January 31, 2019

# Relational Basics, Introduction to SQL and SQLite

What is a relation?

* **A relation corresponds to a type of thing**
  + A database can contain many relations
    - A relation is the structure of the data
      * EG. *Title, year, length, genre* for a movie relation
        + Includes the data types for each of these attributes
  + A relation contains many individual instances of the same thing
  + All instances have the same structure
    - Can have a relation with no instance and a relation with a billion instances, but they must always have the same structure
      * Some attributes can be blank, but they must be present
  + No two instances are the exact same
    - For example, two of the same movies cannot be in our relation
* Actions can be taken on a relation
  + Relational Algebra
  + Constraints

**A table is a relation that is stored on disk**

* This is the exact definition, but be loose about it for now

**An instance is a ‘row’ in our relation**

* Row is intuitive but slightly inaccurate
* Also referred to as a tuple

**An attribute is a ‘column’**

* Straightforward

**Schema is *all* the relations in a database**

* What are the ‘rules’ of the database?
* Very important
  + Schema can make or break a database and make or break the ease of interacting with that database via our application

Schema Shorthand:

*Movies(title, year, length, genre)* -> a description of the form of the data

Rules for attributes:

* **Attributes must be elementary type**
  + Number, string, bool, etc.
* **NO COMPOUND DATA STRUCTURES**
* **Every attribute must only be one type**
  + No flopping between strings and ints and on

SQL is a loose standard:

* ANSI SQL is the closest thing to a true standard

The attribute(s) that identify a row is called a **key**

**Every database has a particular real-world subject matter**

* This is called a **domain**
* It requires domain knowledge to make the schema of that database properly
  + You can’t make a good schema is you don’t understand the domain

Domain knowledge is king

* Without domain knowledge, we would be making sub-optimal choices for our database

Sometimes, there are no truly unique combinations of attributes

* You could declare the **key** to be **all attributes** of a specific relation
  + Beyond that, making unique ids is a last resort
    - Not a bad option, just unfortunate

EXAMPLE:

*Movie(title, year, length, genre)*

*Stars(name, address, dob)* -> there are a few options for key here, but dob alone is bad (name alone is wrong, but we go with it because it is the example in the book)

*Executive(name, address, union id)*

*Studio(name, address)*

**Relationships between Relations**:

One-to-one – Each *x* refers to one *y* and each *y* refers to one *x*

One-to-many – Each *x* can refer to one or many *y*, but each *y* can only be related to one *x*

Many-to-many – Same logic as above

**Modeling Relationships**

* For a one-to-one relationship, we add the key of one of the relations to the other
* For a one-to-many relationship, we add the key of the one, to the relation of the many
* For many-to-many relationships, we must create a NEW relation which contains the keys to both sides

TIP: Redundancy means that we have to write more code and more complex code which is going to create issues down the line

**Creating a table in SQL**

CREATE TABLE *tablename*(attribute type, attribute type, …)

SQL is case *insensitive,* but it is common practice to make keywords all caps and user-defined things proper/lowercase

Example:

CREATE TABLE Movies(title VARCHAR(25), year INT, length INT, genre VARCHAR(15))

**Modifying a schema:**

DROP TABLE Movies

There is no coming back from this unless you have a backup.

ALTER TABLE MOVIES ADD studioName CHAR(30)

ALTER TABLE MOVIES DROP year

Making a default for a column

ALTER TABLE MOVIES ADD studioName CHAR(30) DEFAULT ‘Independent’

CREATE TABLE Star(name VARCHAR(20) PRIMARY KEY, address VARCHAR(255), date\_of\_birth DATE)

CREATE TABLE Movies(title VARCHAR(25), …, PRIMARY KEY(attribute\_1, attribute\_2)

INSERT INTO Movies(title, year, genre)

VALUES(‘Dark Knight’, 2008, action)

**Relational Operations:**

Union, Intersection, and Difference

Projection is a **unary** operator. It allows you to select a particular set of columns and slice off the rest

* We use this because we only want to pull only what we need from the database

Selection is also a **unary** operator. It will pull only specific rows.

* NEVER changes the number of columns

SELECT – for projections

WHERE – for selections

FROM – which relation are we operating on

Basic function:

SELECT \* FROM Movies

SELECT genre FROM Movies

SELECT \* FROM Movies WHERE year == 1998

SELECT genre FROM Movies WHERE year == 1998

Expressions:

AND, OR, <=, <, >, >=, = and <> (not equal)