Introduction to SQL

Ben Winjum
IDRE
July 28, 2020

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?

- **S**tructured **Q**uery **L**anguage

What is SQL?

- **S**tructured **Q**uery **L**anguage
- SQL allows you to create and modify databases and access the information inside of them

What is SQL?

- **S**tructured **Q**uery **L**anguage
- 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?

- **S**tructured **Q**uery **L**anguage
- 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

Social media

Lists of user info, connections, posts

Sports

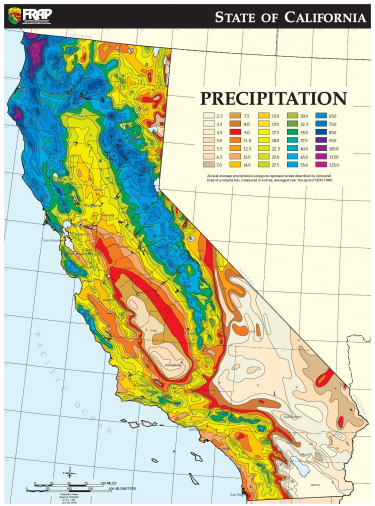
Players, games, statistics, brackets, predictions, injuries, news

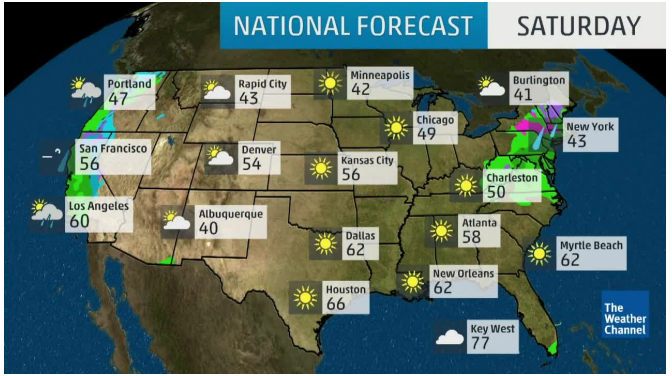
Finance

Accounts, transactions, investments, models

Weather

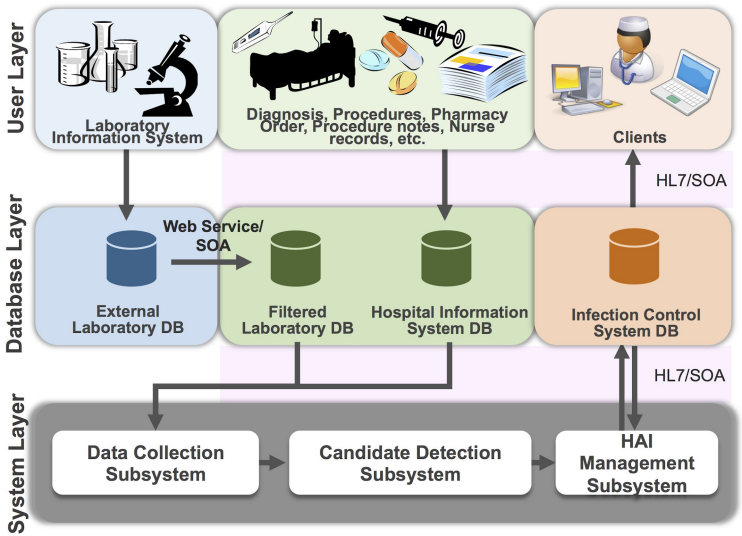
Tracking, historical data, predictions





Healthcare

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



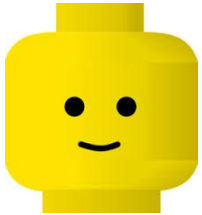
Tseng YJ, Wu JH, Lin HC, Chen MY, Ping XO, Sun CC, Shang RJ, Sheng WH, Chen YC, Lai F, Chang SC. A Web-Based, Hospital-Wide Health Care-Associated Bloodstream Infection Surveillance and Classification System: Development and Evaluation. JMIR Med Inform 2015;3(3):e31.

