Carbon Accounting and Sustainable Designs in Product Lifecycle Management

Prof. Deepu Philip
Department of Management Sciences

Dr. Amandeep Singh Oberoi
Imagineering Laboratory

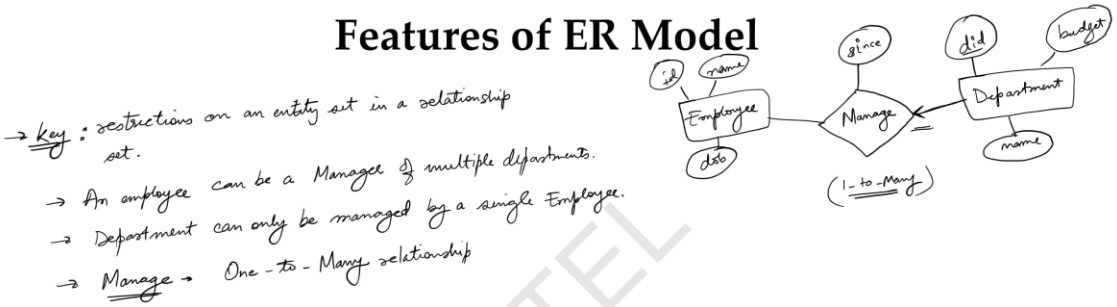
Dr. Prabal Pratap Singh
Department of Management Sciences

Indian Institute of Technology, Kanpur

Week 10

Lecture 47

Terminologies in Database Design (Part-2)



Now let us understand what are the different kinds of keys or the identifying information in a particular database by using the diagram. So again we will try to draw a different er diagram and it will have same things but one extra information in the diagram. So we have this employee entity set it will have id and it will have name and it will have date of

birth similarly we have department, we have department id. We have name and we have budget, right. And we have a relationship between these called as manage.

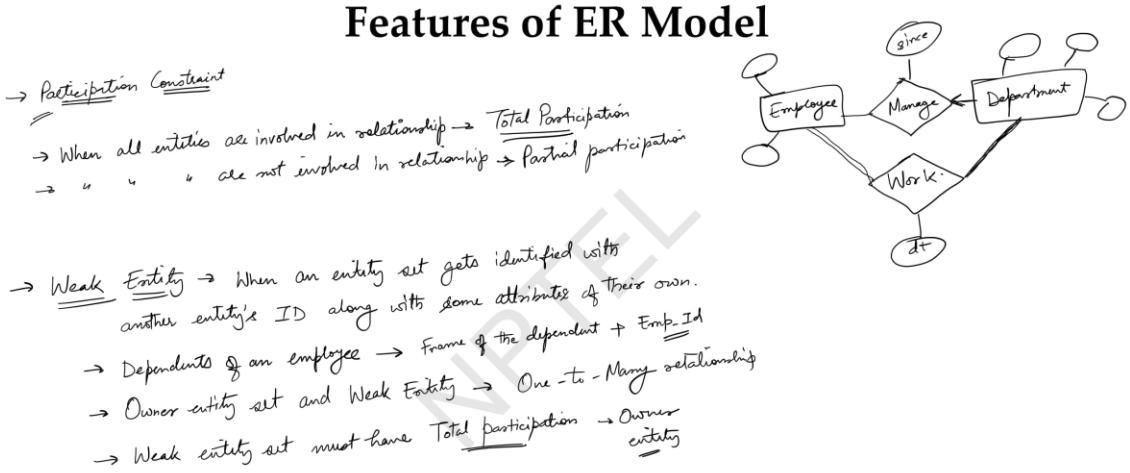
Now this relationship is between these two entity sets and the attribute of this this descriptive attribute of this relation could be since. So what this ER diagram is capturing is an employee from the employee entity set can manage a department. So it could be a manager in the department now a particular employee could be manager of multiple departments but not all employees are managing a department, right. So with this we have a constraint here we need to identify these kinds of constraints that each employee will not be will are not required to manage a department. But a particular employee can be a manager of multiple departments so another representation item in the AR diagram is we can draw an arrow here.

