CRUD (Create Read Update Delete)

CRUD stands for Create, Read, Update, Delete. We are going to be entering a different way to interact with the database with the information we learn here. So far we have really only been interacting with the data which is the R or Read part of this acronym. Read would refer to the SELECT statements that we have been writing to retrieve data that we need out of the database. We are now going to start looking at getting the data into the database in some way (or creating data). We will also look at how we can edit that data or delete data. So we will focus on modifying the database in some way. The create, update and delete types of commands would be only run by admins or users you trust highly

There are many different types of Structured Query Language Commands and we won't learn all of them. But it's good to know that we can do a lot of different things with our data. If you look at the DDL (data definition language) commands and DML (data manipulation language) commands we will be learning more of these commands in this lesson. DCL (Data Control Language) mainly deals with granting rights and permissions and other controls in the database and TCL (transaction control language) deals with transaction within the database to prevent errors when important transactions take place. Again, not everyone should have access to the commands we will be using in this lesson. The Insert, Update, and Delete commands can cause a lot of damage to a database.

As a user who just needs access to data, the SELECT statements can be used to find the data they need. Here a SELECT statement would run that finds all artwork that Leonardo da Vinci created that is in our database. But no data will be added, edited or deleted. We don't want to give that kind of access and power to our end-users.

The manager of the data (or admin) could have access to do these things. They might want to add more artists or artwork to their database or to edit or delete current data. This would be more of the backend of the system that not everyone has access to. You might see a login required to get to this portion of the system. Sometimes this is referred to as the Content Management System (or CMS).

Once logged into the back-end CMS the admin could then edit, delete or add new data to the database. This lesson will cover the different commands that might be used on the back-end of a system. These commands should really only be used by someone that really knows the database and it is always a good idea to be backing up the database so that it can restored in case certain commands mess up your database. So commands like entire deleting tables or deleting the entire database wouldn't even be something that would be included in the back-end CMS. The database administrator would take care of actions like those.

Here's a look at some commands that can be used to create a new database (or schema) and then creating a table in that database.

This command CREATE DATABASE mdb; will create a movie database called mdb with no tables yet.

The next statement would create a table in the mdb database. The table is named actors and the table has 3 columns: actor_id as the primary key, last_name, and first_name. Of course realistically there would be many more columns but I have simplified it greatly. We can see that all columns will require a value from the NOT NULL keyword we added (meaning we will require a value to be entered in that column) and that the primary key actor_id will be auto_incremented since we used the AUTO_INCREMENT keyword.

Here's is what a tool such as Workbench might be displaying for that table.

And when you show everything from that table we can see that it is created with the 3 columns but there is not data in the table yet. So at this point we have created a database with one empty table.

We can use the INSERT statement to put data into our actors table. Here we see 3 rows being added to the table all in one command. We could have used separate commands for each row as well. Also, we could have listed out the column names before the keyword VALUES but since all our values are required we can simply make sure there is a value for each column of the table. We do have to insert the values in the exact same order as the columns in the table. The values also have to match the data type that we defined for the table. For example, actor_id must be an integer and last and first name must be characters (we have put those character values inside single quotes to show they are characters). Notice how instead of a number for the actor_id we have entered the keyword DEFAULT. We could have used an integer here as long as it was one that had not been used yet (and therefore would be unique as primary key) but you can also just enter DEFAULT or NULL and MySQL will automatically enter a unique value for the row. That is because we set up actor_id to be auto-increment. So in this case that gave each actor the unique keys of 1-3 for those 3 actors that were inserted into the table when this ran for the first time.

UPDATE also known as edit will alter your data in tables. This can be a very dangerous command to run because there is no undo button to get your original information back if you mess up. Some tools will not let you update because they will have a Safe Mode setting that restricts UPDATE and DELETE statements since these can be very harmful to the database if not used correctly. You might have to turn safe mode off to run some of these commands.

You should always have a WHERE clause with UPDATE or all the rows will be updated.

Here’s an example of an UPDATE statement that will change the first name to ‘Tom” that used to be ‘Tommy”. If the WHERE clause had been left out then all 3 actors would now have the first name “Tom”. It wouldn’t have been that big of a deal for us to change back the two names, but what if you had thousands or millions of rows of data? Can you see how dangerous an UPDATE could be in the wrong hands.

One way to check that you have the right row to update would be to run a SELECT statement with the same WHERE clause to see if the data you expected comes up first and then run the UPDATE clause after. Because the select statement uses the same WHERE you will get a result set of what row will be effected before in this case all actors with the actor_id of 1 and you can see that it’s Tommy Hanks and know that it’s the right record to change. Then you can run the update and be assured it will only edit that one row.

DELETE can also be a very dangerous command if not used properly. And again, your systems may have a Safe Mode setting that restricts DELETE statements just like it did UPDATE.

Just like UPDATE you should have a WHERE clause with DELETE unless you want every row to be deleted.

The DELETE keyword is used with the FROM keyword to indicate the table. The WHERE clause is needed or else all actors would have been deleted from the table. If it had stated just DELETE FROM actors; and then was run. All rows of data would have been deleted and you would be left with an empty table. Again can you imagine the kind of damage this could do to a database with many records. It could literally ruin a company. So again, be very careful with the delete command.

Again it’s always a good idea to run a SELECT statement first to make sure you’ve selected the right row before you run the DELETE statement.

The DROP statement will delete all the data from a table and delete the definition of the table so it is no longer there. The table will be deleted completely with no way to get it back. unless you’ve first made a copy of the table or database.

This DROP command deletes the entire actors table.

This DROP command deletes the entire mdb database.