

Intro to DB and SQL

CIS1152 Adv Web Dev

Lecture 7

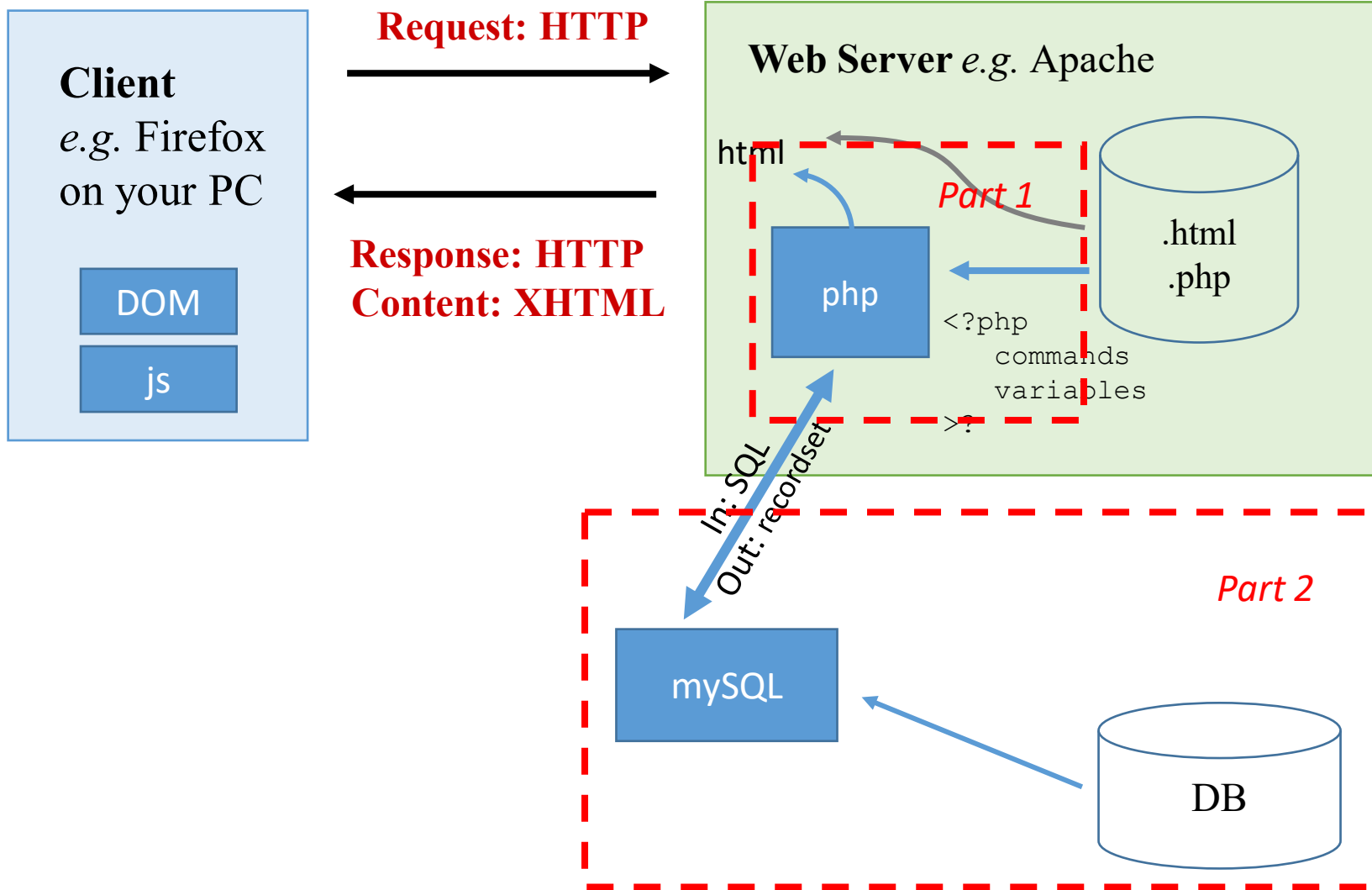
Steve Ruegsegger

Modified with permission by Peter Chapin

Course Perspective

- Whoa!
- We are *leaving behind* PHP, HTML, CSS for a bit.
- This 2nd section of the course is *completely separate* from the 1st section.
- However, we will tie them together in the 3rd half of the course.
- **Part 2:** We are introducing **SQL** and **databases**.
 - SQL is a *very different* language than anything else.
 - DB design and Entity Relationships are very *unique* skills.
 - *(You can get paid a lot to know all this stuff.)*

Back to the Big Picture



Outline

Goal: Introduce what a database is, and the SQL language we use to talk to them.

