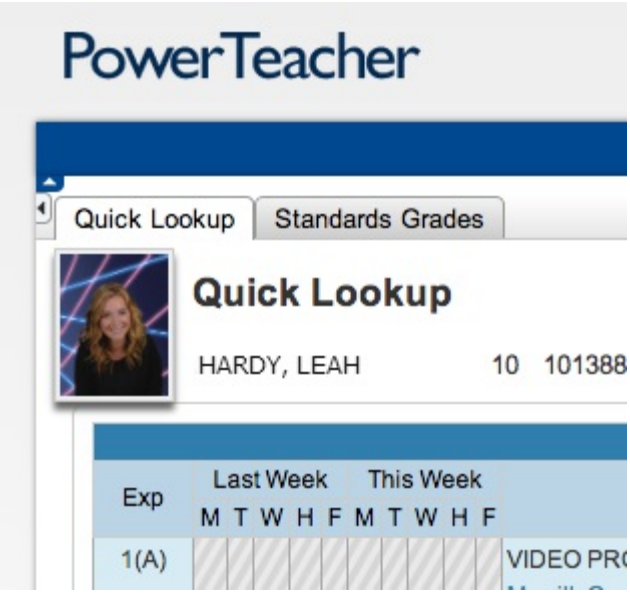
PLTW COMPUTER SCIENCE

Activity 2.2.3

Databases and SQL

Introduction



Data has become a prominent part of almost all businesses in recent years. An online gradebook company could create a database that would automatically send out D and F reports at the mid-quarter mark by cross-referencing its records of student grades with its records of those students' parents' contact information. A clothing company could keep records of what you purchase and use those to suggest items you might be interested in through a bulk email system. None of this data tracking would be possible on the massive scale that is most useful in business without having a good way to store, manage, and retrieve that data.

Database Management Systems (DBMS) are software suites specifically designed to handle the management of data. In previous activities you've already begun to use MySQL®, which is an example of a DBMS.

Materials

- Computer with browser
- Cloud9 workspace with MySQL credentials

Resources

2.2.3 sourceFiles.zip

MySQLFAQ

Procedure

Part I: Create and Query a MySQL Database

To develop an understanding of how a DBMS works, you will make a website with content based on data gathered from users. For this lesson we've chosen to use MySQL, a common tool for managing databases. In this part you will learn some of the basics of MySQL by creating and

manipulating database tables on your Cloud9 workspace. In the next project, you will return to the High School Art Gallery code from the previous activity with the ability to enhance it significantly based on your learning in this activity. You may find it useful to use the MySQL website as a resource to help you answer some questions in this activity.

1. Form pairs as directed by your teacher.
2. Meet or greet your partner to practice professional skills. During this activity, you will collaborate to create
3. Open a Cloud9 workspace. Each student will be responsible for creating a database in their own workspace, but partners will collaborate to achieve a working database in each workspace.
4. At the command line, type `mysql-ctl start` to start MySQL.
5. The prompt will change to `mysql>`. At the prompt, type `mysql -u c9user -p`, where `c9user` is replaced by your Cloud9 user name.
6. When prompted, enter your MySQL password. This signs you in to your MySQL account.

MySQL is a programming language and has its own set of rules and syntax. Some of the important ones for you to know are:

- All commands issued in MySQL must end with a semicolon.
- Pressing the Enter key before the semicolon results in a multi-line command and can be used to make code easier to read.
- If you make a mistake while typing a command, you can always cancel that command by typing `\c` and pressing enter. This is different than using Ctrl-C as you may have in the past to kill processes.
- MySQL commands are not case-sensitive, but the convention is to issue commands with uppercase letters.
- Commands are case-insensitive, but convention is to capitalize keywords. Table and column names are singular and lowercase by convention because case-sensitivity will otherwise require a lot of quotation marks.

6. Type the following command and record the names of the databases that appear.

```
mysql> SHOW DATABASES;
```

You do not currently have any data in your database (the one named with your student account name). To add some data to that database, you need to tell MySQL to use that database. The `USE` command does this. Here is an example of this command: `USE database;` where `database` is the name of the database you want to use. In this case, that is “shoes”.

