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

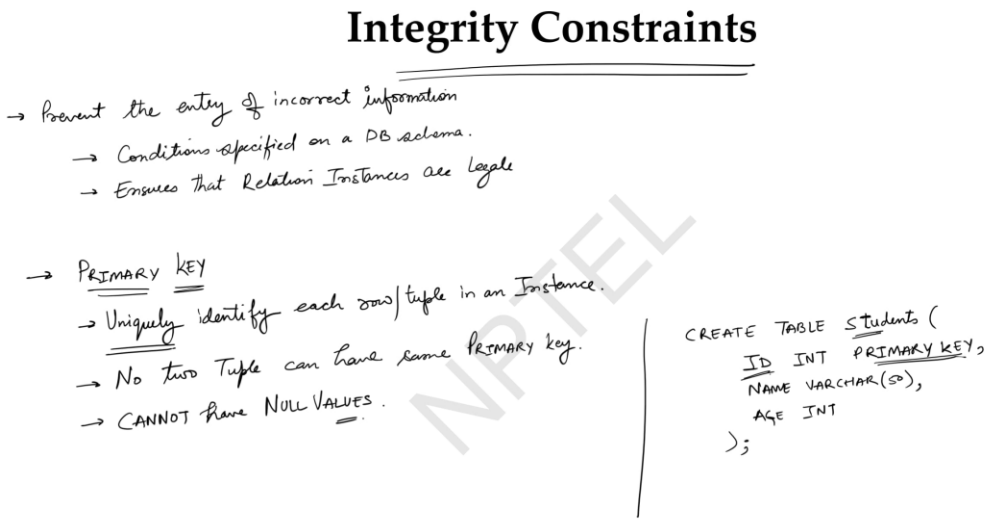
Indian Institute of Technology, Kanpur

Week 11

Lecture 49

Database Schema (Part-2)

So, let us now continue again with our discussion on the database schema and we have touched upon different kinds of commands in the SQL that is structured query languages. And now to connect two different relations with between them in a database, we need to first identify what are the different kinds of constraints that are possible.



So let us talk about the integrity constraints. So database is as good as it has the data. So if we have bad kind of data in a database table, then the database is of no use.

So these constraints, what they do is they try to prevent the entry of incorrect information, right. So how do they do is they try to specify conditions so conditions specified on a DB schema. They use those conditions to check whether the data we are filling is based on those conditions or not. And they ensures that all the tables or the instances relation instances are legal as per the definition of those schemas, right. So, let us first start with the constraint named as primary.

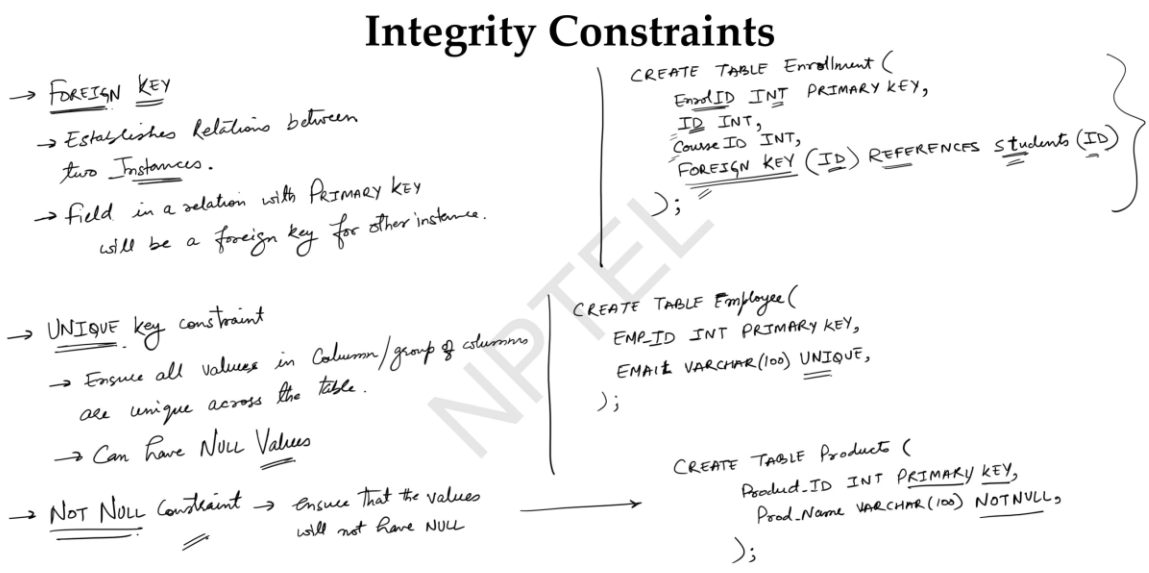
So, primary key. We have already discussed that how we are trying to define a primary key in a ER diagram using an underline, right. So, this is the same key. Now, we are translating it into the database table. So, what they do is they uniquely identify each row or tuple, whatever.