Objectives:

1. Intro DB (terms)
2. Setting up MySQL DB & client(s)
3. CRUD
4. datatypes

Intro to DB

See Welling & Thompson ([purple](#)) Ch 8

See Ullman ([red](#)) Ch 6

Like short tutorial videos? <http://bit.ly/2Nut7tv>

Terms

- A **database** is an ordered collection of information from which a computer program can quickly access information
- A single collection of related data is called a **table**. It has rows and columns like a *spreadsheet*
- Each **row** in a database table is called a *record, tuple, or an observation*
 - A **record** in a database is a single complete set of related information
- Each **column** in a database table is called a *field, attribute or a variable*
 - **Fields** are the individual categories of information stored in a record

example

Employees

table name

a.k.a. Observations, Tuples

The diagram shows a table with 6 columns and 6 rows. A box labeled 'Rows' has arrows pointing to the first two rows. A box labeled 'Fields' has arrows pointing to the first three columns. The table data is as follows:

last_name	first_name	address	city	state	zip
Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
Picard	Raymond	1113 Oakham Road	Barre	MA	01531

a.k.a. Variables, Attributes

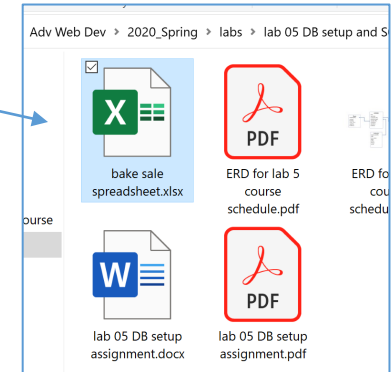
The whole spreadsheet is a table

values (str)

values (num)

Types of DBs

- A **flat-file database** stores information in a single table per file
 - We made some "flat-file" csv db's in Lab 3
- A **relational database** stores information across multiple related tables
 - A table is an *logical entity*
 - I don't really care if it's a *particular* file ...
 - (Who knows if it's one file or not. We don't care.)
 - The DB system keeps the tables separate, but it has the info of how they *relate* to each other!
 - We don't "read" a file (directly). Rather, we use a **client** to make requests from the database app...

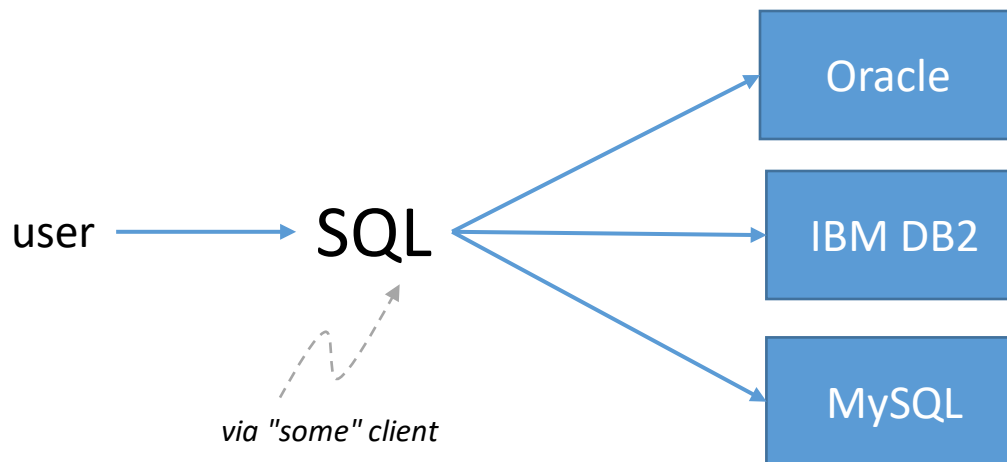


DBMS – database management system

- A **database management system** (or DBMS) is an application or collection of applications used to access and manage a database
 - IBM DB2
 - Oracle
 - Sybase
 - PostgreSQL
 - **MySQL**, SQLite
- Each DBMS could have it's own proprietary client and language to use it's DB! (That would be bad...)
- Fortunately, there is a standard (sort of)... SQL

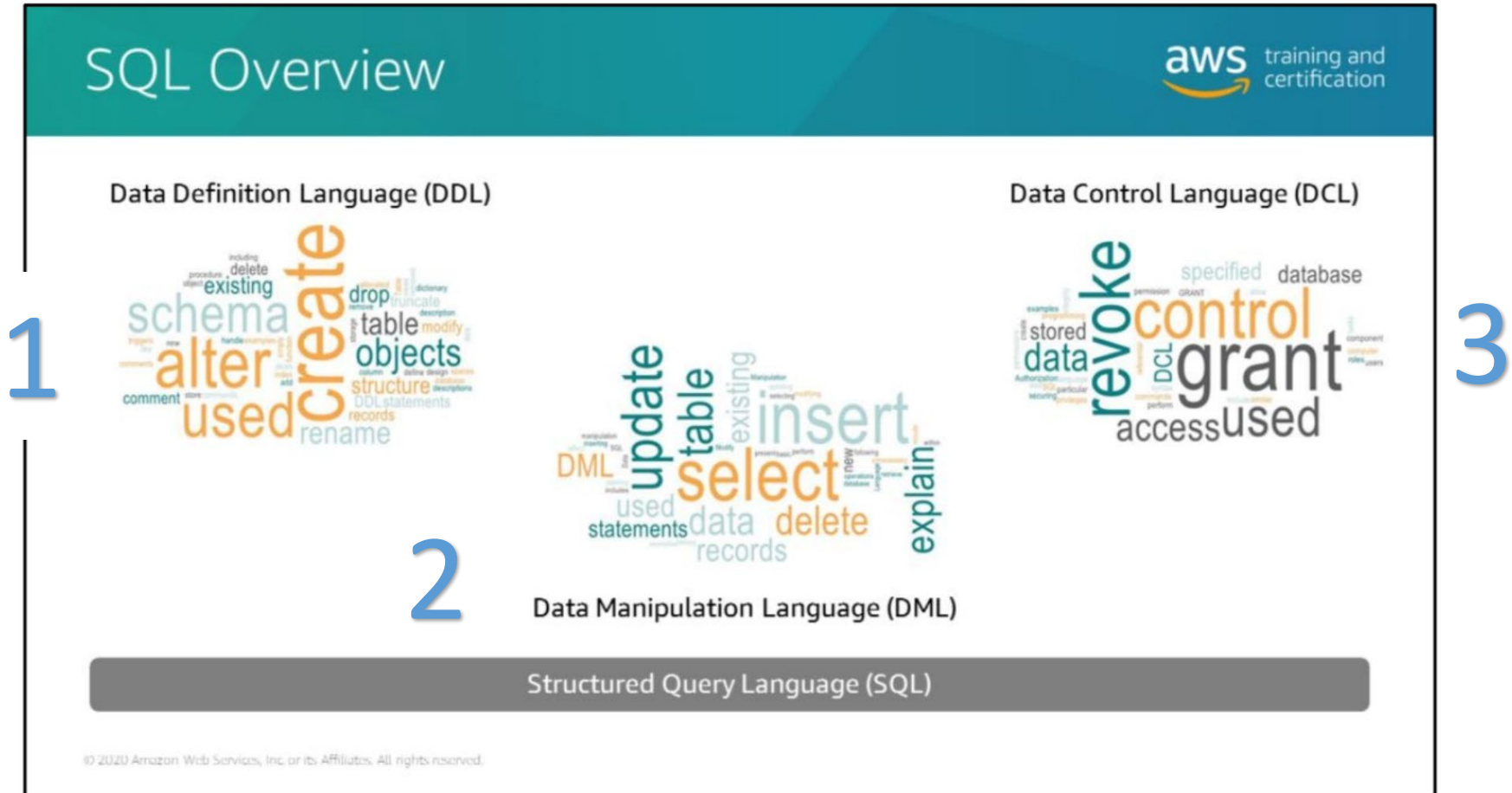
What is SQL?

