



# Introduction to MySQL

# Road Map

- Introduction to MySQL
- Connecting and Disconnecting
- Entering Basic Queries
- Creating and Using a Database

# Attribution

- Most of these slides are based directly on the MySQL Documentation.
- Most information comes from Chapter 3, MySQL Tutorial:
- [http://www.mysql.com/documentation/mysql/bychapter/manual\\_Tutorial.html#Tutorial](http://www.mysql.com/documentation/mysql/bychapter/manual_Tutorial.html#Tutorial)

# MySQL

- MySQL is a very popular, open source database.
- Officially pronounced “my Ess Que Ell” (not my sequel).
- Handles very large databases; very fast performance.
- Why are we using MySQL?
  - Free (much cheaper than Oracle!)
  - Each student can install MySQL locally.
  - Easy to use Shell for creating tables, querying tables, etc.
  - Easy to use with Java JDBC

# Crash Course Fundamentals

- In order to use JDBC, you need:
  - a database.
  - basic understand of SQL (Structured Query Language)
- Some students may have database backgrounds; others may not.
- The purpose of this lecture is to get all students up to speed on database fundamentals.

# Connecting to MySQL

- MySQL provides an interactive shell for creating tables, inserting data, etc.
- On igor, just go to the terminal app and type:

```
mysql -h igor -u ma007xyz -p
```

- (replace ma007xyz with your username)

# Connecting to MySQL

- This command tells *mysql* to connect to the MySQL server on host *igor* (-h igor) as user *ma007xyz* (-u ma007xyz) and to prompt for a password (-p).

# My SQL password

- A password for MySQL will be emailed to you when your MySQL account is created for you (your MySQL password is separate from your normal Goldsmiths network password).



# forgotten your password?

- If you have forgotten your password
  - login in to the department intranet
  - Select MySQL Account Status
  - Type the new password and set it

# Sample Session

```
-bash-3.2$ mysql -h igor -u mas01jo -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4022697
Server version: 5.0.95 Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

To exit the MySQL Shell, just type  
QUIT or EXIT:

```
mysql> QUIT
mysql> exit
```

# Basic Queries

- Once logged in, you can try some simple queries.
- For example:

```
mysql> SELECT VERSION() , CURRENT_DATE;  
+-----+-----+  
| VERSION() | CURRENT_DATE |  
+-----+-----+  
| 3.23.49   | 2002-05-26   |  
+-----+-----+  
1 row in set (0.00 sec)
```

- Note that most MySQL commands end with a semicolon (;)
- MySQL returns the total number of rows found, and the total time to execute the query.

# Basic Queries

- Keywords may be entered in any lettercase.
- The following queries are equivalent:

```
mysql> SELECT VERSION() , CURRENT_DATE;  
mysql> select version() , current_date;  
mysql> SeLeCt vErSiOn() , current_DATE;
```

# Basic Queries

- Here's another query. It demonstrates that you can use mysql as a simple calculator:

```
mysql> SELECT SIN(PI()/4) , (4+1)*5;
+-----+-----+
| SIN(PI()/4) | (4+1)*5 |
+-----+-----+
|      0.707107 |      25 |
+-----+-----+
```

# Basic Queries

- You can also enter multiple statements on a single line. Just end each one with a semicolon:

```
mysql> SELECT VERSION() ; SELECT NOW() ;
```

```
+-----+
| VERSION() |
+-----+
| 3.22.20a-log |
+-----+
+-----+
| NOW() |
+-----+
| 2004 00:15:33 |
+-----+
```

# Multi-Line Commands

- mysql determines where your statement ends by looking for the terminating semicolon, not by looking for the end of the input line.
- Here's a simple multiple-line statement:

```
mysql> SELECT
```

```
    -> USER()
```

```
    -> ,
```

```
    -> CURRENT_DATE;
```

```
+-----+-----+
| USER()                | CURRENT_DATE |
+-----+-----+
| joesmith@localhost    | 1999-03-18   |
+-----+-----+
```

# Canceling a Command

- If you decide you don't want to execute a command that you are in the process of entering, cancel it by typing \c

