

# Introduction to SQL

Ben Winjum

OARC

October 29, 2021

# Overview

- Some basics about SQL and databases
- Create a simple database and execute SQL queries
  - Command line
- A few more details about relational database management systems
- More sophisticated SQL queries utilizing “relations”
  - GUI
- Using SQL with other languages for analysis

# What is SQL?

- **Structured Query Language**

# What is SQL?

- **Structured Query Language**
- SQL allows you to create and modify databases and access the information inside of them

# What is SQL?

- **Structured Query Language**
- SQL allows you to create and modify databases and access the information inside of them
- More specifically, SQL allows you to interact with relational database management systems

# What is SQL?

- **Structured Query Language**
- SQL allows you to create and modify databases and access the information inside of them
- More specifically, SQL allows you to interact with relational database management systems
- But first, a little motivation....

# Online movies and TV streaming

Lists of shows/movies, user preferences, recommendations



# Online retail

Lists of products, user purchases, recommendations

ebay



Rakuten



# Social media

Lists of user info, connections, posts



# Sports

Players, games, statistics, brackets, predictions, injuries, news



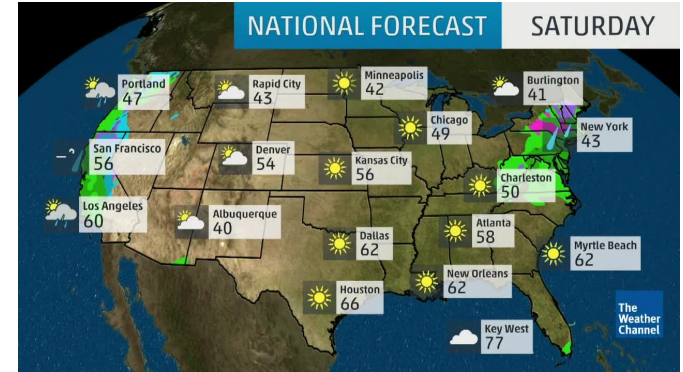
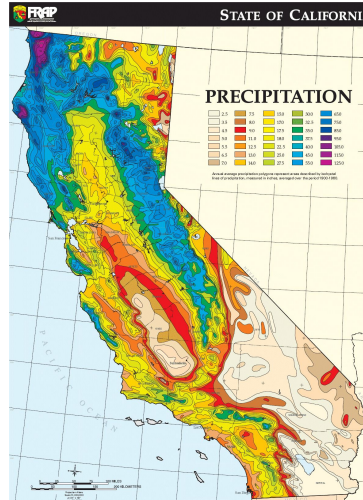
# Finance

Accounts, transactions, investments, models



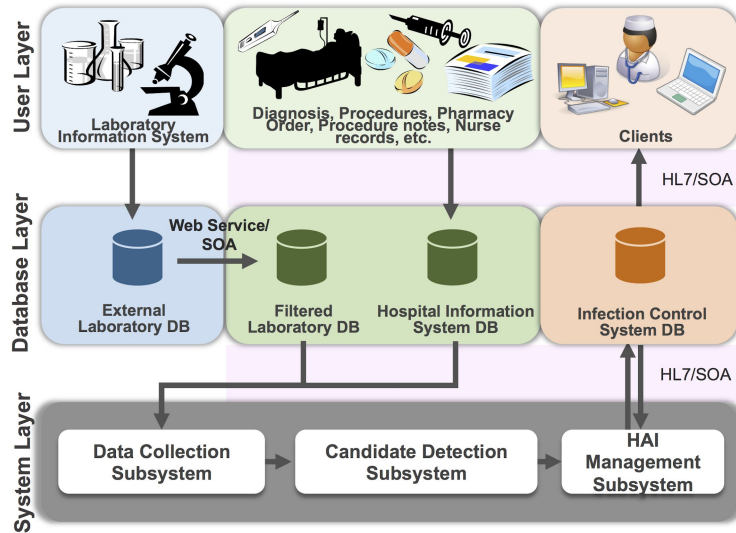
# Weather

Tracking, historical data, predictions



# Healthcare

Patient info, doctor's offices, healthcare orgs, research

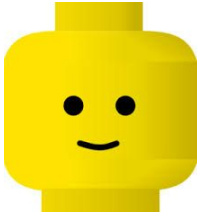


# These systems benefit from databases

- Lots of data
- Frequent updates
- Simultaneous changes to data
- Shared data among a lot of people
- Rapid queries without much analysis

