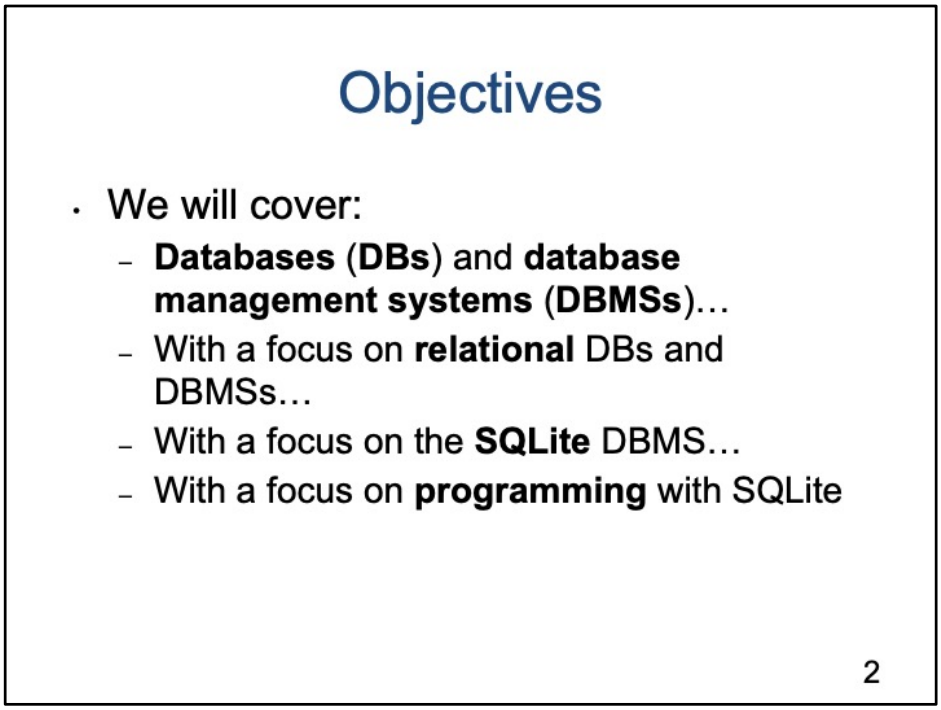
Database Programming (Part 1)

Robert M. Dondero, Ph.D.
Princeton University



Objectives

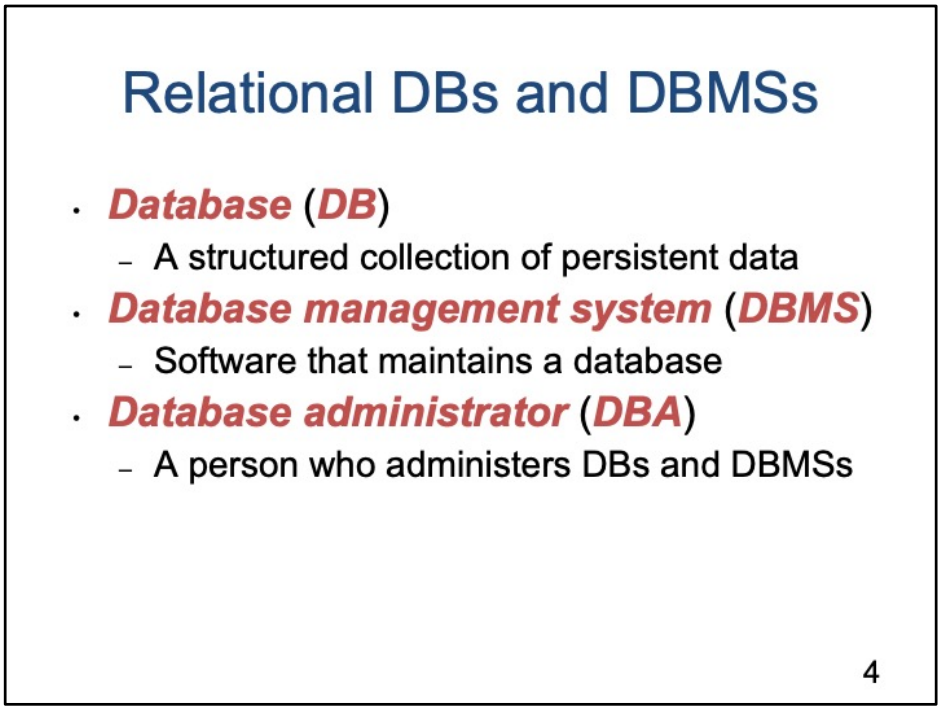
[see slide]

Note:

Comprehensive coverage is impossible!
Please supplement with the reading

Agenda

- **Relational DBs and DBMSs**
- SQL and SQLite
- The SQLite command-line client



Relational DBs and DBMSs

Database (***DB***)

A structured collection of persistent data

An abstract view of a file or collection of files stored persistently (disk, flash drive, cloud, …)

Database management system (***DBMS***)

Software that maintains a database

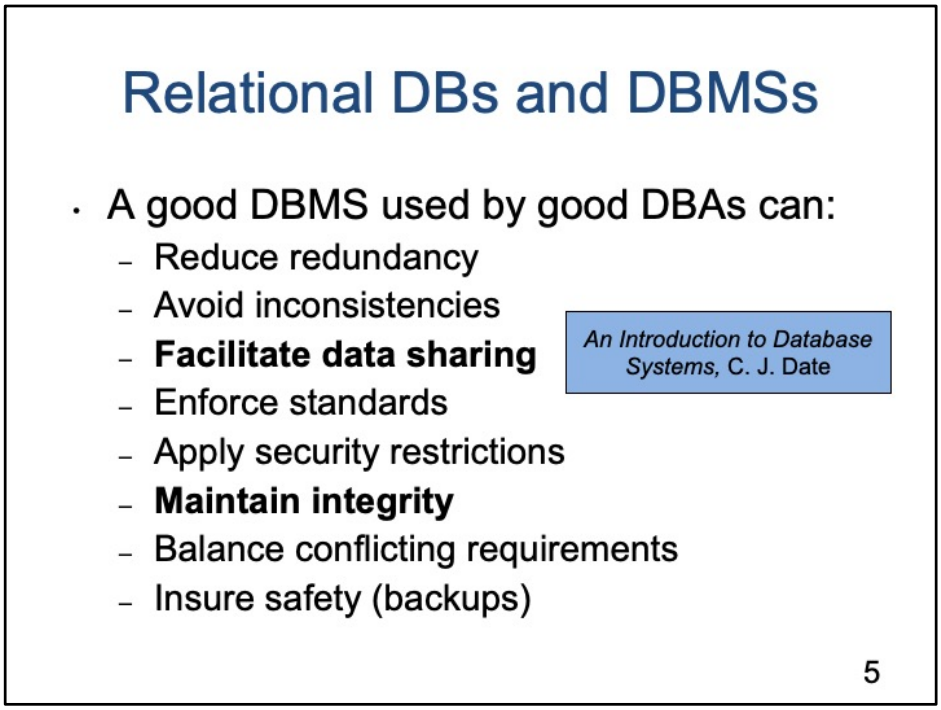
Usually, a server

Listens on a known host at a known port

Clients contact server to perform queries and updates

Database administrator (DBA)

A person who administers DBs and DBMSs



Relational DBs and DBMSs

Question: Why not simply use files?

