Introduction to MySQL
Acknowledge

These slides are a modified version of the slides available online at

- https://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiYlv6D4rfJAhWFORoKHaqlBVgQFgghMAA...%2F%40api%2Fdeki%2Ffiles%2F1830%2F%3DIntro_to_MySQL.ppt&usg=AFQjCNGzdn0C4HYp3JkSNY1_A1T2MSRGfg&sig2=MNbk3ujw91NxZ4vLN_750w
- http://www.slideshare.net/chauhantushar/introduction-to-mysql/4
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.


Introduction to Mysql
MySQL

- MySQL is a very popular, open source DBMS
- MySQL databases are relational
- Officially pronounced “my Ess Que Ell” (not my sequel).
- Handles very large databases;
- very fast performance; reliable.
- MySQL is compatible with standard SQL
- Why are we using MySQL?
  - Free (much cheaper than Oracle!)
  - Each student can install MySQL locally.
  - Multi-user access to a number of databases offered
  - Easy to use Shell for creating tables, querying tables, etc.
  - Easy to use with Java JDBC
  - MySQL is frequently used by PHP and Perl
  - Commercial version of MySQL is also provided (including technical support)
History of MySQL

- Founded and developed by David Axmark, Allan Larsson, and Michael “Monty” Widenius
- Named after Monty’s daughter, My
- MySQL Dolphin logo is “Sakila”, the name of a town in Arusha, Tanzania
- Written in C and C++
- Works on many different platforms
- Sun acquired MySQL AB in Jan 2008 for $1 billion dollars
MySQL Products Overview

**MySQL Server**
- Community Server
- Enterprise Server
- Embedded Server
- Cluster (Standard and Carrier-Grade)

**MySQL GUI Tools**
- Query Browser
- Administrator
- Migration Toolkit
- Visual Studio Plug-in
- MySQL Workbench (New!)

**MySQL Drivers**
- JDBC
- ODBC
- .NET
- PHP
MySQL: characteristics

MySQL Server works in:

- **client/server systems** → a system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

- **embedded systems** → provide MySQL Server as an embedded multi-threaded library that can be linked into an application to get a smaller, faster, easier-to-manage standalone product.
### MySQL: Features & Benefits

<table>
<thead>
<tr>
<th>Features &amp; Benefits</th>
<th>Details</th>
</tr>
</thead>
</table>
| **Scalability and Flexibility**      | Run anything from...  
|                                      | - Deeply embedded applications with a footprint of  
|                                      |   just 1MB, or  
|                                      | - Massive data warehouses holding terabytes of  
|                                      |   information |
| **High Performance**                 | Table and Index Partitioning  
|                                      | - Ultra-fast load utilities  
|                                      | - Distinctive memory caches  
|                                      | - Full-text indexes, and more |
| **High Availability**                | Run high-speed master/slave replication  
|                                      | configurations with Row-Based and Hybrid  
|                                      |   Replication  
|                                      | - Specialized Cluster servers offering instant  
|                                      |   failover |
| **Robust Transactional Support**     | Complete ACID (atomic, consistent, isolated,  
|                                      |   durable) transaction support  
|                                      | - Unlimited row-level locking  
|                                      | - Distributed transaction capability, and  
|                                      | - Multi-version transaction support |
| **Web and Data Warehouse Strengths** | High-performance query engine  
|                                      | - Tremendously fast data insert capability, and  
|                                      | - Strong support for specialized web functions, like  
|                                      |   fast full text searches |
# MySQL: Features & Benefits

| Strong Data Protection | - Powerful mechanisms for ensuring only authorized users have access  
| - SSH and SSL support safe and secure connections 
| - Powerful data encryption and decryption functions |

| Comprehensive Application Development | - Support for stored procedures, triggers, functions, views, cursors, ANSI-standard SQL, and more  
| - Plug-in libraries to embed MySQL database support into nearly any application |

| Management Ease | - Use Event Scheduler automatically schedule common recurring SQL-based tasks to execute on the database server.  
| - Average time from software download to complete installation is less than fifteen minutes. |

| Open Source Freedom and 24 x 7 Support | - Around-the-clock support and indemnification available through MySQL Network  
| - Enterprise quality and enterprise ready, from installation to support |

| Lowest Total Cost of Ownership | - Save on database licensing costs and hardware expenditures, all while cutting systems downtime |
MySQL: Architecture
MySQL: Community & Customers