7. Enter the command to use the shoes database:

```
mysql> USE shoes;
```

MySQL is used for working with **relational databases**, databases in which sometables of data relate data from one table to data from another table. A user or program can execute a **query** to isolate a **derived data set**. A derived data set is a subset of the data contained in a database created by filtering the database through commands like the ones we explore in later steps within this activity. To demonstrate the relational properties of databases in MySQL, you will create a set of three related tables for a shoe company.

In the image above, model numbers are assigned to shoes to reflect many of the non-unique properties of those shoes. Together, those properties form one unique identifying string of characters. The most central idea in working with relational databases is the concept of a **primary key**. Primary keys are extremely useful, because they allow us to fetch a specific row from a table. Among all the rows in a given table, any given primary key must appear at most once. For this reason sequential integers are often chosen as the primary key and are used in this activity.

8. Enter the command that follows to create a table in your database. In this tablethe model_id of a shoe serves as the primary key. Each model_id is correlated with its size, its color, and style.

```
mysql> CREATE TABLE shoe (
  -> model_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  -> size VARCHAR(4),
  -> color VARCHAR(128),
  -> style VARCHAR(128),
  -> PRIMARY KEY (model_id));
```

9. To verify that you entered your information correctly in the previous step, type `DESCRIBE shoe;`. You should see the following. If there are any errors, consult the MySQLFAQ resource for help correcting them.

```
mysql> describe shoe;
+----------+------------------+------+-----+---------+----------------
| Field    | Type             | Null | Key | Default | Extra
+----------+------------------+------+-----+---------+----------------
```

```
| model_id | int(10) unsigned | NO   | PRI | NULL    | auto_increment
| size     | varchar(4)       | YES  |     | NULL    |
| color    | varchar(128)     | YES  |     | NULL    |
| style    | varchar(128)     | YES  |     | NULL    |
+----------+------------------+------+-----+---------+---------------
```

The code that you just typed on lines 2, 3, 4, and 5 each define a field in the table. They also specify the type of data. For example, line 3 tells the table to contain a field called `size` that contains at most four characters, using the form `VARCHAR(number)`. Varchar is a creation type used by MySQL that creates a variable-length string up to the maximum length given by the number.

Answer the following questions:

1. Which of the fields created above has the longest possible maximum string?
2. Which has the shortest maximum string?

Line 2 contains some additional specifications for the `model_id` field. `NOT NULL` means that no cell in that column can be empty. `AUTO_INCREMENT` specifies that the values for that column are provided automatically by MySQL every time a new row is added to the table and that those values are sequential integers. These specifications make `model_id` an ideal candidate to be the primary key for the table, and it has been set as such on Line 6.

10. Use the Internet to help you find three other types of data that can be used to define fields in MySQL, including at least one type that you recognize from another language.

Imagine that our shoe company distributes its shoes to a variety of stores. When a store wants a particular model of shoe, they send a request to the shoe company. The request is stored with a unique request number in a separate table. This table includes the request number, the store's identification number, and the model that the store requested.

Enter the following code to create this table.

```
mysql> CREATE TABLE request (
  ->request_id INT,
  ->store_id INT,
  ->model_id VARCHAR(16));
```

Verify the correctness of your table with `DESCRIBE request;`. It should look as follows. If there are any errors, consult **2.2.3.Aa MySQLFAQ.docx** for help correcting them.

```
mysql> describe request;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| request_id | int(11)     | YES  |     | NULL    |       |
| store_id   | int(11)     | YES  |     | NULL    |       |
| model_id   | varchar(16) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Given that any store could make multiple requests and different stores could request the same model, what should the primary key be for this table?

You may have noticed that you did not define a primary key for the `request` table. Follow these steps to set the primary key for your table.

- Insert your answer from the previous step into the following command where `youranswer` appears, to set that field to be used as the primary key.

```
mysql> ALTER TABLE request ADD PRIMARY KEY (youranswer);
```

- Make that field auto-increment by issuing the following command, again replacing `youranswer` with your answer.

