

|  |
| --- |
| **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. |  |

Equipment

* Computer with:
  + Internet Access
  + Firefox with FireFTP and FireSSH Addons installed
  + Notepad++ (Windows), or TextWrangler (Mac)
* Schoolwide VPS with accounts created for all CS students; students granted access to teacher MySQL database

Procedure

1. Form pairs as directed by your teacher.
2. Meet or greet your partner to practice professional skills.

**Part I: MySQL Basics, a Shoe Store**

In order 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 the VPS. 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. Use FireSSH to log in to your account on the VPS as in previous activities. At the command line type mysql -p. When prompted, enter your MySQL password (it should be the same as your VPS password unless you have changed it). This signs you in to your MySQL account.
2. 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 upper-case letters.
   * Names are case-sensitive in the Linux environment you are using.

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

mysql> SHOW databases;

1. You do not currently have any data in your database (the one named with your student account name). In order 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 accountname; where accountname is your account name. Type that command and your information, then press enter.
2. MySQL is used for working with **relational databases**, databases in which many different tables of data have some fields (columns) in common. These databases allow a user or program to query using those relationships to isolate desired data, referred to as 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 shown above, model numbers have been 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.

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | mysql> CREATE TABLE shoes (  -> model\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT,  -> size VARCHAR(4),  -> color VARCHAR(128),  -> style VARCHAR(128),  -> PRIMARY KEY (model\_id)); |

Now type DESCRIBE shoes; to verify that you entered your information correctly in the previous step. You should see the following. If there are any errors, consult **2.2.3.Aa MySQLFAQ.docx** for help correcting them.

mysql> DESCRIBE shoes; +----------+------------------+------+-----+---------+----------------+

| 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 | |

+----------+------------------+------+-----+---------+----------------+

1. The code that you just typed in 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 4 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:
2. Which of the fields created above has the longest possible maximum string?
3. Which has the shortest maximum string?
4. Line 2 contains some additional specifications for the model\_id field. NOT NULL means that no cell in that column can be empty and 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.
5. Use the Internet to help you list 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.

1. 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.

|  |  |
| --- | --- |
| 1  2  3  4 | mysql> CREATE TABLE requests (  -> request INT,  -> store INT,  -> model VARCHAR(16)); |

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

mysql> DESCRIBE requests;

+------------+-------------+------+-----+---------+-------+

| 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 for this table be?

1. Follow these steps to set the primary key for your table.
2. Insert your answer from the previous step into the following command where youranswer appears in order to set that field to be used as the primary key.

mysql> ALTER TABLE requests ADD PRIMARY KEY (youranswer);

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

mysql> ALTER TABLE requests MODIFY youranswer

-> INT UNSIGNED AUTO\_INCREMENT;

1. It makes sense to create one last table for this database, one that includes information about the stores identified by store in the requests 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  storename  city | INT  VARCHAR(128)  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 that store is located in. When you are finished, your table should look like the one shown below. If there are any errors, consult **2.2.3.Aa MySQLFAQ.docx** for help correcting them. Note that the Field names given in the left column of a DESCRIBE command’s output are the column headers when we begin accessing information in our tables using SELECT.

mysql> DESCRIBE store\_info; +-----------+------------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-----------+------------------+------+-----+---------+----------------+

| store\_id | int(10) unsigned | NO | PRI | NULL | auto\_increment |

| storename | varchar(128) | YES | | NULL | |

| city | varchar(128) | YES | | NULL | |

+-----------+------------------+------+-----+---------+----------------+

Why should store\_id be the primary key in this table?

1. 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 shoes table, enter the following command. 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 6, Line 2.

mysql> INSERT INTO shoes(size, color, style)

-> VALUES('6.5', 'red', 'heel');

Add two more models to the shoes table. Make sure that the model numbers you give them are distinct from any other models already in the table. Write down the data that you used for each field in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| model\_id | size | color | style |
|  |  |  |  |
|  |  |  |  |

1. 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 shoes;. The \* in the command is a wild-card and, as in a Google search query, means to match any string. What does MySQL display?
2. Now use the same techniques as in Step 13 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

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

* Use store numbers from the store\_info table
* Use model numbers from the shoes table
* Include at least two requests from the same store
* Include at least two requests for the same model shoe

1. To demonstrate the relational property of this database we use JOIN commands. 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 on all of the stores that have requested them. 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.

mysql> SELECT model\_id,size,color,style,store\_id,storename,city

-> FROM store\_info NATURAL JOIN requests NATURAL JOIN shoes;

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 number in your shoes table by replacing modelnumber with each model number. Which query or queries returned results and why?

mysql> SELECT model\_id,city

-> FROM store\_info NATURAL JOIN requests

-> NATURAL JOIN shoes WHERE model\_id=modelnumber;

1. Step 17 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.
   1. Find the model numbers of all blue shoes
   2. Given a request number, find the recommended sale price of that shoe
   3. Given a shoe size, list all the cities where that size shoe was requested
2. The queries that you tested and created in Steps 17 and 18 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 17 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 number to retrieve the names of your friends by joining the table that contains user names and user ID numbers with the table that contains pairings of user ID numbers representing “friend” status. Your news feed could be created by joining the previous table with the table containing posts and timestamps with user ID number 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: Automating 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 in order to issue the next queries appropriately. In this part, we automate the shoe company problem posed in Step 14 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.