Community
- MySQL Community Server
- MySQL GUI Management Tools
- MySQL Connectors (JDBC, ODBC, etc.)
- Documentation
- Forums

Customer
- Subscription:
  - MySQL Enterprise
- License (OEM):
  - Embedded Server
  - Support
- MySQL Cluster Carrier-Grade
- Training
- Consulting
- NRE
MySQL: Workbench

MySQL Workbench enables:

• a DBA, developer, or data architect

• to visually design, generate, and manage all types of databases
  • including Web, OLTP, and data warehouse databases
  • It includes everything a data modeler needs for creating complex ER models, and also delivers key features for performing difficult change management and documentation tasks that are normally time consuming
Some characteristics

❖ **Forward and Reverse Engineering**

- A visual data model can easily be transformed into a physical database on a target MySQL Server with just a few mouse clicks.
- It can also import SQL scripts to build models and export models to DDL scripts that can be run at a later time.

❖ **Change Management**
MySQL Workbench

Database Documentation

- Documenting database designs can be a time-consuming process. MySQL Workbench includes DBDoc that enables a DBA or developer to deliver point-and-click database documentation.

- Models can be documented in either HTML or plain text format, and includes all the objects and models in a current MySQL Workbench session.
Resources

• General starting point
  > http://www.mysql.com/

• Developer focused
  > http://dev.mysql.com/
Installing MySQL
MySQL: Installation

Instruction available at Ch. 2 of the MySQL tutorial available at
Connecting and Disconnecting to/from MySQL
Conventions

- commands meant to be executed from within a particular, for example, `shell>`
  - `root-shell>` is similar but should be executed as root
- `mysql>` indicates a statement that has to be executed from the `mysql` client program
- SQL keywords are not case sensitive
Connecting to MySQL

- MySQL provides an interactive shell for creating tables, inserting data, etc.
- On Windows, just go to c:\mysql\bin, and type:
  - `Mysql` or
  - `mysql -u user -p`;
- Or, click on the Windows icon
Sample Session

- For example:

Enter password:  *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 241 to server version: 3.23.49

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

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

mysql> QUIT
mysql> exit
Connecting to MySQL

How to use the mysql client

- **mysql** is an interactive program that enables you to:
  - connect to a MySQL server,
  - Run queries,
  - view the results

- **mysql** may also be used in batch mode:
  - *place your queries in a file beforehand*, then tell **mysql** to execute the contents of the file

- To see a list of options provided by mysql
  - shell> mysql --help
Entering & Editing commands

- Prompt mysql>
  - issue a command
  - Mysql sends it to the server for execution
  - displays the results
  - prints another mysql>

- A command could span multiple lines
- A command normally consists of SQL statement followed by a semicolon
MySQL commands

- help \h
- Quit/exit \q
- Cancel the command \c
- Change database use
- …etc
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:

```sql
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:

```sql
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;
```

```
+--------------------+--------------+
<table>
<thead>
<tr>
<th>USER()</th>
<th>CURRENT_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>joesmith@localhost</td>
<td>1999-03-18</td>
</tr>
</tbody>
</table>
+--------------------+--------------+```
# Command prompt