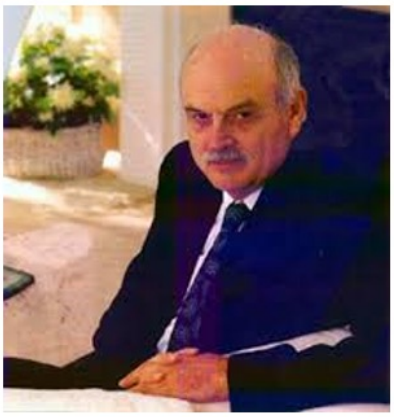
Answer: Centralized control
Database administrators (DBAs) control the data

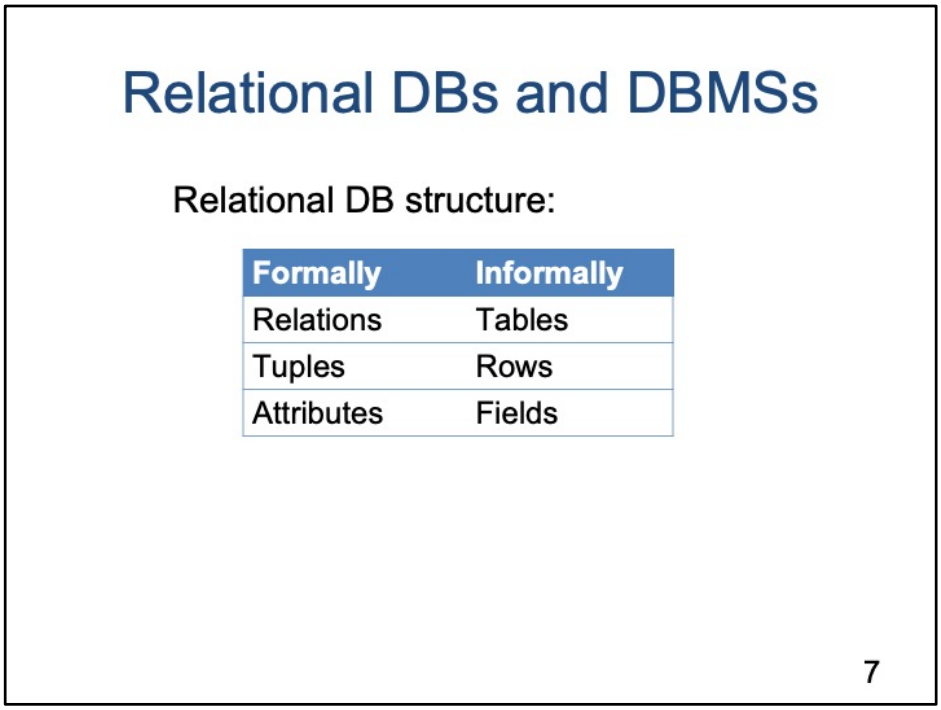
[see slide]

Note:
C.J. Date book is the most popular introductory book

Relational DBs and DBMSs

- **Who**: Edgar Codd
- **When**: 1968-1970
- **Where**: IBM





Relational DB structure

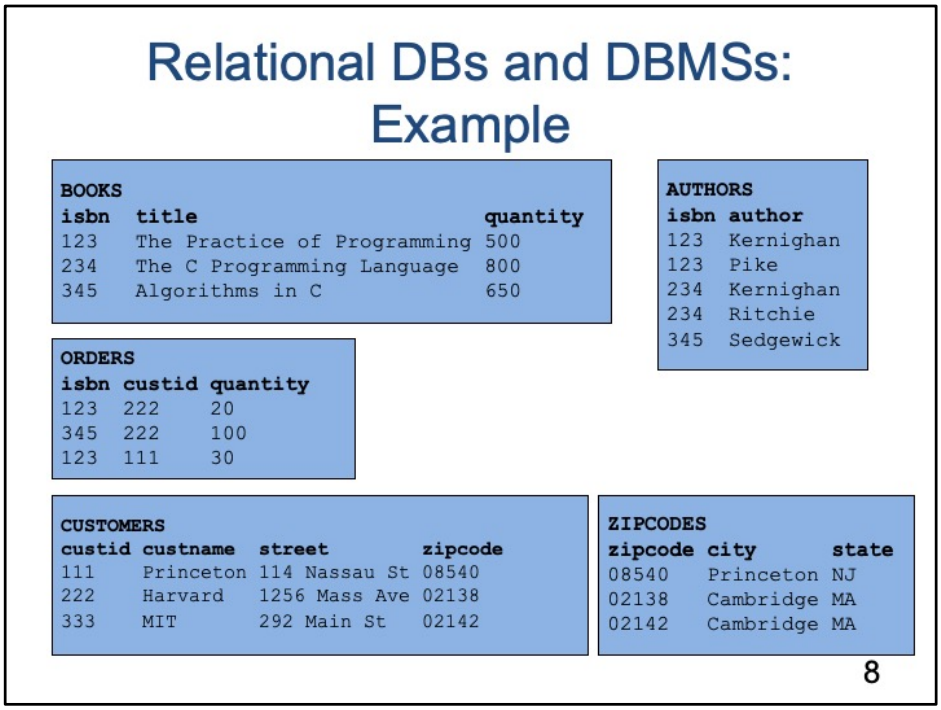
Formally

A relational DB consists of **relations**
Each relation has **tuples**
Each tuple has **attributes**

Informally

A relational DB consists of **tables**
Each table has **rows**
Each row has **fields**

Example...



Relational DBs and DBMSs

An example DB

[see slide]

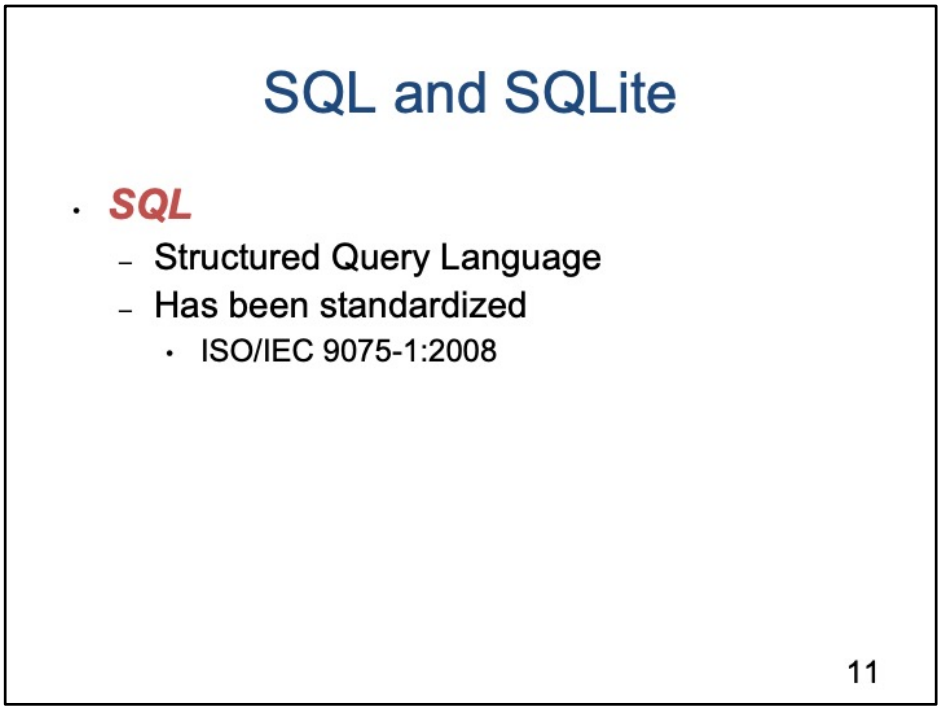
We'll reference this example relational DB often throughout this lecture
So I'm giving you hard copy of it

Agenda

- Relational DBs and DBMSs
- **SQL and SQLite**
- The SQLite command-line client

SQL and SQLite

- **SQL**
  - **Who**: Donald Chamberlin & Raymond Boyce
  - **When**: 1970s
  - **Where**: IBM
  - **Why**: Manipulate data in IBM's System R relational DBMS



