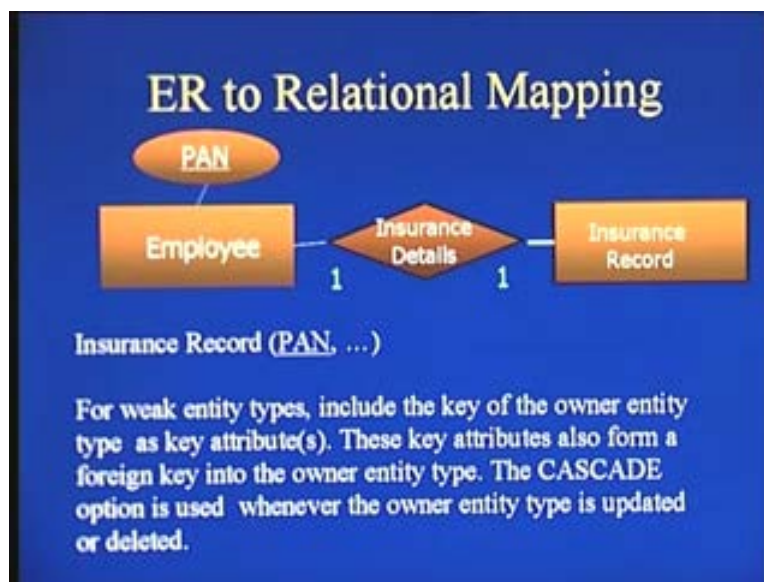
The association or the relationship type is an identifying relationship type that means the relationship type called insurance details shown in the slide here is used to identify or provide an identity for the insurance record by associating it with an employee. And also the insurance record has a total participation in this relationship that is insurance record has no existence without this relationship.

Now how do we translate such entity types? So essentially the idea here is how do we translate insurance record into the relational model? For weak entity types the translation is shown here in the slide below. Just create a relationship or a just create a relation of the same name as the entity type but since it does not have a key to because weak entity types do not have key attributes. Since it does not have a key attribute, use the key attribute from employee with which it is associated with and take that key attribute and make it into the key attribute of insurance record.

However note that since pan number here is also the primary key for employee, this has to be made as a foreign key of insurance record that means whenever we are updating or altering the table, we have to use the cascade option. Whenever let us say the employee type is updated or deleted that means to say that if the employee relation is deleted from the database, this has to in turn delete all the insurance record relations from the database itself because insurance records do not have any existence without the employee records.

So the three step here in order to translate a weak entity type is to first identify or is to first locate the identifying relationship and see which entity type is this weak entity type associated with and use the primary key of that entity type as the key for the weak entity type or the record of or the relation for the weak entity type and make it into a foreign key of this entity type and use cascade options whenever updations or deletions or performed on the strong entity type.

(Refer Slide Time: 00:25:20)
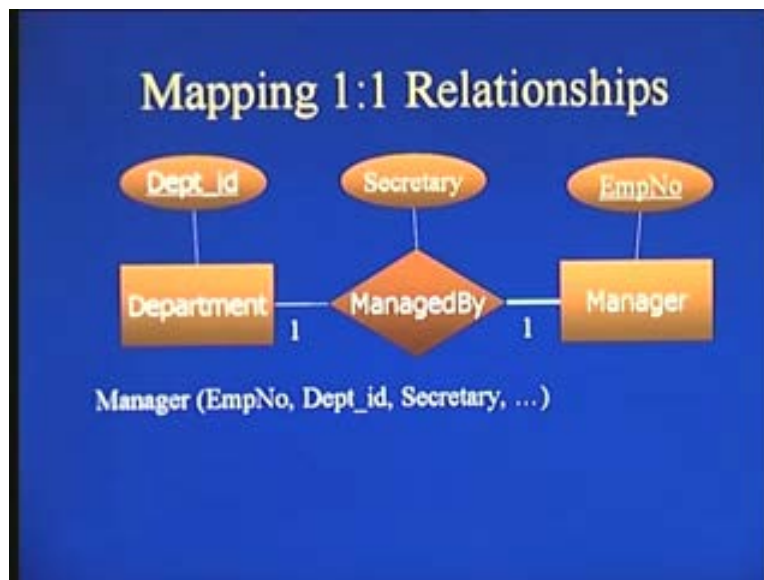
Let us move on with translating relationships. So how do we translate, let us take the simplest form of relationship again the 1:1 relationship. We saw what happens or how do we translate 1:1 relationships, when weak entity types are considered. Now let us consider an entity type which is not weak but still is involved in a total participation, this is shown in the figure here. The figure shows a relationship type called managed by which relates two different entity types that is a department and manager and there are attributes, relevant attributes are shown for each of them. That is the department has a key attribute called department id and a manager has a key attribute called employee number and the relationship itself has a key attribute called secretary that is a secretary is assigned for a department that is managed by a manager.