# ACID: some basic database considerations

AL



\$100,000



JOE



- \$15,000

# ACID: some basic database considerations

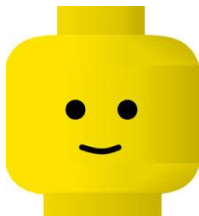


Bank records \$15,000 being transferred  
Also logs to an auditing system for IRS



# ACID: some basic database considerations

AL

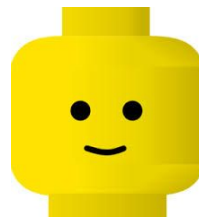


\$100,000  
-\$15,000  
= \$85,000



Bank updates both accounts and its  
IRS-relevant logs

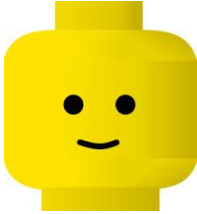
JOE



- \$15,000  
+\$15,000  
= \$ 0

# ACID: some basic database considerations

AL



\$100,000



JOE



- \$15,000

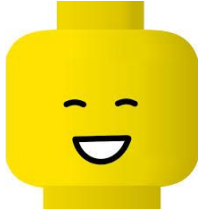
# ACID: some basic database considerations



Another investor makes a simultaneous deposit of  
\$2 while the transaction is being processed  
-- Ledger records are overwritten

# ACID: some basic database considerations

AL

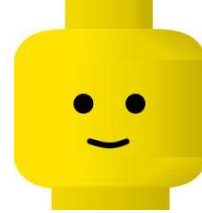


\$100,000  
- \$0 !!  
= \$100,000



Bank gets upset

JOE

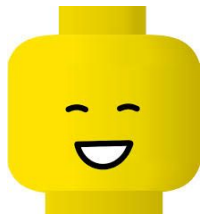


- \$15,000  
+\$15,000  
= \$ 0

# ACID: some basic database considerations

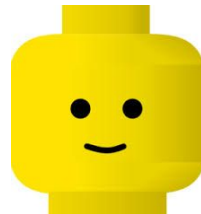
Atomic: All results of a transaction are committed or the whole thing is rolled back

AL



\$100,000  
- \$0 !!  
= \$100,000

JOE



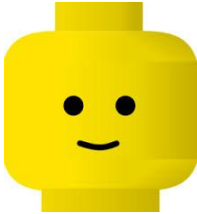
- \$15,000  
+\$15,000  
= \$ 0



Bank gets upset

# ACID: some basic database considerations

AL



\$100,000



JOE



- \$15,000

# ACID: some basic database considerations

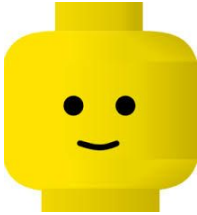


The logs for recording IRS-relevant transactions are off-line

-- Log for this transaction is never written

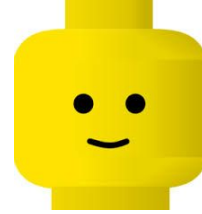
# ACID: some basic database considerations

AL



\$100,000  
-\$15,000  
= \$85,000

JOE



- \$15,000  
+\$15,000  
= \$ 0



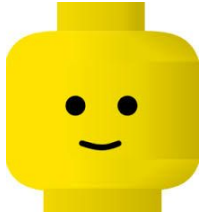
Bank updates accounts but not its  
IRS-relevant logs.... And gets antsy



# ACID: some basic database considerations

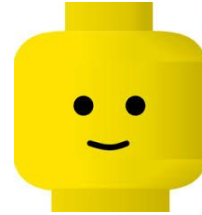
Consistency: If an integrity constraint can't be satisfied, the transaction is aborted