SQL

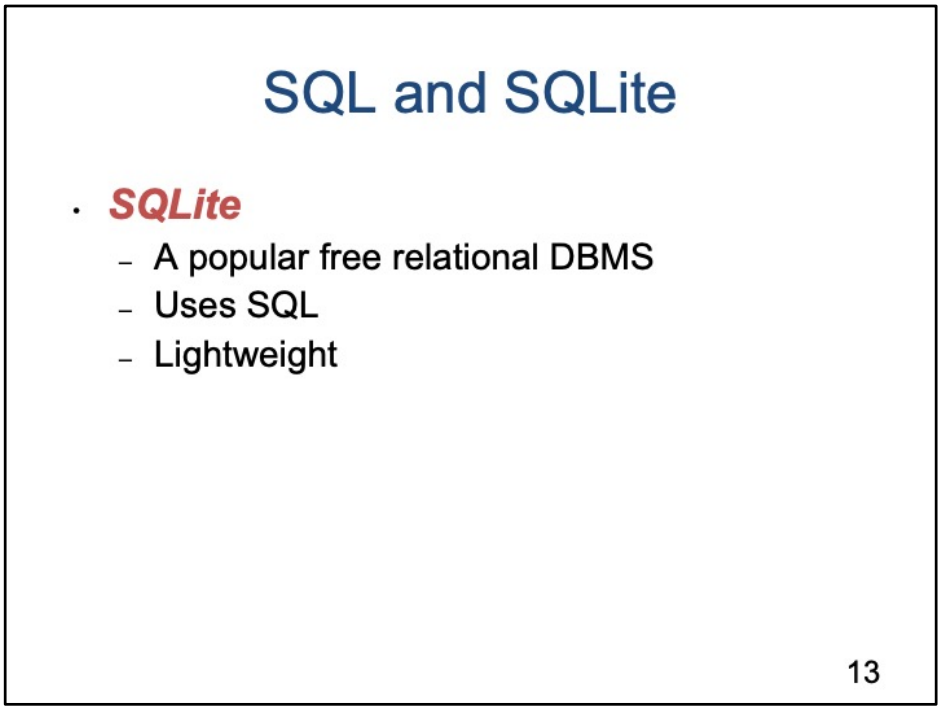
Structured **Q**uery **L**anguage

Has been standardized
ISO/IEC 9075-1:2008

Now the de facto standard for communicating with relational DBMSs

SQL and SQLite

- **SQLite**
  - **Who**: D. Richard Hipp
  - **When**: 2000
  - **Where**: General Dynamics
  - **Why**: Eliminate distinct DBMS process, eliminate DBAs?



SQLite

A popular free relational DBMS

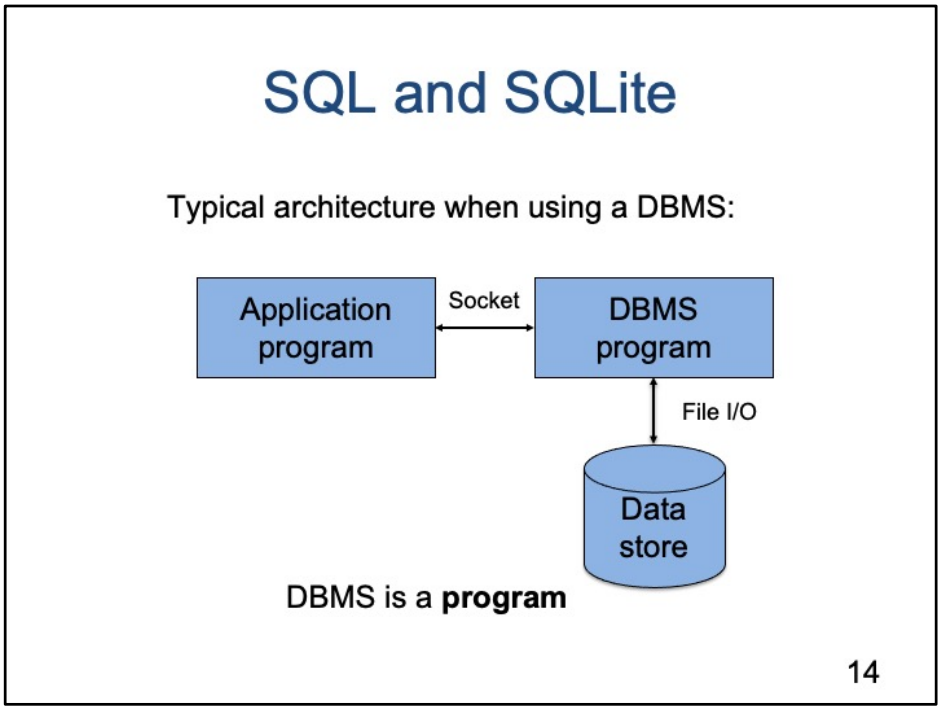
Uses **SQL**

Extends SQL with additional statements

… As DBMSs typically do

Lightweight

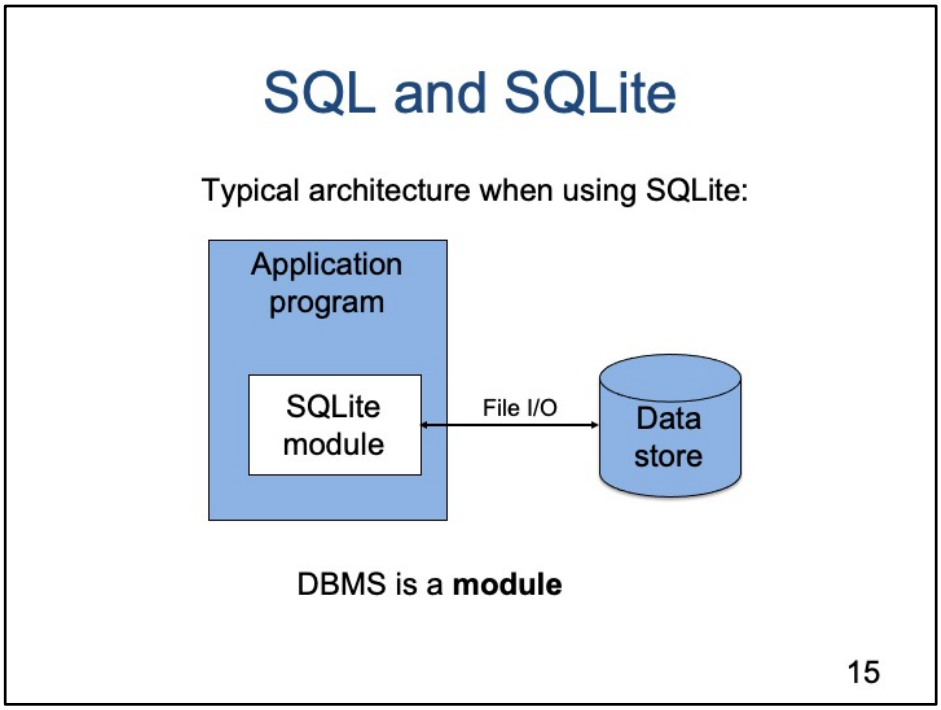
Good choice for learning SQL



SQL and SQLite

[see slide]

DBMS is a **program**
App pgm & DBMS pgm run in **distinct** processes
App process & DBMS process communicate via a **socket** (see upcoming *Networks* lecture)



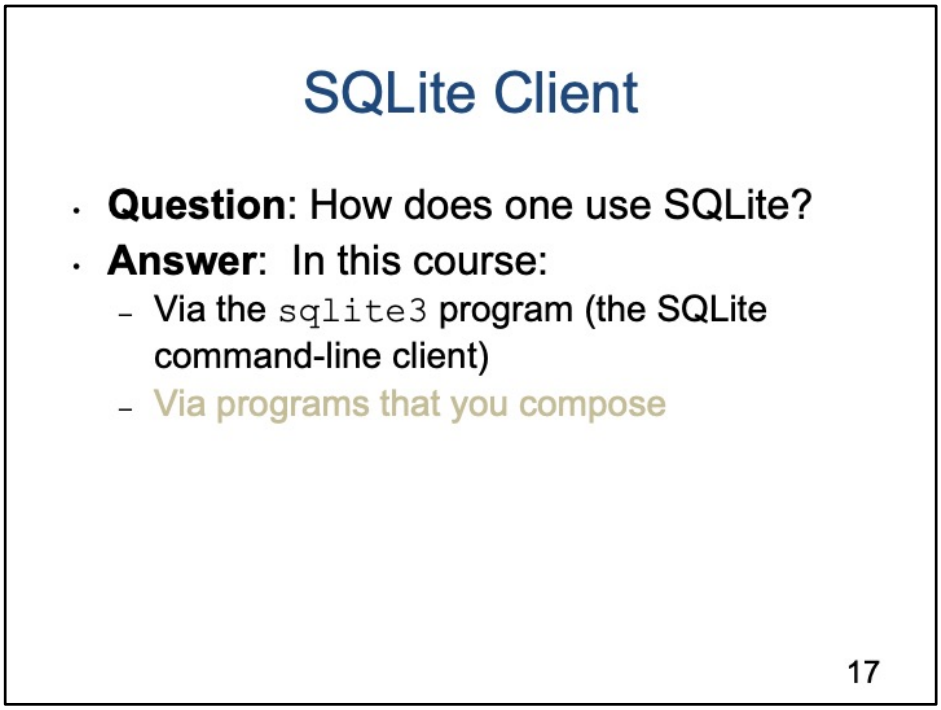
SQL and SQLite

[see slide]