- Structured Query Language
- NIST *forced* all the DBMS companies to write 1 **standard** language which they all recognize as a way to get data to/from their own proprietary db.
- The gov't had no problem with all the different methods the companies had for their product. But the gov't was only going to buy a DBMS if it had a standard **interface** to it.



SQL command subsets

3 "command types" or "command sets" which make up SQL.



SQL overview

- Some rules of the language:
 - Case insensitive
 - White space doesn't matter
 - Commands end in ;
 - Each command runs sequentially when ; seen
 - A command is a 'statement'
 - 3 types of comments:
 - # or -- for a line comment
 - C-style /* */ for multi-line
- 4 primary actions required for SQL DML:
 - C – create new data
 - R – read existing data
 - U – update existing data
 - D – delete existing data

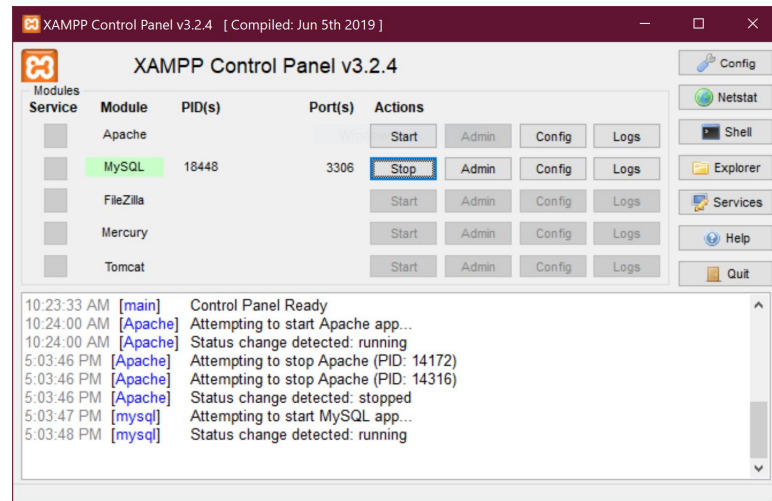
2. Setting up SQL client(s)

Our database management system

Our clients: MySQL client, phpMyAdmin client

Our DBMS app

- MySQL
- Ullman Ch 4; Thomson & Welling Ch 8
- Actually, we will use MariaDB. It is a *fork* of MySQL. But I'll still call it MySQL.
- The MySQL app came with XAMPP control panel installation. So if you installed XAMPP then you should have MySQL also.