That is the secretary attribute does not have an existence without an existence of this relationship that is if a department is not managed by a manager then there is no secretary that is associated with this.
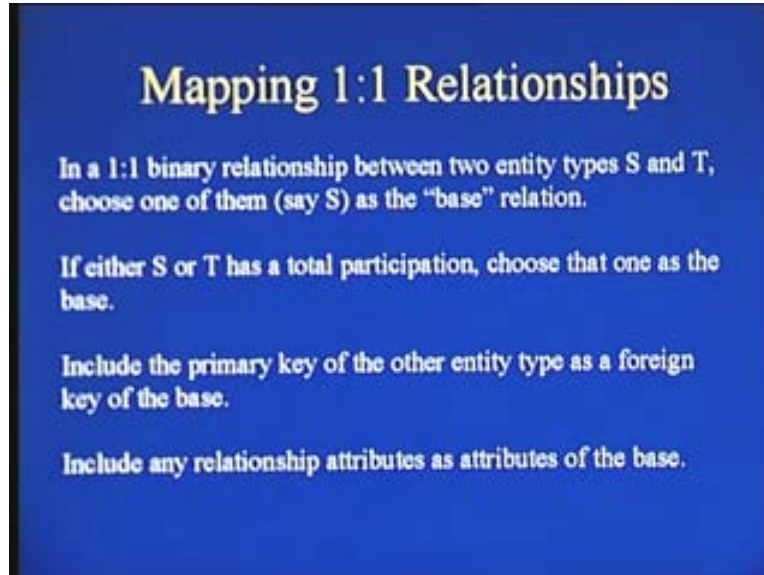
Now how do we translate this? The translation is again shown in the slide below. So first create an entity type or create a relationship, create a relation called manager. Let me repeat this again, for this relationship create a relation in the RDBMS model called manager with the following attributes. Now you might be wondering why should we create a relation called manager, why not department. Now let us think about it a little further. See in this slide here that manager is a strong entity type, it is not a weak entity type. However it is involved in a total participation in this relationship type.

What is a total participation? The total participation is that the entity type does not have any existence without being participating in a relationship type of this kind. So what it essentially means here is that a manager has no existence that is a manager would probably be just an employee, so a manager would have no existence unless here she is associated or is given a department to manage. It is the entity type that is involved in the total participation is taken as a primary entity type or the base entity type, primary relation called manager and then employee number becomes the key here that is the key for manager and the department id which is the primary key for department becomes a foreign key in manager and whatever attributes are associated with the relationship itself become attributes of this relation here that is of the manager relation here.

So therefore the manager relation has a primary called primary key called employee number and a foreign key called department id. Note that this makes sense, when we note that manager does not have any existence without this relationship. That is referred to the problem of referential integrity in relational data model. What is the referential integrity stipulate? Whenever a foreign key refers to a tuple in another relation, the tuple should exist that is it should refer to an existing tuple in the other relation. Now if we had department as the base entity or the base relation here, we cannot use employee number as a foreign key because the manager relation won't even exist before this relation that is managed by is formed.

On the other hand department has an independent existence without whether or not a manager is associated with it. Therefore that forms a rational behind, why we choose the entity type which is involved in total participation as a base entity type for the translation.

(Refer Slide Time: 00:30:03)

Mapping 1:1 Relationships

In a 1:1 binary relationship between two entity types S and T, choose one of them (say S) as the "base" relation.

If either S or T has a total participation, choose that one as the base.

Include the primary key of the other entity type as a foreign key of the base.

Include any relationship attributes as attributes of the base.

