Database Management Systems and the MySQL RDMS

In this week, we will cover the following topics:

- *Relational Database Management Systems (RDMS)*
  - *What they are.*
  - *What they are for.*
- *How RDMS can often refer to different aspects regarding database design, database software, and database querying etc.*
- *Examples of RDMS, including the software environment we use on this course.*
- *Querying: brief look at some MySQL queries.*

… and will result in the following learning outcomes

- *A knowledge of RDMS and the different ways in which they can be used.*
- *An appreciation of how the term RDMS can be used in different contexts.*
- *Knowledge that there are many different kinds (commercial, free) of RDMS available.*
- *An understanding of the development environment used on this course, and a map of where databases are typically installed in terms of:*
  - *real-world set-up and*
  - *developer set-up*
- *Knowledge of how MySQL queries are 'part of' the RDMS.*

Table of Contents

WEEK 6: DATABASE MANAGEMENT SYSTEMS AND THE MYSQL RDMS

6.1 Introduction

During the first 5 weeks of this course, you have learned about the principles of relational databases We starting in the beginning by looking at why data is better stored in a database, as opposed to in a file-based system, through to learning about the design of database systems using the entity-relationship approach to the representation of data entities and relationships between them. Last week we also covered the process of normalisation in arriving at an ER-diagram.

The principles of relational databases are *independent* of the software tools that you use to actually create a relational database. This is fine. It is always best to learn the principles of a subject, in order that you can understand and enjoy the practice of database creation etc.

In this chapter, we will provide a brief introduction to Database Management Systems (DBMSs) before introducing the MySQL Relational Database Management System (RDMS). The latter is the RDMS that you will be using in the course, and within the practical sessions / labs.

When you first start to learn the subject of computing and computer programming, you will be introduced to many 'software environments' (the set of software tools that you use to create, then develop, something for a given purpose… a program running to solve a problem, a database to store data etc.). As your skills become more sophisticated, you will be able to more easily learn new software environments, but in the beginning it is important to spend time on familiarising yourself with the relevant environment / environments.

This is very important. While it is also very important to learn and understand the principles of a subject, knowing how to convert that knowledge into useful creations, which, in the end, is the whole point of undertaking an applied computing course.

This week's material has therefore been written to provide you with a 'first-contact' with DBMSs. Then, in particular, we expand on this in the context of the MySQL RDBMS, the RDBMS that will be the environment of choice for this course. However, we also provide a brief overview of some of the other DBMS 'out there', which you may come across in the future.

6.2 Database management systems and their architecture

People who use a database may do so directly or indirectly, which we come to, but generally we can think of 'users' as database administrators, database designers, and database end users:

- Database administrators: Database administrators are tasked with overseeing the 'usage' of the database. That is, managing the specific way in which various users of the database can access it and, therefore, specifying restrictions on which part of the database a user might have 'access' to etc., depending on the scope of the particular use requirements. This will involve, for example, setting up of user profiles and passwords, deleing them if necessary, but also for maintaining the various other admin related tasks, e.g., making sure the current licence (agreed with a software

vendor) covers all usages, and that the licence is, of course, updated; the software should be being used legally.

- Database designers: Designers are familiar with all of the principles of database design, covered thus far in the course, because they actually design the schema, and all of the many details that go hand in hand with this role.
- Database end users: Any user of the database who accesses data from the database. A user role can vary greatly. It might, for example, be someone who has a relatively infrequent, but important, interaction with the database on a monthly basis, to obtain a report on the total costs of a given set of company actions. Alternatively, it might be a quantitative scientist who depends on large volumes of data to be transferred on a daily basis for analysis.

For the current course, we are not interested in going into all of the different levels of detail that exist between the use of databases, at the external level, and the internal representation of databases in file storage areas within the hardware of a computer. However, you should be aware that this detail exists and whereabouts the DBMS sits in this broader context.

Therefore, in Figure 1, we distinguish between the basic idea of database interaction (A) and that of the database architecture, which includes the database management system. The architecture is drawn in its most basic form (B) and is elaborated on (C).