DBMS is a **module** that is part of (is imported into) the app program
App pgm & DBMS module run in **same** process
App pgm & DBMS module communicate via **function/method calls**

Agenda

- Relational DBs and DBMSs
- SQL and SQLite
- **The SQLite command-line client**



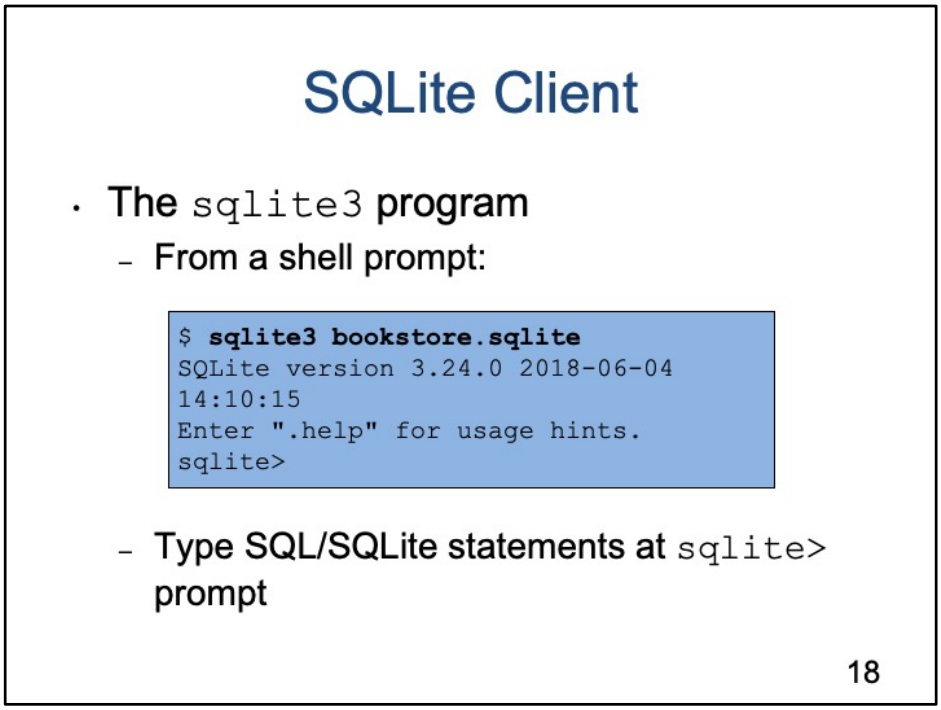
SQL and SQLite

First we'll study how to use SQLite via the SQLite command-line client – whose name is sqlite3

Then we'll study how to use SQLite via programs that you compose

This lecture: Python

Later in the course Java and JavaScript programs too



SQL and SQLite

SQLite command-line client...

From a shell prompt:

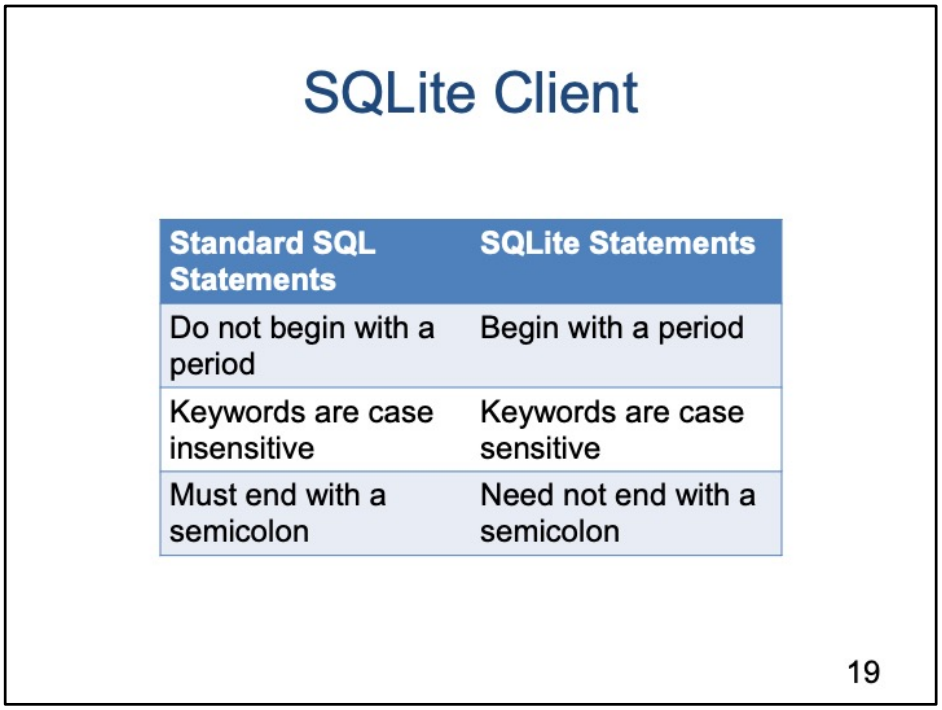
sqlite3 *filename*

If named file exists, uses the database in that file

If named file does not exist, creates it containing an empty database

Writes sqlite> prompt

You then type SQL/SQLite statements at sqlite> prompt

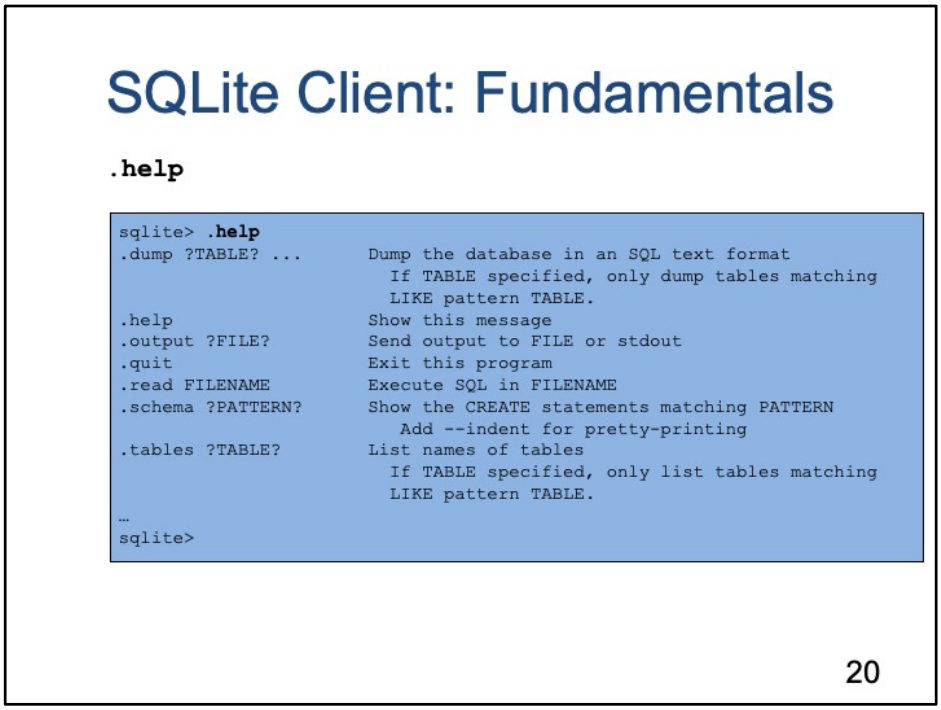


SQL and SQLite

So, what do you enter at the sqlite> prompt?