```
mysql> SELECT  
      -> USER()  
      -> \c  
Mysql>
```



# MySQL prompts

- notice the prompt. It switches back to `mysql>` after you type `\c`, providing feedback to indicate that mysql is ready for a new command.

# MySQL prompts

The following table shows each of the prompts you may see and summarizes what they mean about the state that mysql is in.

Prompt	Meaning
<code>mysql&gt;</code>	Ready for new command.
<code>-&gt;</code>	Waiting for next line of multiple-line command.
<code>'&gt;</code>	Waiting for next line, waiting for completion of a string that began with a single quote ("'").
<code>"&gt;</code>	Waiting for next line, waiting for completion of a string that began with a double quote ("").
<code>`&gt;</code>	Waiting for next line, waiting for completion of an identifier that began with a backtick ("`").
<code>/*&gt;</code>	Waiting for next line, waiting for completion of a comment that began with /*.

# Typos

- Multiple-line statements commonly occur by accident when you intend to issue a command on a single line, but forget the terminating semicolon. In this case, mysql waits for more input:

```
mysql> select user()  
->
```

Enter a semicolon to complete the statement, and mysql executes it

# Wrong statement

```
mysql> SELECT * FROM my_table  
WHERE name = 'Smith AND age <  
30;  
'>
```

- Notice the prompt It tells you that mysql expects to see the rest of an unterminated string

# corrections

- At this point, what do you do? The simplest thing is to cancel the command. However, you cannot just type `\c` in this case, because mysql interprets it as part of the string that it is collecting. Instead, enter the closing quote character

`'\c`

# Using a Database

- To get started on your own database, first check which databases currently exist.
- Use the SHOW statement to find out which databases currently exist on the server:

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| mysql    |  
| test     |  
+-----+  
2 rows in set (0.01 sec)
```

# Using a Database

- To create a new database, issue the “create database” command:
  - `mysql> create database ma007xyz_webdb;`
  - (replacing ma007xyz with your username)
- To the select a database, issue the “use” command:
  - `mysql> use ma007xyz_webdb;`

# Creating a Table

- Once you have selected a database, you can view all database tables:

```
mysql> show tables;
```

```
Empty set (0.02 sec)
```


- An empty set indicates that I have not created any tables yet.



# Creating a Table

- Let's create a table for storing pets.
- Table: pets

➤ name:	VARCHAR(20)
➤ owner:	VARCHAR(20)
➤ species:	VARCHAR(20)
➤ sex:	CHAR(1)
➤ birth:	DATE
➤ date:	DATE



**VARCHAR** is usually used to store string data.

# Creating a Table

- To create a table, use the CREATE TABLE command:

```
mysql> CREATE TABLE pet (  
    -> name VARCHAR(20) ,  
    -> owner VARCHAR(20) ,  
    -> species VARCHAR(20) ,  
    -> sex CHAR(1) ,  
    -> birth DATE, death DATE) ;  
Query OK, 0 rows affected (0.04 sec)
```

# Showing Tables

- To verify that the table has been created:

```
mysql> show tables;
```

```
+-----+
```

```
| Tables_in_test |
```

```
+-----+
```

```
| pet            |
```

```
+-----+
```

```
1 row in set (0.01 sec)
```

# Describing Tables

- To view a table structure, use the DESCRIBE command:

```
mysql> describe pet;
```

Field	Type	Null	Key	Default	Extra
name	varchar(20)	YES		NULL	
owner	varchar(20)	YES		NULL	
species	varchar(20)	YES		NULL	
sex	char(1)	YES		NULL	
birth	date	YES		NULL	
death	date	YES		NULL	

```
6 rows in set (0.02 sec)
```

# Deleting a Table

- Don't do this now ! But to
- delete an entire table, use the DROP TABLE command:

```
mysql> drop table pet;
```

```
Query OK, 0 rows affected (0.02 sec)
```

# Loading Data

- Use the INSERT statement to enter data into a table.
- For example:

```
INSERT INTO pet VALUES  
    ('Fluffy', 'Harold', 'cat', 'f',  
    '1999-02-04', NULL);
```