So let us summarize this previous slide once again. So in any 1:1 binary relationship between types S and T, choose one of them as a base relation. In case one of them is involved in a total participation choose that as the base. If neither department nor manager where to be involved in total participation, it doesn't matter which you are going to choose as the base relation. Include the primary key of the other entity type as a foreign key in the base relation and include any relationship attributes as attributes of the base relation.
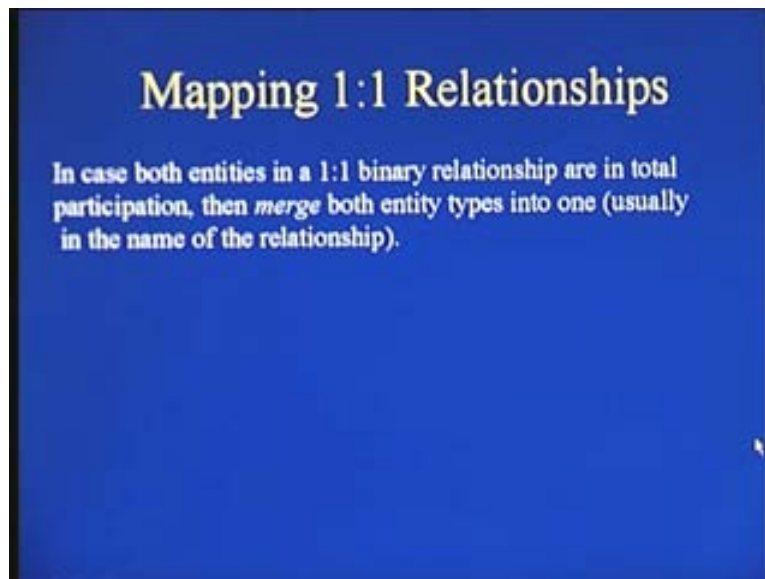
(Refer Slide Time: 00:30:42)

Mapping 1:1 Relationships

Consultation (Proj_id, PAN, Secretary, ...)

Consider the example shown in the slide here. What happens if in a 1:1 relationship both entity types that is both entity types that are participating in this 1:1 binary relationship are involved in a total participation. Take a look at the slide here. The slide shows two entity types project and consultant and each project is uniquely identified by a project id and each consultant is uniquely identified by his or her pan number. And there is a relationship type called consultation which has its own attribute called secretary and it's a 1:1 relationship and both of them are involved in a total participation that is a project has no existence unless it is being consulted by a consultant and a consultant has no existence unless here or she is associated with a project. So neither of them will have independent existence without the other.

In such cases we cannot identify any relation as the base relation. If we identify project as a base relation and try to use pan number of the consultant as the foreign key then referential integrity could be violated. It's the same in the other way around as well. If we use consultant as the base relation and try to use project id as the foreign key, again there is a chance of violating the referential integrity. In such cases the simplest way is to take the relationship type, in this case the consultation as the base relation. That is form a relation called consultation and use project id and pan as the primary key of consultation and then all of the attributes from both of them will become attributes of this relation.

(Refer Slide Time: 00:32:40)

Mapping 1:1 Relationships

In case both entities in a 1:1 binary relationship are in total participation, then *merge* both entity types into one (usually in the name of the relationship).

So in case both entities in a 1:1 binary relationship are both in total participation then we merge both of the entity types into one usually in the name of the relationship that is in the name of the relationship called consultation in the previous example. Now let us see how do we map one to N relationships. What is a 1: N relationship? That is N entities of one of the entity types could be associated with one entity of the other entity type, that is it form some kind of a tree relationship that is one entity being associated with N different entities of the other type.

So the slide here shows such an example that is employees works in department that is employee is an entity type, so N different employees can work in one department that is one department may have several employees but each employee is associated with only one department. And of course there are keys called employee number for employee and department id for department. The slide also shows how we can reduce this to a relation. The simplest way is to take the entity type on the N side of a relationship.