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

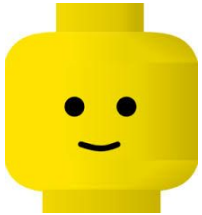
- $15,000

ACID: some basic database considerations

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

ACID: some basic database considerations

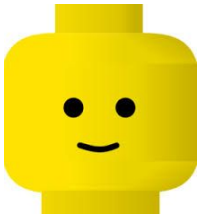
AL



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

Bank updates both accounts and its IRS-relevant logs

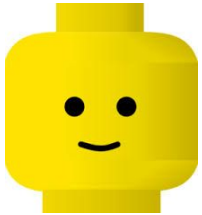
JOE



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

ACID: some basic database considerations

AL



$100,000

JOE

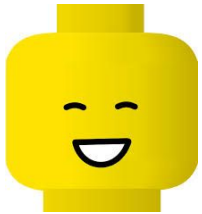
- $15,000

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

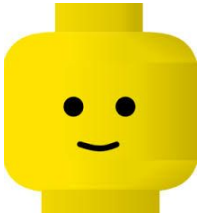
AL



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

Bank gets upset

JOE

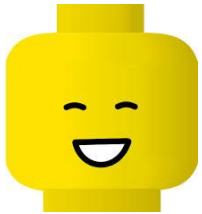


- $15,000
+$15,000
= $ 0

ACID: some basic database considerations

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

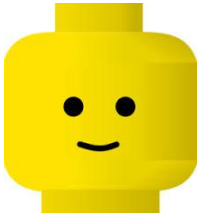
AL



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

Bank gets upset

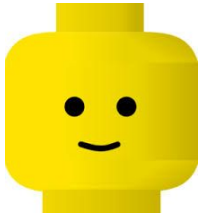
JOE



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

ACID: some basic database considerations

AL



$100,000

JOE

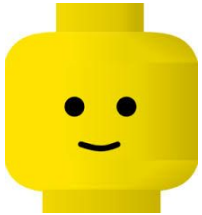
- $15,000

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

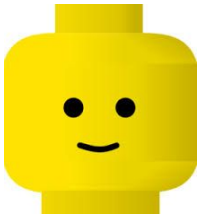
AL



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

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

JOE

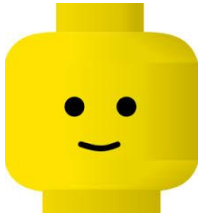


- $15,000
+$15,000
= $ 0

ACID: some basic database considerations

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

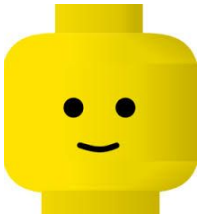
AL



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

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

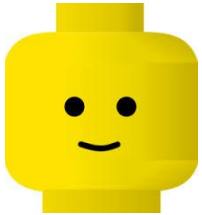
JOE



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

ACID: some basic database considerations

AL



$100,000

JOE

- $15,000

ACID: some basic database considerations

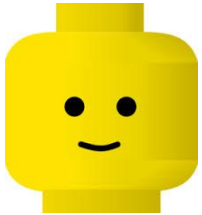
The bank does not let AL or JOE see modified balances until the transaction has finished successfully

ACID: some basic database considerations

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

AL

JOE



$15,000

$15,000

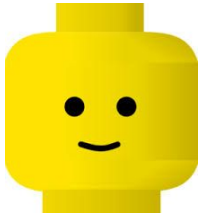
$100,000

- $15,000

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

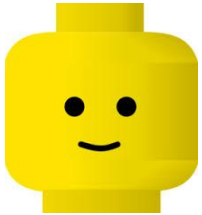
- $15,000

ACID: some basic database considerations

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

ACID: some basic database considerations

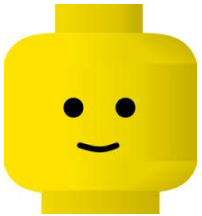
AL



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

Bank updates both accounts and its logs

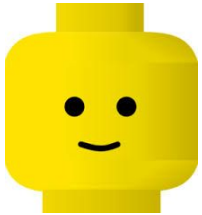
JOE



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

ACID: some basic database considerations

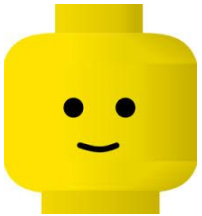
AL



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

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

JOE

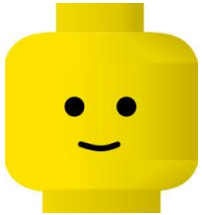


- $15,000
+$15,000
= $ 0

ACID: some basic database considerations

Durability: Results of a transaction survive future failures

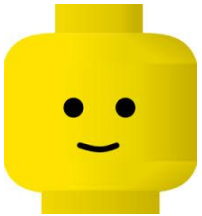
AL



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

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

JOE



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

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

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 create these! Go to the JupyterHub (or Binder) link that's included on the repository page at https://github.com/benjum/idre-intro-to-sql.

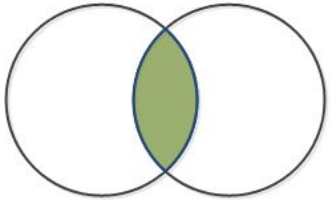
RDBMS: relational database management systems

- 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

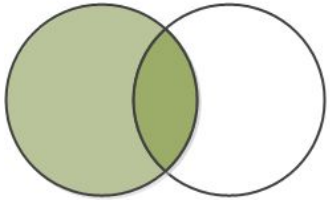
A few simple tables to start

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

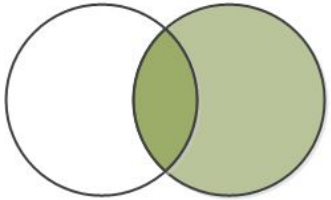
classId	Title
1	Film002
2	Econ243
3	Phys100



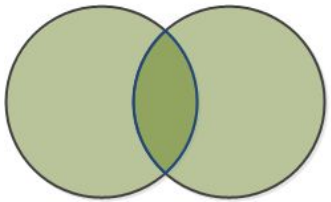
INNER JOIN: include records from both tables where values match



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)
```

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

Though SQLite does not have RIGHT JOIN

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)
```

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

Nor does SQLite have OUTER JOIN

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

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

Final item:
please complete the brief
post-course survey

https://bit.ly/2VmU6ws

Any Future Questions:
Email – bwinjum@idre.ucla.edu