# To see the data in the file

- To see the records in a file we use the select statement

```
mysql> select * from pet;
```

# Deleting a row

- To delete a row from a table

```
mysql> delete from pet where  
name = 'fluffy';
```



# More data...

<b>name</b>	<b>owner</b>	<b>species</b>	<b>sex</b>	<b>birth</b>	<b>death</b>
<b>Fluffy</b>	<b>Harold</b>	<b>cat</b>	<b>f</b>	<b>1993-02-04</b>	
<b>Claws</b>	<b>Gwen</b>	<b>cat</b>	<b>m</b>	<b>1994-03-17</b>	
<b>Buffy</b>	<b>Harold</b>	<b>dog</b>	<b>f</b>	<b>1989-05-13</b>	
<b>Fang</b>	<b>Benny</b>	<b>dog</b>	<b>m</b>	<b>1990-08-27</b>	
<b>Bowser</b>	<b>Diane</b>	<b>dog</b>	<b>m</b>	<b>1998-08-31</b>	<b>1995-07-29</b>
<b>Chirpy</b>	<b>Gwen</b>	<b>bird</b>	<b>f</b>	<b>1998-09-11</b>	
<b>Whistler</b>	<b>Gwen</b>	<b>bird</b>		<b>1997-12-09</b>	
<b>Slim</b>	<b>Benny</b>	<b>snake</b>	<b>m</b>	<b>1996-04-29</b>	

# Loading Sample Data

- We are going to use a text file `pet.txt' containing one record per line.
- Values must be separated by tabs, and given in the order in which the columns were listed in the CREATE TABLE statement.
- We will load the data via the LOAD DATA Command.

# Sample Data File

Fluffy	Harold	cat	f	1993-02-04	\N
Claws	Gwen	cat	m	1994-03-17	\N
Buffy	Harold	dog	f	1989-05-13	\N
Fang	Benny	dog	m	1990-08-27	\N
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	\N
Whistler	Gwen	bird	\N	1997-12-09	\N
Slim	Benny	snake	m	1996-04-29	\N

# Saving the data file

- Go to the teaching webpage for this week's lecture
- Hold down ctrl key and click on the pet.txt file link
- Select Save link as from the menu
- Select igor.gold.ac.uk from the file list on the left hand plane.
- Save the file to to your igor root directory.

# Load data command

- To load the pet.txt file type the following

```
mysql> LOAD DATA LOCAL INFILE  
"pet.txt" INTO TABLE pet lines  
terminated by '\r';
```

The rest of the examples,  
assume the following set of data.

<b>name</b>	<b>owner</b>	<b>species</b>	<b>sex</b>	<b>birth</b>	<b>death</b>
<b>Fluffy</b>	<b>Harold</b>	<b>cat</b>	<b>f</b>	<b>1993-02-04</b>	
<b>Claws</b>	<b>Gwen</b>	<b>cat</b>	<b>m</b>	<b>1994-03-17</b>	
<b>Buffy</b>	<b>Harold</b>	<b>dog</b>	<b>f</b>	<b>1989-05-13</b>	
<b>Fang</b>	<b>Benny</b>	<b>dog</b>	<b>m</b>	<b>1990-08-27</b>	
<b>Bowser</b>	<b>Diane</b>	<b>dog</b>	<b>m</b>	<b>1998-08-31</b>	<b>1995-07-29</b>
<b>Chirpy</b>	<b>Gwen</b>	<b>bird</b>	<b>f</b>	<b>1998-09-11</b>	
<b>Whistler</b>	<b>Gwen</b>	<b>bird</b>		<b>1997-12-09</b>	
<b>Slim</b>	<b>Benny</b>	<b>snake</b>	<b>m</b>	<b>1996-04-29</b>	

# SQL Select

- The SELECT statement is used to pull information from a table.
- The general format is:

```
SELECT what_to_select  
FROM which_table  
WHERE conditions_to_satisfy
```

# Selecting All Data

- The simplest form of SELECT retrieves everything from a table

```
mysql> select * from pet;
```