```
mysql> ALTER TABLE request MODIFY youranswer
  -> INT UNSIGNED AUTO_INCREMENT;
```

It makes sense to create one last table for this database, one that includes information about the stores identified by `store` in the `request` table.

Use the `CREATE` command as you have in the previous two steps to make a table named `store_info` that contains each of the following fields of the specified type.

Field Name	Type
store_id	INT
store_name	VARCHAR (128)
city	VARCHAR (128)

The first field contains the store identification number as in the previous table. The second contains the name of that store and the third contains the name of the city in which the store is located. When you are finished, your table should look like the one below.

```
mysql> describe store_info;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| store_id   | int(11)      | YES  |     | NULL    |       |
| store_name | varchar(128) | YES  |     | NULL    |       |
| city       | varchar(128) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

As you did for the `request` table in step 13, alter the `store_info` table to create a primary key that auto_increments. Your table should now look like:

```
mysql> describe store_info;
+------------+-----------------+------+-----+---------+----------------+
```

```
| Field      | Type              | Null | Key | Default | Extra          |
+------------+-------------------+------+-----+---------+----------------+
| store_id   | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| store_name | varchar(128)      | YES  |     | NULL    |                |
| city       | varchar(128)      | YES  |     | NULL    |                |
+------------+-------------------+------+-----+---------+----------------+
```

If there are any errors, consult **2.2.3.Aa MySQLFAQ.docx** for help correcting them. The `Field` names in a `DESCRIBE` command's output are the same as the column names of the table. We use these column names to access information in our tables using the `SELECT` command.

Why should `store_id` be the primary key in this table?

The tables have been created, but they contain no data. Now use the `INSERT` command to create a row in your table. This command is formatted as follows:

```
INSERT INTO tablename(field1, field2, …, fieldn)
VALUES(value1, value2, …, valuen);
```

To create a row for a new model of `shoe` in the shoe table, enter the following command.

```
mysql> INSERT INTO shoe(size, color, style)
  -> VALUES('6.5', 'red', 'heel');
```

The columns in the table represent the fields we created earlier. Note that we do not specify a model number, as MySQL provides a sequential integer for us because of the commands in Step 8, Line 2.

Add two more models to the `shoe` table.

You now perform your first query of the database. As you saw in the last activity, a query is used to extract data or rows from a database.

Type `SELECT * FROM shoe;`. The `*` in the command is a **wildcard** and, as in a Google search query, means to match any string. What does MySQL display? Make a note of the output here.

model_id	size	color	style

Why do the `model_id`s each have a value even though you did not assign any values?

Now use the same techniques as in steps 17 and 18 to add data to the `store_info` table. Follow these criteria as you add store data:

- - Create at least two stores with the same name.
- - Create at least one store with a different name.
- - Create at least two stores in the same city.
- - Create at least one store in a different city.

Use the same techniques as in the previous steps to add data to the `request` table. Follow these criteria as you add request data:

- - Use store numbers from the `store_info` table.
- - Use model numbers from the `shoe` table.
- - Include at least two requests from the same store.
- - Include at least two requests for the same model shoe.

This command allows us to create one new table from several others based on criteria that we provide.

Enter this example which creates one table based on all of the models of shoe that have been requested and all of the stores that have requested them.

```
mysql> SELECT model_id,size,color,style,store_id,store_name,city
  -> FROM store_info NATURAL JOIN request NATURAL JOIN shoe;
```

Note that a model that has not been requested does not appear in the table. Similarly stores that have not made requests do not appear in the table.

Let's say the owner of our shoe company wants to find out what cities are currently requesting a particular model of shoe. They can now use the relational property of the database to produce a derived data set.

Execute the following command once for each model_id in your `shoe` table by replacing `model_number` with each model number.

```
mysql> SELECT model_id,city
  -> FROM store_info NATURAL JOIN request
  -> NATURAL JOIN shoe WHERE model_id=modelnumber;
```

Which query or queries returned results and why?

Step 23 used a query to derive a specific data set: the cities where a given model of shoe was requested.

Choose two of the following data sets to derive and record the query that you would use to derive that data. Derived data set A should be the easiest, with C being the most difficult.

- Find the model numbers of all blue shoes.
- Given a request number, find the city where that shoe was requested.
- Given a shoe size, list all the cities where that size shoe was requested.

The queries that you tested and created in steps 23 and 24 could be used to create a wide variety of solutions to other problems. This is one of the reasons that DBMS are so popular.

