Databases: SQL

Databases

What are databases?

- A way to work effectively with data
- Allows for **CRUD**
- Every database is broken into 3 parts:
    - The database itself
    - The tables within the database
    - Individual records on a table

What is CRUD?

Interacting with a database is broken down into 4 parts:

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

SQL

What is SQL?

- **S**tructured **Q**uery **L**anguage
- A language that allows us to interact with data in a database
- Made for relational database management systems (RDBMS)
- Was created by Donald D. Chamberlin and Raymond F. Boyce at IBM in the early 1970s

Installation (Mac Only)

```
brew install sqlite3
```

Creating Tables: `CREATE TABLE`

```sql
-- person.sql

CREATE TABLE person
(
  id INTEGER PRIMARY KEY,
  first_name TEXT,
  last_name TEXT,
  age INTEGER
);
```

```
sqlite3 DATABASE_NAME.db < FILE.sql
```

Viewing a Database's Structure

```
sqlite3 database.db
```

Then, in the SQL REPL:

```
.schema
```

Creating a Multi-Table Database

```sql
-- create_person_and_pet.sql

CREATE TABLE person
(
  id INTEGER PRIMARY KEY,
  first_name TEXT,
  last_name TEXT,
  age INTEGER
);

CREATE TABLE pet
(
  id INTEGER PRIMARY KEY,
  name TEXT,
  breed TEXT,
  age INTEGER
);
```

```
sqlite3 DATABASE_NAME.db < FILE_NAME.sql
```

Inserting Data: `INSERT INTO`

```sql
INSERT INTO person
  (id, first_name, last_name, age)
VALUES
  (0, "Jacques", "Cousteau", 42);

INSERT INTO pet
  (id, name, breed, age)
VALUES
  (0, "Roger", "Irish Wolfhound", 14);
```

Selecting Data: `SELECT`

```sql
SELECT *
FROM person;

SELECT name, age
FROM pet;

SELECT name, age
FROM pet
WHERE breed = "Dog";

SELECT *
FROM person
WHERE first_name != "Jacques";
```

Running Queries

```
sqlite3 --echo DATABASE_NAME.db < FILE_NAME.sql
```

Running Queries

```
sqlite3 --echo --header --column DATABASE_NAME.db < FILE_NAME.sql
```

Deleting Data: `DELETE FROM`

```sql
DELETE FROM person WHERE age = 42;

DELETE FROM person WHERE age = 42 AND first_name = "Jacques";
```

Updating Data: `UPDATE`

```sql
UPDATE person
SET first_name = "J"
WHERE first_name = "Jacques";

UPDATE person
SET first_name = "Jacques", last_name = "COUSTEAU"
WHERE first_name = "J";
```

Destroying Tables

```sql
DROP TABLE IF EXISTS person;
```

Altering Tables

```sql
ALTER TABLE person RENAME TO human;

ALTER TABLE person ADD COLUMN email TEXT;
```

Dates, Times and `ORDER BY`

```sql
SELECT *
FROM person
WHERE dob > date("now", "-100 years");

SELECT *
FROM person
ORDER BY first_name ASC;

SELECT *
FROM person
ORDER BY first_name DESC;

SELECT *
FROM person
WHERE dob > date("now", "-100 years")
ORDER BY dob ASC;
```

Aggregate Functions

```sql
SELECT avg(price)
FROM product;

SELECT min(price)
FROM product;

SELECT max(price)
FROM product;

SELECT sum(price)
FROM product;
```

JOINs

What are `JOIN`s?

- A way to combine records from two tables
- It locates related column values
- There are lots of different types of joins
    - Inner Join
    - Left Join
    - Right Join
    - Full Join

Joining Data: JOIN

```
SELECT *
FROM pet JOIN person ON person.id = pet.person_id;

SELECT *
FROM pet JOIN person ON person.id = pet.person_id
WHERE person.first_name = "Jack";

SELECT pet.full_name, pet.breed, person.first_name AS person_name
FROM pet JOIN person ON person.id = pet.person_id
WHERE person.first_name = "Jack";

SELECT *
FROM pet, person
WHERE person.id = pet.person_id;
```

Resources

- Khan Academy
- Codecademy
- Learn SQL The Hard Way
- SQL Zoo
- Visual Join
- SQL Join
- Do Factory: Joins

SQL and Ruby

Let's install a gem

```
gem install sqlite3
```

Let's use it!

```
# Create a connection to the database
db = SQLite3::Database.new 'database.db'

# Ask for the information in a nicer format
db.results_as_hash = true

# Write your SQL command
sql = "SELECT * FROM person"

# Show the SQL that was generated in the logs
puts sql

# Execute a line of SQL and store the result
result = db.execute sql
```