So what does this arrow means we will try to understand so these are different kinds of keys key constraints keys are restrictions on an entity set in a relationship set. So, let us write about this ER diagram an employee can be a manager of multiple departments. But a department can only be managed by a single employee. So that is why we are trying to draw an arrow from the department towards the manage relationship. So manage it is a one to many one to many relationship.

So this is one to many. Why it is one to many? Because a particular employee could be a manager of multiple departments, right. And similarly, you can also see while we were working in this relationship, this is a many to many because an employee can work in multiple departments and the other thing is also true. So that is why this is many to many.

So the next feature of the ER model is participation constraints. We have started identifying the constraints by gathering the data from the real world using the requirement analysis. And so until now here what we have done is we have tried to gather the information from the real world by using the requirement analysis. And currently we are in the conceptual design phase and we have learned how to diagrammatically show what is the actual relationship between different entity and entity sets in our proposed database. And after this we will try to convert it into a schema.

This ER model provides you different kinds of features. So we have already identified few features here like this key feature or the relationship and relationship sets.



Now there is one more feature that is how to identify the participation constraints in the database diagram. So to understand this again let us create this employee entity set and department entity set. And it is connected with the manage relationship, right.

Further we have already said that these two entity sets can have other uh relationship sets as well between them. So the another example of the relationship set between this is work relationships. So a particular employee works in a particular department or more than one department. And both of these relationship sets can have their own database attributes like since when it is being managed by a particular employee and the date since when the particular employee is working. So this is also connected means this is also a relationship between these two entity sets.

And all these entity sets will have their own attributes as we have discussed earlier now what happens is these entity sets can also identify the participation between the with the relationship. So what we can say is when all entities are involved in relationship then it is a total participation. Each employee of the organization will be working in one or more department of the organization, right. So all the entities in this entity set will be related with this entity set of department so this is an example of a total participation. However some of the employees from the employee entity set will not be managing any department of the organization.

So, this manage relationship is an example of a partial participation. So, when all entities are not involved in a relationship that is a partial participation. Now this we have already understood that what this arrow means. Whatever we have drawn here, this diagram is not showing which entity or which relation is having a total participation or partial participation. So, how to show it?

So whenever there is a total participation we can either use double lines between entity and the relationship or we can use a thick line. So just follow one rule either double lines or a thick line. So, thick line could be by thickening this line. So, these are all available when you are trying to draw these diagrams in a software program or by hand. So, you need to follow one out of these representation either a double line or a thick line.

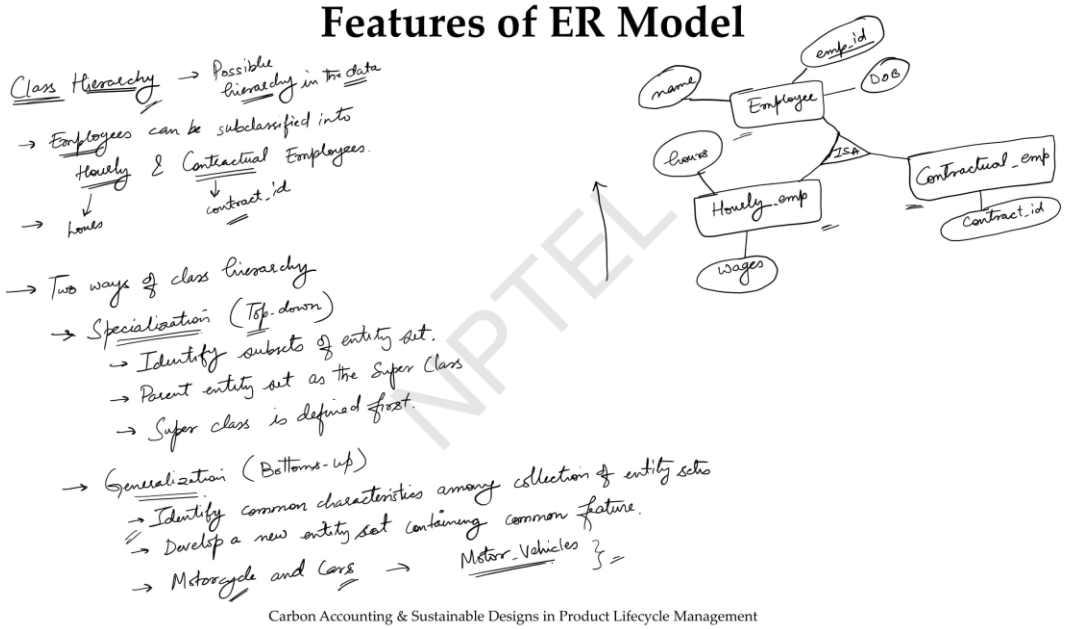
Now, another feature of the ER model is a weak entities. So entity is nothing but a distinguishable simplest entity in the complete database. But we can further identify different kinds of entity in the database. So let us say in the employee database system. There are different dependents of the employee.