You can say and in an instance so the main thing is they uniquely identify the contents of a table. Now what constraint they provide they that no two tuple can have same primary key. So in a student database you can relate this with the id of the student that no two student will have the same id and another constraint that they inherently provide is that they cannot have null values. So a student in a school must have a student ID. It is not possible that a student is not having any ID.

So how to write the commands in the database to create a primary key? So let us take an example of student database only. So we can write create since we are creating a table. So create table. The name of the table let's say it is students now let's define the fields of this table so the first could be id that is the student id and the data type of the id is int that is integer.

And here we will try to define the constraint of the table. So we are trying to define ID as the primary key, so we should write primary key here. The next field could be name. This could be a string and we can specify a string using where char keyword and we can initialize it with the length of 50. Finally, let's say this table has an age attribute as well and this will be an integer, right.

So, this is how you will try to specify that this field of this table will be a primary key that is the unique identifier for each row in this table, right.



The next important constraint is foreign keys. So, what it does is it establishes relations between two different instances, two instances. Instances is nothing but the table in a database so to connect two different kinds of tables you will need to provide foreign key in those tables. So this is nothing but a field in a relation which is a primary key, with primary key constraint will be a foreign key for other instance. So let us go back to our example of creating students. So let's say a particular student in a school is enrolled in a course. Now we have course enrollment table as well. So each student should be related to that course.

So the primary key of this student must be mentioned in that course enrollment row as a foreign key for that table, right. So that is how we connect it. Now we can take it as an example. So if we are specifying the course enrollment, so we can define the table as create table. The name of the table that is an enrollment.

And we can write the enroll ID. This the course enroll id will be an integer. So this is the data type and this will be a primary key for this table so each course will have a course enrollment id and it cannot be null the second thing in this table that we are trying to store is the student id. So we defined our student id using the id keyword only. So we can write id this will also be an integer and then we can have course id.

This will also be an integer and now we define our foreign key constraint so to define it we will use the keyword foreign key. And we will use the name of the field that we are trying to use as a foreign key. So we will write id here and from where it is referenced, so we will use another keyword references students. This is the name of the table which is connected to this enrollment table, right, with id. And we can close this specification now what this complete command will do is it will create a field name with the name enrollment id which will be an integer and this is a primary key of this complete table.

So each row in this table will be having a unique number for this enrollment ID. The next is the student ID and this is coming from the table from the student table which we have already described. Now we can also have multiple attributes for this table. So let's say course id is another attribute finally to specify how this enrollment table is connected with the student table we will use this keyword foreign key. And we will specify which key in this current table is having that field with the foreign key.

So this is id and the other references keyword will try to specify which of the table in the database will have this key. So the DBMS will search for this table. Find the primary key of that table, matches it with this ID that is the field or attribute of this table and then specify that this is a foreign key. So this is all happens by the technologies available or the software capabilities of the database management systems. So the next key is the unique key constraint.