<table>
<thead>
<tr>
<th>prompt</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql&gt;</td>
<td>Ready for new command.</td>
</tr>
<tr>
<td>- &gt;</td>
<td>Waiting for next line of multiple-line command.</td>
</tr>
<tr>
<td>' &gt;</td>
<td>Waiting for next line, waiting for completion of a string that began with a single quote (&quot;'&quot;').</td>
</tr>
<tr>
<td>&quot; &gt;</td>
<td>Waiting for next line, waiting for completion of a string that began with a double quote (&quot;&quot;&quot;&quot;').</td>
</tr>
<tr>
<td>` &gt;</td>
<td>Waiting for next line, waiting for completion of an identifier that began with a backtick (&quot;`&quot;').</td>
</tr>
<tr>
<td>/* &gt;</td>
<td>Waiting for next line, waiting for completion of a comment that began with /*.</td>
</tr>
</tbody>
</table>
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>
```
Creating, Removing and Getting Information for a Database
Info about databases and tables

- **Listing the databases** on the MySQL server host
  - `mysql> show databases;`

- **Access/change database**
  - `mysql> Use [database_name]`

- **Showing the current selected database**
  - `mysql> select database();`

- **Showing tables in the current database**
  - `mysql> show tables;`

- **Showing the structure of a table**
  - `mysql> describe [table_name];`
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 (and for which the user has privileges):

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

View Users (Before MySQL 5.7.6)

- `mysql> SELECT User, Host, Password FROM mysql.user;`

Changing password for a user

- After 5.7.6, use `SET PASSWORD`
  - `mysql> ALTER USER user IDENTIFIED BY 'new_password';`
  - Ex.: `ALTER USER 'root'@'localhost' IDENTIFIED BY 'new_password';`
- Before 5.7.6, use `SET PASSWORD`:
  - `mysql> SET PASSWORD FOR user = PASSWORD('new_password');`
Using a Databases

Create a user

- `mysql> CREATE USER user [IDENTIFIED BY 'new-password']`;
- See “help CREATE USER”

Remove a user

- `mysql> DROP USER user;`
Using a Database

- To create a new database, issue the "create database" command:
  - `mysql> create database webdb;`
  - Note: Database names are case sensitive

- To the select a database, issue the "use" command:
  - `mysql> use webdb;`

  - To see what database is selected
    - `mysql> select database();`
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:

```sql
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:

```sql
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:

```sql
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

- To delete an entire table, use the DROP TABLE command:

```sql
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
  ('Puffball','Diane','hamster','f',
  '1999-03-30',NULL);
```
- The next slide shows a full set of sample data.
## More data...

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

- You could create 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.
- Then load the data via the LOAD DATA Command.
# Sample Data File

<table>
<thead>
<tr>
<th>Name</th>
<th>Owner</th>
<th>Species</th>
<th>Sex</th>
<th>Birthdate</th>
<th>Deathdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluffy</td>
<td>Harold</td>
<td>cat</td>
<td>f</td>
<td>1993-02-04</td>
<td>\N</td>
</tr>
<tr>
<td>Claws</td>
<td>Gwen</td>
<td>cat</td>
<td>m</td>
<td>1994-03-17</td>
<td>\N</td>
</tr>
<tr>
<td>Buffy</td>
<td>Harold</td>
<td>dog</td>
<td>f</td>
<td>1989-05-13</td>
<td>\N</td>
</tr>
<tr>
<td>Fang</td>
<td>Benny</td>
<td>dog</td>
<td>m</td>
<td>1990-08-27</td>
<td>\N</td>
</tr>
<tr>
<td>Bowser</td>
<td>Diane</td>
<td>dog</td>
<td>m</td>
<td>1979-08-31</td>
<td>1995-07-29</td>
</tr>
<tr>
<td>Chirpy</td>
<td>Gwen</td>
<td>bird</td>
<td>f</td>
<td>1998-09-11</td>
<td>\N</td>
</tr>
<tr>
<td>Whistler</td>
<td>Gwen</td>
<td>bird</td>
<td>\N</td>
<td>1997-12-09</td>
<td>\N</td>
</tr>
<tr>
<td>Slim</td>
<td>Benny</td>
<td>snake</td>
<td>m</td>
<td>1996-04-29</td>
<td>\N</td>
</tr>
</tbody>
</table>