So all of these dependents of the employees are connected with the employees and they are not distinguishable in the complete data set with their own unique key or unique identifying information. They need the identifying information of the employee of the organization to find out who's dependents are there. So these kind of entities that need strong relationships in the actual database are known as weak entities. So when an entity set gets identified with another entity's id. Along with some attributes of their own.

So they do have their own attributes as well but the actual identifying information is related with another entity in the database. So an example of a weak entity is dependence of an employee so how do we identify them we use the first name or f name of the dependent plus the employee id. So this is a unique identifying information in the database and with this we are connecting the first name of the dependent. So owner entity set that is the actual employee of the organization which is the owner entity set of this weak entity set. Owner entity set and weak entity set.

They will have a one to many relationship also we have learned about participation constraints so weak entity set must have total participation. Total participation with whom? The actual owner entity or the employee of the organization. So whenever you are trying to capture the weak entity in a database. You need to create it in a way that is you need to create a diagrammatic representation in a way that any user or the database developer once read the diagrammatic representation.

They can understand that this is a weak entity and this week the owner entity of this weak entity is a particular entity that is already being defined in the database, right.



So other feature of entity relationship model is a class hierarchy. Now we are trying to understand these major concepts because once you will have a real world problem with you. And you need to translate that real world problem with your own database you need to find out when to introduce which kind of feature of the entity relationship diagram. So that is why these different concepts like class hierarchy, partial or total participation, weak entity, relationship sets, entity sets, these needs to be actually first understood and then needs to be implemented in your own problem.

So when we try to develop our own carbon accounting database, we will try to understand how different kinds of organizations are connected with each other. So if we are trying to develop a global carbon accounting database, we need to define different kinds of database schemas and try to identify all these weak entities in those schemas. So that when we are trying to translate it further into the actual database schema using a relational database model. Then it will be an easy process, right. So let us continue with the class hierarchy so since we are defining different kinds of entities so each these entities can have possible hierarchy in the data.

Now, we need to show this hierarchy using the diagrammatic representation. So, first let us take an example. So, employees can be subclassified into hourly and contractual

employees right. So these are two sub classification of the employee entity set now these classifications can have their own inherent attributes. So hourly sub classification can have hours worked and contractual employees can have the contract id.

So how can we show this kind of representation. So let us say we have employees which have their own employee id and it is a unique key further they will have a date of birth and they will have a name. Now this there are two more sub classifications that we have identified in our data. So this could be hourly employees and this is another entity set because they have this is a collection of different entities what are those entities employees that are working on hourly basis similarly they are contractual employees. And this is another entity set and it is a sub classification.