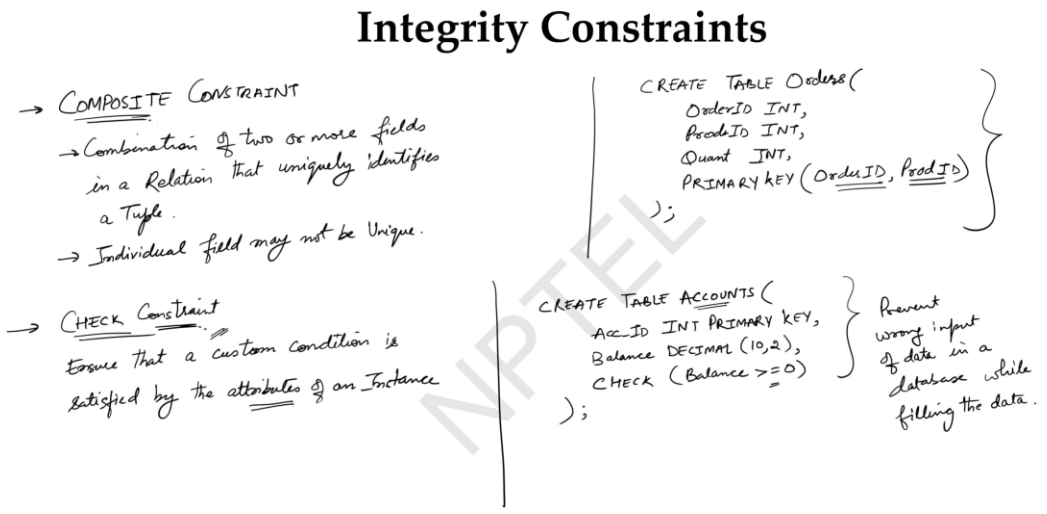
So, this constraint will ensure all values in a particular column, or group of columns across the table ,right. So the distinguishing factor between a unique key and a primary key is that a unique key can have null values right. Now an example of this is we can create an employees table. Create table with the name employee and it will have an EMP ID which will be an integer and it will be a primary key.

And we can have an email for this employee this will be a string. So we can use where care and now let us specify that it will be unique. So we can close this definition here. So this email will stay unique across the table, right. And it can have null values.

That is, an employee may have no emails assigned to them, right. But if an email is assigned, then across the column, the value of that email will stay unique in the table, right. So that is how we use this unique key constraint. The next is not null constraint. So as the name suggests, it will ensure that the values filled by the user will not have null.

So the user must specify the value for that particular column where we specify this constraint, right. So we can have a different example for this we can write create table let's say the name of the table is products and since it is a table about product we can have a product id. This will be an integer and a primary key for this table and the product name this will be a string so let's say we will use. Where char and now we can specify not null so in this table. If we are specifying a product table then the name of the product should not be null we must specify a name for each product, right.

And all these things we need to specify as a keyword, so we need to write these things in our console and we will see in the upcoming lectures. How we provide these commands to the console of the MariaDB database management system. And then we will see whether these are executing successfully or not.



The next constraint could be the composite constraint. So these constraints are useful. When we have a combination of two or more fields in a relation, that uniquely identifies a tuple or a record in the relation instance. So these are useful when let's say the individual field available in a database table may not be. So an example of this could be let us define a table create table with the name orders. And this will have order id as an integer product id with the data type integer quantity of the order with the data type integer.

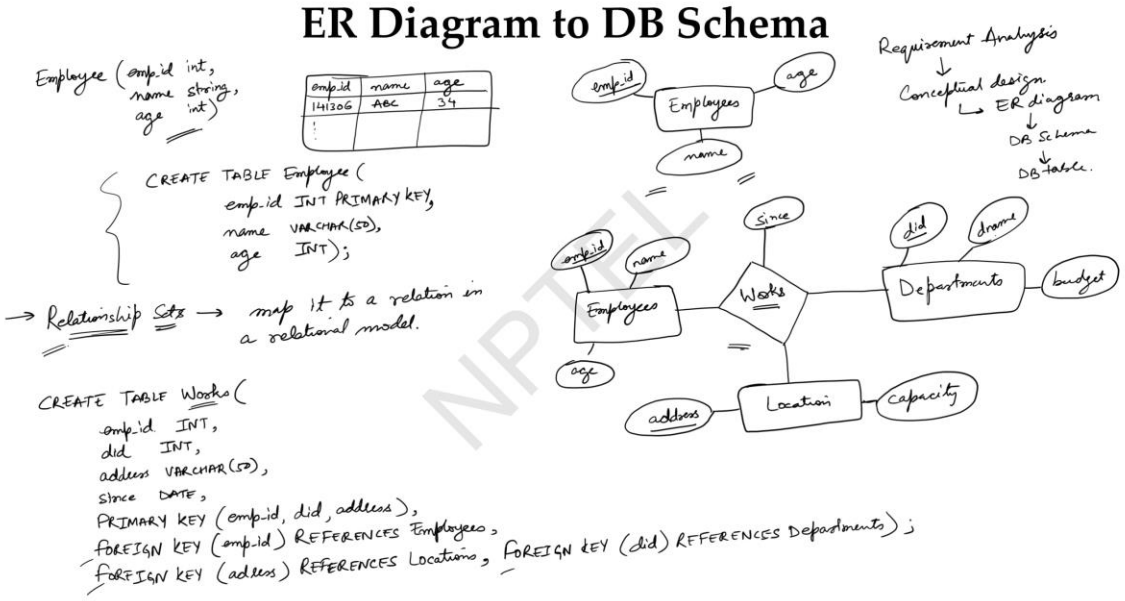
And now let us define our primary key a composite primary key using two fields of the same table namely order ID and product ID. So what this specification is telling us that in this table, each row will be uniquely identified by the combination of order ID and product ID. So if this kind of specification is provided to the database, then the uniqueness of a particular row is based on the composite values of these two fields. So, these are also available in all the RDBMS as a constraint. The next constraint could be check constraint.

