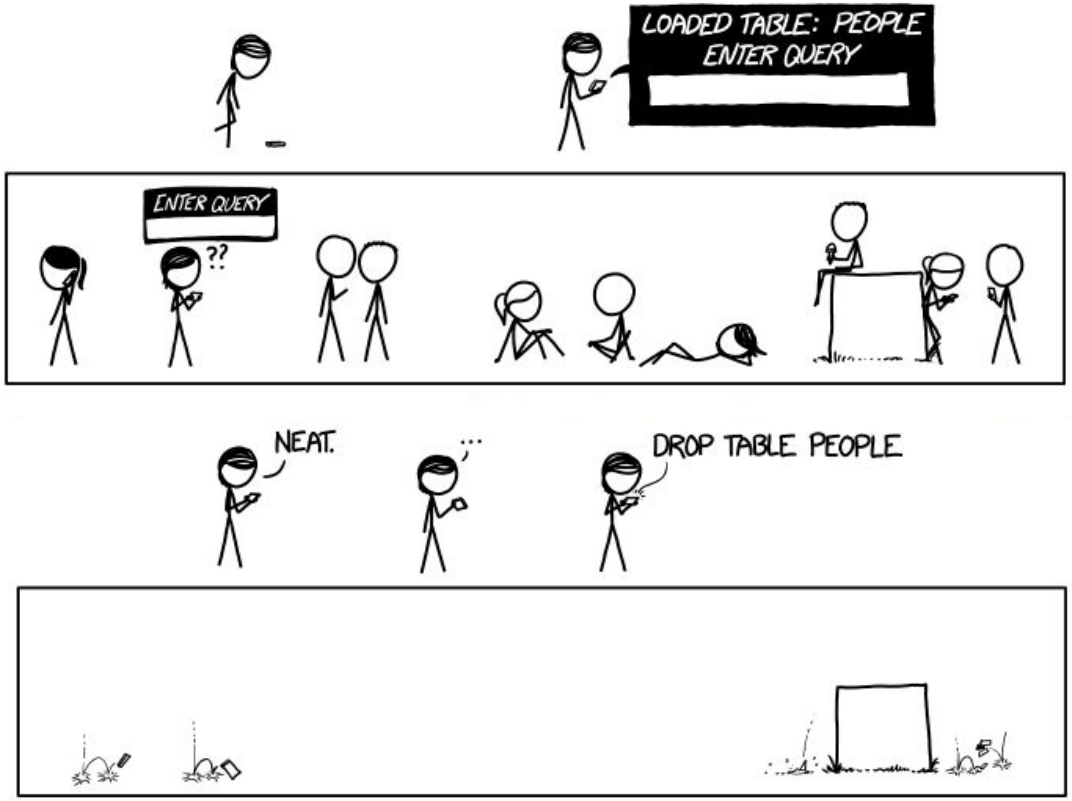
03_sql_basics

Kyle Shannon
kshannon@ucsd.edu
HDSI & Dept. of Cognitive Sciences
UC - San Diego



Source: XKCD

Relational Model Review for SQL 1/3

- A database consists of several **tables (relations)**
- Each table has one or more columns named by attributes
- Each attribute has an associated **domain (set of allowed values)**
- Data in each table consists of a set of **rows (tuples)** providing values for the attributes
- Each table has a **primary key** that uniquely identifies each row (tuple)
- Tables are related to other tables through **foreign keys**