So how this er diagram can show that this is a sub classification by using a triangular representation. So we can say this is connected with the parent entity class and these are the hierarchical two subclassifications. So if you see a particular triangle with this is a or has a sub classification then we can read this er diagram as employee is a entity set. That can be sub classified into hourly employees and contractual employees and these entity sets can have their own attributes like hours worked or the wages hourly wages similarly contractual employee can have contract id. So this is how we can represent a class hierarchy if you have that in your requirement by getting from requirement analysis.

So there are two different kinds of class hierarchy. Class hierarchy can be seen as a top-down or bottom-up. So we can have two ways of class hierarchy. The first is top-down that is specialization. It is termed as a specialization.

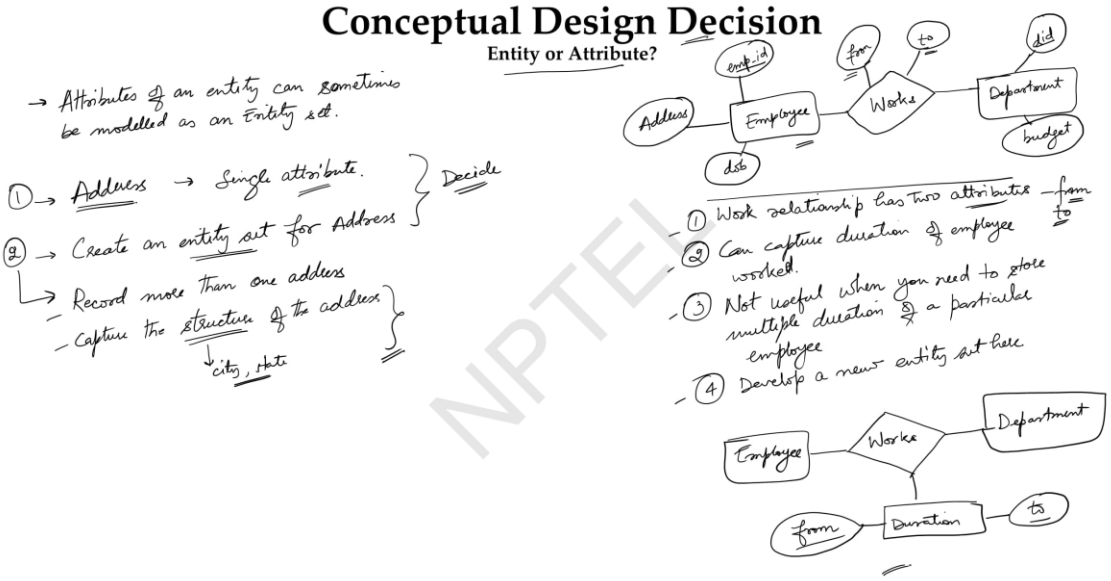
This is a top-down approach. So what happens is developers will try to identify subsets of an entity set. So we identified subsets as hourly employees and contractual employees from the employee entity set here. So we are actually doing specialization hierarchy process in this. And we term parent entity set as the superclass.

And the important thing is why it is top down because superclass is defined first. Another way of looking at class hierarchies could be bottoms up and it is known as generalization. So it is a bottoms up approach. So let's say you initially defined hourly employees and contractual employees as two different entity sets. So the first task is to identify common characteristics among all the collection of entity sets.

So we have two different entity sets, hourly employees and contractual employees. And we need to first identify what are the common characteristics between these two sets,

right. Then the next step is to develop new entity set containing common features. So we may develop the employee entity set after studying the first step that is after identifying the common characteristics. So this is that is why when we are going from bottom to up we are saying that this is a generalization process of doing class hierarchies an example could be motorcycles and cars.

So let's say you are working on a different database schema and you have already defined motorcycle schema of motorcycles and cars. Now you need to define a superclass. So this could be a generalized motor vehicles class motor vehicles class. So when you already have defined the schema for motorcycle and cars and you are identifying the common characteristics out of these and then creating a super class then this is actually the generalization process, right. So, now we have covered most of the important features of ER models and what are the different kinds of diagrammatic representations, how to show all these relations between attributes, entity sets and the relationship sets.