So, what it does is it ensures that, a custom condition is satisfied by the attributes of an instance. So let's say in the student database management system, we want to check whether a particular student is legally valid for a particular class or not. So we may assign age as a particular class. Attribute of this table so that we can check whether he or she has passed a particular age let's say age of 15 to appear in a particular class.

So these kinds of constraints when the database users are filling the data in that database management system the DBMS will automatically check whether the particular row is following this check constraint or not. Specified on this table, so let us see how to specify this constraint so in a banking sector we can create this constraint easily using this table create table accounts, right. And we can have an account id this will be an integer. And the primary key the balance of this account this could be a decimal the specification of 10, 2. And finally we can specify our check constraint.

So the DBMS will check whether the balance is greater than equal to zero or not so this command will only fill those legal instances those legal tuples in the database table accounts where the balance is greater than equal to zero. If a user provides less than zero that is in negative then the DBMS will throw an exception or an error to the user and will ask whether the user is correctly filling the data or not.

So this way we can wrong input of data in a database while filling the data. So now we understood that there are different kinds of structured query languages syntax and what are the different kinds of integrity constraints. Now it is time to again switch back to how to translate from an entity relationship diagram to the actual database schema. And then create the database tables inside a database.



So let us again study how to translate these er diagrams to database schema. So the very most basic entity relationship diagram we studied in our previous lectures was the employees entity set so what was that this employees. They will have an employee id they will have an age and they will have a name so this is the er diagram now we need to translate it to a database schema. So this could be employee empid as integer name as string and age again as integer.

So this is a schema. Now we need to define a table. So when we try to define or create a table in the database, the DBMS will first try to define the fields of the table. So let us create a small table and it will have fields like EMP ID, then name and then age. So let's say the ID for the first employee could be 141306.

Name could be ABC and age could be 34, right. And this is how there are different kinds of employees. So to create this table from this database schema, we can write these SQL commands. So create table with the name employee. And they will have an empid which will be an integer and it will be a primary key.

So, we have not yet shown that it is a primary key in our ER diagram. So, we can just underline this, right. So, now it is a primary key and we can write name another field as this where care 50 and age as integer, okay. So this is how from the requirement analysis towards the conceptual design where we studied the ER diagrams. Then we tried to

develop the schema and so let us write here requirement analysis that this was the first step then we move to the conceptual design.

Where we developed the er diagram then we developed the DB schema and now we have created the DB table for this simplest example right. The next thing we know about er diagrams is that they have relationship sets. So :et us see how we can translate these relationship sets. So, what they do is they map it to a relation in a relational model. Let us draw our example of the participating constraint that we defined previously in our earlier lectures.

So it had employees as an entity set and departments As another entity set, right. And these two entity sets. Were connected with the works relationship. So to define a relation.

We use diamond as the diagrammatic representation. We can connect this. Now the attribute of this. The descriptive attribute of this relation is. Since when a particular employee is working in a particular department and employee usually will have an EMP ID and all other attributes we define like age and name.

Similarly departments will have a department ID which will be a primary department name. Budget of the department, right. And so we will have another entity set named as location which will store the addresses of each employee. So it will have address as the primary key and let's say it can have capacity as another attribute. So to translate this er diagram with a relationship named as works into a relationship table inside the database we can write this the following SQL command.

So it can be create since we are creating different kinds of tables. And we are specifying only the relationship, here this works. So we already understood how to create an entity set inside a database table. So we can define these employees and departments easily but now we have this relationship to define. So let us define this relationship with a table so we can write create table works.