There are two kinds of statements that you can enter

[see slide]



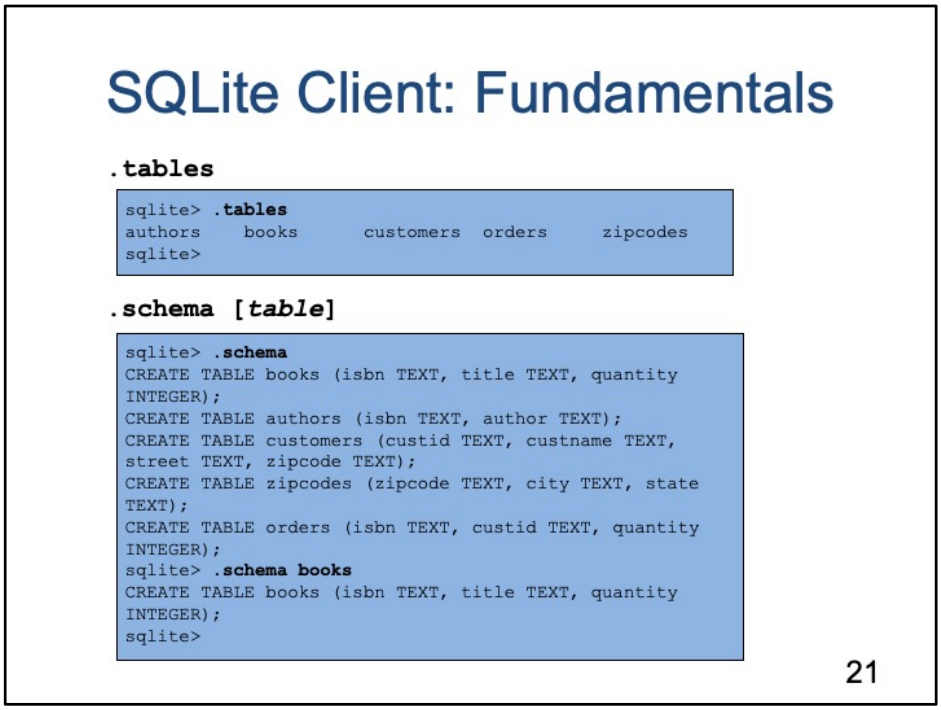
SQL and SQLite

This slide and the following slides show some SQL/SQLite statements

[see slide]

.help

Writes a description of each SQLite statement



SQL and SQLite

[see slide]

.tables

Shows the names of the tables in the DB

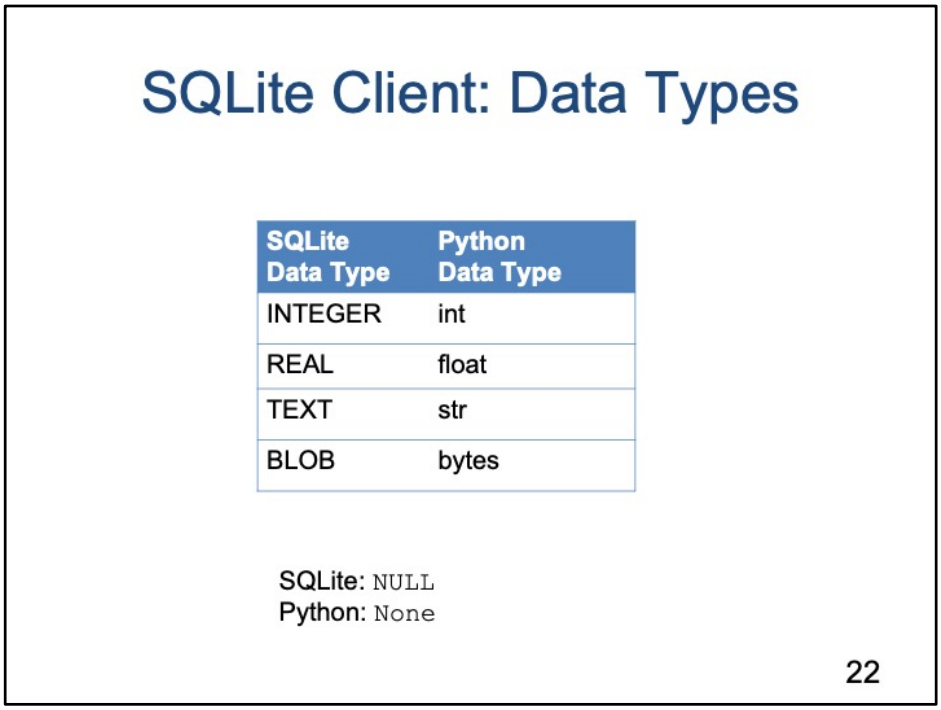
.schema [*table*]

Shows the CREATE TABLE statements that were issued to create the tables of the DB

We'll cover the CREATE TABLE statement in a few minutes

Thereby shows each table, each field, and the data type of each field

Or can specify any one table

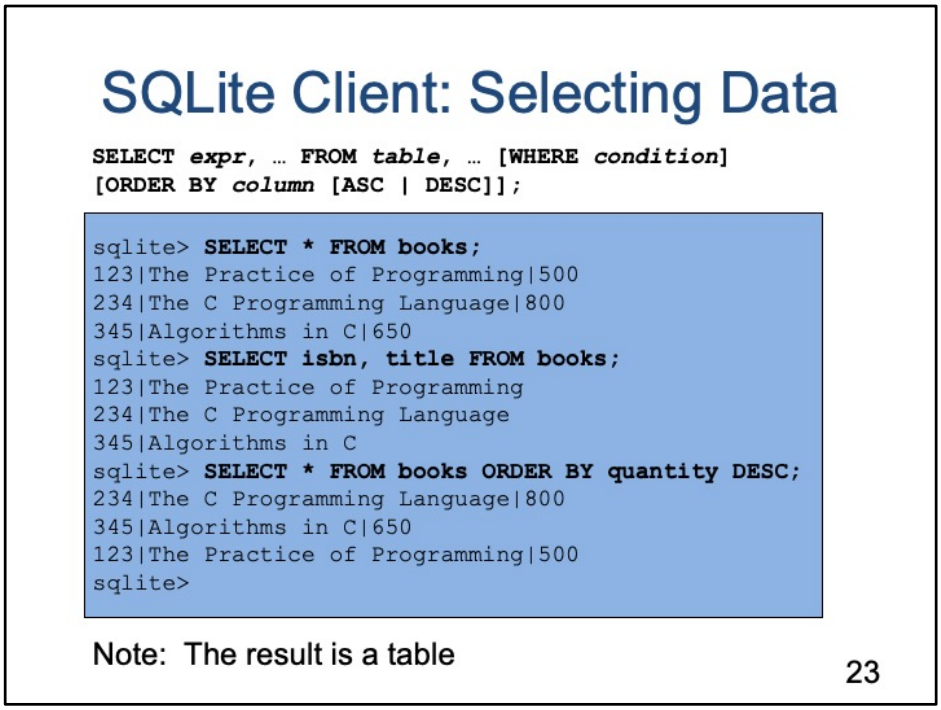


SQL and SQLite

Speaking of data types…

[see slide]

BLOB is an abbreviation for “binary large object”



SQL and SQLite

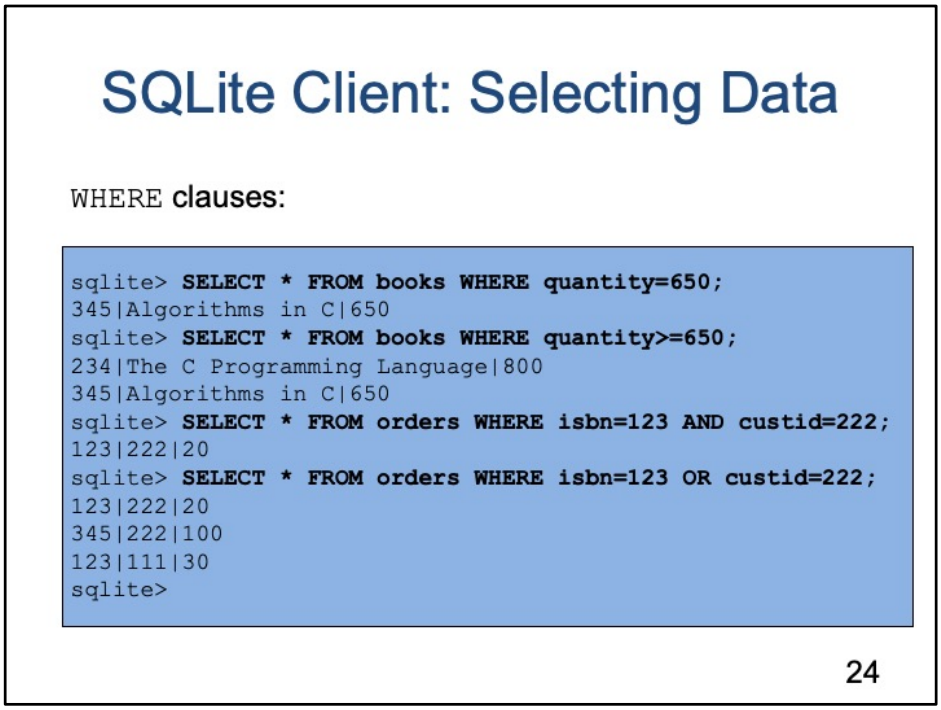
[see slide]

