

Hi.  
  
In this video we're going to discuss how to take the subtyping relationships from the enhanced entity relationship diagrams and represent those relationships adequately in our tables.  
  
So let's go and see one example.  
  
So I'm not going to go for the full diagram.  
  
I'm just looking at one particular subtyping example.  
  
And the example is from a university or.  
  
Yeah, from the university.  
  
We have an entity called person, which could be somebody in the university, basically.  
  
And now that a person can be either a member of staff or a student.  
  
So there is this disjunction between staff members and students.  
  
And in addition, staff members, there's a subtype which is called research assistant.  
  
So we have research assistant that are a particular type of staff members.  
  
We have staff and students, and the person is either a staff member or a student.  
  
So now we want to turn this into our tables, how we want to represent the situation in tables.  
  
How do we do this?  
  
There are three options.  
  
Basically, the first option is we just create separate tables for any entity that we have in the diagram.  
  
So we create a separate table for the super type and the subtype entities and we post the identifier from the super type to the subtype.  
  
Meaning we have a person entity which has, in this example a national insurance number.  
  
Then we have a staff which has a staff number and a student which has a student number.  
  
But then we copy the national insurance number to both staff and students to make sure that we know that this member of staff corresponds to this person, basically.  
  
And then finally we have also this relationship of research assistant, where the research assistant has just the staff number from the staff and the grant number, for example, for the grant they are working on.  
  
So the first option is like the basic one where we just introduce for each entity a new table.  
  
Advantage is that there are no null values.  
  
Disadvantage is we ended up with many tables, namely four tables.  
  
Another option could be option two.  
  
We create a single relation for each subtype and we take the attributes from the supertype to the subtypes.  
  
So this only works, of course, if the supertype is really a disjunction of the subtypes.  
  
So in this example it works because a person has to be either a student or a member of staff.  
  
In principle, there's no point of having the person entity represented in tables because we just represent the students in one table and the staff members in the other table and then we have all the persons.  
  
So the option here is just put the student information in one table and then all the information, including the things that we had previously in the person table and similarly put the stuff information in the other table, including that stuff from the person table, and then we ending up with two tables, student and stuff.  
  
So if we come to the research assistant, the research assistant is just a subtype of staff.  
  
So we deal with this in the same way as previously because we already copied down the staff number from staff and we have the grant number and that's it.  
  
The main difference here is really how we resolve this disjunction between students and staff in person.  
  
And so now again, we have no null values and we have slightly less tables, but we still have three tables.  
  
So how can we deal with less tables even?  
  
Well, that is the more radical options in terms of less tables.  
  
So the option here is just create a single relation for all of those entities, because after all, they are all persons, and then just post all the information about the various persons into each row.  
  
So basically we have now in each row all the information stored about a person.  
  
And so in a sense this is the option where we could say, okay, if we do this, then the whole subtyping idea was useless because now we ended up with one table anyway.  
  
So we have a person which has an I number, national insurance number, first name, last name and a student number.  
  
If it was a student and and then other attributes, or if it is a staff member, we have again, of course, a national insurance number, first name, last name.  
  
But then suddenly we don't have a student number because it's not a student.  
  
What do we do?  
  
We put null values.  
  
And in the end what we end up with is the big table with loads of null values for those attributes that do not apply for a given row.  
  
Because students and staff members will have different attributes, general staff members and research assistants will have different attributes.  
  
So whenever this differs, we will have null values.  
  
So the advantage now is we have one single table.  
  
The disadvantage is we have loads of null values.  
  
So it's a matter of taste.  
  
In a way.  
  
It also depends whether the sub entities have relationships to other entities, because then it's better to create separate tables and it also depends on how to query the data.  
  
If you often query all of those, if you often query for general persons, it's probably better to have them all in one table.  
  
If you normally write queries about members of staff or queries about students which are separate, then it's definitely better to split them into two tables.  
  
To summarise, maybe the entity relationship modelling provides an approach for generating this conceptual design of a database.  
  
To some extent, this data modelling is kind of a black art, so you always have to have some, but there's always some liberty in deciding exactly, for example, what's the degree of a relationship and what is the optionality and much more than of how to represent sub entities.  
  
Do we need the sub entity relationship?  
  
And then how exactly do we move into the tables?  
  
There are lots of small design questions you have to think about and there's often not a clear cut answer.  
  
But the goal should be always keep the numbers of tables small and the number of null values as well.  
  
Thanks for listening.  
  
See you next time.