Week 4 Lecture 11

Applied

Helpful Resources

- http://www.sqlite.org/docs.html
- http://souptonuts.sourceforge.net/readme_sqlite_tutorial.html

What's in this lecture?

- Introduction to Databases
- SQLite

The Basics

- Need a way to persist (lots of) data
- Must allow for fast access
- Division of data typically maps to models
- Reads and writes should not corrupt data

The Problem

- Let's imagine Mr. PHB, our boss, decides that for every person in our application we create a file based on their username: username.txt
- What happens when we want :
  - to add an attribute to the person object?
  - search on a specific attribute?

The Solution

- Relational databases store data in such a way that:
  - querying by attributes is really easy
  - restructuring data is possible in a clean way

The Organizational Structure

Table: ModelName

Primary Key	Column 1	Column 2
1	“foo”	“bar”
2	“baz”	“qux”

Structure Notes

- Primary keys are unique
- Tables map to models
- Columns are model attributes
- Rows are records of data

Notes:

- Our applications use multiple databases
- A database can have multiple tables
- A table can have multiple columns
- Columns are mapped to data types
- A table has a row for each instance of the data type

Actual Record Data:

- Multiple records may have similar data:

ID	Name	Age
1	“Kip”	24
2	“Joe”	24

- Attributes have specified data types:
  ID -- INTEGER PRIMARY KEY
  Name -- TEXT
  Age -- INTEGER

Why this structure?

- Able to index on columns, allowing for fast and efficient retrieval
- Duplicate data is reduced by specialization
- Suited to queries based on attribute values

Introducing SQLite

- Uses SQL syntax
- Codebase is small and efficient
- Extremely well tested
- Writes data to flat file

Creating a database

- From scratch:
  $ sqlite new_database_name.sqlite3
- The filename and extension are arbitrary -- Rails will use .sqlite3 (3 for the version number) so let's stick with that
- Creates a flat file from which we can...

Create tables

- sqlite> CREATE TABLE table_name (
  ------->col_name1 DATATYPE,
  ------->col_name2 DATATYPE);

- Available datatypes:
  TEXT, REAL, INTEGER, BLOB

Queries I

- Getting data out:
  sqlite> SELECT * FROM table_name;
- Getting specific data out:
  sqlite> SELECT * FROM table_name
  WHERE col_name >= value;

Queries II

- Updating a record:
  sqlite> UPDATE table_name SET col_name = value WHERE another_col = some_val;

- Inserting a record:
  sqlite> INSERT INTO table_name (col_a,col_b) VALUES (val_a, val_b);

- Deleting a record:
  sqlite> DELETE FROM table_name WHERE condition;

Schema

```
sqlite> .schema
CREATE TABLE "graph_points" ("id" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL, "x_coord" float, "y_coord" float,
"created_at" datetime, "updated_at" datetime);
CREATE TABLE "schema_migrations" ("version" varchar(255)
NOT NULL);
CREATE UNIQUE INDEX "unique_schema_migrations" ON
"schema_migrations" ("version");
```

Stuck?

- sqlite> .help <return>

Exercises

- Choose two of the following and build a one table schema in SQLite for each:
  - TODO list
  - contact database
  - web bookmark list
  - product catalog
- Populate each with ~15 rows of data