AL



\$100,000  
-\$15,000  
= \$85,000

JOE



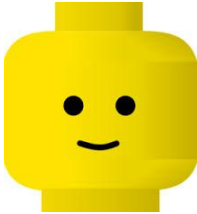
- \$15,000  
+\$15,000  
= \$ 0



Bank updates accounts but not its  
IRS-relevant logs.... And gets antsy

# ACID: some basic database considerations

AL



\$100,000



JOE



- \$15,000

# ACID: some basic database considerations



The bank does not let AL or JOE see modified balances until the transaction has finished successfully

# ACID: some basic database considerations

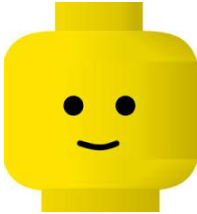
Isolation: Results of a transaction are invisible until it's completed



The bank does not let AL or JOE see modified balances until the transaction has finished successfully

# ACID: some basic database considerations

AL



\$100,000



JOE



- \$15,000

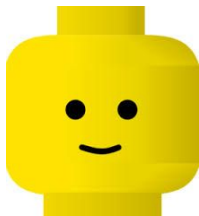
# ACID: some basic database considerations



Bank records \$15,000 being transferred  
Also logs to an auditing system for IRS

# ACID: some basic database considerations

AL

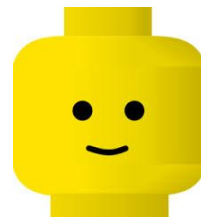


\$100,000  
-\$15,000  
= \$85,000



Bank updates both accounts and its logs

JOE



- \$15,000  
+\$15,000  
= \$ 0

# ACID: some basic database considerations

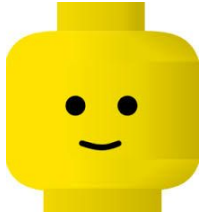




# ACID: some basic database considerations

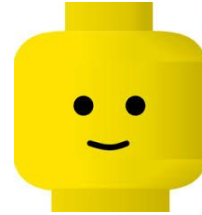
Durability: Results of a transaction survive future failures

AL



\$100,000  
-\$15,000  
= \$85,000

JOE



- \$15,000  
+\$15,000  
= \$ 0



Bank updates both accounts and its logs  
... right before lightning strikes and a  
power surge crashes the banks computers

# Database systems

- Database systems should ideally be ACID-compliant and also very fast
- There's a lot of heady math (relational algebra, normal form, ...) to database development and research
- One of the biggest innovations was the use of a declarative language (SQL)

# SQL as a language

- Imperative languages: Python, C++, Java, Fortran....
  - “Computer, here is how I want you to change your state ...”
- Declarative query language like SQL:
  - “Computer, I want data that meets the following criteria ...”
  - The RDBMS can store the data how it wants, and its query planner can figure out how to get the data

# RDBMS: relational database management systems

- Database systems should ideally be ACID-compliant and also very fast
- There's a lot of heady math (relational algebra, normal form, ...) to database development and research
- But to start working with databases, all you really need to know to start is:
  - A relational database is like a big spreadsheet that several people can update simultaneously

# RDBMS: relational database management systems

- Each table in a database is like one spreadsheet
- Databases are collections of tables
- Tables have rows and columns
  - Unlike spreadsheets, columns have definite data type, and rows are not ordered
- Rows can have a unique identifier, and you can refer to rows in other tables by these unique IDs → relations between tables

# A few simple tables to start

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

classId	Title
1	Film002
2	Econ243
3	Phys100

# Let's start working with SQL!

Go to <https://github.com/benjum/oarc-fall21-intro-to-sql>

And click on the the JupyterHub (or Binder) link

# RDBMS: relational database management systems

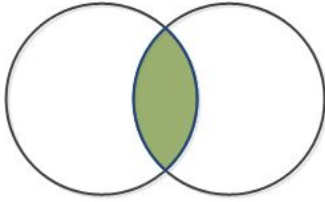
- Rows can have a unique identifier, and you can refer to rows in other tables by these unique IDs → relations between tables
  - OR between one table and itself
  - OR between one table and the result table of a query
  - OR between two tables that are the results of other queries
  - OR between a table in one database and a table in a different database...
- This is the basis for JOINS