(Refer Slide Time: 00:33:02)
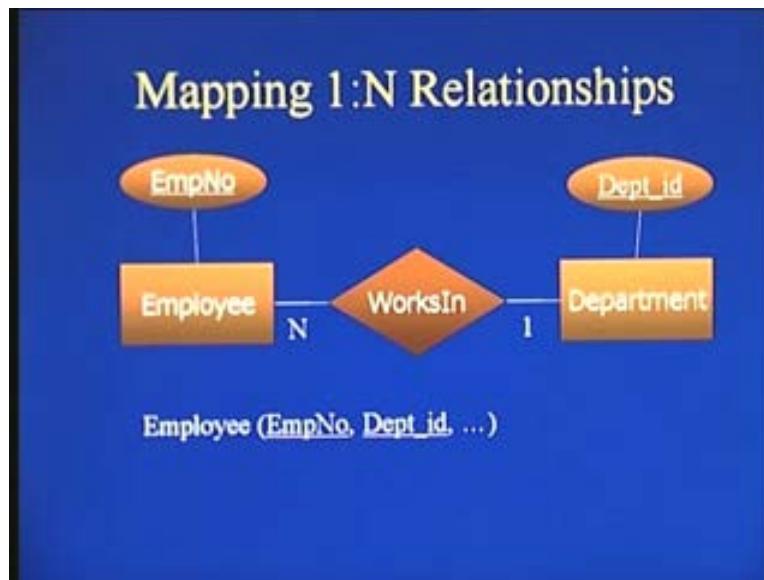
So in this case the employee, so take this as the base that is translate it into a relation called employee and the primary key of employee becomes the key of the employee entity type here and the department id becomes a foreign key here.
(Refer Slide Time: 00:34:19)
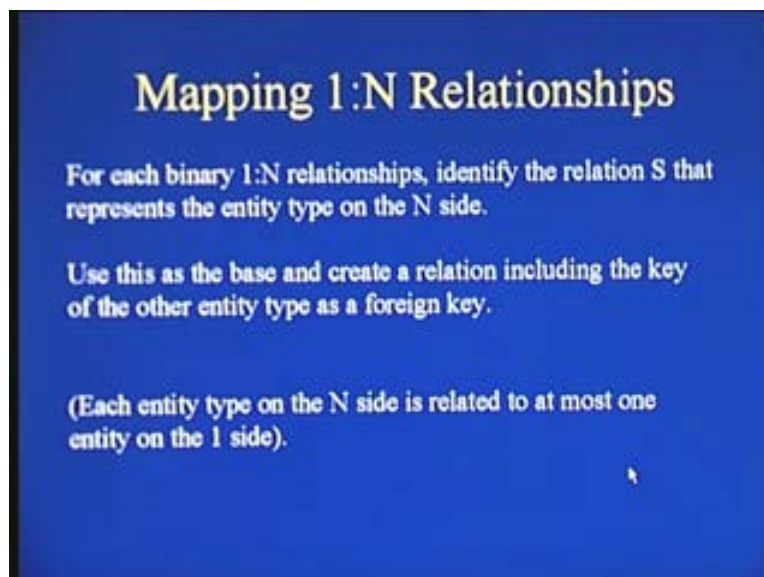
So this is as simple as that. That is for each binary one is to N relationships identify the relation S that represents the entity type on the N side. Why is this so? Because each entity type on the N side uniquely identifies a department that is uniquely identifies the entity type on the other side. Therefore we can use the primary key of the entity type on the other side as a foreign key in the base relation. Therefore use this as the base relation and create a relation including the key of the other entity type as the foreign key.

(Refer Slide Time: 00:35:03)
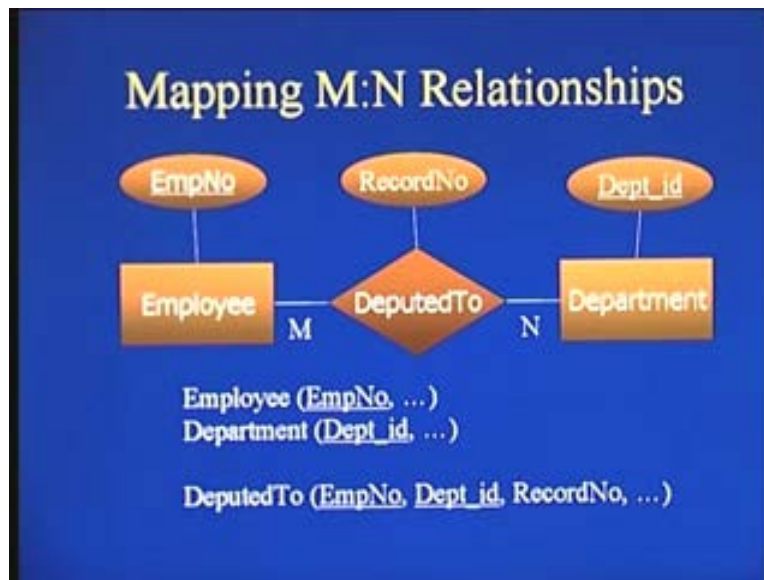
How do we map M N relationships? Now what is an M N relationship? An M N relationship essentially says that M different entities of the first type can be associated with N different entities of the second type. Therefore there is no unique identification that is given an employee in the previous case, one could uniquely identify the department with which the employee is working in because each employee can work in at most one department.

