CS 240: Creating and Dropping Tables Transcript

[00:00:00] SQL can be used to create and drop tables.

[00:00:03] So this is an example of a statement for creating a table.

Start visual description. The professor demonstrates an example of a statement for creating a table in SQL. The screen shows the general syntax for creating a table, using the keywords "CREATE TABLE" followed by the name of the table. End visual description.

[00:00:07] So first of all, we'll look at this, uh this shows the general syntax.

[00:00:12] So to create a table, you use the keywords, create table followed by the name of the table that you want to create by convention. People often will put um SQL or yeah SQL, keywords in all caps.

[00:00:25] I'm not doing that here, but often you would see create table as all caps but that's not required.

[00:00:31] So we have a create table statement and this is the statement that creates the book table from the book club example that I used when I first introduced the relational model.

Start visual description. The professor demonstrates the "CREATE TABLE" statement that creates the book table from the book club example. The screen shows the specification of primary keys and whether attributes are nullable or not. End visual description.

[00:00:42] So we create the table, we can specify things like what are the primary keys.

[00:00:47] We can specify whether attributes are knowable or not.

[00:00:50] Um We'll talk about auto increment and foreign keys.

[00:00:54] So if we look in this create table statement, um we have specified all of the columns that are going to be included in this table.

[00:01:03] So we specify the name of the column followed by its data type and then we can specify that it's not null. You could specify null or not null.

Start visual description. The professor demonstrates specifying columns in the "CREATE TABLE" statement. The screen shows the name of the column followed by its data type and the "NOT NULL" constraint. End visual description.

[00:01:13] But if you just leave this off completely, then it's assumed to be null.

[00:01:16] Or in other words, you could, you could leave that column blank when you do an insert.

[00:01:20] But these columns are all specified as not null, which means they all have to be specified when a row is inserted, then we have primary key.

[00:01:28] So that's saying that this ID column is the primary key for the table.

[00:01:32] There are actually two ways to specify a primary key.

[00:01:35] So if we have just one attribute is the whole primary key, then we do it this way.

[00:01:40] But if you have a composite key where you have multiple attributes that make up the primary key, then we have a separate primary key clause and you're not seeing that here, but we'd have a separate row that would say primary key.

[00:01:52] And then in parentheses, it would have a comma separated list of the columns that make up the key, then we have auto increment. So, I need to talk about that a little bit.

[00:02:01] So often if you remember from the book club example, I talked about artificial primary keys.

Start visual description. The professor demonstrates the concept of auto increment for primary keys. The screen shows how different databases specify auto increment differently, such as "AUTO_INCREMENT" in MySQL and "AUTOINCREMENT" in SQLite. End visual description.

[00:02:08] So sometimes we have primary keys that really have no meaning outside of the relational model. They don't mean anything to people.

[00:02:15] Um out there in the real world, the keys are just there. So, we can do joins, for example, and so if you have that situation, then it's often useful to just have the ID E generate the primary key value for you.

[00:02:30] So you can do that by specifying auto increment.

[00:02:33] And unfortunately, this isn't completely standardized.

[00:02:35] So different databases um have different ways that you specify auto increment.

[00:02:39] So in my sequel, it's auto underscore increment or sorry. And yeah, in MySQL, it's auto underscore increment.

[00:02:48] And another database SQL light, for example, is just auto increment without the underscore.

[00:02:52] So you have to kind of know the exact syntax of auto increment for your database.

[00:02:58] But what this says is when a table is inserted and I'll show you how to do inserts in a, in a separate video.

[00:03:04] But when I'm sorry, when a, when a book is inserted, you don't specify the ID value even though it says not null because the database will supply that.

[00:03:15] And the way it does it is it just keeps an internal counter.

[00:03:17] And every time you insert a row, it just increments the counter and gives that row that you're inserting the next the next value.

[00:03:24] OK? So that's what auto increment does. And primary key then for the other columns, we are specifying that these three are bar char and we're saying that they're not null.

[00:03:35] And for this last one, we're saying it's a, it's an integer column that's not null.

[00:03:41] OK? You can also see here that we have foreign keys, foreign key constraints.

[00:03:47] These are optional but really useful.

[00:03:51] So you can, you can have a database with that has tables and you can do joins without having foreign key constraints.

[00:03:59] So what this does, it doesn't say anything about how the different tables are joined in the queries when you learn how to do those.

[00:04:06] Um what it really says that tells the database that this is a foreign key. So, then the database can help you by making sure that, that we maintain what's called referential integrity.

