Intro to SQL

Focusing on Sqlite

Richard Morrill

Fordham University CS Society

Wednesday, February 26th, 2020

Game Plan

- Installing Sqlite
- Why you need SQL
- What is SQL
- What is Sqlite
- Basics of SQL syntax and database design
- How adding a SQL[1] database to an existing project can simplify your code

[1] If pronounced "sequel", "a" is the correct article.

CS SOCIETY
FORDHAM UNIVERSITY™

Let's get it Installed!

Mac and Linux

There's a 99% chance it's already installed. Open a terminal window and try `sqlite3`, and if that doesn't work, `sqlite`. If for some reason your OS didn't come with it, install it at `https://www.sqlite.org`

Windows

You could have unknowingly installed Sqlite when installing something else. Give it a try in Powershell or CMD, but it's likley you'll need to install Sqlite at: `https://www.sqlite.org`

If it works you will see:
(The specific version does not matter.)

```
SQLite version 3.30.0 2019-10-04 15:03:17
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

CS SOCIETY
FORDHAM UNIVERSITY™

What’s the point of this, anyways?

Non-CS majors that don’t know SQL say, “Can’t I just use a GUI tool like MS Access?”

CS majors that don’t know SQL say, “There’s plenty of tools in the language I’m already using, why do I need another one?”

The power of SQL

- SQL is a *declarative language*.

The power of SQL

- SQL is a *declarative language*.You don't need to worry about *how* the database engine is going to do something, just tell it what you *want it to do*.

The power of SQL

- SQL is a *declarative language*.You don't need to worry about *how* the database engine is going to do something, just tell it what you *want it to do*.
  - SQL lets you think about your data on a higher level than if you were dealing with it in objects and arrays.
  - Learning to think *declaratively* can help you write more readable code, even when working in languages other than SQL.

The power of SQL

- SQL is a *declarative language*. You don't need to worry about *how* the database engine is going to do something, just tell it what you *want it to do*.
  - SQL lets you think about your data on a higher level than if you were dealing with it in objects and arrays.
  - Learning to think *declaratively* can help you write more readable code, even when working in languages other than SQL.
- Even if your language provides its own set of data-manipulation tools, they're probably just different names for things SQL already does.

The power of SQL

- SQL is a *declarative language*.You don't need to worry about *how* the database engine is going to do something, just tell it what you *want it to do*.
    - SQL lets you think about your data on a higher level than if you were dealing with it in objects and arrays.
    - Learning to think *declaratively* can help you write more readable code, even when working in languages other than SQL.
- Even if your language provides its own set of data-manipulation tools, they're probably just different names for things SQL already does.
    - If you learn to use the real thing, you'll be able to figure out pretty much any pseudo-sql stuff you come across.

CS SOCIETY
FORDHAM UNIVERSITY™

The power of SQL

- SQL is a *declarative language*.You don't need to worry about *how* the database engine is going to do something, just tell it what you *want it to do*.
    - SQL lets you think about your data on a higher level than if you were dealing with it in objects and arrays.
    - Learning to think *declaratively* can help you write more readable code, even when working in languages other than SQL.
- Even if your language provides its own set of data-manipulation tools, they're probably just different names for things SQL already does.
    - If you learn to use the real thing, you'll be able to figure out pretty much any pseudo-sql stuff you come across.
- SQL gets to the point.

CS SOCIETY
FORDHAM UNIVERSITY™

A Bit About SQL

- *Structured Query language*

A Bit About SQL

- *Structured Query language*
- Developed by IBM in the 1970s

A Bit About SQL

- *Structured Query language*
- Developed by IBM in the 1970s
  - Like other languages developed by committee, has its weird bits.
  - Syntax isn't always consistent with itself.

A Bit About SQL

- *Structured Query language*
- Developed by IBM in the 1970s
    - Like other languages developed by committee, has its weird bits.
    - Syntax isn't always consistent with itself.
- The gold standard for managing *relational databases*.

A Bit About SQL

- *Structured Query language*
- Developed by IBM in the 1970s
  - Like other languages developed by committee, has its weird bits.
  - Syntax isn't always consistent with itself.
- The gold standard for managing *relational databases*.
  - A bit of a victim of its popularity, SQL isn't always implemented identically across database software from different vendors.
  - 90% of syntax is the same, but some things you "get away with" on one database you won't in others.