There are an number of objectives to this architecture, which known as the ANSI-SPARC three-level architecture. Therese objectives are to:

- decouple the view of the data from the data itself. In other words, two different end users of the database might have different views of the data, depending on how they would like to use the data, even though the data itself is a resource that is common to all users (it is the same data).
- ensure users do not deal with internal physical storage details.
- decouple structures from user views. In other words, if a database administrator alters the representation of data structure, this should not affect the any user views.
- separate the structure of the database from the details of the storage device.

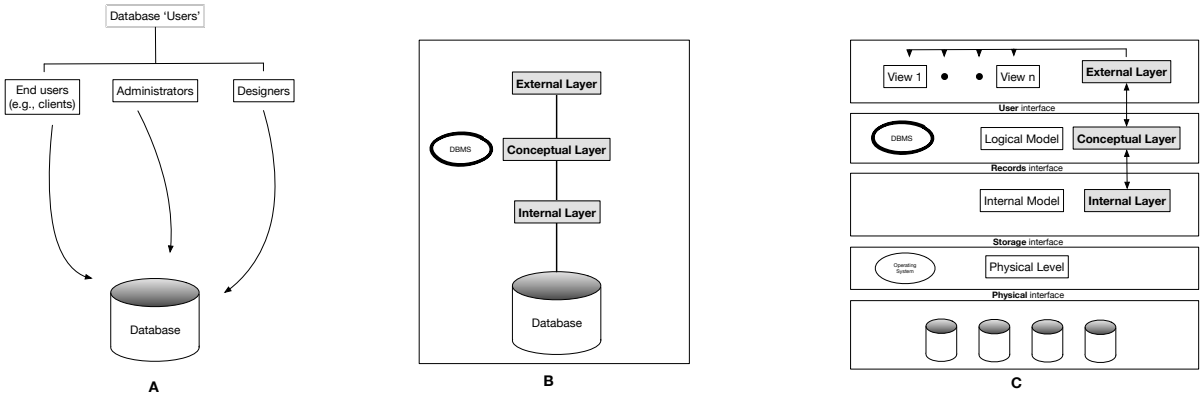

Figure 1: Database interaction v database architecture

6.3 Architecture levels

During the process of database development, you will be involved, to different degrees, with operations relevant to each of the three different levels – internal, conceptual, external – of the ANSCI-SPARC architecture. We now briefly consider each of these in turn.

6.3.1 Internal level

The internal level is concerned with the storage of data at the physical level. The engineering and design of the physical level is not something that directly concerns us, but it is worth noticing the interaction between the operating system and the file system that is required. For example, when the database needs to store a given datum, the DBMS communicates with the operating system, which in turn communicates to the physical layer whose operations allow the storage and retrieval of data.

The internal level is therefore the place where operations concerning the memory allocation and writing of data to locations in memory.

For example, when we create a table (we will see how to do this soon) we can define exactly the amount of storage that is required for a given field. The physical process of actually creating this memory is triggered from within the internal layer. Notice that the DBMS sits higher up in the architecture and, therefore, from a practical point of view, knowing detail about the lower-level is not essential to a database administrator or, of course, an end user. Nevertheless, the design of this layer is key to the performance of the database, because it will effect the speed of data retrieval.

6.3.2 Conceptual level

This layer is most relevant to the first part of the course (principles of relational databases) and therefore covers those aspects, i.e., the 'outputs' from the process of the design stage:

- Definitions of entities, of relations, and relationships.
- Any boundaries and constraints on the data, e.g., ranges and heuristics that are imposed to reflect constraints in the real world.
- Descriptions about the data, what the entities mean, related designs etc.

Following our example, if we create a table at this level, certain details will be necessary, such as an Address table containing a postcode, which might be defined as a String. However, there is no need to define at this stage that the String field 'postcode' requires # bytes for storage.

The best way to think of the conceptual layer is as an overall birds-eye view of the entire database in relation to the requirements of the client as have been established throughout the process of design.

6.3.3 External level

On the other hand, whereas the birds-eye view will be necessary to know, especially for the database administrator and, by definition, the database designer him/her/themselves, certain clients might only want to parts of the data.

During the process of database design, as we experienced during the process of normalisation, data becomes optimally arranged into tables, minimising the amount of storage, reducing the likelihood of anomalies, and improving ease of administration etc.

One of the striking features of this process, if we consider the arrangement of data in its original file-based form, for example, is that this process of optimisation (from the perspective of a database design) might look (from the perspective of a specific user) like a process of fragmentation. This is because a user will have grown used to seeing the data they work with, day to day, in a specific format, and usually from their own point of view, not that of the entire organisation.