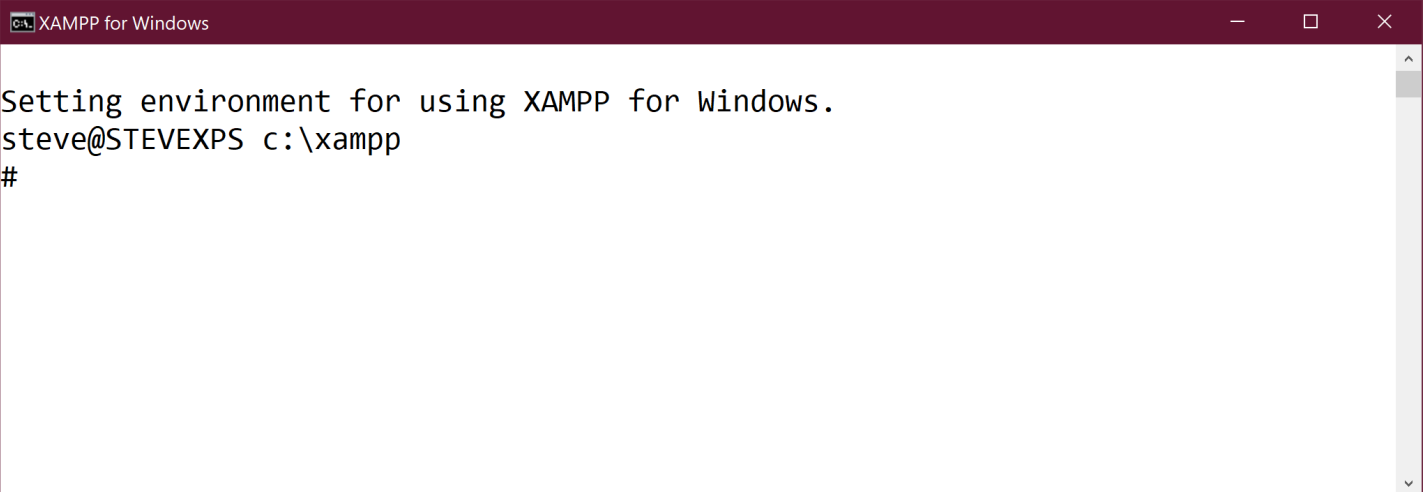


Accessing MySQL

- There are 3 main ways to send commands to MySQL:
 1. MySQL CLI – command line interface
 2. phpMyAdmin (web app)
 3. MySQL Workbench
- Prof Comments:
 - You will *have* to code in SQL from PHP. A UI (#2, #3) **won't** be helpful for that skill. Therefore, I will be teaching the **CLI** (#1)
 - phpMyAdmin is a nice web app. You have to fill out forms and push buttons. It's OK to use it, but I'm not teaching it.
 - **MySQL Workbench** is a *separate* download and install from the MySQL team (not XAMPP). I am not using this.


Accessing MySQL CLI

- In XAMPP control panel, click on "Shell"
- Right click on **border** to change **Properties** like font & colors. They are saved for *future* uses.
- The prompt is #.
- This is a Windows Command window.



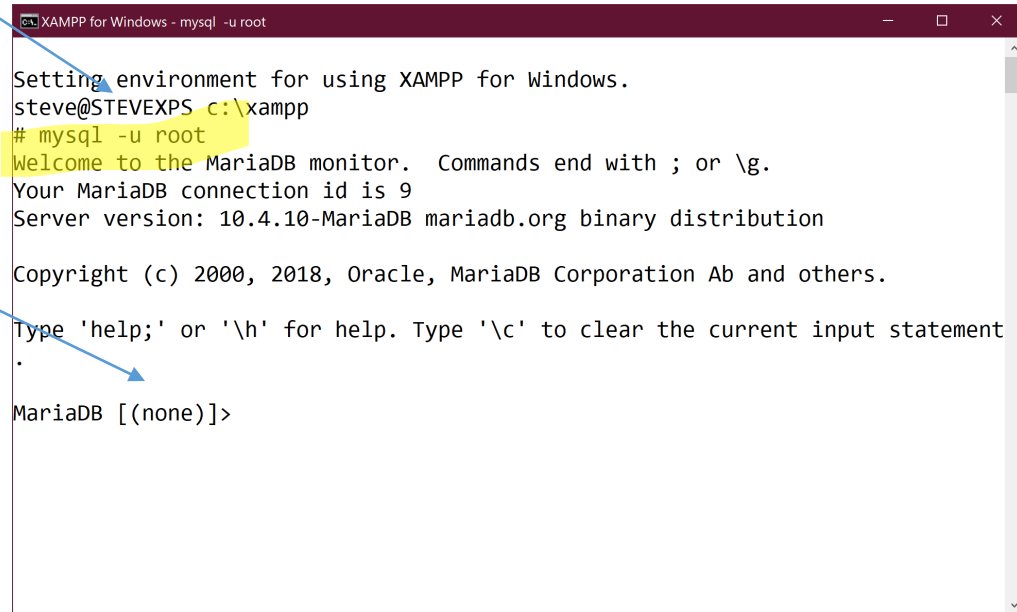
```
Setting environment for using XAMPP for Windows.  
steve@STEVEXPS c:\xampp  
#
```


Accessing MySQL CLI

- Access MySQL with command "`mysql -u root`" 
- (use "-u root" to be the 'root' user, so the phpMyAdmin can also see these tables)
- In my XAMPP installation, "root" didn't have any passwords. (whoa!) You will just go right to the MariaDB command prompt.

Key notes

- The "[(none)]" prompt is displaying the particular **database** being used right now in the DBMS.
- All commands end in ;
- Commands can span multiple lines. The ; tells mysql to execute.
- # is a comment



```
XAMPP for Windows - mysql -u root
Setting environment for using XAMPP for Windows.
steve@STEVEXPS c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.10-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.
MariaDB [(none)]>
```

Web Client UI

- phpMyAdmin is a very popular web app UI for MySQL.
- Click on "Admin" button on XAMPP CC or <http://localhost/phpmyadmin>

