COMP353 Project #1
Warm-up Project

Mair Elbaz, François David, Daniel Vigny-Pau,
Alexandre Therrien, Charles-Antoine Guité

Creating the Tables

Below are the lines of code that we have put in our program to create the various tables required for this project.

```
//query the database with the command in (). Creates table 'people'
  $mysqli->query("CREATE TABLE people (id smallint unsigned not null auto_increment, lastname varchar(20) not null, firstname varchar(20) not null, middle_name varchar(20), userID int not null, password int not null, constraint pk_people primary key (id) );");
```

This code queries our database to create a new table with the name ‘people’. That is, assuming you are already connected to your database and that you have selected a database to use. The $mysqli variable is the variable used to store the connection to our database. The characters in blue compose the query.

We are to create a table that has a variable id, which is never to be attributed the value null (from the constraint “not null”), that is always positive (“unsigned”), and that increments automatically (“auto_increment”) so that there are never two same id values. It is also a small int value, which means it has a smaller range than an int value. The last statement of this query (constraint pk_people primary key (id)) indicates that this variable will be used as the primary key.

The rest of the statements between commas are giving the different columns this new table will have. The varchar type is a type of data element that contains a series of characters. We have the ‘lastname’ element, which has the constraint to be not null and a maximum length of 20. We have the ‘firstname’ element, which has the constraint to be not null and a maximum length of 20. We have the ‘middle_name’ element, which has a maximum length of 20 and can be null.

Then we have two int values, userID and password. These are integer types, both of which have the constraint to be not null.

```
//query the database with the command in (). Creates table 'event'
  $mysqli->query("CREATE TABLE event (id smallint unsigned not null auto_increment, Event varchar(20), EventID int, start_date varchar(20), end_date varchar(20), AdminUserID int, constraint pk_event primary key (id));");
```

Similar to the previous table, this statement creates a table called ‘event’ and has the same primary key statements as in the previous example.

It creates the different columns Event (varchar of maximum length 20 that can be null), EventID (integer value that can be null), start_date (varchar of maximum length 20 that can be null), end_date (varchar of maximum length 20 that can be null), and AdminUserID (integer value that can be null).

```
//query the database with the command in (). Creates table 'role_of_people_in_the_event'
  $mysqli->query("CREATE TABLE role_of_people_in_the_event (id smallint unsigned not null auto_increment, userid int, EventID int, constraint pk_role primary key (id));");
```

Similar to the previous table, this statement creates a table called ‘role_of_people_in_the_event’ and has the same primary key statements as in the previous example.

This table has a userid column (integer, could be null) and an EventID column (integer, not null).

Prepare and Read Input Data

To start reading the input data, some operations had to be done before.

```php
//open the file
$fn = fopen("db19s-P1.csv", "r");
//read line. go over line of +---+
$result = fgets($fn);
//read line. go over line of headers of table
$result = fgets($fn);
//get to the first line of data to put in our database
$result = fgets($fn);
```

First, I needed to open the file in php with read authorization. Then, I placed my $result variable to the first line of data of the first table.

```php
//checks if the substring from index 0-1 is a + (indicates end of table)
while(substr($result, 0, 1) !== '+'){
  //separate the line of data using the pipe |
  $results = explode("|",$result);
  //then query database to add the data into the table. If we get an error, print it.
  if($mysqli->query("INSERT INTO people (id, lastname, firstname, middle_name, userID, password)
  VALUES (null, '".$results[1]."', '".$results[2]."', '".$results[3]."', ".$results[4].", ".$results[5].")")){
    echo '<p>'.$mysqli->error.'</p>';
  }
  //print the data being put in the table people.
  echo '<p>'.$results[0].' '.$results[1].' '.$results[2].' '.$results[3].'</p>';
  echo '<p>'.$results[4].' '.$results[5].'</p>';
  //go to the next line of data.
  $result = fgets($fn);

}
```

Then, I am doing a loop until I read a line starting with the + value (which is the line that separates the different tables from one another). To read all the different data to go in the different columns of the table, I separate the line of data received by using the pipe (|) character, which is what separates those values. Because the line starts with a pipe character, the first array element we are interested in is the index 1, as index 0 will always be an empty string.

I follow that with an if condition that does the query and verifies that the output is not FALSE, which would indicate that there is an error in the query. If there is an error, I would print the error in the browser. This query indicates that we want to add some data to the table ‘people’ in the following order: (id, lastname, firstname, middle_name, userID, password). The INSERT INTO query adds that data into the table ‘people’. Then, we use VALUES (…) with the values in the same order as they were mentioned before.

Then the results of my separation using the pipe character are printed to the screen, and I follow that by going into the next line of the file doing $result = fgets($fn);

```
//pass through next lines -- the + and the column title lines
$result = fgets($fn);
$result = fgets($fn);

//verify if the line starts with a +
while(substr($result, 0, 1) !== '+'){
  //separate the line with |
  $results = explode("|", $result);
  //ensure there is no error from the query. Print the error otherwise.
  if($mysqli->query("INSERT INTO event (id, Event, EventID, start_date, end_date, AdminUserID)
  VALUES (null, '".$results[1]."', ".$results[2].", '".$results[3]."', '".$results[4]."', ".$results[5].")")
  === FALSE){
    echo '<p>'.$mysqli->error.'</p>';
  }
  //go to the next line
  $result = fgets($fn);
}
```

Next, once we have reached a line starting with a + (which separates the tables), we exit the loop from the previous image. We then go over that + line and the column titles' line using fgets, and we enter a new loop which is very similar to the previous one. The only difference is that the results are not getting printed and the query has been adapted to add the data into the 'event' table. This time I put the first result from the split as Event, then EventID, start_date, end_date, and finally AdminUserID.

I then go to the next line.

```
//pass through the next lines -- the + and the column title lines
$result = fgets($fn);
$result = fgets($fn);

//ensure the line does not start with a +
while(substr($result, 0, 1) !== '+'){
  //separate the line with |
  $results = explode("|", $result);
  //ensure no error in query. Print error otherwise.
  if($mysqli->query("INSERT INTO role_of_people_in_the_event (id, userid, EventID)
  VALUES (null, ".$results[1].", ".$results[2].")") === FALSE){
    echo '<p>'.$mysqli->error.'</p>';
  }

  //go to the next line.
  $result = fgets($fn);
}
```

Same process here. Once we reach the line starting with + separating the tables, we exit the loop in the previous image and enter this one after going over the + and the column titles' lines. This time I add the content of the split into the 'role_of_people_in_the_event'.