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1999-02-04	NULL
Claws	Gwen	cat	f	1994-03-17	NULL
Buffy	Harold	dog	f	1989-05-13	NULL
Fang	Benny	dog	m	1999-08-27	NULL
Bowser	Diane	dog	m	1998-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	NULL
Whistler	Gwen	bird		1997-12-09	NULL
Slim	Benny	snake	m	1996-04-29	NULL

```
8 rows in set (0.00 sec)
```



# Selecting Particular Rows

- You can select only particular rows from your table.
- For example, if you want to verify the change that you made to Bowser's birth date, select Bowser's record like this:

```
mysql> SELECT * FROM pet WHERE name = "Bowser";
```

name	owner	species	sex	birth	death	
Bowser	Diane	dog	m	1998-08-31	1995-07-29	

```
1 row in set (0.00 sec)
```

# Selecting Particular Rows

- To find all animals born after 1998

```
SELECT * FROM pet WHERE birth >= "1998-1-1";
```

- To find all female dogs, use a logical AND

```
SELECT * FROM pet WHERE species = "dog" AND sex = "f";
```

- To find all snakes or birds, use a logical OR

```
SELECT * FROM pet WHERE species = "snake"  
OR species = "bird";
```

# Selecting Particular Columns

- If you don't want to see entire rows from your table, just name the columns in which you are interested, separated by commas.
- For example, if you want to know when your pets were born, select the name and birth columns.
- (see example next slide.)

# Selecting Particular Columns

```
mysql> select name, birth from pet;
```

name	birth
Fluffy	1999-02-04
Claws	1994-03-17
Buffy	1989-05-13
Fang	1999-08-27
Bowser	1998-08-31
Chirpy	1998-09-11
Whistler	1997-12-09
Slim	1996-04-29

8 rows in set (0.01 sec)

# Queries

- to find out who owns pets, use this query:

```
mysql> SELECT owner FROM pet;
```

- To retrieve each unique output record just once by adding the keyword **DISTINCT**:

```
mysql> SELECT DISTINCT owner  
FROM pet;
```

# Sorting Data

- To sort a result, use an ORDER BY clause.
- For example, to view animal birthdays, sorted by date:

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
```

```
+-----+-----+
| name      | birth      |
+-----+-----+
| Buffy     | 1989-05-13 |
| Claws     | 1994-03-17 |
| Slim      | 1996-04-29 |
| Whistler  | 1997-12-09 |
| Bowser    | 1998-08-31 |
| Chirpy    | 1998-09-11 |
| Fluffy    | 1999-02-04 |
| Fang      | 1999-08-27 |
+-----+-----+
```

```
8 rows in set (0.02 sec)
```

# Sorting Data

- To sort in reverse order, add the DESC (descending keyword)

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
```

```
+-----+-----+
| name      | birth      |
+-----+-----+
| Fang       | 1999-08-27 |
| Fluffy     | 1999-02-04 |
| Chirpy     | 1998-09-11 |
| Bowser     | 1998-08-31 |
| Whistler   | 1997-12-09 |
| Slim       | 1996-04-29 |
| Claws      | 1994-03-17 |
| Buffy      | 1989-05-13 |
+-----+-----+
```

```
8 rows in set (0.02 sec)
```

# Working with NULLs

- NULL means missing value or unknown value.
- To test for NULL, you cannot use the arithmetic comparison operators, such as =, < or <>.
- Rather, you must use the IS NULL and IS NOT NULL operators instead.



# Working with NULLs

- For example, to find all your dead pets (what a morbid example!)

```
mysql> select name from pet where death  
      >IS NOT NULL;
```

```
+-----+
```

```
| name  |
```

```
+-----+
```

```
| Bowser |
```

```
+-----+
```

```
1 row in set (0.01 sec)
```

# Pattern Matching

- MySQL provides:
  - standard SQL pattern matching; and
  - regular expression pattern matching, similar to those used by Unix utilities such as vi, grep and sed.
- SQL Pattern matching:
  - To perform pattern matching, use the LIKE or NOT LIKE comparison operators
  - By default, patterns are case insensitive.
- Special Characters:
  - \_ Used to match any single character.
  - % Used to match an arbitrary number of characters.