These are the defined databases

SQL tab is another CLI

The screenshot displays the phpMyAdmin web interface in a browser window. The browser's address bar shows the URL 'localhost/phpmyadmin/'. The interface features a navigation menu on the left side, which is highlighted with a red box. This menu lists several databases: 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', and 'test'. The main content area is divided into several sections. At the top, there are tabs for 'Databases', 'SQL', 'Status', 'User accounts', 'Export', 'Import', 'Settings', 'Replication', 'Variables', 'Charsets', and 'More'. The 'SQL' tab is highlighted with a red box. Below the tabs, the 'General settings' section shows the 'Server connection collation' set to 'utf8mb4_unicode_ci'. The 'Appearance settings' section shows the 'Language' set to 'English' and the 'Theme' set to 'pmahomme'. The 'Database server' section provides details about the server, including the server name '127.0.0.1 via TCP/IP', the server type 'MariaDB', the server connection 'SSL is not being used', the server version '10.4.10-MariaDB - mariadb.org binary distribution', the protocol version '10', the user 'root@localhost', and the server charset 'UTF-8 Unicode (utf8mb4)'. The 'Web server' section lists the server software 'Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.3.12', the database client version 'libmysql - mysqlnd 5.0.12-dev - 20150407 - \$Id: 7cc7cc96e675f6d72e5cf0f267f48e167c2abb23 \$', the PHP extension 'mysqli', and the PHP version '7.3.12'.

Hierarchy of data objects

- Hierarchy:
 - A. Database → collection of tables
 - B. Tables → data in 2-D rows (obs) & cols (variables)
- The first thing we need to know is a few database commands
- Database
 - We can only "use" 1 database at a time
 - > `show databases;`
 - > `create database <db>;`
 - > `use <db>;`

You should memorize these few SQL commands

3. CRUD

Using SQL for the 4 basic db actions

Create new data

Create new table & cols (DDL)

Insert new obs

Import from csv

The C in CRUD

Creating a table

- table name
- Columns
 - Required!
 - Type and size is required!

```
> create table <table>
  ( col1 type,
    col2 type
  ... ) ;
```

- Col names shouldn't have spaces (if you do, you'll have quote or escape them every time)
- Every col must have a type. They come in pairs.
 - *numeric*: INT
 - *string* : VARCHAR(N)

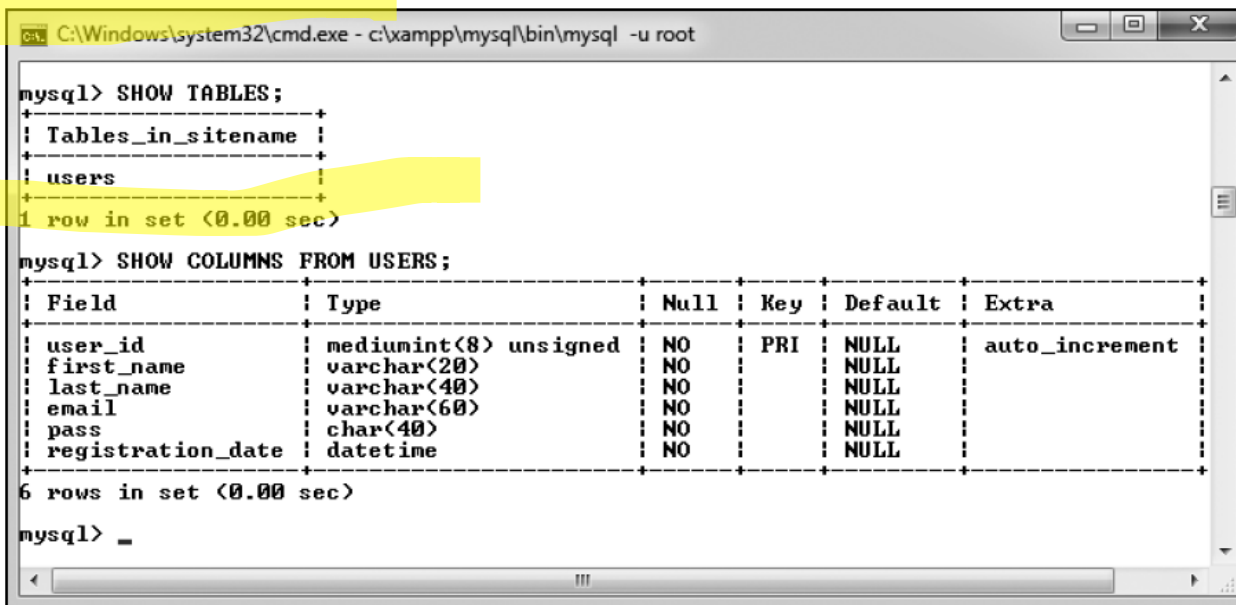
```
> create table students (
  id          int,
  firstname   varchar(20),
  surname     varchar(40),
  init_year   int );
```

cols

type

Show defined tables and their col definitions

```
> show tables;  
> show columns from <table>;  
> describe <table>; # same output
```



```
C:\Windows\system32\cmd.exe - c:\xampp\mysql\bin\mysql -u root  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_sitename |  
+-----+  
| users |  
+-----+  
1 row in set (0.00 sec)  
mysql> SHOW COLUMNS FROM USERS;  
+-----+-----+-----+-----+-----+-----+  
| Field          | Type                | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| user_id        | mediumint(8) unsigned | NO   | PRI | NULL    | auto_increment |  
| first_name     | varchar(20)         | NO   |     | NULL    |                |  
| last_name      | varchar(40)         | NO   |     | NULL    |                |  
| email          | varchar(60)         | NO   |     | NULL    |                |  
| pass           | char(40)            | NO   |     | NULL    |                |  
| registration_date | datetime            | NO   |     | NULL    |                |  
+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)  
mysql> _
```

D Confirm the existence of, and columns in, a table using the **SHOW** command.