on the other hand here let us say a relationship called deputed to, so an employee could be deputed to several departments let us say. So M different employees can be deputed to N different departments and of course employee has employee number as the key and department has department id as the key and so on. And deputed to also has an attribute called record number which maintains a record of which employee is deputed where and so on.
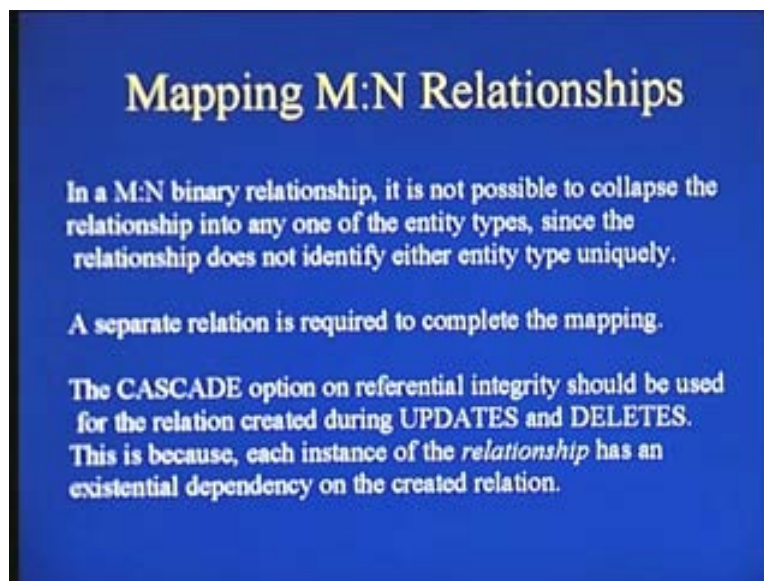
So in order to translate such relationships, note the steps that are shown in the slide here. There are three different relations that are formed, one is the employee relation that is one of the entity types of this relationship type. So the employee relation is formed with employee number as the primary key and all the other attributes that form the employee entity type. Similarly department relation is found with department id as the primary key and all other relations and then a separate relation is created for the relationship type itself.

So deputed to becomes a separate relation by itself and then uses employee number and department ids as foreign keys and also as the primary keys of this relation and whatever attributes that belong to the relation becomes part of or belong to the relationship type becomes part of the relation that is created here. That is the record number attributes become one of the attributes of deputed to relation. Note that we cannot move this record number that is the attribute of this relation either the employee or a department because it does not uniquely identify either employee or department.

Each employee could be associated with N different record numbers because they could be associated with N different departments and similarly each department could be associated with M different record numbers because M different employees could be working in that department. So this slide shows summarizes, how M is to N relationships are translated. In a M: N binary relationship, it is not possible to collapse the relationship into one of the entity types because neither of the entity types uniquely identifies the other entity type.

Therefore a separate relation is required and usually this is in the name of the relationship type itself. So a separate relation is required in order to complete the mapping and of course the cascade option should be used whenever updates are performed on any of the relations pertaining to the entity types.

(Refer Slide Time: 00:37:57)

That is whenever the entity type called employee or department is updated or deleted then these changes should be cascaded so that they are reflected in the deputedto relationship as well. That is if employee entity type is deleted then of course the deputed to relationship should also be deleted. Now one might ask the question is it possible to use the strategy that is use the relationship type as the base relation rather than any of the participating entity types.

Can we use this strategy for mapping 1:1 and 1: N relations as well because M is to N is simply a generalization of 1:1 and 1: N relationships. So of course it is possible that is take the example of employee works in department that is N different employees working in one department. We can still create a separate relation called works in where it can use the employee number and the department id as the foreign keys of this relation. However it just creates an extra relationship or extra relation in the database that is totally unnecessary but this is sometimes actually attractive to use than collapsing the relationship into one of the relations especially where, especially if we have to avoid null values.

