JOHNS HOPKINS
UNIVERSITY

Zanvyl Krieger School of Arts and Sciences
Advanced Academic Programs

410.712.81
Advanced Practical Computer Concepts For Bioinformatics

Topic: MySQL

Instructor: Joshua Orvis

MySQL and Chado

We have already reviewed general databases and SQL, including basic principles of data modeling and normalization. This lecture will extend this into the realm of the practical by first discussing a particular RDBMS vendor (MySQL) and then discussing its basic features.

There's even one cartoon.

MySQL introduction

There is a wide variety of database vendors in modern use – these vary greatly in cost and feature sets. Empirical usage comparisons between them are difficult because some are commercial, with well-tracked install base statistics, while others are open-source and more difficult to track since they are not distributed from a single source. Some of the more popular include:

DB2, MySQL, Oracle, PostgreSQL, Sybase

So why are we using MySQL? It is a popular choice for developers of web applications because:

- It's free
- It has a relatively low overhead
- Integrates well with Linux, Apache, Perl/PHP/Python
- Commercial support is available for paying customers
- Many high-profile sites use it:
    - Flickr, Google, YouTube, Facebook, Wikipedia

Oracle currently owns MySQL after acquiring Sun, the previous backer, in 2009.

MySQL features

Most developers have a few advanced favorite features they rely on and will harshly judge a new RDBMS if they aren't present. So, here's the neat stuff MySQL can do:

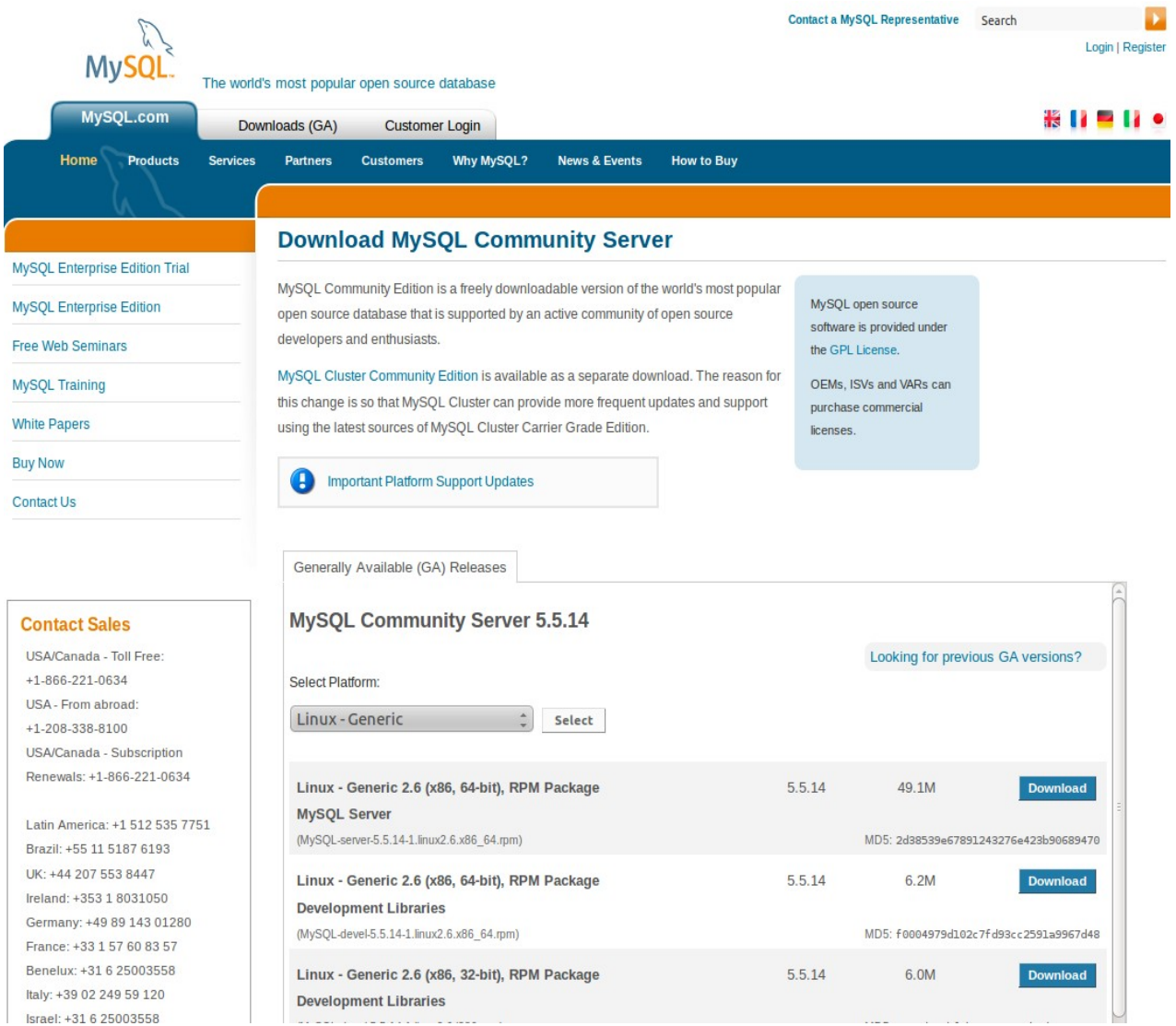
- Partitioned tables and database clustering
- Query caching
- Sub-select queries
- Cursors, triggers and stored procedures
- Updatable views (but not materialized views)
- Cross-platform support (Windows, OS X, Linux, etc.)
- Full text indexing
- Customizable storage engines
- Transactions

