Introduction to SQL on GRAHAM

ED ARMSTRONG

SHARCNET

AUGUST 2018

Background Information

Background Information
What is a (Relational) Database

- Dynamic collection of information.
- Organized into **tables**, **rows**, and **columns**.
- Often indexed to improve access time.
- They exist in a variety of flavours.

Background Information

Types of Databases

- Dynamic collection of information.
- Organized into **tables**, **rows**, and **columns**.
- Often indexed to improve access time.
- They exist in a variety of flavours.

- Distributed / NoSQL
- Object Oriented / PostgreSQL
- Relational / MySQL, MariaDB

Background Information

What is SQL

- **S**tructured **Q**uery **L**anguage
- The standard for accessing & manipulating relational databases.
- There is a standard for how SQL works.

Requesting a Database

Send a request to support@computecanada.ca with the following information:

- Your Compute Canada username.
- Amount of database space needed for your project.
- The system you would like an account on (Graham / Cedar).

We will create an account with a randomly generated password. The necessary information will be stored in a '.my.cnf' file in your home directory.

MySQL Configuration

```
[client]
ssl
ssl-cipher=DHE-RSA-AES256-SHA:AES128-SHA
user=your_username
password=YyG1ZJYRxkmdfV0U
database=your_username
host=199.241.163.99
```

Create, Use, Delete Databases

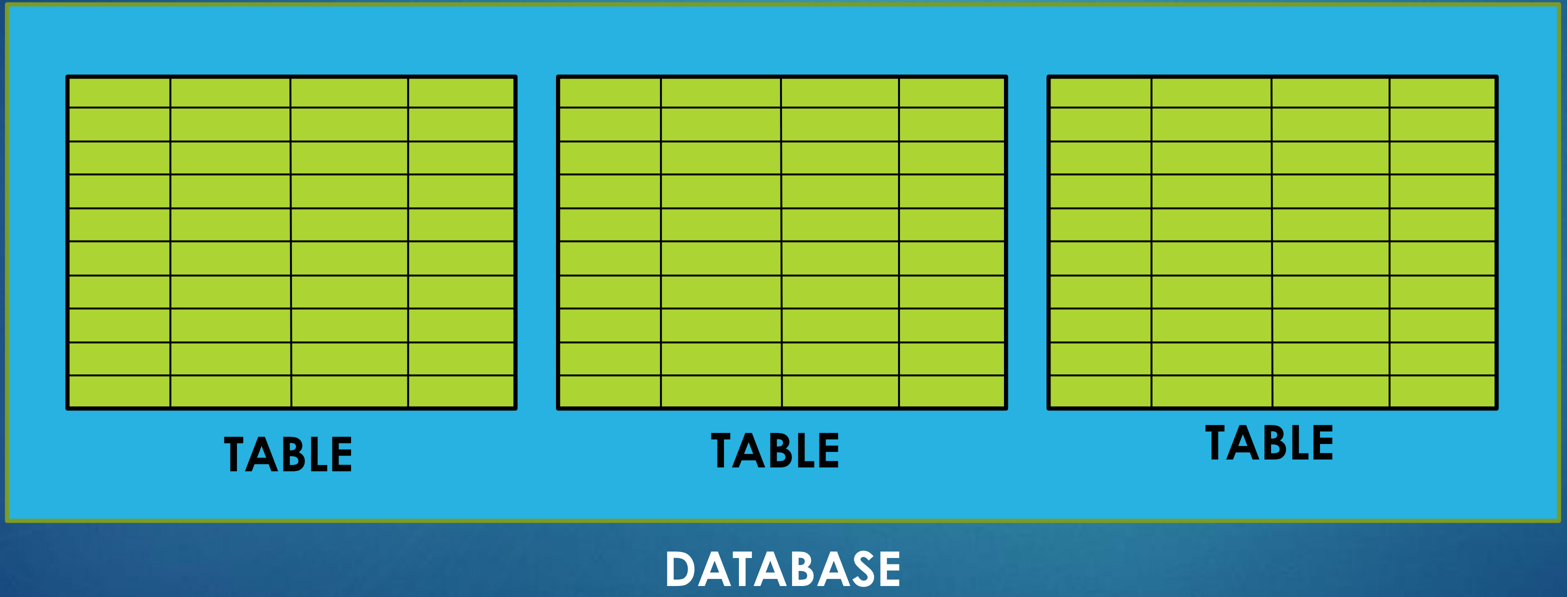
Create, Use, Delete Databases

```
$ ssh graham@computecanada.ca
$ mysql (with .my.cnf)
$ mysql -h hostname -u username (w/o .my.cnf)
$ mysql --local-infile=1
CREATE DATABASE my_database;
SHOW DATABASES;
USE my_database;
DROP DATABASE my_database; * You won't receive a warning
```