To Load pet.txt:

```sql
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```
For each of the examples, assume the following set of data.

<table>
<thead>
<tr>
<th>name</th>
<th>owner</th>
<th>species</th>
<th>sex</th>
<th>birth</th>
<th>death</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluffy</td>
<td>Harold</td>
<td>cat</td>
<td>f</td>
<td>1993-02-04</td>
<td></td>
</tr>
<tr>
<td>Claws</td>
<td>Gwen</td>
<td>cat</td>
<td>m</td>
<td>1994-03-17</td>
<td></td>
</tr>
<tr>
<td>Buffy</td>
<td>Harold</td>
<td>dog</td>
<td>f</td>
<td>1989-05-13</td>
<td></td>
</tr>
<tr>
<td>Fang</td>
<td>Benny</td>
<td>dog</td>
<td>m</td>
<td>1990-08-27</td>
<td></td>
</tr>
<tr>
<td>Bowser</td>
<td>Diane</td>
<td>dog</td>
<td>m</td>
<td>1998-08-31</td>
<td>1995-07-29</td>
</tr>
<tr>
<td>Chirpy</td>
<td>Gwen</td>
<td>bird</td>
<td>f</td>
<td>1998-09-11</td>
<td></td>
</tr>
<tr>
<td>Whistler</td>
<td>Gwen</td>
<td>bird</td>
<td></td>
<td>1997-12-09</td>
<td></td>
</tr>
<tr>
<td>Slim</td>
<td>Benny</td>
<td>snake</td>
<td>m</td>
<td>1996-04-29</td>
<td></td>
</tr>
</tbody>
</table>
Manipulating Instances of Tables of a Database
Manipulating Table Instances

- **Remove records of a table**
  
  - `mysql> DELETE FROM tableName;`
  
  - `[WHERE where_condition];`

- **Update records of a table**
  
  - `UPDATE pet SET birth = '1989-08-31' WHERE name = 'Bowser';`
Querying Tables of a Database
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)
```

Try the same select:
- without the last ("")
- without the last ("") and (;)
- Try with \c
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";
  ```

  AND has higher precedence than OR → Use parenthesis if necessary
Selecting Particular Columns

- For having only a selection of columns of a table, just name the columns you are interested, separated by commas.
- Example: you want to know when your pets were born
  - select the name and birth columns.
- (see example next slide.)
Selecting Particular Columns

```sql
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)
```
Sorting Data

- To sort a result, use an **ORDER BY** clause.
- Example: view animal birthdays, sorted by date:

```sql
mysql> SELECT name, birth FROM pet ORDER BY birth;
+----------+------------+
<table>
<thead>
<tr>
<th>name</th>
<th>birth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffy</td>
<td>1989-05-13</td>
</tr>
<tr>
<td>Claws</td>
<td>1994-03-17</td>
</tr>
<tr>
<td>Slim</td>
<td>1996-04-29</td>
</tr>
<tr>
<td>Whistler</td>
<td>1997-12-09</td>
</tr>
<tr>
<td>Bowser</td>
<td>1998-08-31</td>
</tr>
<tr>
<td>Chirpy</td>
<td>1998-09-11</td>
</tr>
<tr>
<td>Fluffy</td>
<td>1999-02-04</td>
</tr>
<tr>
<td>Fang</td>
<td>1999-08-27</td>
</tr>
</tbody>
</table>
+----------+------------+
8 rows in set (0.02 sec)
```
Sorting Data

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

```sql
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)
```
Sorting Data

- Sorting on multiple columns in different directions
  - Get name, species, birth with animals in ascending order and date (within animal type) in descending order (youngest first)
  - `mysql> SELECT name, species, birth FROM pet ORDER BY species, birth DESC;`
  - Try the opposite
Selecting Particular Rows

- Find out who owns pets
  SELECT owner FROM pet;

- Find out who owns pets (without duplicate)
  SELECT DISTINCT owner FROM pet;

- Get birth dates for male dogs and female cats
  SELECT name, species, birth FROM pet WHERE (species = "dog" AND sex="m") OR (species = "cat" AND sex="f");
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

- Find all your dead pets