Of course, while the process of database design should be geared towards creating a single common resource, free of redundancy, and optimal in terms of performance, the use wants to see the kind of data they are used to seeing. This latter requirement is served by the external layer of the database architecture.

However, rather than each separate user having their 'own copy' of the data, a *view* of the organisations common database is created. In this way, the data relevant to a particular role within an organisation (pay roll administrator, truck driver, department manager, sales executive, business analyst etc.) can be served-up as a specific view. This shield the user from most of the data, which from their perspective is irrelevant, but where data is common across views it is the *same* data, not a local copy. A view might also contain data that is not stored, but which can be calculated from other data, such as the commission owed to an employee who has worked a given number of hours (stored) and who is payed according to one of several rates (stored).

6.4 Relational database management systems

Which database management system is the best one to use? There is no simple answer to this question because it depends on what it is you want to use your database for. This section is really to give you a quick map of the different kinds of database software that are 'out there', before we go on to look at a relational database management system in more detail – i.e., the MySQL RDMS.

6.4.1 Desktop applications

Desktop applications aimed at non-developers are popular with people who do not necessarily know how to program or query a database, but are very interested in data, its organisation and have a need access to it in an efficient and ordered way. The two main examples are *Microsoft Access* and *FileMaker*.

From a user point of view, desktop applications are easy to install (they are just like any other application in this respect). As a consequence of the target customer, the applications are relatively easy to use and have numerous 'typical' business solutions readily available. However, these desktop applications do not scale for larger amounts of data, and are limited in terms of the number of users who can read and write to the database. Therefore, the applications are aimed at relatively small businesses (SMEs), which do not have masses of

data, or a large set of staff who need to update and query the database. Such limitations rule out the use of such systems for larget companies and/or projects where many more users will be accessing and updating a dense set of data reflecting the business activity of a much larger enterprise, for example.

6.4.2 RDMS

Relational database management systems support the concepts that we have been looking in, thus far, in this course; data stored in tables, which have rows and columns, keys and relationships etc. etc.

An immediate difference with RDMS, compared to desktop applications, is that they are less straightforward to install. Firstly, the apparently singular RDMS is actually composed of a database engine, the core part of the software, which is a separate installation to the associated administrative tool. For example, when a given installation takes place, such as an install of MySQL, then the database engine is installed, typically on a server. Once this is done it is possible to interact with the engine via the command line, but a separate application called MySQL Workbench can then be installed, which is the software that the database administrator will use to access and manage the database.

A number of RDMS can be obtained, either on a commercial or Open Source licence. Examples are presented in Table 1.

RDMS Name	Company Vendor	Licence
Oracle	Oracle	Commercial
DB2	IBM	Commercial
SQL Server	Microsoft	Commercial
MySQL	Oracle	Open Source
SQLite	Public Domain	NA
Terradata	Terradata Corporation	Commercial
Sybase	Sybase Inc.	Commercial

Table 1: RDMS examples.

6.5 Software environment for this course

For this course, it will be useful to install the following software.

- **Apache web server**
  - This is web server software. Web server software is installed on a computer, which then has the capacity to be a web server, i.e., to process requests via the network protocol HTTP.
- **MySQL Engine**
  - The software that allows databases to be created.
- **MySQL Workbench**
  - Application software, which provides various tools that allow creation of databases, querying, database administration etc.
- **PHP**

- A server side programming language typically used in conjunction with HTML to provide functionality to websites, and the ability to connect to databases.

- **JAVA SDK**
  - Javas software development kit, which includes tools to edit, compile and run java programs, in this course as desktop applications.
- **ECLIPSE**
  - Chosen integrated development environment (used in the development of java applications).

We will return to the various items in this software in future weeks, but for now we focus on the MySQL items (MySQL and MySQL Workbench).

6.6 MySQL and MySQL Workbench

Once installed, MySQL and MySQL Workbench allow you to run a database. As mentioned, a database in the real-world will be stored on a server (or a set or servers). On the desktop, you can have installed a GUI and a command line client, which can get access to the database by providing the username and password for that database. This general set-up is illustrated in Figure 2 (A).