(Refer Slide Time: 00:39:06)
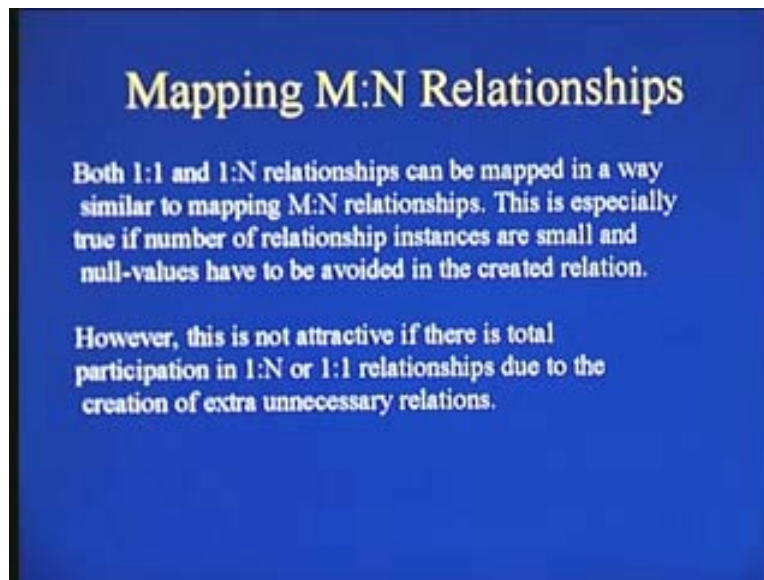
That is especially if we have to, especially if we have cases where there are some employees who do not work in any department. if N different, if an employee can be associated with at most one department it means that an employee can also be associated with zero departments. So in that case the department id field of employee would be null. It does not violate referential integrity because remember that referential integrity says that a foreign key should refer to an existing tuple or else should be null. Therefore it does not violate referential integrity but creates a lot of null values in the database schema, in the database itself.

So if we have to avoid null values, it is actually preferable to use the relationship type as the base relation when performing the translation. How do we map multi-valued attributes? We have seen how to map composite attributes and a simple attributes and keys and so on but what happens if there are multi-valued attributes. Composite attributes are different from multi-valued attributes in the sense that each of them can have several different domains that is it is just a combination of several simple attributes. so we just open up the combination when we are translating a composite attribute and then include all the simple attributes that form part of the composite attribute.

(Refer Slide Time: 00:41:11)
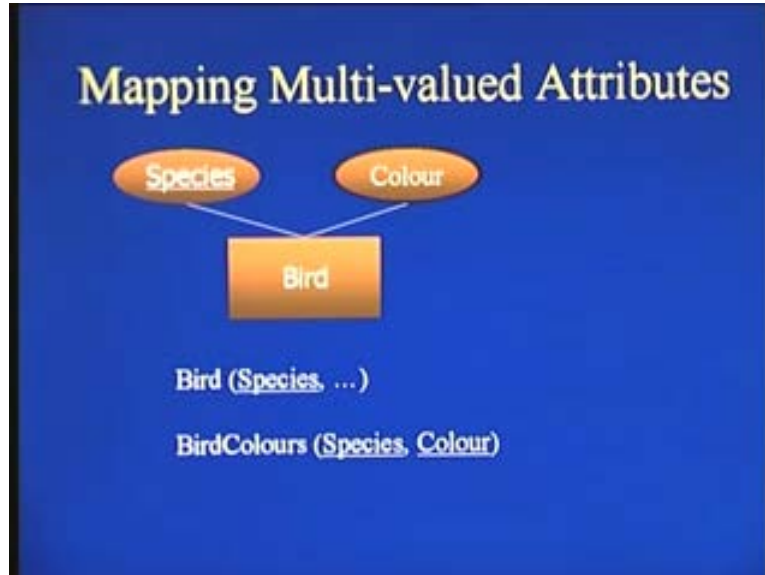
On the other hand a multi-valued attribute is not composition of several sub attributes instead it is an attribute that can take on several values instead of one value and the example, the slide shows is that of a bird. So a bird has a multi-valued attribute called color. So what is the color of this bird? A bird could have several colors it need not have just one color and of course there is a primary key called species which identifies each bird uniquely.

So in order to translate multi-valued attributes, take a look at the lower half of the slide which shows two different relations which make up this translation. The first relation shows a relation called bird with species as a primary key and all other attributes except the color attribute, all other attributes of the entity type called bird.