Inserting records into a table

- 2 SQL formats to do this with INSERT command:

1. Insert values into *every* column in proper order
2. Or *specify* the columns and then the values

```
> insert into <table> values (v1, v2, v3);
```

```
> insert into <table> (col1, col3) values (v1, v3);
```

- Inserting multiple observations in one command is also allowed:

```
> insert into table values  
    (va1, va2, va3),  
    (vb1, vb2, vb3),  
    (vc1, vc2, vc3);
```


Load data from csv file

- Or you can use LOAD command
- Prep:
 - Table needs to be defined in SQL
 - (that means columns are defined and proper var type)
 - The col order in text file must match table order
- This can be tricky. Use the *template* below for the SQL command in the client



```
LOAD DATA local INFILE 'c:\\data\\country.csv'  
INTO TABLE country  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\\r\\n'  
IGNORE 1 ROWS
```

example

	A	B	C	D	E	F	G	H	I	J	K	L
1	make	model	trany	UCity	UHighway	VClass	year	cylinders	displ	drive	engld	fuelType
2	Alfa Romeo	Spider Veloce 2000	Manual 5-spd	23.33	35.00	Two Seaters	1985	4	2	Rear-Wheel Drive	9011	Regular
3	Ferrari	Testarossa	Manual 5-spd	11.00	19.00	Two Seaters	1985	12	4.9	Rear-Wheel Drive	22020	Regular
4	Dodge	Charger	Manual 5-spd	29.00	47.00	Subcompact Cars	1985	4	2.2	Front-Wheel Drive	2100	Regular
5	Dodge	B150/B250 Wagon 2V	Automatic 3-spd	12.22	16.67	Vans	1985	8	5.2	Rear-Wheel Drive	2850	Regular
6	Subaru	Legacy AWD Turbo	Manual 5-spd	21.00	32.00	Compact Cars	1993	4	2.2	4-Wheel or All-Wheel	66031	Premium
7	Subaru	Loyale	Automatic 3-spd	27.00	33.00	Compact Cars	1993	4	1.8	Front-Wheel Drive	66020	Regular
8	Subaru	Loyale	Manual 5-spd	28.00	41.00	Compact Cars	1993	4	1.8	Front-Wheel Drive	66020	Regular
9	Toyota	Corolla	Automatic 3-spd	29.00	37.00	Compact Cars	1993	4	1.6	Front-Wheel Drive	57005	Regular
10	Toyota	Corolla	Manual 5-spd	30.00	43.00	Compact Cars	1993	4	1.6	Front-Wheel Drive	57005	Regular
11	Toyota	Corolla	Automatic 4-spd	29.00	42.00	Compact Cars	1993	4	1.8	Front-Wheel Drive	57006	Regular
12	Toyota	Corolla	Manual 5-spd	30.00	42.31	Compact Cars	1993	4	1.8	Front-Wheel Drive	57006	Regular
13	Volkswagen	Golf III / GTI	Automatic 4-spd	23.00	36.00	Compact Cars	1993	4	2	Front-Wheel Drive	59007	Regular
14	Volkswagen	Golf III / GTI	Manual 5-spd	27.00	41.00	Compact Cars	1993	4	2	Front-Wheel Drive	59007	Regular

```

MariaDB [advwebdev]> create table cars (
  -> make varchar(24),
  -> model varchar(32),
  -> trany varchar(24),
  -> Ucity float,
  -> uhighway float,
  -> vclass varchar(32),
  -> year int,
  -> cylinders int,
  -> dist float,
  -> drive varchar(24),
  -> engld int,
  -> fueltype varchar(16),
  -> highway08 float,
  -> id int not null unique)
  -> ;
Query OK, 0 rows affected (0.02 sec)

```

```

XAMPP for Windows - mysql -u root
MariaDB [advwebdev]> describe cars;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| make       | varchar(24)   | YES  |     | NULL    |       |
| model      | varchar(32)   | YES  |     | NULL    |       |
| trany      | varchar(24)   | YES  |     | NULL    |       |
| Ucity      | float         | YES  |     | NULL    |       |
| uhighway   | float         | YES  |     | NULL    |       |
| vclass     | varchar(32)   | YES  |     | NULL    |       |
| year       | int(11)       | YES  |     | NULL    |       |
| cylinders  | int(11)       | YES  |     | NULL    |       |
| dist       | float         | YES  |     | NULL    |       |
| drive      | varchar(24)   | YES  |     | NULL    |       |
| engld      | int(11)       | YES  |     | NULL    |       |
| fueltype   | varchar(16)   | YES  |     | NULL    |       |
| highway08  | float         | YES  |     | NULL    |       |
| id         | int(11)       | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
14 rows in set (0.01 sec)

```

example

```
MariaDB [advwebdev]> LOAD DATA LOCAL INFILE 'c:\\Users\\steve\\Box Sync\\VTC\\Adv Web Dev\\2019_Spring\\labs\\lab 06 SQL\\cars.txt' INTO TABLE cars FIELDS TERMINATED BY '\t' lines terminated by '\n' IGNORE 1 ROWS;
Query OK, 40903 rows affected, 50017 warnings (2.69 sec)
Records: 40903 Deleted: 0 Skipped: 0 Warnings: 50017

MariaDB [advwebdev]>
```

```
MariaDB [advwebdev]> select count(*) from cars;
+-----+
| count(*) |
+-----+
|    40904 |
+-----+
1 row in set (0.04 sec)
```

