

poloclub.github.io/#cse6242

CSE6242/CX4242: **Data** & **Visual** Analytics

# Simple Data Storage; SQLite

## Duen Horng (Polo) Chau

Associate Professor, College of Computing  
Associate Director, MS Analytics  
Georgia Tech

## Mahdi Roozbahani

Lecturer, Computational Science & Engineering, Georgia Tech  
Founder of **Filio**, a visual asset management platform

**How to store the data?**  
**What's the easiest way?**

# Easiest Way to Store Data

As comma-separated files (CSV)

But may not be easy to parse. Why?

```
1997,Ford,E350
```

# Easiest Way to Store Data

```
1997,Ford,E350
```

- Any field *may* be *quoted* (that is, enclosed within double-quote characters). Some fields *must* be quoted.

```
"1997","Ford","E350"
```

- Fields with embedded commas or double-quote characters must be quoted.

```
1997,Ford,E350,"Super, luxurious truck"
```

- Each of the embedded double-quote characters must be represented by a pair of double-quote characters.

```
1997,Ford,E350,"Super, ""luxurious"" truck"
```

- Fields with embedded line breaks must be quoted (however, many CSV implementations do not support this).

<http://www.sqlite.org>



**Most popular** embedded database in the world

Well-known users: <http://www.sqlite.org/famous.html>  
iPhone (iOS), Android, Chrome (browsers), Mac, etc.

**Self-contained:** one file contains data + schema

**Serverless:** database right on your computer

**Zero-configuration:** no need to set up!

See more benefits at <http://www.sqlite.org/different.html>



# SQL Refresher

# SQL Refresher: create table

```
>sqlite3 database.db
```

```
sqlite> create table student(id integer, name text);
```

```
sqlite> .schema
```

```
CREATE TABLE student(id integer, name text);
```

Id	name

# SQL Refresher: insert rows

```
insert into student values(111, "Smith");  
insert into student values(222, "Johnson");  
insert into student values(333, "Lee");  
select * from student;
```

id	name
111	Smith
222	Johnson
333	Lee



# SQL Refresher: create another table

```
create table takes  
(id integer, course_id integer, grade integer);
```

```
sqlite> .schema
```

```
CREATE TABLE student(id integer, name text);
```

```
CREATE TABLE takes (id integer, course_id integer,  
grade integer);
```

id	course_id	grade

# SQL Refresher: joining 2 tables

More than one tables - **joins**

E.g., create roster for this course (6242)

id	name
111	Smith
222	Johnson
333	Lee

id	course_id	grade
111	6242	100
222	6242	90
222	4000	80

# SQL Refresher: joining 2 tables + filtering

```
select name from student, takes
where
    student.id = takes.id and
    takes.course_id = 6242;
```

id	name
111	<b>Smith</b>
222	<b>Johnson</b>
333	Lee

id	course_id	grade
111	6242	100
222	6242	90
222	4000	80

## Summarizing data:

Find **id** and **GPA** (a summary) for each student

```
select id, avg(grade)
from takes
group by id;
```

Id	course_id	grade
111	6242	100
222	6242	90
222	4000	80

id	avg(grade)
111	100
222	85

# Filtering Summarized Results

```
select id, avg(grade)
from takes
group by id
having avg(grade) > 90;
```

id	course_id	grade
111	6242	100
222	6242	90
222	4000	80

id	avg(grade)
111	100
<del>222</del>	<del>85</del>

# SQL General Form

```
select a1, a2, ... an  
from t1, t2, ... tm  
where predicate  
[order by ...]  
[group by ...]  
[having ...]
```

A lot more to learn! Oracle, MySQL, PostgreSQL, etc.

Highly recommend taking

**CS 4400 Introduction to Database Systems**

**Beware of Missing Indexes**

SQLite easily scales to multiple GBs.

## What if slow?

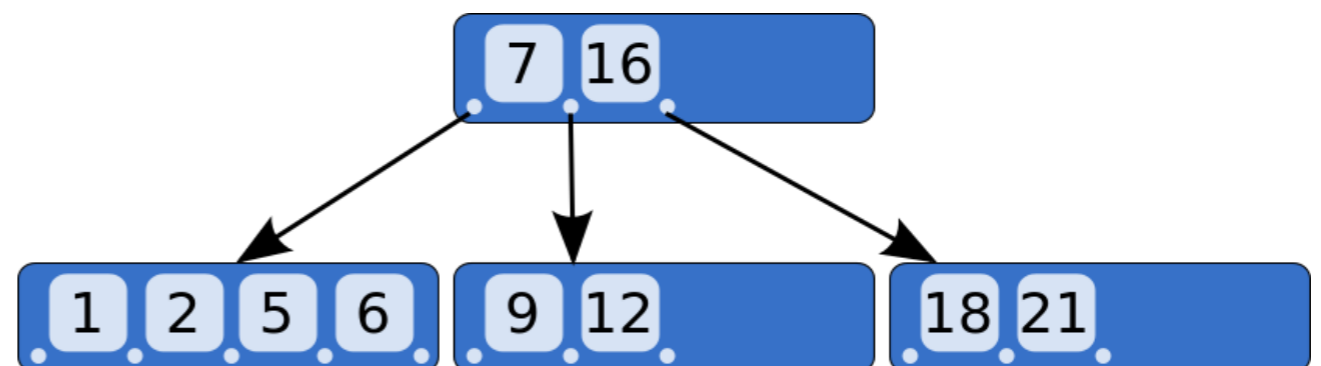
**Important sanity check:**

Have you (or someone) created appropriate **indexes**?

SQLite's indices use **B-tree** data structure.

**$O(\log n)$  speed** for adding/finding/deleting an item.

```
create index student_id_index on  
student(id);
```





# How to Store Petabytes++ ?

Likely need “No SQL” databases

HBase, Cassandra, MongoDB, many more

**HBase** covered in Hadoop/Spark modules later  
this semester