The result generated by a SELECT statement is a table

Can nest SELECT statements

(Beyond our scope)

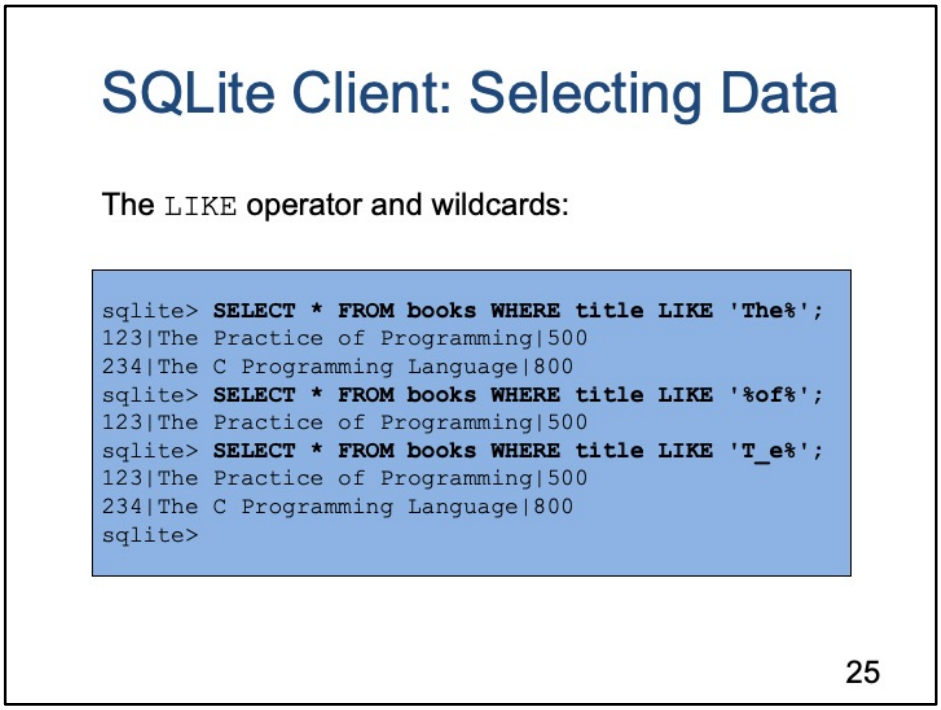
You can specify the columns/fields that you want in the result table



SQL and SQLite

[see slide]

You can use a WHERE clause to specify the rows that you want in the result table

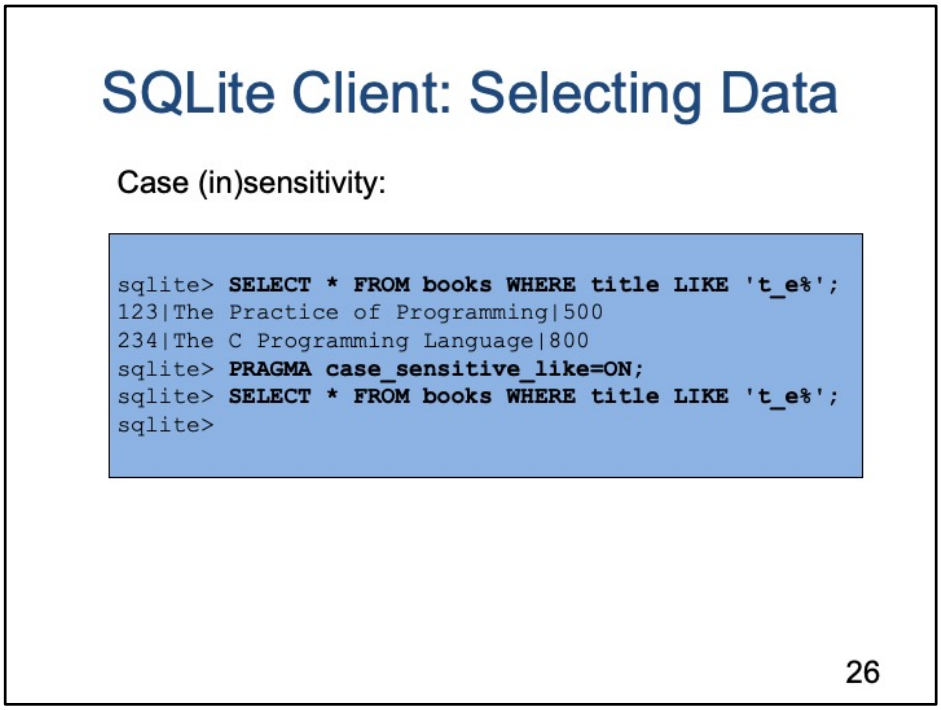


SQL and SQLite

You can use wildcards:

- % matches any 0 or more characters
- _ matches any one character

[see slide]



SQL and SQLite

Normally the SQL LIKE operator is case-insensitive
You can issue a PRAGMA statement if you want the LIKE operator to be case sensitive

[see slide]

Aside: Escape Char

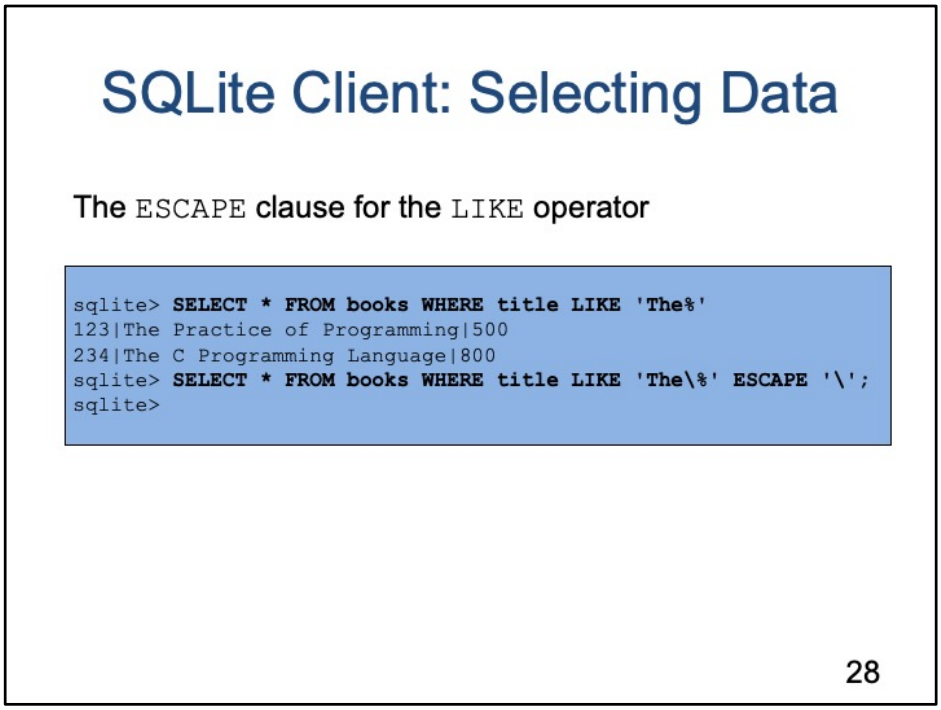
C, Java, and Python define backslash as the ***escape char***

Within a string literal, the char following the escape char is not a special char

```
"abc\"def"
```

The second double quote char doesn't delimit the string, but instead is an ordinary char within the string