Notes:

- Export from XL as *.txt tab delimited
- Use double slashed in filename
- Use the word LOCAL to get any directory
- '\t' is TAB and '\n' is newline

Read data

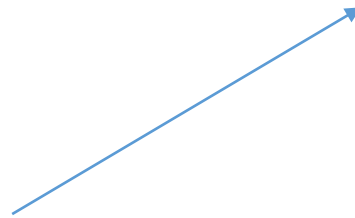
Retrieving observations (row) – i.e. viewing data

The R in CRUD

SELECT

- We use this command ALL the time!
- Write once... Read **many**
- It has a very set, fixed, predictable format
- Think of it as one *sentence*, but with many **optional phrases**.
- These phrases have a fixed mandatory order

Clauses **always** this order



```
Select    ...  
From      ...  
Where     ...  
Group by  ...  
Having    ...  
Order by  ...  
;
```

SELECT

- Use the `SELECT` statement to retrieve records from a table:

```
SELECT columns FROM table_name;
```

- Use the **asterisk** (*) wildcard with the `SELECT` statement to retrieve **all fields** from a table
- To return multiple fields, separate field names with a comma
- e.g.

```
select * from students ;
```

Sorting Query Results

- Use the `ORDER BY` keyword with the `SELECT` statement to perform an alphanumeric sort of the results returned from a query

```
SELECT make, model  
FROM inventory  
ORDER BY make, model;
```

- To perform a *reverse* sort, add the **DESC** keyword after the name of the field by which you want to perform the sort

```
SELECT make, model, year  
FROM inventory  
ORDER BY year desc, make, model;
```

Rename col (variable)

- You can have SQL rename the cols in the output

```
SELECT model_year as year ,  
       mileage as odometer  
FROM company_cars ;
```


Filtering Query Results

- You can also specify **which** *records* (observations) to return by using the `WHERE` keyword
- The Boolean `WHERE` clause is examined *for each row*.
- If **T**, the row is returned.
- But if the Where evaluation is **F**, then the row is not returned!

```
SELECT *  
FROM inventory  
WHERE make='Martin';
```

Boolean

Filtering Query Results

- Use the keywords `AND` and `OR` to specify more detailed conditions about the records you want to return
- `AND` Boolean logic:

```
SELECT *  
FROM company_cars  
WHERE model_year = 2007  
      AND mileage < 60000;
```

Filtering Query Results

- OR Boolean logic
- The WHERE clause can be pretty complex

```
SELECT *  
FROM company_cars  
WHERE ( make='Toyota' OR make='Honda' )  
      AND mileage < 60000  
ORDER BY mileage ;
```

In () operator

- A list of possible, accepted strings
- Where this or this or this or this...
- Very cool.

```
SELECT *  
FROM company_cars  
WHERE make IN ('Toyota', 'Honda')  
ORDER BY mileage ;
```

Wildcard strings

- The '%' character in a **LIKE()** means "any number of other characters"
- % is the SQL *wildcard*

```
SELECT make, model, year
FROM cars
WHERE make = 'Ford'
      AND model like ('Focus%')
      and year > 2011
ORDER BY year ;
```

UPDATE data

UPDATE to edit cols of existing rows

The U in CRUD

Updating Records

- To update records in a table, use the UPDATE statement
- The syntax for the UPDATE statement is:

```
UPDATE table_name
SET column_name=value
WHERE condition;
```
- The UPDATE keyword specifies the name of the table to update
- The SET keyword specifies the value to assign to the fields...
- *only in the records that match the condition in the WHERE keyword*



```
UPDATE company_cars
SET mileage=31568.2
WHERE make='Ford' AND model='Fusion'
and model_year=2015;
```

DELETE data

The D in CRUD

Deleting Records

- Use the `DELETE` statement to delete records in a table
- The syntax for the `DELETE` statement is:

```
DELETE FROM table_name
WHERE condition;
```

- The `DELETE` statement deletes **all records that match the `WHERE condition`**
- Note: to delete all the records in a table, leave off the `WHERE` keyword

```
DELETE FROM company_cars
WHERE model_year=2006
      AND make='Honda'
      AND model='Accord';
```