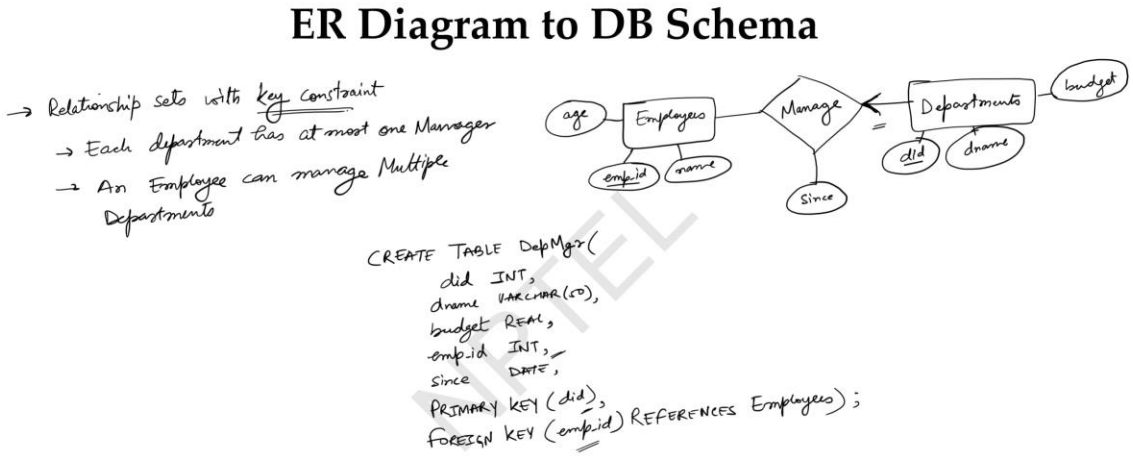
And it will have a emp id because an employee is working in a particular department. So we will use the emp id of the employee and let's say it will integer we can have the department id which will also be an integer. We can have address. Let's say it is a worker. And the descriptive attribute that is since.

So it will be a date time object. So let's say it is a date. Now, let us define our constraints here. So, EMP ID will be a primary key, DID is a primary key. So, primary key will be

EMP ID, DID and the address which is also a primary key for the location entity set, right.

The next is the connections that is how we are translating this relationship between three different entity sets or three different tables in the database. So now we actually utilize our foreign key. So foreign key will let us first define the foreign key as EMPID. And it will reference employees so references employees table the other foreign keys for this works table is address. It will reference location location table another foreign P for this table is DID.

And it references department right. So this way this works table will have three different foreign keys this one this one and this one. Because it is connected with three different tables and each row or each tuple in this works table is will have the information of all the three tables connected to it. So that we can understand that how a particular employee is connected with a particular department and the actual location of that employee where he or she is residing. So that is how we can translate this complex ER diagram to a table in a database.



Now let us talk about another kind of ER diagram which we already discussed and try to translate it. So we saw that there was a relationship of managing different kinds of

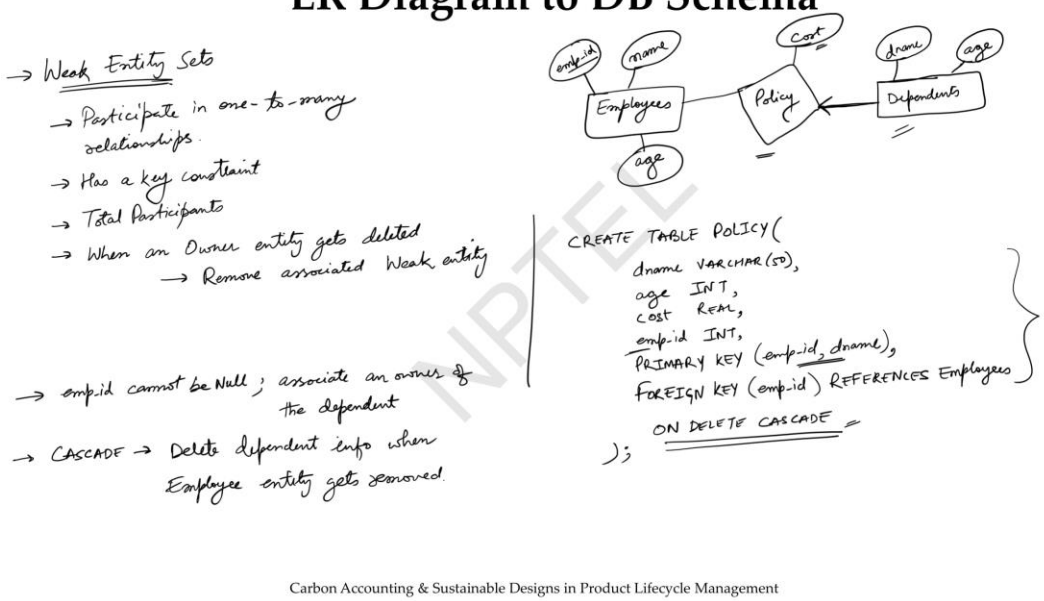
departments. So let us draw that ER Diagram. Employees and departments. And the relation we are discussing now is manage.

