Database concepts (CBD)

Class 3

MySQL (suite)

Creating database elements

Up to now, we've been working on tables that already existed and were imported, but it is possible to create your own elements such as tables. Although, take a moment to familiarize yourself with existing tables as there are a few important details to consider.

Content type

As you already know, each entry is composed of values from different categories (fields), and each of these categories store a specific type of data. For instance, ID will usually use numbers so its type would be integer, name would be a simple string, etc. Most of these content type will be filled automatically by the application.

Although, the **"Null"** column is important. This field is a constraint, a require condition. If the value is "YES", the user must supply content, if the value is "NO", the field can be left empty.

Also, the **"Extra"** column is where you may add *AUTO_INCREMENT* for the automatic generation of a primary key number.

Creating a new table

To create a table, it is first needed to use the *CREATE* command, followed by the *TABLE* command, to specify the name of the new table, and then to specify the columns name between parenthesis. After writing the query, hit the GO button and the new table should show in the left panel.

```
CREATE TABLE `comments` (
        comment_id INT NOT NULL AUTO_INCREMENT,
        comment_title VARCHAR(255) NOT NULL,
        comment_description TEXT(1000) NOT NULL,
        users_id INT NOT NULL,                  // To link to specific user
        product_id INT NOT NULL,                // To link to users table
        PRIMARY KEY (comment_id)
)
```

Explanation of the code:

Deleting a table

To delete a table, the command DROP will be used followed by the name of the table to be deleted.

```
DROP TABLE `table's name`
```

Adding a field to a table

We already have seen how it is possible to modify a table's data and how to add entries. We are now going to see how to modify a table. To do so, the ALTER command will be used, followed by TABLE, the name of the table to modify and one of the following commands : ADD, DROP or MODIFY. It is also important to specify the data type and constraint if any.

```
ALTER TABLE `table's name` ADD `new_column` VARCHAR(255);
```

Modifying a table's field

Just like we did for adding a field, we will now use the ALTER command to modify a table's field data type from VARCHAR(255) to TEXT(1000).

```
ALTER TABLE `table's name` MODIFY `new_column` TEXT NOT NULL;
```

Deleting a table's field

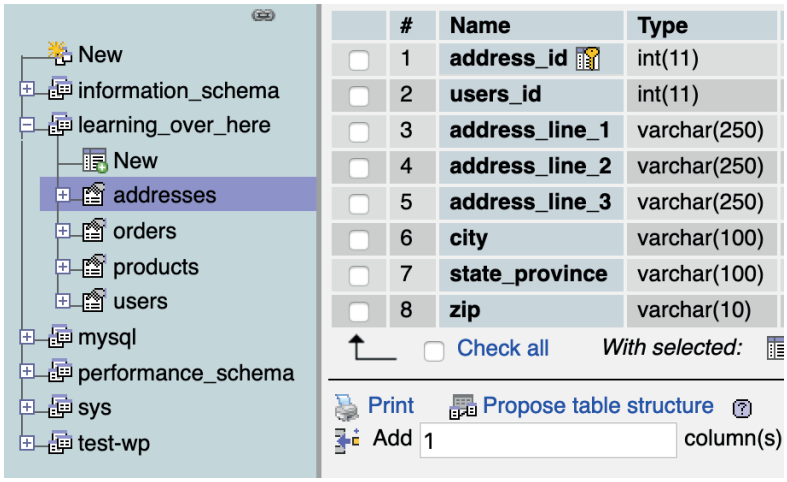
As you can guess, it is also possible to delete a table's field using the command DROP.

```
ALTER TABLE `table's name` DROP `new_column`;
```

Linking tables

In the preceding lesson, we added a field which related to information contained in another table. For instance, when creating a comments table, we want for the comment to be linked to a specific user. Instead of repeating the information, we can manage to have this information automatically set using what is called a *foreign key*.

Foreign keys



If you look at the addresses table above, You will first notice the primary key located in **field #1** (*address_id*) followed by another primary key in **field #2** (*users_id*) linking the table to the users field of the users table. This actually link the users and addresses tables so each users are linked to their corresponding addresses.

```
SELECT users.`username`, users.`first_name`, users.`last_name`, addresses.`state_prov-
ince` FROM addresses JOIN users ON addresses.`users_id` = `users_id`
```

users

#	Name	Type
1	id	int(11)
2	username	varchar(150)
3	password	varchar(35)
4	user_email	varchar(150)
5	first_name	varchar(150)
6	last_name	varchar(150)

adresses

#	Name	Type
1	address_id	int(11)
2	users_id	int(11)
3	address_line_1	varchar(250)
4	address_line_2	varchar(250)
5	address_line_3	varchar(250)
6	city	varchar(100)
7	state_province	varchar(100)
8	zip	varchar(10)

Explanation :

SELECT is used to select information from the specified fields. **FROM** i used to indicate from which initial table to search from. **JOIN** (or INNER JOIN/LEFT JOIN/RIGHT JOIN /FULL JOIN) is used to indicate the second table that is linked to the first one. **ON** is used to tell to retrieve information of entries in which these two fields match.

Types of JOIN

(INNER) JOIN
Returns records that have matching values in both tables

LEFT (OUTER) JOIN
Returns all records from the left table, and the matched records from the right table

RIGHT (OUTER) JOIN
Returns all records from the right table, and the matched records from the left table

FULL (OUTER) JOIN
Returns all records when there is a match in either left or right table