```sql
mysql> select name from pet where death >IS NOT NULL;
+--------+
| name   |
+--------+
| Bowser |
+--------+
1 row in set (0.01 sec)
```
Working with NULLs

- Two NULL values are regarded as equal in a GROUP BY
  - Ex.: create a query with a group by on an attribute having NULL values

- NULL values are presented:
  - first with ORDER BY ... ASC
  - last with ORDER BY ... DESC
Pattern Matching

- MySQL provides:
  - standard SQL pattern matching
  - regular expression pattern matching

- 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’:

```sql
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

- Find names ending with `fy':

```sql
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

- Find names containing a ‘w’:

```sql
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

- Find names containing exactly five characters
  - use the _ pattern character:

```sql
mysql> SELECT * FROM pet WHERE name LIKE "_____";
+-------+--------+---------+------+------------+-------+
<table>
<thead>
<tr>
<th>name</th>
<th>owner</th>
<th>species</th>
<th>sex</th>
<th>birth</th>
<th>death</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claws</td>
<td>Gwen</td>
<td>cat</td>
<td>m</td>
<td>1994-03-17</td>
<td>NULL</td>
</tr>
<tr>
<td>Buffy</td>
<td>Harold</td>
<td>dog</td>
<td>f</td>
<td>1989-05-13</td>
<td>NULL</td>
</tr>
</tbody>
</table>
+-------+--------+---------+------+------------+-------+
```
The other type of pattern matching provided by MySQL uses extended regular expressions.

Testing 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:
  - “.” matches any single character.
  - A character class [...] matches any character within the brackets.
    - Example: [abc] matches a, b, or c.
    - To name a range of characters, use a dash.
    - [a-z] matches any lowercase letter
    - [0-9] matches any digit.
  - “*” matches zero or more instances of the thing preceding it.
    - Example: x* matches any number of x characters
    - [0-9]* matches any number of digits
    - .* 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

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

```sql
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

- Find names ending with `fy',
  - use `$' to match the end of the name:

```sql
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 often used to answer the question,
  - "How often does a certain type of data occur in
    a table?"
  - Example: 1) how many pets are sored
  - 2) 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:

```sql
mysql> SELECT COUNT(*) FROM pet;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```
Counting Rows Example

- Finding how many pets each owner has:

```sql
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
```

<table>
<thead>
<tr>
<th>OWNER</th>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benny</td>
<td>2</td>
</tr>
<tr>
<td>Diane</td>
<td>2</td>
</tr>
<tr>
<td>Gwen</td>
<td>3</td>
</tr>
<tr>
<td>Harold</td>
<td>2</td>
</tr>
</tbody>
</table>
Selecting Particular Rows

- Find out number of animals per species
  
  ```
  SELECT species, count(*) FROM pet GROUP BY species;
  ```

- Find out number of animals per sex
  
  ```
  SELECT sex, count(*) FROM pet GROUP BY sex;
  ```

- Find out number of animals per combination of species and sex
  
  ```
  SELECT species, sex, count(*) FROM pet GROUP BY species, sex;
  ```
Selecting Particular Rows

- Find out number of dogs and cats per combination of species and sex
  
  ```sql
  SELECT species, sex, count(*) FROM pet WHERE species = 'dog' or species = 'cat' GROUP BY species, sex;
  ```
  
  - Try what happens changing `OR` with `AND`

- Find out number of animals per sex only for animals whose sex is known
  
  ```sql
  SELECT species, sex, count(*) FROM pet WHERE sex IS NOT NULL GROUP BY species, sex;
  ```
Batch Mode

- MySQL used interactively
  - to enter queries and view the results.
- MySQL can be run in batch mode.
  - put the commands you want to run in a file, then tell mysql to read its input from the file:
    - The created file is script file that is requested to be executed

- `shell> mysql < batch-file`
- `shell> mysql -t < batch-file`
Exercise 1

- Create a new DB or a table *shop* in an existing DB

```sql
CREATE TABLE shop (  
    article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,  
    dealer VARCHAR(20) DEFAULT '' NOT NULL,  
    price DOUBLE(16,2) DEFAULT '0.00' NOT NULL,  
    PRIMARY KEY(article, dealer));