Tables

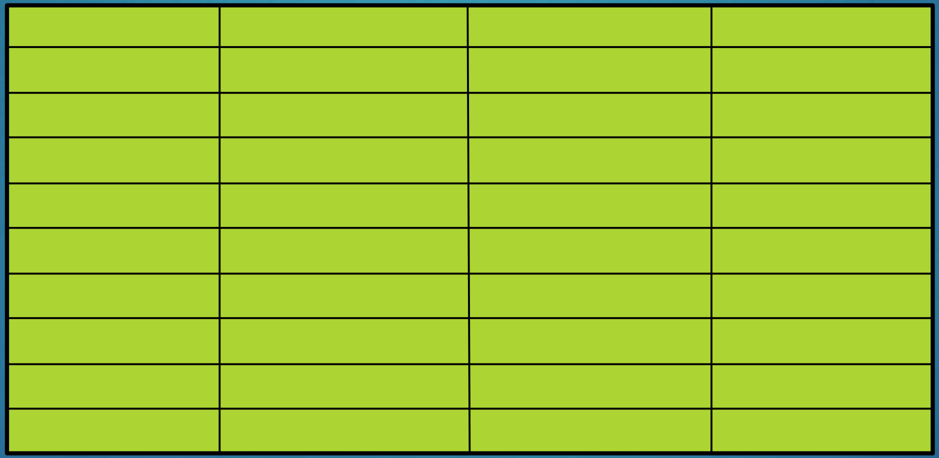
Tables

A database is a collection of tables.



Tables

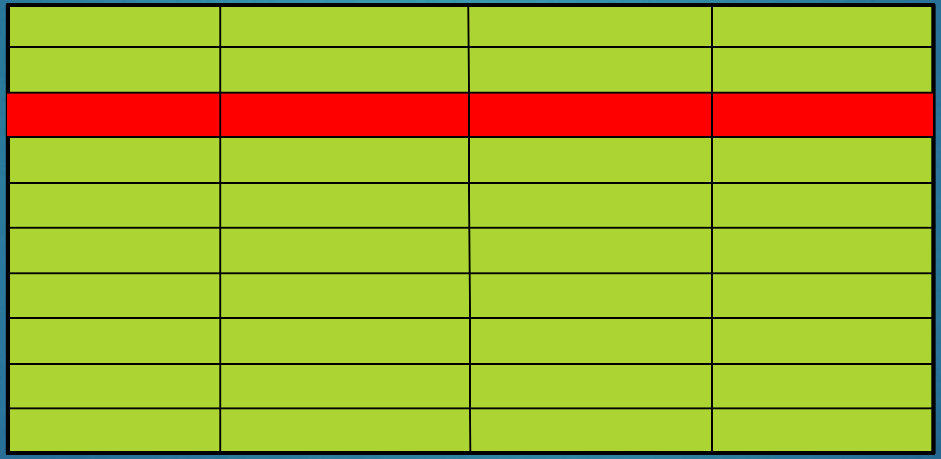
A database is a collection of tables.
A table is a collection of data entries (tuples).