Relational Model Review for SQL 2/3

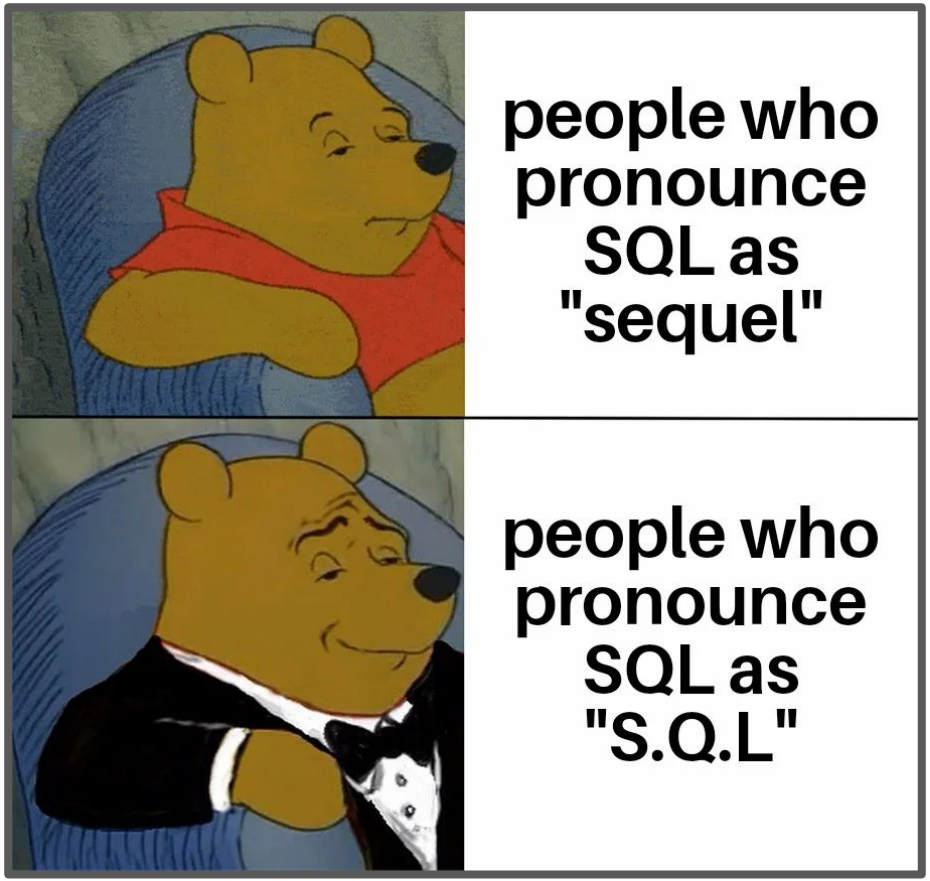
- A database's tables are **unordered** collections of data (sets)
- Likewise, rows **are not** considered to be ordered
  - Do not get confused just because the PK is often {1,2,3, …}
- Tuples are often **ordered in a query** via SQL statements
- Row ordering has no impact on functionality
- The RDMS determines the stored order of rows, not the PK or user.
  - This is to say the on disk physical storage

Relational Model Review for SQL 3/3

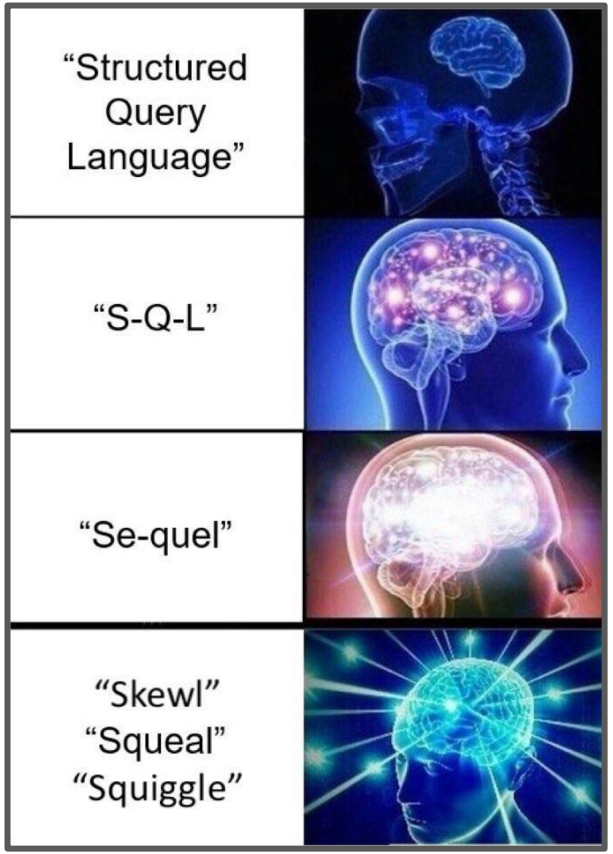
- Physical data independence
  - Changes can be made to the **physical storage** of the data **without affecting the logical schema** or applications that use the data
  - Allowing **performance improvements** can be made
- 1NF - all relations must be flat
  - Recall the example of courses:

StudentID	StudentName	Courses	Email
1	Alice	Math, Science	a@ucsd.com
2	Bob	History	b@ucsd.com
3	Charlie	Math, History, Geography	c@ucsd.com
4	Areeb	Science, History	ab@ucsd.com

OK moving on… First thing first. Names matter.



But do not worry. “They” can’t decide either.



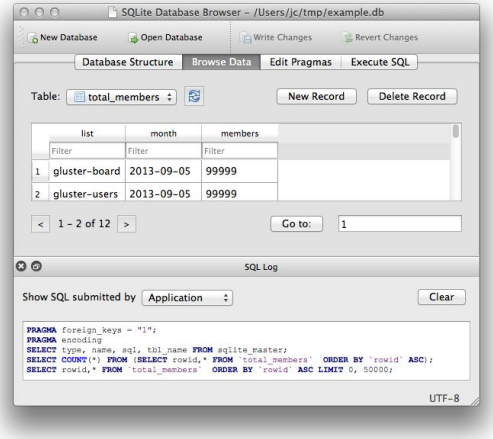
SQL

- Structured Query Language
- Most popular language for query relational data
- **2nd most important language** for data scientists iff you count python/R as 1st.
- A **declarative** language, unlike python/R which is **imperative**