CS SOCIETY
FORDHAM UNIVERSITY™

A Bit About SQL

- *Structured Query language*
- Developed by IBM in the 1970s
  - Like other languages developed by committee, has its weird bits.
  - Syntax isn't always consistent with itself.
- The gold standard for managing *relational databases*.
  - A bit of a victim of its popularity, SQL isn't always implemented identically across database software from different vendors.
  - 90% of syntax is the same, but some things you "get away with" on one database you won't in others.
- Technically a fully-fledged programming language, although rarely used this way[2].

[2] With one important exception we'll see later.

Relational Databases

Very few pieces of data exist in a vacuum

Relational Databases

Very few pieces of data exist in a vacuum

Department

<u>Name</u>	Manager	Building

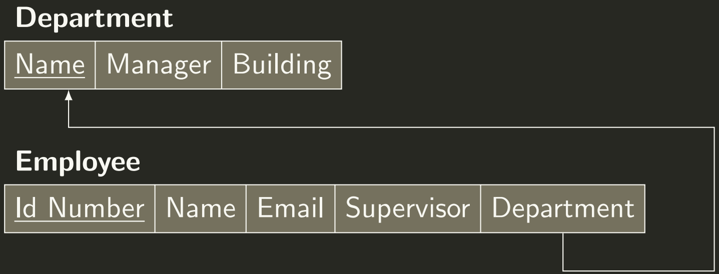
Employee

<u>Id Number</u>	Name	Email	Supervisor	Department

Insert about 100 more tables here . . .

Relational Databases

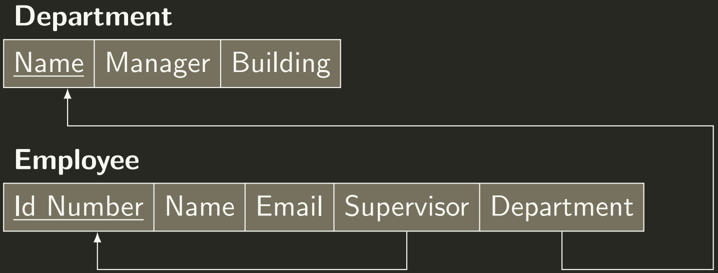
Very few pieces of data exist in a vacuum



Insert about 100 more tables here . . .

Relational Databases

Very few pieces of data exist in a vacuum



Insert about 100 more tables here . . .

Consistency

- Consistency is key to making relational systems work smoothly.
  - Imagine what would happen if half of the employees called their department “facilities”, while the other half called it “maintenance”.

Consistency

- Consistency is key to making relational systems work smoothly.
    - Imagine what would happen if half of the employees called their department “facilities”, while the other half called it “maintenance”.
- With many different sources of data and users, maintaining consistency can be very difficult.

Consistency

- Consistency is key to making relational systems work smoothly.
  - Imagine what would happen if half of the employees called their department "facilities", while the other half called it "maintenance".
- With many different sources of data and users, maintaining consistency can be very difficult.
  - You can educate users $\rightarrow$ Like that's gonna happen...

Consistency

- Consistency is key to making relational systems work smoothly.
  - Imagine what would happen if half of the employees called their department "facilities", while the other half called it "maintenance".
- With many different sources of data and users, maintaining consistency can be very difficult.
  - You can educate users $\rightarrow$ Like that's gonna happen...
  - You can write code that "gaurds" your data store $\rightarrow$ Closer, but what if you make a mistake?

CS SOCIETY
FORDHAM UNIVERSITY™

Consistency

- Consistency is key to making relational systems work smoothly.
  - Imagine what would happen if half of the employees called their department "facilities", while the other half called it "maintenance".
- With many different sources of data and users, maintaining consistency can be very difficult.
  - You can educate users $\rightarrow$ Like that's gonna happen. . .
  - You can write code that "gaurds" your data store $\rightarrow$ Closer, but what if you make a mistake?
- SQL builds consistency protection directly into the data storage[3].

[3] This is assuming the SQL database is set up properly.

Consistency

- Consistency is key to making relational systems work smoothly.
  - Imagine what would happen if half of the employees called their department "facilities", while the other half called it "maintenance".
- With many different sources of data and users, maintaining consistency can be very difficult.
  - You can educate users → Like that's gonna happen...
  - You can write code that "gaurds" your data store → Closer, but what if you make a mistake?
- SQL builds consistency protection directly into the data storage[3].
  - Bad data is rejected... No Exceptions!
  - Strict column types
  - Foreign keys: Can't insert an employee if his department is non-existent.
  - Triggers: Arbitrary code that can stop mistakes in their tracks.

[3]This is assuming the SQL database is set up properly.

Basics of SQL: Making a Table

1. Open a terminal.
2. Navigate to a directory you can use for temporary files.
3. Type in `$ sqlite demo.db`
4. On the `sqlite>` prompt that will come up, enter the commands below.

IN SQL WE ALWAYS SHOUT!

```
CREATE TABLE "employee" (
    "id" INTEGER PRIMARY KEY,
    "name" VARCHAR,
    "dob" VARCHAR,
    "joinDate" VARCHAR
);
```

To see if it worked:

```
.tables
.schema "employee"
```

Basics of SQL: Inserting and Reading Data

To make things more readable:

```
.mode column
.nullvalue NULL
.headers on
```

Add Some Data:

```sql
INSERT INTO "employee"
     VALUES (12, "Sam Smith", "1980-05-06", "2008-06-22");
INSERT INTO "employee" (id, "dob")
     VALUES (44, "1968-04-18");
```

Read It Back:

```sql
SELECT * FROM "employee";
SELECT "id", "name" FROM "employee";
```

CS SOCIETY
FORDHAM UNIVERSITY™