1. Obtain the source code for Activity 2.2.3 as directed by your instructor.
2. Modify the file **223login.php** as you did in the previous activity, inserting your own database name on line 3 (again, this should be the same as your user name).
3. Upload **223login.php** to the server using FireFTP as you did in the previous activity.
4. Upload **223index.php** to the VPS.
5. Using your browser, navigate to the new index file on the server. If your student name was bkia8cp, and your school subdomain was samplehs, then your URL would look like this: http://samplehs.pltwcs.org/students/bkia8cp/223index.php . What do you see on this page?
6. 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 17 of this activity with just a single query. This fact exhibits the power of a relational database. Answer the following questions.
7. On what lines are query strings stored?
8. What role does $number\_of\_requests serve?
9. What does $request represent?
10. What would the statement in line 29 do if you changed the $store[2] to $store[1]?
11. In line 13, modify the code so that instead of model=1, the model number is one that you added in Step 13. Save the file, upload it using FTP, 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. 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?
12. 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 18. When you are done, describe the role of PHP and MySQL in generating this web page.

**Part III: Practical Databases**

1. 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 | |

+-----------+--------------+------+-----+---------+-------+

1. 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 in order to add that field. Remember to consider all languages involved. Document your discussion here.
2. In the real world relational databases are often most useful if they are normalized. **Normalization** is the process of deciding where all of your data needs to go, into which tables, and with which fields being used as primary keys. Edgar Codd described three hierarchical sets of rules for normalization in a 1971 paper. The rules were called the First, Second, and Third Normal Forms. Normalization following these rules can lead to decreased memory usage as well as faster queries to the database. It can also help you avoid errors when you need to update your database because any changes will only need to be made in one place.

The **First Normal Form** consists of three requirements:

1. No two columns in a table should contain the same kind of data. Having a column for a second artist in the images table to account for the possibility of collaborative works of art would fail this requirement. Instead we might choose to have a separate table that has one row for every artist for every piece of work they contributed to, with fields for username and filename .
2. Columns should contain only one value, for example a version of the artists table where firstname and lastname were combined into one field called name would fail this requirement.
3. Each table should have a primary key. In the solution proposed in requirement 1 above, we would need to add a third column with some unique identifier to the new table in order to meet this requirement.

Next to each of the tables below write the number(s) of the requirement(s) of the First Normal Form that it fails to meet and explain your reasoning.

A.

|  |  |
| --- | --- |
| firstname | lastname |
| John | Mohammed |
| Abdul | Mohammed |
| Rashida | Smith |
| John | Smith |

B.

|  |  |  |
| --- | --- | --- |
| Vet customer number | Pet 1 | Pet 2 |
| 1 | Fluffy | Kelly |
| 2 | Spike | Mischief & Crocadelio |

C.

|  |  |
| --- | --- |
| Composer | Song |
| Swift | Love Story |
| Zimmerman and Swire | Ghosts ‘n’ Stuff |

1. To achieve the **Second Normal Form** you must already have a database that meets First Normal Form requirements and then remove columns that contain duplicate data to separate tables leaving only the necessary columns. It is all about getting rid of repeated information.

For example, the following table would violate second normal form because the store name and city are repeated every time a given store ID number appears.

|  |  |  |  |
| --- | --- | --- | --- |
| request\_id | store\_id | Storename | city |
| 1 | 2 | Payless | Albuquerque |
| 2 | 1 | TJ’s | Santa Fe |
| 3 | 2 | Payless | Albuquerque |

To align this database with Second Normal Form we make a second table like store\_info from Step 12 that contains store\_id, storename, and city as fields, eliminating storename and city from the table above. In this way, a query could tie the store\_id number from the table containing request\_id to the store\_id number in the store\_info table and retrieve the city and name data. This process is sometimes referred to as factoring out information. **Third Normal Form** is not always necessary and will not be covered here.

With your partner, design a table structure for the High School Art Gallery site project that satisfies the requirements of the first two forms and includes 3 – 6 new fields. Working through example insertions into the database that you design on paper may help you evaluate it for correctness. Describe your final design.

1. 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).
2. Use the Internet to help you write a paragraph comparing and contrasting relational database management systems with **NoSQL**, a non-relational DBMS.
3. Record the name of one local business or company that likely uses databases and explain how it might use them.

**Conclusion**

1. What are the advantages of adhering to the first two of the normal forms?
2. Describe a way that a school might use a relational database.
3. What additional functionality do you achieve by airing up PHP with MySQL as opposed to using MySQL from the command line?