

# MySQL Workshop

Scott D. Anderson

# Workshop Plan

- Part 1: Simple Queries
- Part 2: Creating a database:
  - creating a table
  - inserting, updating and deleting data
  - handling NULL values
  - datatypes
- Part 3: Joining tables
- Part 4: complex queries with groups, subqueries and sorting
- Reference: <https://cs.wellesley.edu/~cs304/mysql-workshop/>



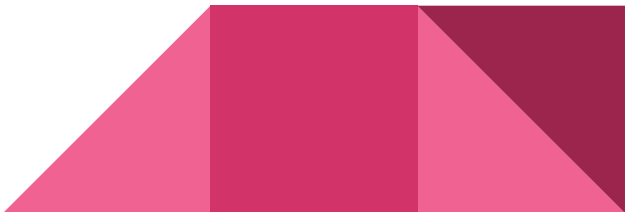
# Files for this part

Do this with me:

```
$ curl -O https://cs.wellesley.edu/~cs304/mysql-workshop/part2.tar  
$ tar xf part2.tar  
$ cd part2  
$ ls
```

If the database server is not running, start it:

```
$ mysqlctl start
```



# Creating a table

Defining a database table requires a lot of information:

- a **name** for the table
- one or more columns
- each column as a **name** and a **datatype**
- each table should have a **primary key**



# Creating the Person table

You'll find this in `person-table.sql`:

```
CREATE TABLE person (  
    nm int,  
    name varchar(30),  
    birthdate date,  
    addedby int,  
    primary key (nm)  
);
```

Use it and describe the table



# Creating the Movie table

Our next example is the Movie table:

```
CREATE TABLE movie (  
    tt int,  
    title varchar(30),  
    `release` char(4),  
    director int,  
    addedby int,  
    primary key (tt)  
);
```

the **release** column has backquotes because it's a reserved word



# Create table syntax

```
CREATE TABLE name (  
    colname datatype,  
    colname datatype,  
    colname datatype,  
    PRIMARY KEY (colname)  
);
```

There is much more, but this is a good start.



# Datatypes

- determine the storage requirements; e.g. 4 bytes for an integer
- determine the meaning of operations. Numbers are different from strings.

Compare

- `SELECT 3 < 11;`
- `SELECT '3' < '11';`
- You should choose a datatype that
  - Can represent all the values you (reasonably) want to represent, and
  - Doesn't (unreasonably) waste space
- Space is increasingly cheap, but wasting space can also waste time, as unneeded storage is searched, sorted, transmitted, etc.





# Simple Datatypes

- int for integer values, like class size or number of children
- float for floating point values: numbers with a decimal point, like 3.14
- char for character strings. E.g. char(4) for release year
- varchar for character strings of variable length, up to some maximum. E.g. varchar(30) for a person's name
- Note that int is for numbers that we might do arithmetic with. Numbers like zip codes and social security numbers would typically be char(5) and char(9)



# Dates and Times

- MySQL has several useful fourth-dimension types:
- `date` is like 2018-08-31
- `time` is like 13:01:30
- `datetime` is like 2018-08-31 13:01:30



# Exercise: Create a table

Create a table for a vet office.

Create a batch file for it.

It'll go in your personal database, which is c9 on cloud9, so put a "use" statement at the top.

My table has columns for name, date of birth, and weight



# Insert Data

Let's put some data into our table (`insert-pets1.sql`):

```
insert into pet(name)
values ("Santa's Little Helper");
```

```
insert into pet(name)
values ("Snowball"), ("Snowball II");
```


Do a select statement to show the rows. What are the NULLS?