TABLE

Tables

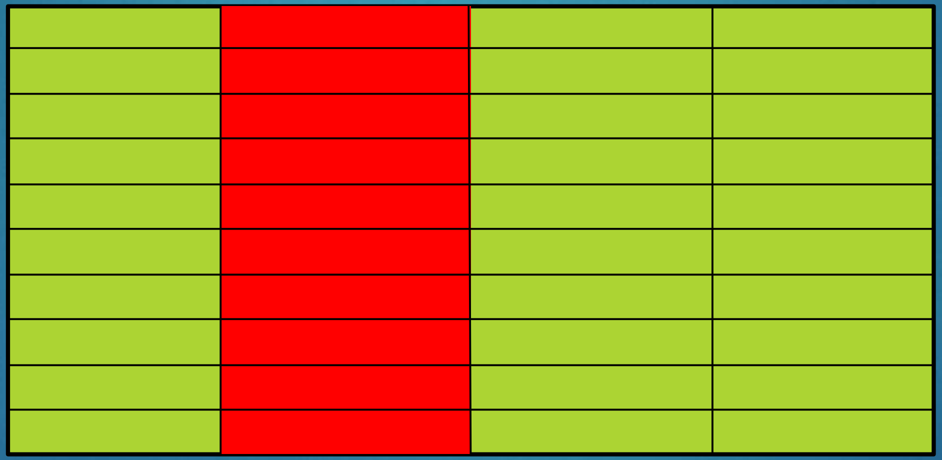
A database is a collection of tables.
A table is a collection of data entries (tuples).
An entry is a row.



TABLE

Tables

A database is a collection of tables.
A table is a collection of data entries (tuples).
An entry is a row.
A data point (type) is a column.



TABLE

Table Schema

The table schema describes the contents of a table.

NAME	AGE	BREED	COLOR

DOGS

Table Schema

The table schema describes the contents of a table.

NAME: STRING	AGE: NUMBER	BREED: STRING	COLOR: STRING

DOGS

SQL Data Types

SQL Data Types

Text Data Types	Number Data Types	Date Data Types
CHAR	TINYINT	DATE
VARCHAR	SMALLINT	DATETIME
TINYTEXT	MEDIUMINT	TIMESTAMP
BLOB	INT	TIME
MEDIUMTEXT	BIGINT	YEAR
LONGTEXT	FLOT	
LONGBLOB	DOUBLE	
ENUM	DECIMAL	
SET		

SQL Data Types

Text Data Types	Number Data Types	Date Data Types
CHAR	TINYINT	DATE
VARCHAR	SMALLINT	DATETIME
TINYTEXT	MEDIUMINT	TIMESTAMP
BLOB	INT	TIME
MEDIUMTEXT	BIGINT	YEAR
LONGTEXT	FLOT	
LONGBLOB	DOUBLE	
ENUM	DECIMAL	
SET		

SQL Data Types

- CHAR
- VARCHAR
- INT
- DOUBLE
- DATE

If your content is a fixed size, you'll get better performance with CHAR.

SQL Data Types

- **CHAR / VARCHAR**
- INT
- DOUBLE
- DATE

Variables in CHAR are fixed length string up to 255 characters in length.
Variable in VARCHAR are variable length strings up to 65,535* characters in length.

*Shared across all columns.

SQL Data Types

- CHAR/VARCHAR
- **INT**
- DOUBLE
- DATE

Type	**Storage (Bytes)**	**Minimum Value Signed**	**Minimum Value Unsigned**	**Maximum Value Signed**	**Maximum Value Unsigned**
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

Required Storage and Range for Integer Types Supported by MySQL
https://dev.mysql.com/doc/refman/8.0/en/integer-types.html

SQL Data Types

- CHAR/VARCHAR
- INT
- **DOUBLE**
- DATE

The FLOAT and DOUBLE data types are APPROXIMATE. If you require and exact decimal value, such as for currency, use DECIMAL.

SQL Data Types

- VARCHAR
- INT
- DOUBLE
- **DATE**

The DATE data type represents a calendar value. There are a number of interpretation rules that MySQL uses, as such you should stick to the SQL standard format (YYYY-MM-DD).

Creating a Table

Creating a Table

```
CREATE TABLE employees (
   name varchar(64),
   id int,
   start date
);
```

- SQL command
- Table name
- Column names
- Column data types

Setting and Getting Values

- A PREVIEW

```
INSERT INTO employees VALUES(
   'Adam',
   1,
   '2018-07-07'
);
```

```
SELECT * FROM employees;
```

```
INSERT INTO employees (name) VALUES (‘Adam’);
```