SQLite

- **Lightweight**, file-based database engine
- No separate server process, **easy to set up**
- Ideal for **small-scale applications**, embedded systems, and **prototyping**
- Supports most of the ‘modern’ SQL92 standard
- Atomic commit and rollback support
- Public domain source code, free for use

DB Browser for SQLite – a great tool



- Open-source SQLite database viewer and editor
- Features:
  - Browse and edit databases
  - Run SQL queries and view results
  - Create, define, modify, and delete tables
  - Import and export data
- User-friendly interface; great for beginners and pros
- **Does not replace** being able to work with a DB in the terminal

SQLite + DB Browser for SQLite

- Simplest “off the shelf” RDMS 🔥
  - https://www.sqlite.org/index.html
  - linux, macos, windows
- Simplest “off the shelf” SQLIte DB viewer 🔥🔥
  - https://github.com/sqlitebrowser/sqlitebrowser
  - https://sqlitebrowser.org/
  - Linux, macos, windows

Some features of SQL

- Data Definition Language (**DDL**)
  - Create/alter/delete tables and their attributes
- Data Manipulation Language (**DML**)
  - Query one or more tables insert/delete/modify tuples in tables
- Triggers and Advanced Constraints
  - Actions executed by DBMS on updates and specify complex integrity constraints

Overview of a Basic SQL Statement

The `**FROM**` statement may specify 1 or more tables, if more than 1 you will join them! <=====[to be continued]==||=\\=| in next lecture.

```
2
3   -- SELECT, the first statement, is used to specify the columns in the result set.
4   SELECT [DISTINCT] <column expression list>
5
6   -- FROM is used to specify the tables from which the columns will be taken.
7   FROM <list_of_tables>
8
9   -- WHERE is used to specify the conditions for a tuple to be in the result set.
10  WHERE <predicate>
11
```

The result will be a new table (in memory) of the **selected columns from the table(s) specified where some or no tuples** have been filtered

Projection in SQL

```
| name            | year | genre             |
|-----------------|------|-------------------|
| Apocalypse Now  | 1979 | War               |
| The Godfather   | 1972 | Crime             |
| Planet Earth II | 2016 | Nature doc        |


SELECT name, genre
FROM   Movies

-- results in:

| name            | genre             |
|-----------------|-------------------|
| Apocalypse Now  | War               |
| The Godfather   | Crime             |
| Planet Earth II | Nature doc        |
```

Our Query Must

(i.) Return all movie names and their genres

Notes

- No `WHERE` clause, thus no filtering of tuples
- `SELECT` specified the attributes we wanted

Selection in SQL

```
| name             | year | genre             |
|------------------|------|-------------------|
| Apocalypse Now   | 1979 | War               |
| The Godfather    | 1972 | Crime             |
| Planet Earth II  | 2016 | Nature doc        |


SELECT * --we want all columns
FROM   Movies
WHERE year > 2000

-- results in:

| name             | year | genre             |
|------------------|------|-------------------|
| Planet Earth II  | 2016 | Nature doc        |
```

Our Query Must

(i.) Return movies produced after 2000

Notes

- `WHERE` clause specifies our filter on an attribute
- `SELECT` specified all attributes via `*` somewhat dangerous for large DBs

Selection + Projection in SQL

```
| name             | year | genre              |
|------------------|------|--------------------|
| Apocalypse Now   | 1979 | War                |
| The Godfather    | 1972 | Crime              |
| Planet Earth II  | 2016 | Nature doc         |


SELECT name
FROM   Movies
WHERE year > 2000

-- results in:

| name             |
|------------------|
| Planet Earth II  |
```

Our Query Must

(i.) Return movie names produced after 2000

Notes

- `WHERE` clause specifies our filter on an attribute
- `SELECT` specified only a subset of attribute(s)

Joins in SQL

```
| name            | year | genre             |
|-----------------|------|-------------------|
| Apocalypse Now  | 1979 | War               |
| The Godfather   | 1972 | Crime             |
| Planet Earth II | 2016 | Nature doc        |


| Actor Name    | Movie Name       |
|---------------|------------------|
| Marlon Brando | Apocalypse Now   |
| Al Pacino     | The Godfather    |
| Marlon Brando | The Godfather    |


SELECT DISTINCT genre
FROM   Movie, ActedIN
WHERE  Movie.name = ActedIN.moviename

-- results in: ???

```

Our Query Must

(i.) what do you think it must do?