And then a separate relation is created called bird colors where species and color are both included and are both of part of the key that is combinedly define the key of this bird. therefore we say that birds species eggs has color Y, eggs has color Z, eggs has color A and so on. So the color attribute may be repeated in several or several tuples or rather the species attribute may be repeated in several tuples, once for each different color that the bird can take and both of them that is species and color become the primary key for bird colors.
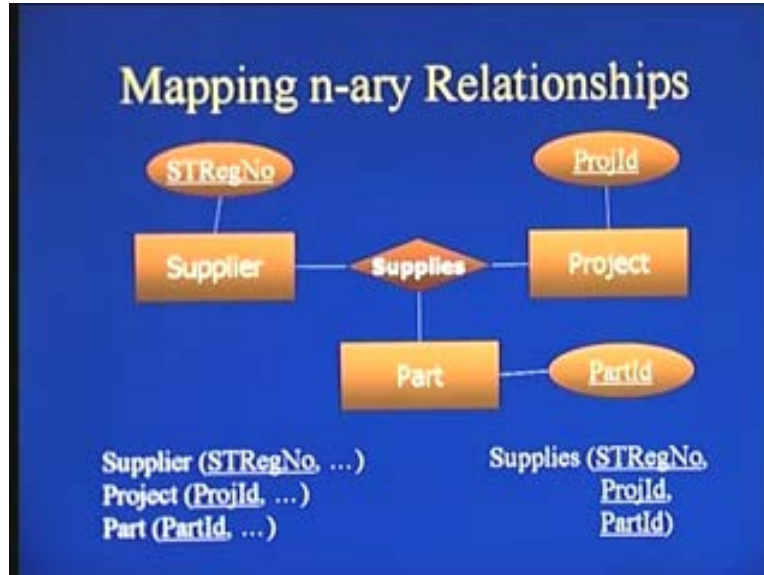
(Refer Slide Time: 00:43:34)
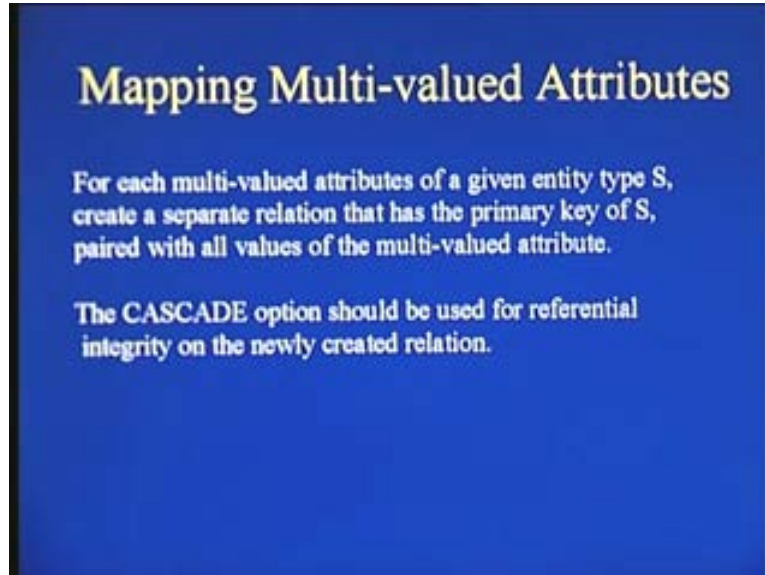
So for each multi-valued attributes of a given entity type, we have to create a separate relation that has a primary key of S paired with all possible values that the multi-valued attribute can take. and of course the cascade options should be used for referential integrity on the bird relations that is whenever the bird relation is deleted or updated, the corresponding changes has to be made in bird colors as well.

How do we map n-ary relationships? Until now we have been looking at binary relationships, what happens when there are n-ary relationships and different entities forming part of the relationship. The slide here shows such an example that is the standard example of suppliers supplies part to project. So there is a supplier who is uniquely identified by the supplier or the sales tax registration number or something STReg number and there is a project that is uniquely identified with project id and there is a part that is uniquely identified by part id and then the supplies relation which relates all of these three different entity types.

So the simple way of translating this is to use a separate relation called supplies as the base and of course separate relations each for supplier, project and part with their corresponding primary keys and the supplies relation which has the primary keys of each of these relations has the foreign key of this relation.