In Figure 2 (B), we see the 'MySQL' specification of this general set-up. That is, we have a MySQL database on the server, then on the Desktop we have installed MySQL Workbench (this can be done for either Windows or Mac), and the command line clients are already installed, of course, on the relevant operating system (Windows, Mac OS, Linux etc.).

In the following two sections we will introduce some commands and some screenshots, which illustrate some basic commands in SQL and allow us to have a first look at the interfaces (command line, MySQL Workbench) to connect to the database.

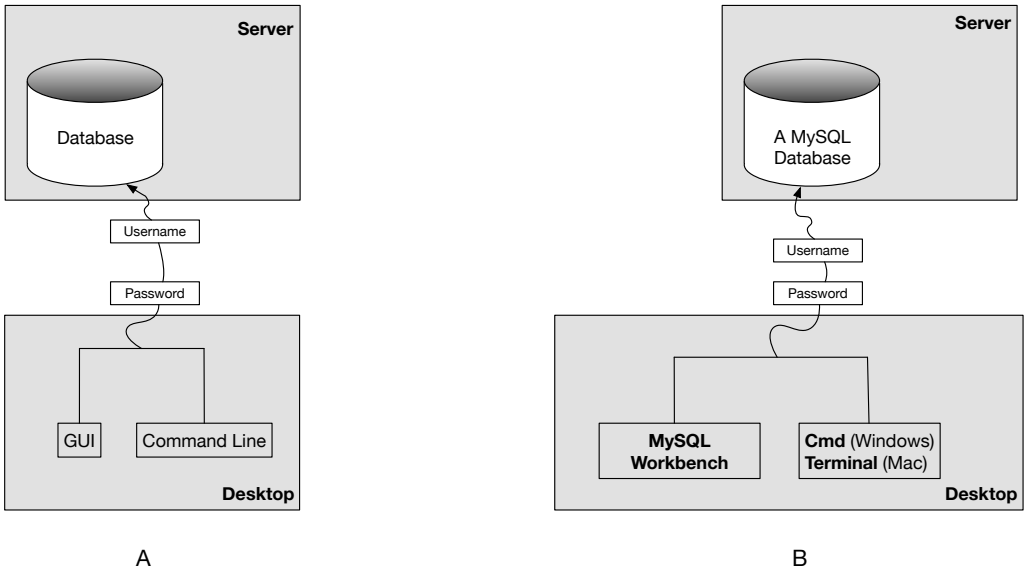

Figure 2: DB - relationship between Desktop and Server.

6.7 Initial steps and some simple SQL

In the examples that follow, we will assume that a database engine is on a server, and is running. In fact, for these examples, the database engine has been installed on the local machine and the database administrator has created for us a *username* and *password*.

6.7.1 MySQL on the command line

Depending on the operating system you are working on, you will be either using the <cml.exe> program on windows or, e.g., if you are on Mac OS you will be using the Terminal program.

In order to start the MySql engine for a specific user who has been given a username and password, you type into Terminal the following commands:

```
>mysql –u root –p
```

… and then you will be asked to enter your password:

```
[ Enter password:
```

What does this mean? The <mysql> part of the command is the name of the program we are calling on the command line. The program has been coded such that it can accept as arguments a value <*root*> associated with the username option <-*u*> and a value <your password> associated with the password option <-*p*>. Once the password is entered correctly we are then *inside* the MySQL engine. The way we know this is that the command line is now prefixed with <mysql>:

```
[mysql> quit
```

If you enter the *quit* command, then you will be returned to the control of the Terminal program. This makes sense – we are using Terminal to launch the mysql program, so, when we quit mysql we are returned to the calling program.

We have not actually done anything yet on the command line. We just started mysql, confirmed that we were inside the mysql program, then quit the program. This is fine. We are not intending to do much here other than become a little familiar with our environment (seen from the point of view of the command line). However, let us have a look at what databases exist in 'on the server':

```
[mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| codepoint_db       |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

It does not matter what these are. We are just demonstrating some very simple commands. The **output** from these commands is in bold.

In order to create a database called <temp_db> we execute the following:

```
[mysql> CREATE DATABASE temp_db;
Query OK, 1 row affected (0.00 sec)
```

…after which the <show databases;> command will reveal temp_db in the list of databases. To drop < temp_db>, i.e., to delete / get rid of the <temp_db> database we use the command:

```
mysql> DROP DATABASE IF EXISTS temp_db;
Query OK, 0 rows affected (0.00 sec)
```

…after which the <show databases;> command will reveal <temp_db> to have been removed from the list of databases.

The preceding set of commands are also given in Figure 1.

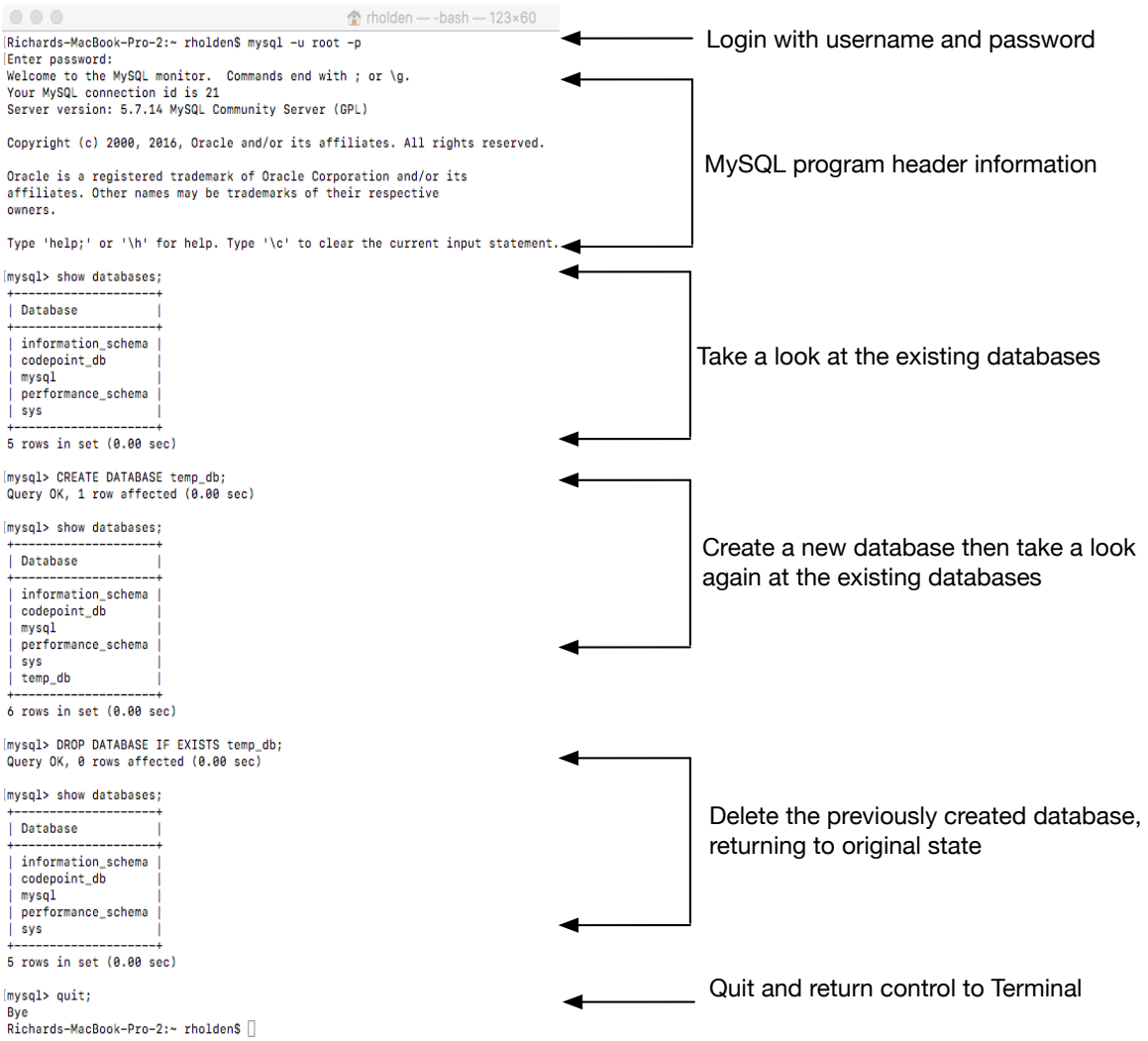

Figure 3: Simple set of SQL commands in Terminal

6.7.2 MySQL and the MySQL Workbench

MySQL Workbench is the Graphical User Interface (GUI) interface to MySQL. If is presented in Figure 4. When launched a dark view can be seen. Double clicking on the required connection will prompt a password dialougue. Once entered correctly we are 'inside MySQL', equivalent to being:

```
[mysql>
```

… on the command line. All of the things we can do from the command line can be done from within the GUI. In fact, the GUI is designed to help MySQL development. Some features are displayed at the bottom of Figure 4:

1. By right-clicking in this area, you can create a schema (i.e., database).
2. This area acts as a command window and a script editor. So, we are able to execute SQL statements from within this window much like we can in the <cmd.exe> or <Terminal> programs.
3. The results of the commands executed are displayed here.
4. When typing commands, a context sensitive help window provides feedback as to which options you might want to include when executing an SQL statement.

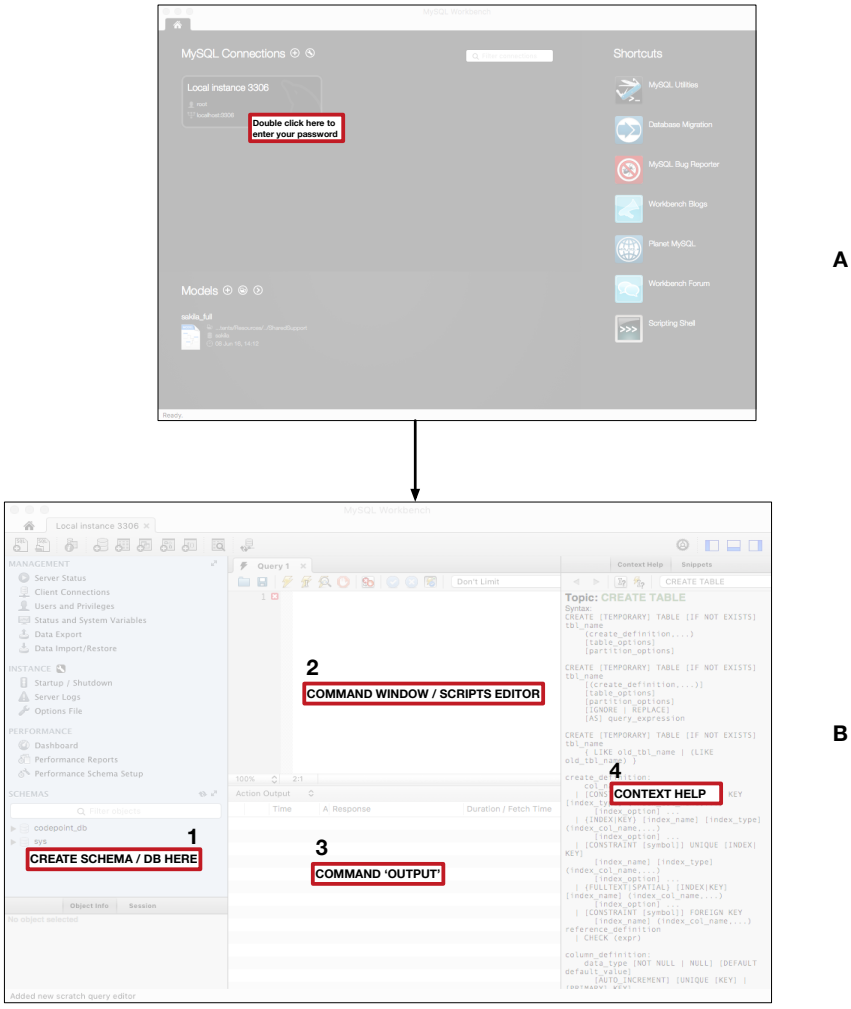

Figure 4: MySQL Workbench.

6.8 Summary

- We introduced the idea of database management systems.
- Several general properties of database management systems were covered, relating to the *internal* level, the *conceptual* level, and *external* level.
- We then summarized some of the software that is relevant to the current course.
- As a budding database developer, it is expected of you within industry that you have a good knowledge not only if the theory of databases, but the pragmatic aspects of database development, which we overviewed.
- Finally, we presented a brief introduction to MySQL workbench, a GUI for MySQL databases.

~~~~~
~~~~~