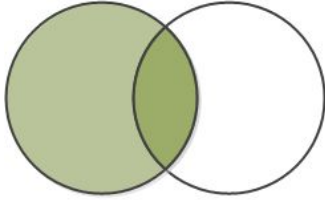
# A few simple tables to start

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

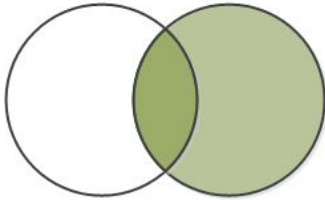
classId	Title
1	Film002
2	Econ243
3	Phys100



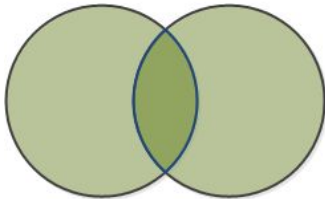
INNER JOIN: include records from both tables where values match



LEFT JOIN: include all records from the first data set even if there are no matches in the second



RIGHT JOIN: include all records from the second data set even if there are no matches in the first



OUTER JOIN: include all records from both tables even if there are no matches in either

Students

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

Classes

classId	Title
1	Film002
2	Econ243
3	Phys100

```

SELECT *
FROM Students
INNER JOIN Classes
USING (classId)

```

studentId	Name	classId	Title
1	Dora	1	Film002
2	Daniel	2	Econ243
3	Mamdooh	1	Film002
4	Lana	2	Econ243

Students

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

Classes

classId	Title
1	Film002
2	Econ243
3	Phys100

May also use:

SELECT \*

FROM Students

INNER JOIN Classes

ON Students.classId = Classes.classId

studentId	Name	classId	classId	Title
1	Dora	1	1	Film002
2	Daniel	2	2	Econ243
3	Mamdooh	1	1	Film002
4	Lana	2	2	Econ243

Students

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

Classes

classId	Title
1	Film002
2	Econ243
3	Phys100

May also use:

SELECT \*

FROM Students s

INNER JOIN Classes c

ON s.classId = c.classId

studentId	Name	classId	classId	Title
1	Dora	1	1	Film002
2	Daniel	2	2	Econ243
3	Mamdooh	1	1	Film002
4	Lana	2	2	Econ243

Students

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

Classes

classId	Title
1	Film002
2	Econ243
3	Phys100

```

SELECT *
FROM Students
LEFT JOIN Classes
USING (classId)

```

studentId	Name	classId	Title
1	Dora	1	Film002
2	Daniel	2	Econ243
3	Mamdooh	1	Film002
4	Lana	2	Econ243
5	Ben	4	

Students

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

Classes

classId	Title
1	Film002
2	Econ243
3	Phys100

```

SELECT *
FROM Students
RIGHT JOIN Classes
USING (classId)

```

*Though SQLite does not have RIGHT JOIN*

studentId	Name	classId	Title
1	Dora	1	Film002
2	Daniel	2	Econ243
3	Mamdooh	1	Film002
4	Lana	2	Econ243
		3	Phys100

Students

studentId	Name	classId
1	Dora	1
2	Daniel	2
3	Mamdooh	1
4	Lana	2
5	Ben	4

Classes

classId	Title
1	Film002
2	Econ243
3	Phys100

```
SELECT *
FROM Students
OUTER JOIN Classes
USING (classId)
```

*Nor does SQLite have OUTER JOIN*

studentId	Name	classId	Title
1	Dora	1	Film002
2	Daniel	2	Econ243
3	Mamdooh	1	Film002
4	Lana	2	Econ243
5	Ben	4	
		3	Phys100



# Varieties of RDBMs

- We've been using SQLite, but there are many other options
- SQLite is:
  - Very light-weight
  - Unique in its operation without a database server (so doesn't require configuration)
  - A database is stored entirely in a file



PostgreSQL



# Ways to use SQL

- We've used SQLite at the command line
- There are also a variety of application programs with GUIs
- Many languages have interfaces to RDBMs and SQL
  - Python, R, Perl, PHP

# Ways to use SQL

- We've used SQLite at the command line
- There are also a variety of application programs with GUIs

Let's try one with the GUI: DBeaver

- Many languages have interfaces to RDBMs and SQL
  - Python, R, Perl, PHP

Any Future Questions:

Feel free to email me – [bwinjum@oarc.ucla.edu](mailto:bwinjum@oarc.ucla.edu)