SQL doesn't define an escape char, but…



SQL and SQLite

Normally the LIKE operator has no escape char

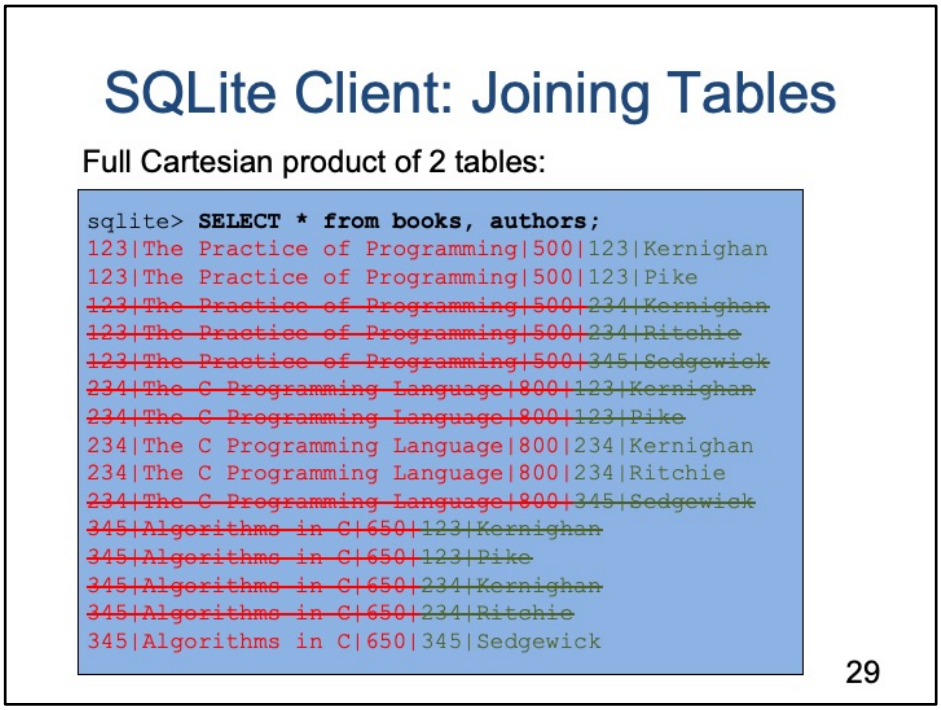
For example, a backslash char has no special meaning to the LIKE operator

You can specify an ESCAPE clause associated with a LIKE operator

Defines an escape char

For example, you could define backslash as the escape char

[see slide]



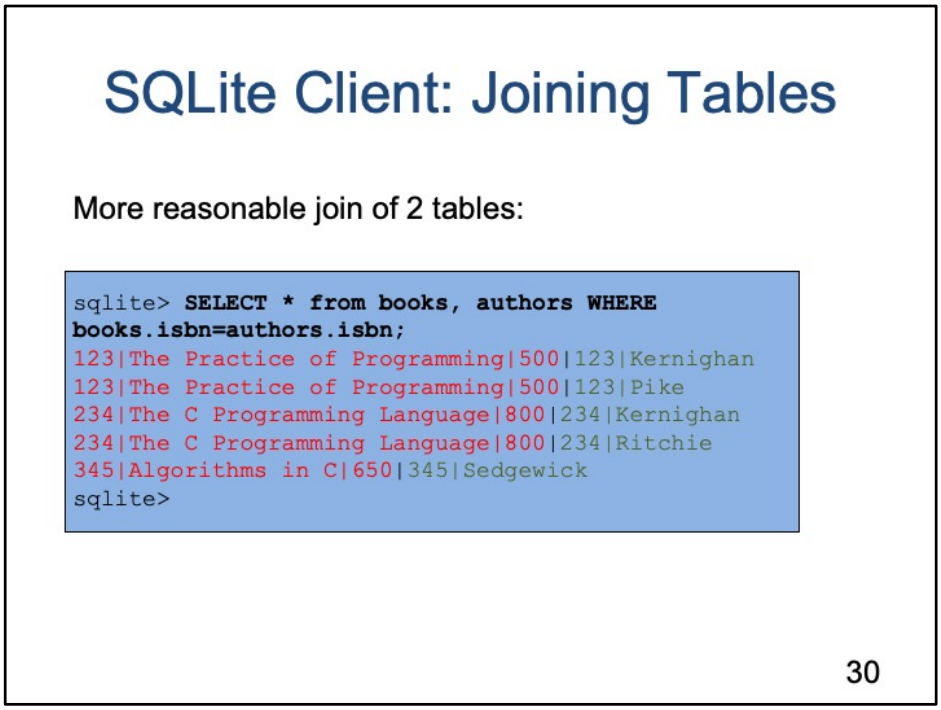
SQL and SQLite

You can join two tables

In the absence of a WHERE clause, the resulting table is the Cartesian product of the two specified tables

[see slide]

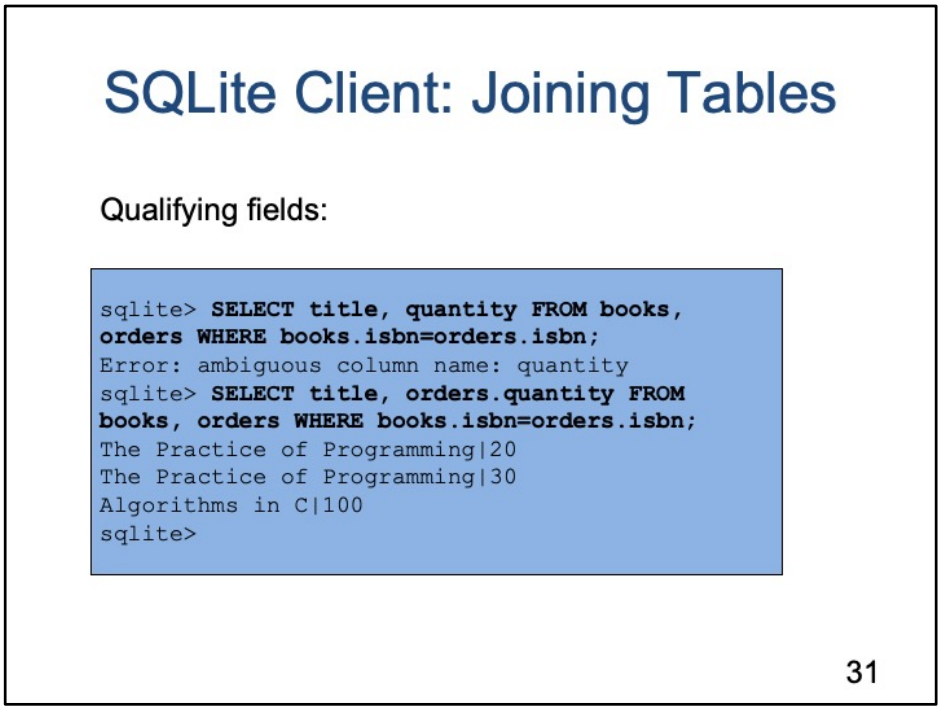
The Cartesian product contains many meaningless rows



SQL and SQLite

So it's common to specify an appropriate WHERE clause

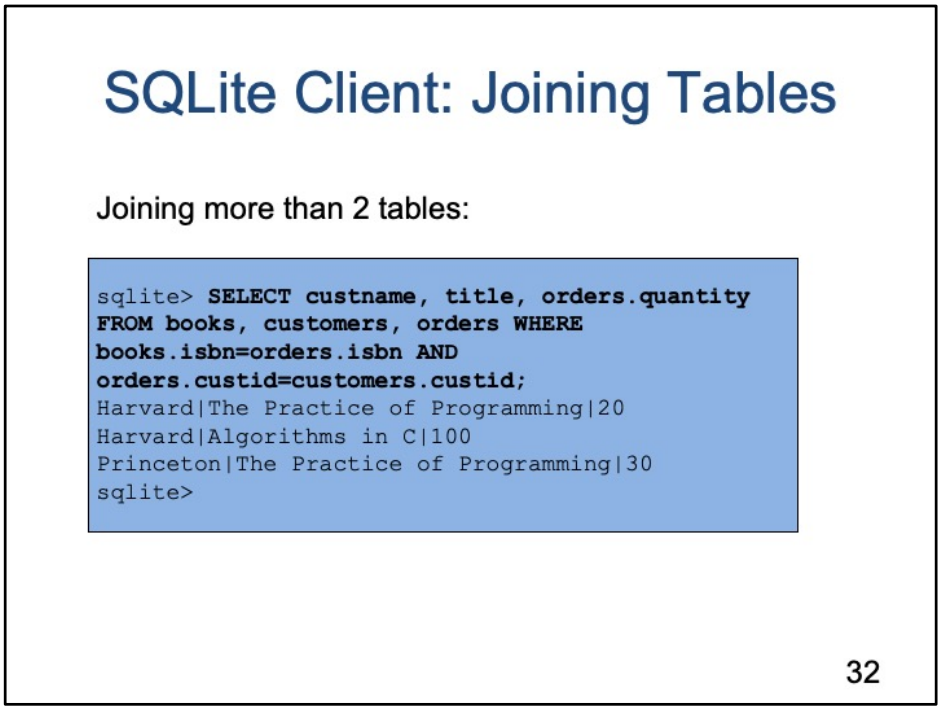
[see slide]