Retailers are not the only ones with a use for DBMS. For example, a query similar to the one outlined in Step 23 could be used by a site like Facebook. When you log in to your account on a social media site, a query is made to its database using your `user_id` to retrieve the names of your friends by joining the table that contains user names and `user_id`s with the table that contains pairings of `user_id`s representing "friend" status. Your news feed could be created by joining the previous table with the table containing posts and timestamps with user_id as its primary key.

Brainstorm with your partner to decide upon some context, other than the shoe company and the social media site, where a similar algorithm could be used and explain the differences and similarities.

Part II: Automate the Database Query Process

As you've seen in previous lessons, the real power of computer science is often found in the automation of processes. In the last part of this activity, you had to write down some data at intermediate steps to issue the next queries appropriately. In this part we automate the shoe company problem posed earlier using PHP. We focus as closely as possible on the details of the interaction between PHP and MySQL to help you better understand the High School Art Gallery site that you will modify and extend in the next project.

27. If you cloned your Cloud9 workspace from a git repository, you already have the file **223index.php** in a folder for 223. Otherwise, create a folder 223 in your Cloud9 workspace, download the resource source files and upload them to the 223 folder in your Cloud9 workspace.
28. Launch Apache if it is not already running.
29. Using your browser, navigate to the new index file on the server. If your Cloud9 username is `justme` and your workspace is `myc9workspace`, then the URL for the page will look like this:

    `https://myc9workspace-justme.cloud9users.io/223/223index.php`

    What do you see on this page?

30. Open the file **223index.php** in your text editor and read through the code and comments.

    This PHP document executes an algorithm to return the same results that we were able to get in step 23 of this activity with just a single query. This fact exhibits the power of a relational database.

    Answer the following questions.

    - On what lines are query strings stored?
    - What role does `$number_of_requests` serve?
    - What does `$request` represent?
    - What would the statement in line 29 do if you changed the `$store[2]` to `$store[1]`?

31. In line 13 modify the code so that instead of `model_id=1`, the model number is one of the other `model_id`s that you added in step 19
32. Save the file and then refresh the browser tab in which you are viewing **223index.php**.

You should see the cities in which stores have requested that shoe model.

33. Open **223indexAlt.php** which is a script that does the same thing as the last one but with only one query to the database.

    Which lines from **223index.php** now appear unnecessary?

34. Modify the code in either **223index.php** (less complex MySQL) or **223indexAlt.php** (less complex PHP) so that it automates the work you did in step 24.

    When you are done, describe the role of PHP and MySQL in generating this web page.

Part III: Normalize a Database

35. Thinking back to the High School Art Gallery site, now that you know more about how relational databases work, what additional metadata fields might you want to add to the images table shown below? Look at the code from 2.2.2 if necessary.

```
+-----------+------------------+------+-----+---------+-------+
| Field     | Type             | Null | Key | Default | Extra |
+-----------+------------------+------+-----+---------+-------+
| username  | varchar(128)     | YES  |     | NULL    |       |
| filename  | varchar(128)     | YES  |     | NULL    |       |
| thumbname | varchar(128)     | YES  |     | NULL    |       |
+-----------+------------------+------+-----+---------+-------+
```

36. Choose one of the fields that you thought of in the previous step and discuss with your partner what alterations would need to be made to the website code to add that field. Remember to consider all languages involved. Document your discussion here.

    Relational databases are most useful if they are **normalized**. A normalized database follows the rules below, described by Edward Codd in 1971. Following these rules can decrease memory usage and allow faster queries to the database. The rules also help you avoid errors when updating your database because any changes will only need to be made in one place.

    Rules for a normalized database:

    - Every table has a primary key. Usually each table has a column named after the thing that each row of the table describes, concatenating with `_id`
    - No two columns in a table should contain the same kind of data. Having a column for a second artist in the`image` table to account for the possibility of collaborative works of art would fail this requirement. Instead, use a relational table to connect two of something to one of something. For the art database, separate the data about the artist from the data about their work, and have a **relational table** that relates artists to work:
        - Each row of the `image` table has a unique `image_id`
        - Each row of the `artist` table has a unique `artist_id`
        - Each row of the `contribution` table has a unique `contribution_id`. If artist 1 and artist 2 collaborated to create image 1, the contributions table might look like this:

`contribution`

contribution_id (primary)	image_id	artist_id
100	1	1
101	1	2

- Columns should contain only one value. For example, a version of the `artist` table where `firstname` and `lastname` were combined into one field called `name` would fail this requirement.
- Tables should only contain information about the concept numbered by the primary key. A database about students can have a table about students and a table about parents, but data about parents should not go in a table that has data about students. A relational table can match students to parents using the student_id and the parent_id.