# NULL values

- Database rows always have the same number of columns
- missing values (columns) are usually represented with NULL
- NULL is a special value. It is *not* a string ('NULL').
- Must be searched with special expressions:

```
select * from pet where weight = null;    -- WRONG
select * from pet where weight <> null;    -- WRONG
select * from pet where weight is null;    -- RIGHT
select * from pet where weight is not null; -- RIGHT
```




# More inserts and a problem

Use the `insert_pets2.sql` file to insert some more pets.

```
insert into pet(name,dob,weight) values
("Tiger","2011-01-01",11),
("Tiger","2013-01-01",8);    -- a different cat named Tiger
```

You'll see a problem: you can't have two primary keys with the same value:

```
$ mysql -u scottdanderson < insert-pets2.sql
ERROR 1062 (23000) at line 3: Duplicate entry 'Tiger' for
key 'PRIMARY'
```



# Primary Keys


- database tables are organized so that *searching* for rows based on the **primary key** is guaranteed to be *as fast as possible*.
- **but**, primary key values must be unique, like SSN
- Duplicate key errors are a good thing: you don't want data entry errors to cause your database to have bad data in it.
- Two solutions here:
  - We could add an "owner name" and make a two-part key, combining owner name and pet name: `primary key (owner, name)`
  - Or, use a made-up integer as the "pet id" or PID.



# Auto\_increment

- An integer primary key field can be declared "auto\_increment" and then a counter will be created and the value of the counter will be used.

```
drop table if exists pet;  
create table pet (  
    pid int auto_increment,  
    name varchar(30),  
    dob date,  
    weight float,  
    primary key (pid)  
);
```





# Exercise: revise your table

- revise your batch file to have a PID
- include the "drop table if exists" which makes it easy to re-run the batch file
- use the batch file to re-create the table
- re-insert the data using the two batch files (insert-pets1.sql and insert-pets2.sql)
- select \* from the table to see the PID values.
- You can also specify NULL or 0 for the PID value in the insert statement
- You can specify a PID value and override the auto\_increment feature



# The UPDATE Statement

See `update-pet1.sql`. Looks like some of the pets have gotten heavier:

```
update pet
set weight = 11.5
where name = 'Fluffy';
```

Better to use a PID. This updates the younger Tiger:

```
update pet set weight = 8.9 where PID = 6;
```

The WHERE clause is the same as in SELECT



# The DELETE statement

If we want to get rid of some rows, it's easily done:

```
DELETE FROM pet WHERE name = 'Tiger';  -- both of them
```

```
DELETE FROM pet WHERE pid = 4;
```

```
DELETE FROM pet;  -- I hope you meant to delete all rows
```

Be careful! MySQL deletes **all** matches and there is **no** "undelete."



# More Integer Datatypes

- all integers have limited range
- trade-off between number of bytes of storage and size of the range
- can also make them "**unsigned**" which means 0 to 2x instead of -x to +x
- See: [integer types](#)



# Sets and Enums

- enum is 1 or 2 bytes

```
sex enum('male','female')
```

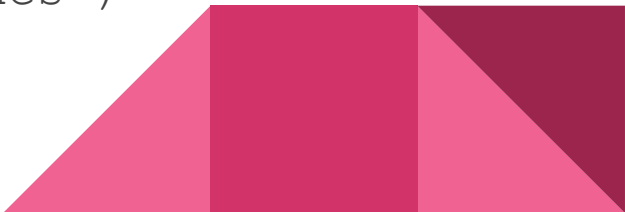
```
neutered enum('yes','no')
```

```
species enum('dog','cat','rabbit')
```

- set allows more than one choice:

```
behaviors set('bites','hisses','scratches')
```

- insert like a string



# Exercise


- Revise your pet table, adding some of the extra columns we talked about.
- Insert some rows
- Maybe create an "owner" table with some reasonable columns



# mysqldump

- Not everything is done with batch files:
  - We could update rows using the MySQL shell.
  - We might have an online application that updates our database via the web.
- So, we can use mysqldump to **dump** the current state of the database to a SQL batch file that can re-create it in one go:

```
mysqldump -u scottdanderson --add-drop-database --databases  
c9 > c9-after-part2.sql
```

- You can restore it just by loading the batch file
  - Experiment!
- 

# Summary of Part 2

- Create a MySQL table by defining the table name, column names and types, and a **primary key**
  - primary key requires uniqueness but guarantees speed
  - Datatypes like int, char, varchar, date
  - Datatypes determine operations and storage costs
  - `insert into table(col1,col2) values (val1,val2)`
  - `update table set col1 = val1, col2 = val2 where expression`
  - `delete from table where expression`
  - `mysqldump database > file.sql`
- 