So employee will have an empid name and age. Department will have a D.I.D. A name Henry Bajat right. And we can have a descriptive attribute of since. So this will be a date kind of data where we are trying to specify that a particular employee is managing a particular department since when right.

So here we are trying to translate an ER Diagram with key constraints so relationship sets with key constraint. So what was the key constraint in this er diagram let us revise it each department has at most. One manager for department will have only one manager and an employee can manage multiple departments. Now, to show this key constraint, we define an arrow here in the ER diagram. So, now we need to translate this ER diagram into a database table.

So, we can write create table department manager. And we can specify the attributes of this table as did that will be an integer d name where care or string and budget that would be real the empid. This is an integer or this since descriptive attribute that will be date now the primary key here is the id only not this employee id and the foreign key. That is how it is related. So this is EMP ID which is referencing employees table, right.

So what this definition is saying is that since it is having a key constraint, the actual identifying information, uniquely identifying information is the department ID for a particular manager. So which department a particular manager with the EMP ID is managing is the actual primary. And this manage is also related with the foreign key using the employee ID. With the employee entity set or the employee table in the actual database, right.



Now let us talk about the last translation that is the how to translate weak entity sets so we talked about this weak entity set using the example of using employees database.

And how the dependence the information about dependence of an employee is stored in a particular database of employees management system. So let us redraw the entity relationship diagram for defining a weak entity set so start with the employees entity set it will have an id. This will be an EMP ID. Employee will have a name and an age. Now let us say we have a dependence table.

So each employee will have some dependents and the organization where this employee works is providing different kinds of policies for their dependents. So a policy, a relation, let's say define a relation policy and this relation is connected with these two entity sets, right? And dependents will have d name and age. Now to identify a particular dependent in the employee management system, we must have the owner employee with whom this dependent is related to. So that is why we have this.

We are specifying this as a weak entity set with full participation. And this will also have a policy as cost of the policy. As the descriptive attribute for this relationship right now. Let us jot down the various features of a weak entity set so these kinds of entity sets will participate in a one to many relationships. They will have a key constraint and they provide total participation.

We already talked about all these terms in our previous lectures. And the difference between this ER Diagram and all the previous diagrams which we have translated is that. Let's say if an owner entity or the actual employee with whom these dependents are related to when an owner entity gets deleted. That is let's say they switched the organization, so when they get deleted we should remove all the associated dependents or weak entities. So, these were the characteristics of weak entity sets.

Now, we need to translate these characteristics using our database SQL language. So, let us define here. Create, we need to define this relationship. We need to define this relationship. So we need to define a table for this so create table.

Let us say name policy with the name d name as a dependent name with where care 50 h integer cost this cost. This could be real the employee id it is a primary key. So we should underline it in ER Diagram employee id is integer and primary key is a composite key here. So we should use employee id and this d name so together these two informations will identify the unique rows or the unique dependents. And how they are connected is specified by foreign key.

Foreign key for this table is empid which is a primary key for employees entity set and it references employees references employees. And now we need to specify other keywords as well on delete cascade So until here, the definition was mostly the same, right. But with this keyword, what happens is whenever a particular employee leaves this employee entity set. All the information associated with that particular employee using his or her employee ID will be removed from the tables of the dependents as well automatically by the DBMS.

So, this keyword will specify that we need to perform all these tasks declaratively to the DBMS. So, other important information about this definition is that this EMP ID cannot be null. So now we can understand why we are trying to define it as a primary key in the employees table as well. Because we must have an employee unique ID so that we can associate the dependents with them. Employee ID cannot be null.

So this will associate an owner of the dependents. Further the cascade keyword will delete dependent info when employee entity gets removed. So with this, we complete a basic introduction towards the translation from an entity relationship diagram to the database schema. And how to specify those into database tables using structured query languages. In the upcoming lectures, we will try to install MariaDB on our system.

It could be Windows or Mac. And then we will try to understand what are the different kinds of database normalizations and we will try to specify them in the MariaDB itself.

Thank you.