(Refer Slide Time: 00:44:05)

(Refer Slide Time: 00:45:15)
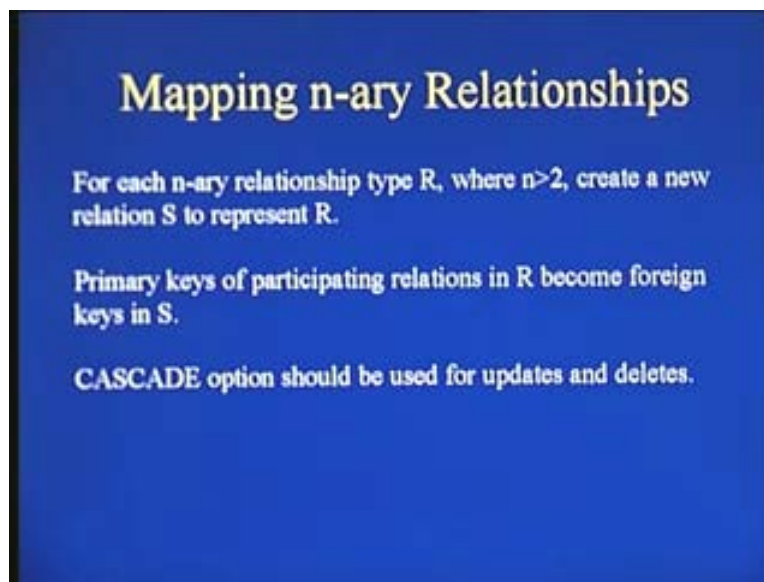
So for each for each n-ary relationship of any type R where n is greater than 2, we have to create a new relation S to represent this relationship, to represent this relationship type R. and of course the primary keys of the participating relations become foreign keys in this new relation and the cascade option should be used for all of the relations that correspond to the entity types that participate in this relationship type. What happens if one of the relations in an n-ary relationship type is a weak entity type? That is let us say part is a weak entity type, there is no existence for part unless it is associated with a supplier supplying it to some project.

In that case we have to identify, we have to first identify the entity type which gives an existence to part and the part relation here has only foreign keys it does not have any primary keys but the supplies relationship does not change. That is it doesn't have any part id, it just has a, it simply has the supplier primary key and the project primary key without the part primary key. Therefore we get two different relations that have foreign keys and they don't have their own primary keys.

(Refer Slide Time: 00:46:56)

So that bring us to the end of this session that talked about, that gave a, that talked about how we can map ER models into relational database models using several different rules. and of course this is not a comprehensive set of rules because there are several other sets of rules used for example in derived attributes or enhanced ER models like generalization and specialization which are not covered here but all of them in totality are used to create the basis for any kind of a tool, software tool that can translate between given ER schema and its corresponding relational schema.

So this slide shows a summary of each of this mappings that is it gives the set of thumb rules saying if this is what is given in the ER model then what happens in the relational model. So in an ER model if an entity type is given then a corresponding entity relation that is a relation in the name of the entity type is created. If a 1:1 or a 1: N binary relationship is given then we create corresponding foreign keys from the N side to the one side or from the weak entity type to the strong entity type in this relation. So we create a relation and create appropriate foreign keys from them. If a M: N relationship type is given as shown in the slide here then we create a relation with the name of the relationship type, so there is the within quotes which shown as relationship relation with two foreign keys that is one for each entity type that participates in this relation.

(Refer Slide Time: 00:46:57)

Summary of Mappings

ER Model	Relational Model
N-ary relationship type	"Relationship" relation with n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple attributes

If an n-ary relationship type is given, it is still the same strategy that is we create a relationship relation with N different foreign keys that is one for each entity type or rather N different foreign keys as long as these entity types are strong entity types. If there is a simple attribute in a ER model that simply becomes an attributes in one of the relations in the relational model. if it is a composite attribute in the ER model then it becomes the set of simple attributes that is you take the simple part of all composite attributes that is just go on finding the simple attributes that form the composite attribute and then make all of them as part of this relation.

(Refer Slide Time: 00:49:47)

Summary of Mappings

ER Model	Relational Model
Multi-valued attribute	Relation and foreign key
Value Set	Domain
Key attribute	Primary or secondary key

If it is a multi-valued attribute then we need to create a separate relation and a foreign key that is you have to associate the primary key of the base relation with each possible value of the multi-valued attribute. If it is a value set it becomes a domain, value set in the ER model it becomes the domain and a key attribute in the primary in the ER model will become either a primary or a secondary key in the relational model. So that brings us to the end of this session.