Creating a Table: Unsigned Integer

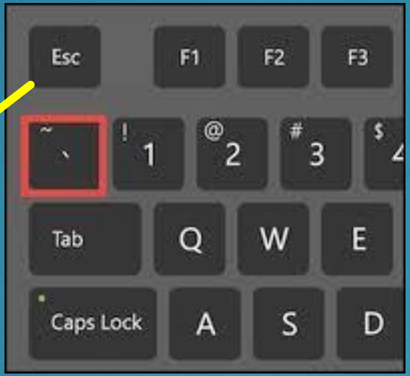
- Table name
- Table column names
- Table column data types

```
CREATE TABLE employees (
  name varchar(64),
  id int UNSIGNED,
  start date
);
```

Creating a Table:
- Reserved Keywords & Spaces

```
CREATE TABLE employees (
  `first name` varchar(64),
  `index` int UNSIGNED,
  start date
);
```

Creating a Table:
- Reserved Keywords



```
CREATE TABLE employees (
   `first name` varchar(64),
   `index` int UNSIGNED,
   start date
);
```

Inserting Data Into a Table

Inserting Data Into a Table

```
INSERT INTO employees (name, id, start)
values ('Adam', 1, '2018-07-07');
```

Inserting Data Into a Table

```
INSERT INTO employees(
  name,
  id,
  start
)VALUES(
  'Adam',
  1,
  '2018-07-07'
);
```

Inserting Multiple Data

```
INSERT INTO employees(name, id, start)
VALUES('Adam', 1, '2018-07-07')
     ,('Steve', 2, '2016-06-04')
     ,('Craig', 3, '2016-06-04');
```

Retrieving Your Data
SELECT

Retrieving Your Data

```
SELECT * FROM employees;
```

Retrieving Your Data

```
SELECT * FROM employees;
```

SQL Keywords

Retrieving Your Data

```
SELECT * FROM employees;
```

Column Selector

* *means all*

Table Selector

* *means all*

Retrieving Your Data

```
SELECT * FROM employees;
SELECT name FROM employees;
SELECT name, id FROM employees;
SELECT name, id, start FROM employees;
SELECT id, start, name FROM employees;
```

Terminating a Command

\c

Selecting Rows by Content
WHERE

Selecting Rows by Content

```
SELECT * FROM employees WHERE id = 3;
SELECT * FROM employees WHERE name = 'Adam';
SELECT * FROM employees WHERE name = 'ADAM';
SELECT * FROM employees WHERE name = 'A%';
SELECT * FROM employees WHERE binary name='Adam';
SELECT * FROM employees
   WHERE name='Adam'
   AND id = 7;
```

Change Existing Data

UPDATE

Change Existing Data

```
UPDATE employees SET name=‘Chris’
WHERE name=‘Adam’;
UPDATE employees SET start=‘2018-05-09’
WHERE id=‘1’;
UPDATE employees SET start=‘2000-01-01’
WHERE start IS null;
```

Removing Data

DELETE

Removing Data

```
DELETE FROM employees WHERE name='Adam';
DELETE FROM employees WHERE start=end;
DELETE FROM employees;
```

Executing SQL Files

SOURCE

Running SQL Files

```
$ mysql < instructions.sql
$ mysql --verbose < instructions.sql
$ mysql -e < “select * from table”
mysql> source instructions.sql
```

Importing Data

LOAD DATA

Load Data

```
LOAD DATA LOCAL INFILE 'file' INTO TABLE table
```

Load Data

```
LOAD DATA LOCAL INFILE ‘file’ INTO TABLE table
   fields
      terminated by ‘\t’
      enclosed by ‘’
      escaped by ‘\\’
   lines
      terminated by ‘\n’
      starting by ‘’;
```

Load Data

```
LOAD DATA LOCAL INFILE ‘file’ INTO TABLE table
  fields
    terminated by ‘,’
    enclosed by ‘’
    escaped by ‘\\’
  lines
    terminated by ‘\n’
    starting by ‘’;
```

Load Data

```
CREATE TABLE employees(
   first_name VARCHAR(64),
   last_name VARCHAR(64),
   id int AUTO_INCREMENT,
   start DATE,
   finish DATE,
   PRIMARY KEY(id)
)

LOAD DATA LOCAL INFILE 'employeeData.tab’
  INTO TABLE employees
  (first_name, last_name, @ignore, start, finish);
```

Save Data

```
mysql -ss -e "select * from employees" > data.tab
```