Now, once you have defined the first blueprint of your ER diagram, you may need to take some decisions. Why you are defining a particular thing as an entity or why it cannot be defined as an attribute. So these kinds of decisions are important while performing your translation from requirement analysis to conceptual design. So let us first create the ER diagram to understand the issues with entity versus attribute decisions. So again we have employee entity set.

And we have department entity set and what we are doing is we are studying the works relationship and we are now defining two descriptive attributes of this relationship set. And these are from and to so a particular employee from an employee entity set will start working from a particular date. And will work until a particular date and they have their own emp id dov department id which is unique and let's say the budget. So let's say that while defining this employee entity set, we can define the address of a particular employee. So the address of an employee can be a single string or we can further divide it into different subdivisions of attributes like the actual address, then the city, then the state and then the country of the employee.

What we are doing here is we are actually defining we are actually dividing the attributes the single attribute of an entity into multiple attributes. And we can identify it as a complete entity set. So we can initialize address as an entity set like the employees as an entity set. And what happens after defining an address as an entity set is each employee will be related to the address entity set. And address entity set will be further divided with its own attributes.

So there are advantages and disadvantages of doing this. That is why this is a decision stage. So attributes of an entity can sometimes be modeled as an entity set, right. Now the example which we are talking about is the address so address can be a single attribute containing the complete address as a string, right. Or we can do is, these are the two different options, option one and option two of storing an address.

Address can be stored by creating an entity set for address. So we need to decide out of these two options that is available to a developer. Now this could be a simple option and we can directly provide a new attribute called address here and the user will start storing the complete address as a string, right. But we can also perform this option. So what are the advantages of using this option to decide anything we need to first identify the pros and cons of the options available.

So the second option what it will provide it can record more than one address further it can capture the structure of the address. So with structure what we mean is the city state these can be the actual simple attributes that can be stored distinguishably in the database schema. So the advantage of using this feature is that let's say the after developing the complete schema. The developers try to identify that what are the number of employees in a particular city that are living in in their organization. So if we are using the first

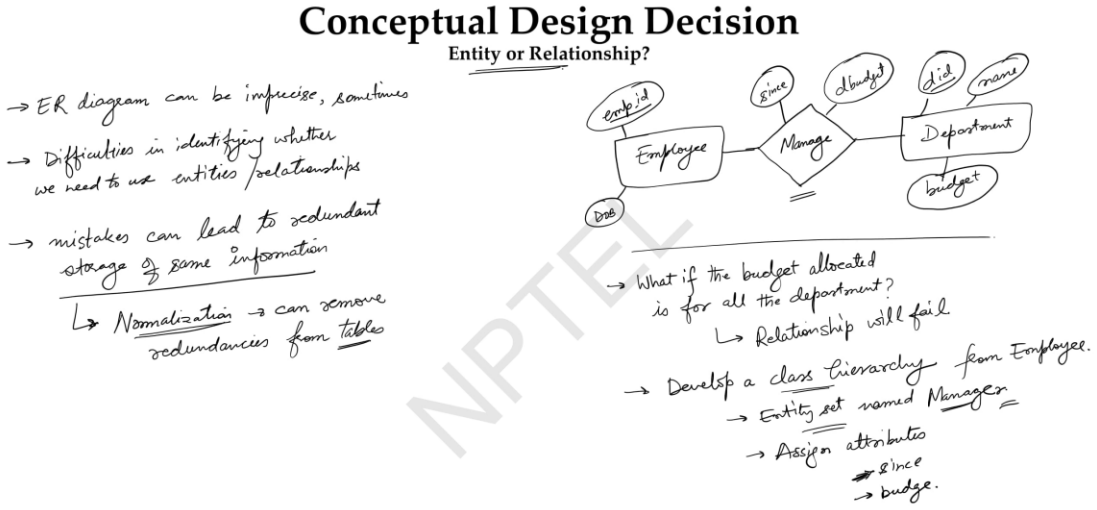
option then we need to further develop more sophisticated analysis tools using natural language processing to dissect the complete string of the data.