Null hypothesis

http://xkcd.org/892/

Getting MySQL

The free version of MySQL is known as the MySQL Community Server.



http://www.mysql.com/downloads/mysql/

MySQL is already set up on the class server, but you can go here if you'd like to install it on your own machine.

Connecting to MySQL

Our specific server information was given in the Introduction sent at the beginning of the semester, but generically, you connect to a MySQL server with the following simple command:

```
mysql -u jsmith -h localhost -p
```

It will then prompt you for your password. Once logged in, you'll see the MySQL terminal, where you can enter commands and SQL statements.

Note that you cannot just connect to ANY computer running a MySQL server. The server must first have the external port open that the MySQL daemon listens on. This is 3306 by default, but can be changed by the server administrator. If changed, you can specify the custom port by adding the -P option to your connection string.

Useful commands

SQL was covered in a previous lecture, so I'm only going to cover MySQL-specific commands here. Some of the most common:

mysql> show databases
Shows a list of all databases on the system you have permission to use.

mysql> use *database*
Changes you command context to operate within a specific database.

mysql> show tables
Lists all the tables within your current database.

mysql> describe *table*
Lists all the columns in a table, their datatypes, and more.

mysql> create database *database*
Creates an empty database with the passed name.

Exporting a database

MySQL has a quick database exporting tool called 'mysqldump'. Depending on the options you use it can do a complete dump of the data and object definitions. This can be used to back up a database or just send a copy to someone else. A simple example backs up a database called 'annotation' into a file called 'annotation.sql'

```
$ mysqldump -u jorvis -h localhost -p annotation > annotation.sql
```

Notice that the output is a plain-text SQL file. This can be imported easily into other MySQL server instances (see next slide.)

The mysqldump command has a great many options controlling what is exported, the format used, etc. Please read the manpage for more information.

Importing data into MySQL

If you have a text file of MySQL commands there are two primary ways to import it into your database. The first is to use the *source* command from the mysql terminal and providing the path to your SQL file:

```
mysql> source /home/jorvis/somefile.sql
```

That will execute all SQL in the file but should carry a warning: as each statement is executed there is a bit of feedback on the command line showing you that the command was either successful or failed. It scrolls by very quickly, so if there are failures they will be hard to see.

The other method is to redirect standard input into the mysql client, specifying the database on the command line. Here's an example:

```
$ mysql -u jorvis -h localhost -p < somefile.sql
```

This would restore the contents of a mysqldump. If you had regular SQL statements that you wanted executed this way but *within* a database you'd just need to add the -D parameter.

Granting permissions

For these examples let's assume you have a database called 'annotation'. The MySQL GRANT command allows you to specify the exact permissions any user from any host can have on any database object. Its generic syntax is:

```
GRANT priv list ON database.table TO user@'host' IDENTIFIED BY 'password';
```

If we had just created our annotation database and I wanted to grant full permissions for myself from any host I'd do this:

```
GRANT ALL PRIVILEGES ON annotation.* to jorvis@'%' IDENTIFIED BY 'foo';
```

If I wanted to grant another power user the ability to read and modify data within my tables, I could do something like this:

```
GRANT SELECT,INSERT,UPDATE,DELETE ON annotation.* to shenson@'%';
```

Notice I left off the password definition there, since I knew she was already a user in the system. The '%' sign means 'all hosts' and the '*' means 'all tables'.

For a full list of the grantable rights and all possible options, see the official documentation here:

http://dev.mysql.com/doc/refman/5.1/en/grant.html

Storage engines

One very nice feature of MySQL is that you can choose a different back-end storage engine for each different table in your database (see the CREATE TABLE documentation). Here are a few of note:

MyISAM: This is the default engine and is optimized for relatively quick read access and has support for full text searches.

InnoDB: Best for transactional databases, this engine has support for commits, rollbacks, crash recovery and foreign key constraints.

Memory: Fastest engine for reads, since MySQL keeps data for all tables of this type in memory.

There is an excellent page here that describes more of the engines and has a feature comparison chart between them.

http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html

phpMyAdmin

I believe that you should always start with a new technology getting as close as you can to it, avoiding any tools or shortcuts that prevent you from really learning the system.

However, this tool is so very popular that it deserves a mention here. It can be used to completely manage your databases with a web-based point and click interface.

It is well designed and documented, but I believe you should spend the time first creating and adding tables, etc. manually and then, once you're a pro, occasionally use a tool like phpMyAdmin to save time.

Further reading

Comparison of relational database management systems
http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

Google Code University introduction to databases and MySQL
http://code.google.com/edu/tools101/mysql.html