Want a jump on next lecture

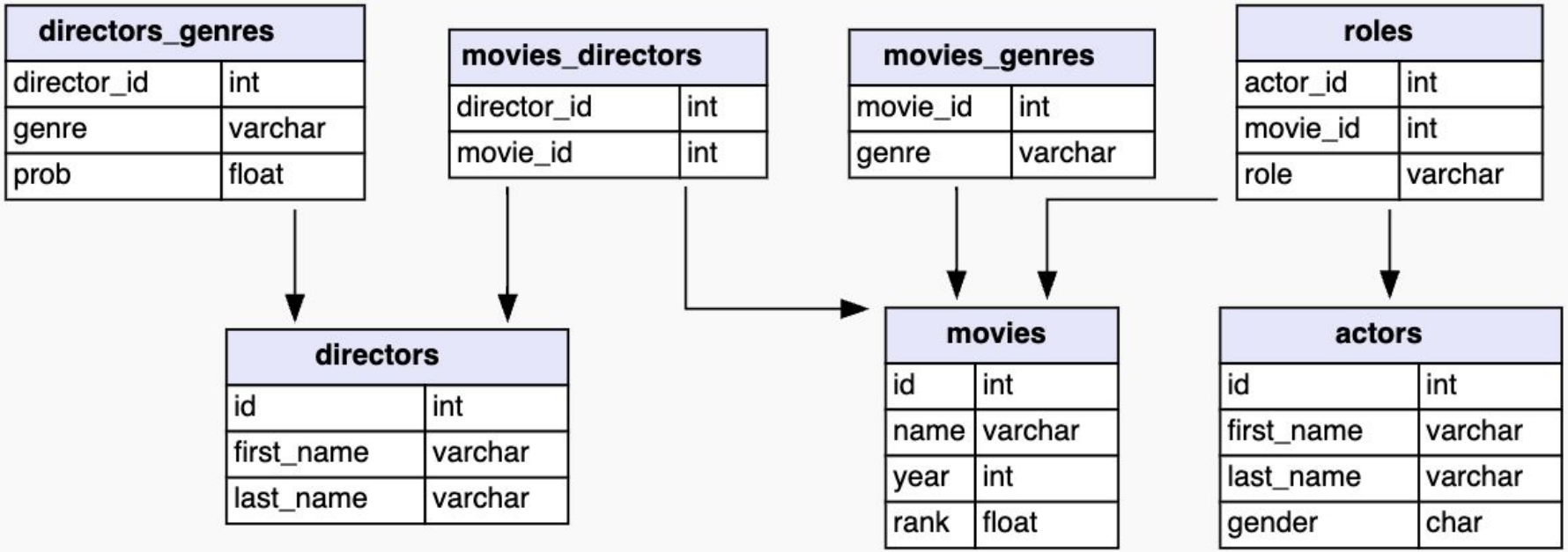
Check out the IMDb database, we will be using that in class and for our demo. It is a reasonably sized database

- 21 columns, 7 tables, 5.7 million rows (full set) we will begin by working with the small version
- Data types: numeric, strings, dates (sort of)
- https://relational.fit.cvut.cz/dataset/IMDb (attribution)
  - you can get the data from here, but I do not recommend it
  - It is a MySQL DB and I needed to write a custom `.sql.` script to load the data into SQLite, as well as a painful Awk script and many regex statements to transform the data. You're welcome
  - I have made all available for you on the class website, cause I'm a nice guy

Demo Time

Let’s use a RDMS and write some SQL

IMDb (small) Database we will be using



Imdb_ijs (by Janez Kranjc)

Ok … but how is knowing about “SQL, sequel, skewl” applicable to my future career?

- **Data Retrieval**: SQL is the standard language for retrieving data from databases, which is a foundational step in any data science project. Knowledge of SQL enables data scientists to create complex queries to interact with large and complex datasets.
- **Data Cleaning**: SQL provides functions and queries that can be used for cleaning and transforming raw data into a usable format, a significant part of a data scientist's role.
- **Job Requirements**: Proficiency in SQL is a common requirement in data science job postings. Knowing SQL demonstrates to employers that you can handle databases and understand key aspects of data manipulation and retrieval.
- **Cross-Disciplinary Tool**: SQL is used in various domains (finance, healthcare, tech, etc.), making your skillset versatile and increasing your potential job opportunities. It’s not just a data science tool; it’s a data tool.
- **Scalability and Efficiency**: For large datasets, SQL can be more efficient than working with data in memory using tools like Pandas. SQL allows you to work with big data in a scalable way, directly within the database.
- **Working in the Cloud**: Many DS operations are performed using SQL within a cloud compute environment (e.g. AWS, GoogleCloud, Azure), you need to be comfortable with this workflow.

What did you learn?

- ❑ Reviewed **relational model**
- ❑ Introduced formally to **SQLite**
- ❑ Learned **basic SQL statements** and keywords
- ❑ Comfortable using the shell to perform **simple DB operations**
- ❑ How to **observe data/DB** in a DB viewer application