Deleting a table

- The term to delete a table is to *drop* it.

```
XAMPP for Windows - mysql
corresponds to your MariaDB server version for the right syntax to use near '//
foobar' at line 1
MariaDB [(none)]> use test;
Database changed
MariaDB [test]> show tables;
Empty set (0.00 sec)

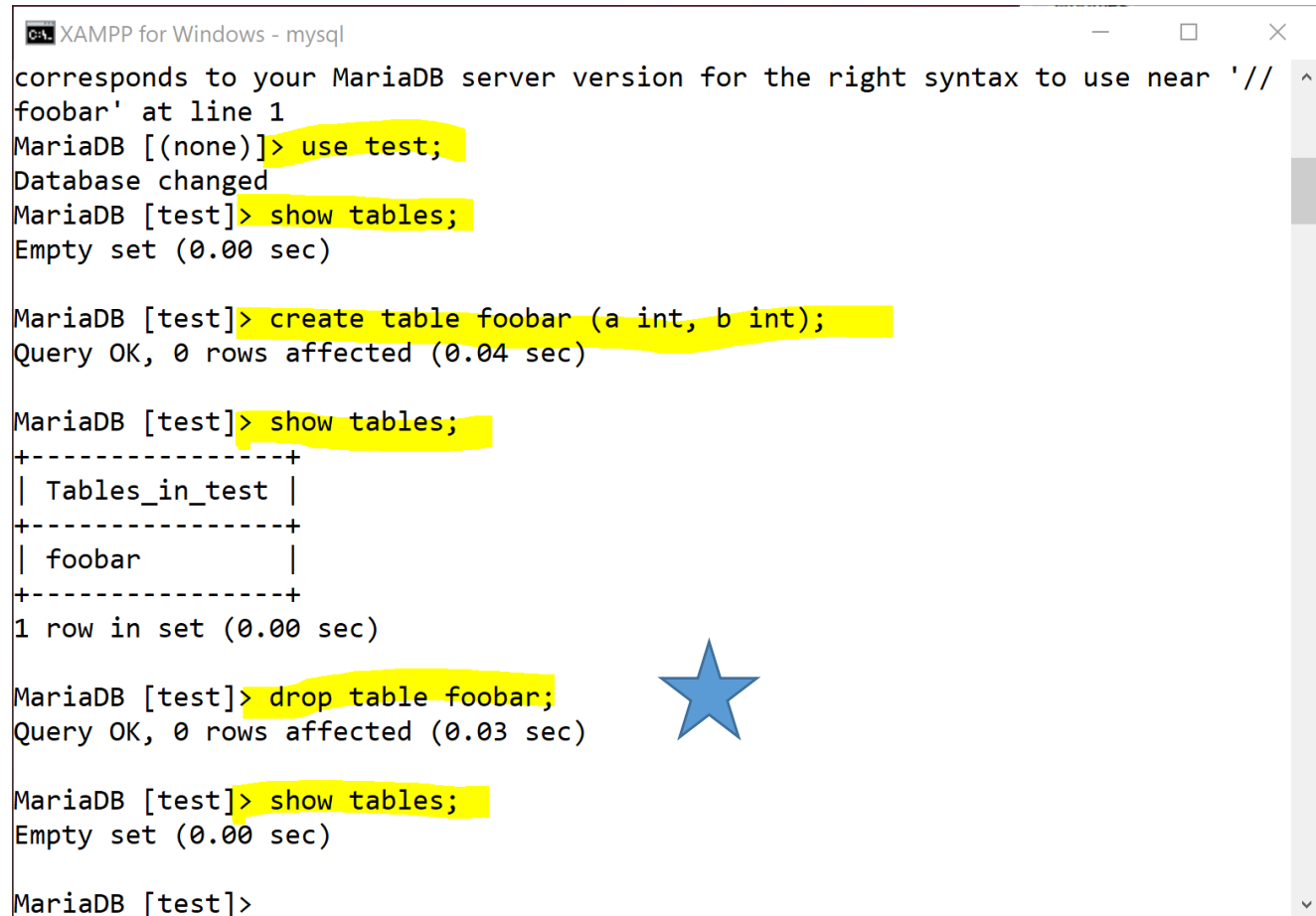
MariaDB [test]> create table foobar (a int, b int);
Query OK, 0 rows affected (0.04 sec)

MariaDB [test]> show tables;
+-----+
| Tables_in_test |
+-----+
| foobar          |
+-----+
1 row in set (0.00 sec)

MariaDB [test]> drop table foobar;
Query OK, 0 rows affected (0.03 sec)

MariaDB [test]> show tables;
Empty set (0.00 sec)

MariaDB [test]>
```



A screenshot of a MySQL terminal window titled "XAMPP for Windows - mysql". The terminal shows a sequence of commands and their outputs. The commands are: "use test;", "show tables;", "create table foobar (a int, b int);", "show tables;", "drop table foobar;", and "show tables;". The output of the first "show tables;" command shows an empty set. The output of the second "show tables;" command shows a table named "foobar" with columns "a" and "b", both of type "int". A blue arrow points to the "foobar" table in the output. The output of the "drop table foobar;" command shows "Query OK, 0 rows affected (0.03 sec)". The output of the final "show tables;" command shows an empty set. A blue star is placed to the right of the "drop table foobar;" command output.

4. Datatypes

Data types (columns)

- The more popular types:
 - char(n) or varchar(n)
 - text, mediumtext, longtext
 - smallint, int or bigint
 - float or double
 - date, datetime or timestamp
 - enum
- What to pick? You decide! You are the *architect*!

TABLE 4.5 users Table

Column Name	Type
user_id	MEDIUMINT
first_name	VARCHAR(20)
last_name	VARCHAR(40)
email	VARCHAR(60)
pass	CHAR(40)
registration_date	DATETIME

- **TINYINT** (-128, 128)
- **SMALLINT** (-32768, +32768)
- **INT** or INTEGER (2 Billion)
- **BIGINT** - (10**18 ish)

- **FLOAT** (32-bit) 10**38 with seven digits of accuracy
- **DOUBLE** (64-bit) 10**308 with 14 digits of accuracy

char vs varchar



- **char** – a fix width. Faster. Takes more HD space.
- **varchar** – variable width. Slower. Takes less HD space.
- Thoughts:
 - If the strings are 'always' the same length, go with **char**
 - If the lengths of the strings change quite a bit... And you need a "long length" just to handle a few observations, then definitely go with **varchar**.

Lab 6 DB setup and SQL

- Setup SQL client
- Single table creation and data insertion