[00:04:18] So let me talk about that a little bit.

[00:04:20] So referential integrity means that if you have a row in one table that refers to a row in another table, that row exists in the other table.

Start visual description. The professor demonstrates the concept of referential integrity. The screen shows an example of a foreign key constraint ensuring that a row in one table refers to an existing row in another table, preventing orphaned rows. End visual description.

[00:04:32] Um Another way to say that is, it ensures that we don't end up with orphaned rows.

[00:04:38] So let me give you an example of that. Let's say that we have um some table that has a bunch of rows in it.

[00:04:44] And then we have another table that is using one of those rows as a foreign key.

[00:04:50] So maybe I can use that in the con or explain that in the context of the book club.

[00:04:54] So we have um the table or we have the book table that can have rows in it. Now, once we execute this SQL and that references the genre table.

[00:05:06] So if you remember the genre table specifies whether a book is fiction or nonfiction or historical fiction or whatever, whatever it is.

[00:05:15] Um So imagine now that we have our genre table that has some rows in it, and we create some books, and we specify which, which uh genres the different books are for.

[00:05:28] But then if we go into the genre table and delete one of the genres.

[00:05:31] So if we create a fiction book, for example, we create a, a fiction book row and then we delete the genre um the row in the genre table. Now we have orphaned these rows.

[00:05:43] We've orphaned all of the books because they, they now have a foreign key that doesn't actually refer to a primary key in some other table.

[00:05:49] That's an orphaned row.

[00:05:51] If you don't specify foreign key constraints, the database doesn't know what the foreign keys are. And so, it can't help you.

[00:05:59] Uh It won't prevent you from creating orphaned rows.

[00:06:02] But if you do tell the database what the foreign keys are, then if you try to do what I just described where you have some books that are referencing a row on the genre table and then you delete that row on the genre table, that delete will fail if you've specified a foreign key constraint.

[00:06:19] So the database will actually make the SQL fail for that and you won't end up orphaning the row.

[00:06:25] Another thing that you can do to cause an orphaned row is you could update the row.

[00:06:29] So you could go into the category table and if you change the spelling of one of the categories, it's not going to match what's in the book table anymore.

[00:06:38] So that's another way to create an orphaned row and the database can help you avoid that with foreign key constraints.

[00:06:45] So I'm going to come back to this in a minute, but let's go talk more about foreign key constraints. So, this creates a foreign key constraint with or two foreign key constraints with some default settings.

[00:06:57] So what this says is the genre column in this table is a foreign key that references the genre column in the genre table.

[00:07:07] And this says the category ID column right here is a foreign key that references the category or the ID column in the category table.

[00:07:17] In other words, um this is a foreign key, and this is the primary key that it references, this is a foreign key in this table, and this is the primary key that it references.

[00:07:26] And by default, we have something called um delete restrict update cascade.

[00:07:33] And what that says is if I tried to, to delete the reference row in the other table, the database would restrict that it will disallow it, but it will allow me to update that other row.

[00:07:45] But if I update it, it will also update the rows that were affected in the referencing table. So, if I go into, let's say that I, I have my genre table and I noticed that I misspelled fiction I got the I MS C reversed or something like that, I'd probably want to go in and change that.

[00:08:02] And if I have my foreign key set up to cascade updates, then I will be allowed to go and change that primary key in the genre table.

[00:08:11] And then the database will automatically update every row that references it.

[00:08:15] So every row in the book table will be changed as well.

[00:08:19] So they still reference that foreign key so that can be really useful to do.

[00:08:22] So here are some more details about foreign key constraints.

[00:08:26] First of all, I want you to be really clear that they're not required, you can query and access your data without them, you can do joins without them.

[00:08:33] So they don't actually have anything to do with joints.

[00:08:36] And what they do is they enforce that the values used as foreign keys actually exist in their parent tables.

[00:08:43] Um and I've already talked about this.

[00:08:45] So let's um look now at the different options you have.

[00:08:48] So in the previous slide, we created default foreign keys, and this is the def this is what a default foreign key looks like.

[00:08:56] So it's uh it's set to cascade updates and restrict deletes.

[00:09:02] But if you, if you write it out in long form, you can specify something different for both updates and deletes.

[00:09:09] So here are the different options you have; you can say no action so you could make it. So, um maybe um you have no action on update, but you restrict, deletes, it's probably not very useful to do that.

[00:09:22] But you can do that. You can specify that either one of these are any of these options.