# Pattern Matching Example

- To find names beginning with 'b' :

```
mysql> SELECT * FROM pet WHERE name LIKE "b%";
```

name	owner	species	sex	birth	death
Buffy	Harold	dog	f	1989-05-13	NULL
Bowser	Diane	dog	m	1989-08-31	1995-07-29

# Pattern Matching Example

- To find names ending with `fy`:

```
mysql> SELECT * FROM pet WHERE name LIKE "%fy";
```

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	NULL
Buffy	Harold	dog	f	1989-05-13	NULL

# Pattern Matching Example

- To find names containing a 'w' :

```
mysql> SELECT * FROM pet WHERE name LIKE "%w%";
```

name	owner	species	sex	birth	death
Claws	Gwen	cat	m	1994-03-17	NULL
Bowser	Diane	dog	m	1989-08-31	1995-07-29
Whistler	Gwen	bird	NULL	1997-12-09	NULL

# Pattern Matching Example

- To find names containing exactly five characters, use the `_` pattern character:

```
mysql> SELECT * FROM pet WHERE name LIKE "_____";
```

name	owner	species	sex	birth	death
Claws	Gwen	cat	m	1994-03-17	NULL
Buffy	Harold	dog	f	1989-05-13	NULL

# Regular Expression Matching

- The other type of pattern matching provided by MySQL uses extended regular expressions.
- When you test for a match for this type of pattern, use the REGEXP and NOT REGEXP operators (or RLIKE and NOT RLIKE, which are synonyms).

# Regular Expressions

- Some characteristics of extended regular expressions are:
  - `.` matches any single character.
  - A character class `[...]` matches any character within the brackets. For example, `[abc]` matches a, b, or c. To name a range of characters, use a dash. `[a-z]` matches any lowercase letter, whereas `[0-9]` matches any digit.
  - `*` matches zero or more instances of the thing preceding it. For example, `x*` matches any number of x characters, `[0-9]*` matches any number of digits, and `.*` matches any number of anything.
  - To anchor a pattern so that it must match the beginning or end of the value being tested, use `^` at the beginning or `$` at the end of the pattern.



# Reg Ex Example

- To find names beginning with b, use ^ to match the beginning of the name:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^b";
```

name	owner	species	sex	birth	death
Buffy	Harold	dog	f	1989-05-13	NULL
Bowser	Diane	dog	m	1989-08-31	1995-07-29

# Reg Ex Example

- To find names ending with `fy`, use `\$` to match the end of the name:

```
mysql> SELECT * FROM pet WHERE name REGEXP "fy$";
```

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	NULL
Buffy	Harold	dog	f	1989-05-13	NULL

# Counting Rows

- Databases are often used to answer the question, "How often does a certain type of data occur in a table?"
- For example, you might want to know how many pets you have, or how many pets each owner has.
- Counting the total number of animals you have is the same question as "How many rows are in the pet table?" because there is one record per pet.
- The COUNT() function counts the number of non-NULL results.

# Counting Rows Example

- A query to determine total number of pets:

```
mysql> SELECT COUNT(*) FROM pet;
```

```
+-----+  
| COUNT(*) |  
+-----+  
|          9 |  
+-----+
```

# Batch Mode

- In the previous sections, you used mysql interactively to enter queries and view the results.
- You can also run mysql in batch mode. To do this, put the commands you want to run in a file, then tell mysql to read its input from the file:
- **shell> mysql < batch-file**

# Is that all there is to MySQL?

- Of course not!
- Understanding databases and MySQL could take us several weeks (perhaps months!)
- For now, focus on:
  - using the MySQL shell
  - creating tables
  - creating basic SQL queries

# Summary

- SQL provides a structured language for querying/updating multiple databases.
- The more you know SQL, the better.
- The most important part of SQL is learning to retrieve data.
  - selecting rows, columns, boolean operators, pattern matching, etc.
- Keep playing around in the MySQL Shell.