However if we are using the second option that is creating an entity set. Then the database will further optimize the search query and it will be very easy to identify and count the number of employees in a particular city that are living in. So these kinds of decisions will play a very important role while creating an efficient database schema, right. So, this can also be understood by using another by implementing this entity or attribute decision in a different way. So, let us say we have this year diagram and currently the worst relationship is having two descriptive attributes that is from and to.

Now this ER diagram can be read as that a particular employee from the employee entity set can work in a particular department. And these can be multiple departments and it can store that since when he or she started working in the department and until when they worked there. But this ER diagram cannot store when a particular employee is working multiple times with breaks in between. So that is why a good thing here is to instead of using these attributes from or to as the descriptive attributes of the relationship set we can develop an entity set named as duration. So we can recreate this ER diagram to have better representation is we can use employee connected with or related with works with another entity set for department.

Now instead of creating from or to we will create a new entity set named as duration and duration will have from and two as an attribute. So with this representation uh what we can further do with our schema is we can store the multiple times a particular employee worked in a particular department with breaks in between. So we can write this as let's say initially we have works relationship, that has two attributes. That's from and to has two attributes from to now these attributes can only capture duration of employee worked. But it is not useful,

when you need store multiple duration of a particular employee. So then we decided that we need to develop a new entity set here. So this is how you have a decision process while creating your conceptual design, right. So this is one decision entity or attribute. Now the next decision we can have is whether we can create an entity or a relationship.



Now the next decision is whether we can create an entity or relationship. So again, why this is important is because ER diagram can be imprecise sometimes sometimes it can be imprecise. So we can have difficulties in identifying whether we need to use entities or relationships. So these let's say actually we need to define a relationship between two different entities entity set but we actually modeled uh. It as a entity set so these kind of faults in your database schema could have serious implications while doing the optimization process of the database during in the database design process.

So that is why these mistakes can lead to redundant storage of same information. So, to remove this redundancy we have the objective mathematical process known as normalization which we talked earlier in brief normalization this can remove redundancies from tables. So once you have created your own schema and translated it into tables, we need to perform this important step that is normalization. And this step can identify whether we need to create entity or relationship in between. So to understand this, let us create another representation of the same employee management system.

So we have an employee. And we have a department employee will have an id it will have a date of birth and now we are trying to relate these two entity sets using a manage relationship. And this relationship will also have two attributes named as sense and discrete budget or the budget and department have the department id, which is a unique variable and it have a name and the budget. So, what is happening in this instance is, we

are saying with this representation is that employee entity set will have employees and one or more employees will be a manager of a particular department.

But each department will have at most one manager, right. And each manager will have their own budgets assigned to them. And they are working as a manager since a particular date and we are storing this as an descriptive attribute of the relationship side. Now let's say that a budget for each department is identified. And they are allotted to a particular department.

So the another possibility is that a manager could have a total budget for all the departments that they are managing or they have the budget for each department that they are managing. So these two different kinds of instances can have different interpretations while designing this representation.

So that is why we need to create a particular relationship between the budget and the manage. So we can say the question is what if the budget allocated is for all the departments. So if the budget is allocated for all the departments then how we will distinguish between the actual budget, so the relationship here will fail and we may need to create an entity.

So another option is to develop a class hierarchy which we have already discussed. And what it will do is class hierarchy from employee entity set it will create new entity set name manager. And we will assign attributes like since that are already assigned here already defined here since and budget.

So, Instead of creating this relationship, when we will develop a class hierarchy and create an entity set named manager, that could be more beneficial decision while creating the database schema. So these kinds of decisions are important and this is how we need to translate a real world problem into first into a database schema.

Then into the relational data tables. So, in the upcoming lectures we will try to translate it into different kinds of database schemas for each entity set and further we will start with the coding of these into relational tables.

Thank you.