[00:09:27] So no action means that part of the foreign key constraint is, is basically turned off.

[00:09:33] If you say restrict, then the thing that you would do is disallowed.

[00:09:37] So delete is disallowed if it would cause an orphaned row, you could say that updates are disallowed by saying update restrict, you can say set null. So, you can make it. So, if you do either an update or a delete the referencing value in the in the other table gets set to null.

[00:09:55] So that could be a reasonable thing to do in certain cases.

[00:09:59] Another thing that you can do is columns, and I don't show that in this example, but columns can have default values.

[00:10:05] So you could say the word default followed by some value.

[00:10:09] And that would mean that if I don't specify the value, the default value gets inserted. When I do an insert.

[00:10:15] If you have a default value, then you can make the foreign key constraints set things to their default values if they need to, to prevent orphan rows and then cascade, you've already seen that that's the default for update.

[00:10:28] You could also make delete cascade, but you probably want to be careful with that because if you have all the, all the tables kind of referencing each other through foreign key constraints.

[00:10:38] You could end up in a situation where you delete one row, and it deletes your entire database or things from lots of your tables.

[00:10:45] So it's not necessarily the wrong thing to do.

[00:10:47] You just need to understand what it does and be careful with that.

[00:10:51] OK.

[00:10:52] So back to this other example, one thing that can be useful and this will be particularly useful in your, in your assignment that you're doing in your, in your chess project, you can make it. So, creating a table happens optionally.

[00:11:09] So if you try to do create table book and the table already exists, that SQL statement will fail.

[00:11:16] So if you have a script that's trying to execute a bunch of statements, if any one statement in that script fails, the script stops executing.

[00:11:23] So it could be useful to say I want to create the table if it doesn't exist.

[00:11:27] But if it does exist, I just want to not create the table and I don't want that to be considered an SQL error.

[00:11:33] So that'll, that'll be useful.

[00:11:35] And we'll talk about that in a video where we describe the different, the different um phases of your project.

[00:11:43] But um just be aware that that's one thing that you can do, you can make it so that create statement will that statement won't fail.

[00:11:51] If the table already exists, it just won't be executed.

[00:11:55] OK? Last thing I want to show you here is how to drop tables. That's pretty easy to do.

[00:12:00] So you can just specify drop table book and that will just wipe the table out.

Start visual description. The professor demonstrates how to drop tables in SQL. The screen shows the "DROP TABLE" statement and the "DROP TABLE IF EXISTS" statement to avoid SQL errors when the table does not exist. End visual description.

[00:12:06] So you want to be careful with that.

[00:12:07] Um SQL assumes you knew what you were, what you were talking about when you asked it to do something.

[00:12:13] So it will just do it if you ask for it.

[00:12:16] Um Again, with scripts, it can be really useful to say to use the statement drop table if exists book, what that will do is it will um if, if I just do drop table book, then that will work.

[00:12:31] If the book table exists, it will drop it.

[00:12:33] But if the book table doesn't exist, this will generate an SQL error.

[00:12:37] Um Again, when I'm using scripts, if I have a script file and I want to generate my entire database, it can be really useful to drop table if exists at the top of your script and drop all the tables.

[00:12:50] So let me kind of describe that scenario.

[00:12:52] So imagine a case where you have a, a file and I'll show you this in a later video of a file that generates an entire database.

[00:13:01] But if you have a file that generates an entire database, the first time you run that you're probably going to find an SQL error somewhere.

[00:13:08] So imagine a file that has a bunch of create table statements.

[00:13:11] So maybe it creates two or three tables and then on one table somewhere in the middle of your script, it fails because you had an error.

[00:13:19] So what are you going to do? You're going to go in and fix your error and then you're going to want to rerun your script.

[00:13:24] But when you rerun your script, now the first statement is going to fail because that first create statement is can't execute because there's already a table by that name.

[00:13:34] So what is useful in a situation like that where you want to want to run a bunch of create table statements in a script is just say drop table that exists at the top.

[00:13:45] And that means every time you find an error and have to correct, correct it, you can just rerun your script file, and it will drop any tables that it created the last time.

[00:13:54] And since you have drop table if exists, if there are any tables that it didn't get to, so it didn't create them because you had an error. Last time, those drops table statements won't be in error. So, your script will keep running.

[00:14:06] So that just allows you to go in and change or fix your script as you find errors and then be able to rerun it the next time without needing to manually delete anything to, to be able to rerun your script.