INSERT INTO shop VALUES  
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),  
(3,'B',1.45),  
(3,'C',1.69),(3,'D',1.25),(4,'D',19.95);  
```
Exercise 1

- Find out the highest item article
  
  ```sql
  SELECT MAX(article) AS article FROM shop;
  ```

- Find the article, dealer, and price of the most expensive article.
  
  ```sql
  SELECT article, dealer, price
  FROM shop
  WHERE price=(SELECT MAX(price) FROM shop);
  ```

  ```sql
  SELECT article, dealer, price
  FROM shop
  ORDER BY price DESC
  LIMIT 1;
  ```
Exercise 1

• Find the highest price per article.

SELECT article, MAX(price) AS price
FROM shop
GROUP BY article;

• For each article, find the dealer(s) with the most expensive price.

SELECT article, dealer, price
FROM shop s1
WHERE price=(SELECT MAX(s2.price)
            FROM shop s2
            WHERE s1.article = s2.article);
Exercise 2

- Create the following tables

```sql
CREATE TABLE person (  
id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
name VARCHAR(60) NOT NULL,  
PRIMARY KEY (id) );
```

```sql
CREATE TABLE shirt (  
id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
style ENUM('t-shirt', 'polo', 'dress') NOT NULL,  
color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,  
owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),  
PRIMARY KEY (id) );
```
Exercise 2

- Populate the tables

```sql
INSERT INTO person VALUES (NULL, 'Antonio Paz');
SELECT @last := LAST_INSERT_ID();
INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', @last),
(NULL, 'dress', 'white', @last),
(NULL, 't-shirt', 'blue', @last);

INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');  
SELECT @last := LAST_INSERT_ID();
INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', @last),
(NULL, 'polo', 'red', @last),
(NULL, 'dress', 'blue', @last),
(NULL, 't-shirt', 'white', @last);

See Sec. 3.6.5, 3.6.9 of the tutorial for more details
```

SELECT @last := LAST_INSERT_ID();
Exercise 2

• Find out person names containing “Lilliana” as a string and having a shirt of any color but not white

```sql
SELECT s.*
FROM person p INNER JOIN shirt s ON s.owner = p.id
WHERE p.name LIKE '%Lilliana%' AND s.color <> 'white';
```
Exercise 3

- Build the DB having the following tables

<table>
<thead>
<tr>
<th>Matricola</th>
<th>Name</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Mario Rossi</td>
<td>34</td>
<td>4000</td>
</tr>
<tr>
<td>103</td>
<td>Mario Bianchi</td>
<td>23</td>
<td>3500</td>
</tr>
<tr>
<td>104</td>
<td>Luigi Neri</td>
<td>38</td>
<td>6100</td>
</tr>
<tr>
<td>105</td>
<td>Nico Bini</td>
<td>44</td>
<td>3800</td>
</tr>
<tr>
<td>210</td>
<td>Marco Celli</td>
<td>49</td>
<td>6000</td>
</tr>
<tr>
<td>231</td>
<td>Siro Bisi</td>
<td>50</td>
<td>6000</td>
</tr>
<tr>
<td>252</td>
<td>Nico Bini</td>
<td>44</td>
<td>7000</td>
</tr>
<tr>
<td>301</td>
<td>Sergio Rossi</td>
<td>34</td>
<td>7000</td>
</tr>
<tr>
<td>375</td>
<td>Mario Rossi</td>
<td>50</td>
<td>6500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Head</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>210</td>
<td>101</td>
</tr>
<tr>
<td>210</td>
<td>103</td>
</tr>
<tr>
<td>210</td>
<td>104</td>
</tr>
<tr>
<td>231</td>
<td>105</td>
</tr>
<tr>
<td>301</td>
<td>210</td>
</tr>
<tr>
<td>301</td>
<td>231</td>
</tr>
<tr>
<td>375</td>
<td>252</td>
</tr>
</tbody>
</table>
Exercise 3: Queries

Q: Find out matriculation number, name, age, salary of the employees earning more than 4000 Euro
(Write the SQL query)

In Tuple relational calculus:

\{ e.* | e(Employees) | e.Salary > 4000 \}