37. Describe what needs to be fixed in each of the tables to use the table in a normalized relational database.
    -

first_name	last_name
John	Mohammed
Abdul	Mohammed
Rashida	Smith
John	Smith

    -

vet_customer_id	pet_name	second_pet_name
1	Fluffy	Kelly
2	Spike	Mischief & Crocadelio

    -

composer	song
Swift	Love Story
Zimmerman and Swire	Ghosts 'n' Stuff

request_id	store_id	store_name	city
1	2	Payless	Albuquerque
2	1	TJ's	Santa Fe
3	2	Payless	Albuquerque

38. With your partner, design a table structure for the High School Art Gallery site project that is normalized and includes three to six new fields. Describe your final design.
39. Does the High School Art Gallery site use the MVC pattern? Describe what elements of this website are like what parts of the MVC pattern and which are not (if any). For more information on MVC see *1.5.3 MVCTkinter.pptx*
40. Use the Internet to help you write a paragraph comparing and contrasting relational database management systems with **NoSQL**, a non-relational DBMS.
41. Record the name of one local business or company that likely uses databases and explain how it might use them.

Conclusion Questions

1. What are the advantages of normalizing a database?
2. Describe a way that a school might use a relational database.
3. What additional functionality do you achieve by pairing PHP with MySQL as opposed to using MySQL from the command line?

2.2.3.Aa MySQLFAQ

If you have made any mistakes in 2.2.3.A step 6 or later, you may need to use one of the following `ALTER` commands in order to fix the table. Each command will be described, followed by the general form of the command and then a specific example:

- - To change the name of a column, use the following format:
  `ALTER TABLE tablename CHANGE columnname newcolumnname DATATYPE;` For example, if we had mistyped the name of the model column as “mofel”, we could correct that by issuing

```
mysql> ALTER TABLE shoes CHANGE mofel model VARCHAR(16);
```

- - To remove a column, use the following format:
  `ALTER TABLE tablename DROP columnname;` For example, if we had added in a column called “treads” that we didn't want, we would issue

```
mysql> ALTER TABLE shoes DROP treads;
```

- To modify the data type of a column, use the following format:
`ALTER TABLE tablename MODIFY columnname DATATYPE;` For example, if we had mistyped the data type of the model column, we could correct that by issuing

```
mysql> ALTER TABLE shoes MODIFY model VARCHAR(16);
```

- - To add a column, use the following format:
  `ALTER TABLE tablename ADD columnname DATATYPE;` For example, if we had forgotten to add the price column, we could correct that by issuing

```
mysql> ALTER TABLE shoes ADD price VARCHAR(16);
```

- If we forgot to add one of the columns before `price` and we wanted to preserve the ordering, we would need to drop the columns following the one that we missed, add that one in, and then add all the others back in sequentially.

- - To rename the table, use the following format:
  `ALTER TABLE tablename RENAME newtablename;` For example, if we had accidentally entered “shoe” as the name of the table, we could correct that by issuing

```
mysql> ALTER TABLE shoe RENAME shoes;
```

- - There is another command to delete tables; however, it has been disabled for your student account.

In order to make changes to individual rows within a table, you will use the `UPDATE` command. The format for changing the value of a field or fields within a table for one or more rows is `UPDATE tablename SET field1='value1' WHERE field2='value2';` . For example, changing the price of your first model of shoe might look like the following:

```
mysql> UPDATE shoes SET price='200.00'
  -> WHERE model='A123-1';
```

Again, there is a command to remove rows, but it has been disabled for your student account.

Advanced: You can set a particular field within a table to be recognizable by MySQL as being the primary key with `ALTER TABLE tablename ADD PRIMARY KEY(fieldname);`