

Hi.  
  
This video is going to be about the details of the relational data model.  
  
So we've discussed data models already and we discussed what's a data model?  
  
A data model, first of all, is a way of representing data.  
  
It specifies what are the operations that we can use to manipulate the data.  
  
And finally, we specify constraints on the data, kind of integrity constraints that make sure that whatever data we store is of the right shape.  
  
And today we are really going to discuss, first of all, again, what the structure is in which we store data and then what the constraints are that we put on the data.  
  
So concerning the representation of the data, we recall that everything is a relation.  
  
And so in the relational database, what we specify is what's called the relational schema.  
  
And so then we have a collection of tables that adheres to the schem.  
  
What is a schema?  
  
Well, a schema of a table is just the name of the table and the name of its attributes.  
  
And then an instance is just indeed a table that adheres to exactly the specification.  
  
So, for example, we would have a table schema called.  
  
We have a table film, which has title and release here as attributes, that is the schema, and then we have the concrete table adhering to that schema.  
  
Given a table, what are the properties?  
  
First of all, there's a notion of a degree of a table which says how many columns we have, basically.  
  
So if it's just two columns, we have a degree of two, and there's the cardinality of the table, which is the number of rows.  
  
So if it has four rows, then the cardinality is four.  
  
And important to note is that we are talking really about relations.  
  
So relations are sets.  
  
So that has two consequences.  
  
First of all, there is no repetition of rows.  
  
And second of all, there is no order on the rows, even though if you look at it, it looks like as if they are ordered.  
  
Also, I should say that for now, we will always use table and relation interchangeably.  
  
We are not going to make a distinction between the two.  
  
And as I said previously, we always have to make sure that every attribute is in a relation has a unique name.  
  
And later on we will also see that we not only want to specify the name, but we also want to specify what type of thing we are storing there.  
  
So we will see that we have to say this attribute holds a number, this attribute holds a string.  
  
But more about this later.  
  
So this is the rough structure of the data.  
  
We have tables with names and attribute names.  
  
And then later on we will see that each of these attributes has a type.  
  
So Everything looks very neat and kind of uniform.  
  
However, any representation of data that you will come across in your life has some issues.  
  
We've seen already in the past this hierarchical data model with the trees.  
  
You figure out that not everything is a tree.  
  
If you have the graphs, you will figure out that graphs are not suitable as well, because then you don't know how to access the data.  
  
In the case of the relational data model, we have the problem that everything that you store in a relation has to have exactly the same structure.  
  
But often you store information about, for example, students and PhD students and undergraduate students, let's say, and you will have attributes that will only apply to PhD students and you will have algebra that only applies to undergraduate students.  
  
Now, if you still want to store them all in one table, you will have to.  
  
Exactly.  
  
Then say this is not applicable.  
  
And for the non applicable, you just say we leave it unspecified.  
  
And leaving things unspecified in SQL is via null values.  
  
The other thing is that often you just don't know all the information, but you still want to put the information that you do know.  
  
Then you would just say, and those attributes I don't know.  
  
And again, I put null values.  
  
So null values are there for saying, this is a missing value, I just don't know it yet.  
  
Or you say for this row, this attribute doesn't apply, we leave it unspecified.  
  
In any case, we just put null.  
  
And this gives us a way of storing several things that look not exactly the same in one table.  
  
So null values give us a more flexible way of storing data in a table.  
  
Okay, so we've talked about the representation of the data.  
  
We've seen previously what the operations are.  
  
Let's move on to the constraints in the relational data model.  
  
Most fundamentally, this brings us to the notion of key.  
  
What is a key?  
  
Now, firstly, somehow, if we have a table, clearly one of the fundamental things we want to be able to do is we want to address rows in the table somehow, because a row, so a table somehow describes something like an employee student, it describes an object, and each row is an instance of this.  
  
And so we want to be able to point as certain instances.  
  
And the key is exactly this kind of pointer, or as we call it, identifier, that allows us to, to point at a row.  
  
Now, more generally, there's a notion of a super key.  
  
So super key is just a collection of attributes in a table which allows us to identify the row.  
  
So basically, if you have a table, in this case we have a table with three attributes.  
  
Film id, title and release year.  
  
If we know the value for filmid, for title and for release year, we know in which row we are.  
  
So clearly, if I take just all the attributes, then I have a super key because it's a collection of attributes that allows me to identify in which row we are.  
  
However, just saying all of the attributes is too much.  
  
We can in this case just do with a film id, as you see, because if we know the ID of the film, we already know which film we are talking about, we know in which row we are.  
  
So the ID determines already the title and the release here.  
  
So while the super key, for example, all the attributes allows us where we are, the key is something that is minimal with this property.  
  
So a key is a minimal super key.  
  
So in this case, film ID is a key.  
  
Sometimes it happens that there are more than one key of a relations.  
  
So it could be some film id.  
  
And if we have by chance a case where the title would be unique, then the title might be also a key.  
  
So if we have several keys, then we just call any key of the table a candidate key.  
  
So anything that is a key in the table is called a candidate key.  
  
And in the relational data model, what we always have to pick for a table is what is called the primary key.  
  
So the primary key is something that we say to SQL, this is the thing with which you should identify rows of the table.  
  
So it's chosen key for the table.  
  
And whenever we write a table, we write the schema of the table, the name of the attributes, we underline the primary key attribute.  
  
So indeed in this case we would write film.  
  
And then in brackets we name film ID title and release year, and then we underline the film ID to indicate that this is a primary key.  
  
And possibly, indeed the title here would be a candidate key.  
  
So the primary key of the table is also often referred to as the key, because we just say, okay, that is the key of the table.  
  
There's always a confusion between the key and a key.  
  
And so why are keys mentioned as a constraint?  
  
Because we started talking about constraints.  
  
Well, keys are mentioned in the constraints because as soon as you pick attributes or one attribute as the primary key of the table, then you are not allowed to have null values there, because SQL is supposed to identify the row using the values there.  
  
And if null means it's unknown, so we cannot really identify the rows there.  
  
So indeed.  
  
So the attributes of the key need to contain unique values for each row.  
  
And this is what we call an entity integrity constraint.  
  
So this is one of the key constraints, one of the constraints in the relational data model.  
  
In the next video, I'm going to discuss the second important one that concerns constraints between more than one table.  
  
Okay, thanks for listening.  
  
See you next video.