SQL and SQLite

[see slide]

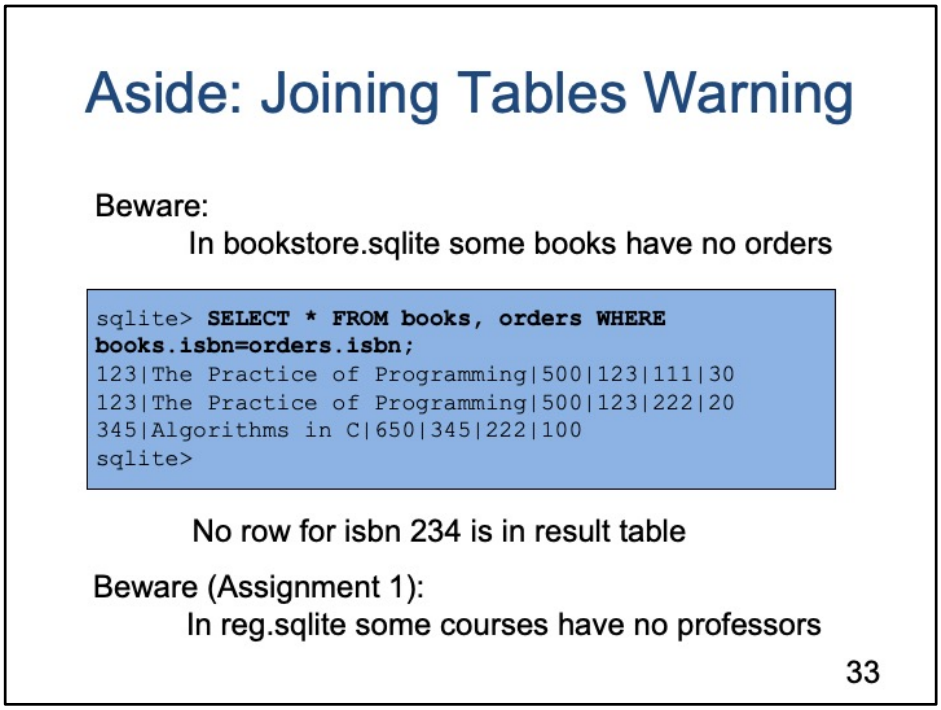
When joining tables you can specify the columns/fields that you want in the result table Generally, you must qualify each field name with the name of the joined table in which it resides



SQL and SQLite

[see slide]

You can join more than two tables



SQL and SQLite

[see slide]

That's a hint for Assignment 1

SQLite Client: Changing Rows

UPDATE ***table*** **SET** ***column1=expr1*** **[,** ***column2=expr2*** **…]**
[WHERE ***condition*****]**

```
sqlite> UPDATE books SET quantity=60 WHERE isbn=123;
sqlite> SELECT * from books;
123|The Practice of Programming|60
234|The C Programming Language|800
345|Algorithms in C|650
sqlite> UPDATE books SET quantity=quantity+1 WHERE isbn=123;
sqlite> SELECT * from books;
123|The Practice of Programming|61
234|The C Programming Language|800
345|Algorithms in C|650
sqlite> UPDATE books SET quantity=500 WHERE isbn=123;
sqlite> SELECT * from books;
123|The Practice of Programming|500
234|The C Programming Language|800
345|Algorithms in C|650
sqlite>
```

SQLite Client: Inserting Rows

INSERT INTO *table* (*column*, …) VALUES (*expr*, …);

```
sqlite> INSERT INTO books (isbn, title, quantity) VALUES
('456', 'Core Java', 120);
sqlite> SELECT * from books;
123|The Practice of Programming|500
234|The C Programming Language|800
345|Algorithms in C|650
456|Core Java|120
sqlite>
```

SQLite Client: Deleting Rows

```
DELETE FROM table [WHERE condition];
```

```
sqlite> DELETE FROM books WHERE isbn=456;
sqlite> SELECT * from books;
123|The Practice of Programming|500
234|The C Programming Language|800
345|Algorithms in C|650
sqlite>
```

```
DELETE FROM books;  -- Be careful!!!
```

SQLite Client: Destroying Tables

DROP TABLE [IF EXISTS] ***table***

```
sqlite> DROP TABLE books;
sqlite> .tables
authors    customers  orders     zipcodes
sqlite>
```

SQLite Client: Creating Tables

CREATE TABLE [IF NOT EXISTS] ***table***
(*column datatype*, …);

```
sqlite> CREATE TABLE books (isbn TEXT, title TEXT,
quantity INTEGER);
sqlite> INSERT INTO books (isbn, title, quantity)
VALUES ('123','The Practice of Programming',500);
sqlite> INSERT INTO books (isbn, title, quantity)
VALUES ('234','The C Programming Language',800);
sqlite> INSERT INTO books (isbn, title, quantity)
VALUES ('345','Algorithms in C',650);
sqlite> SELECT * FROM books;
123|The Practice of Programming|500
234|The C Programming Language|800
345|Algorithms in C|650
sqlite>
```

SQLite Client: Altering Schema

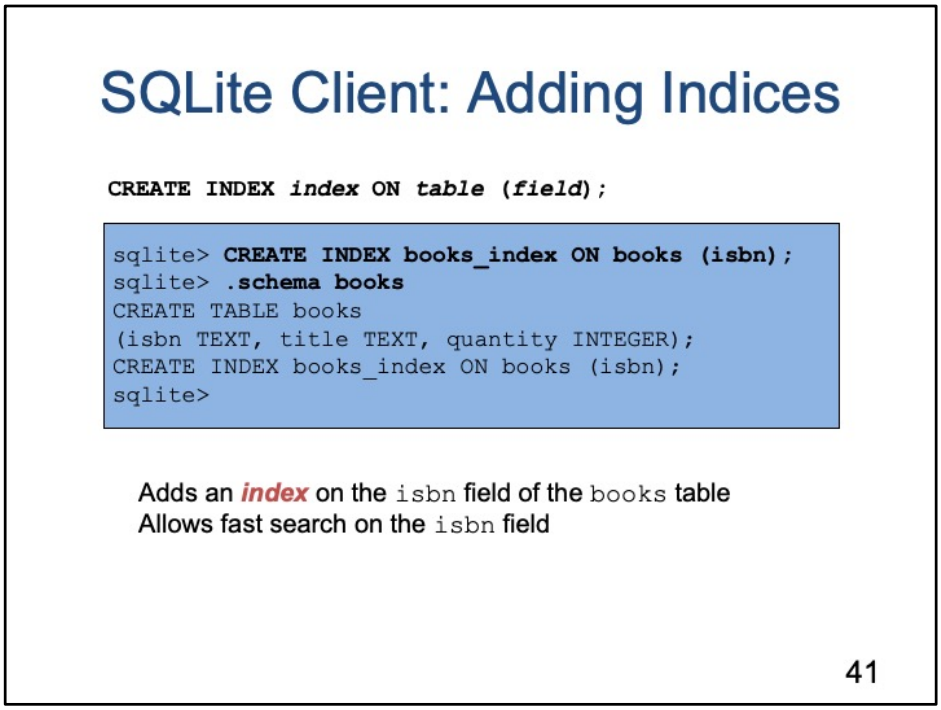
ALTER TABLE ***table specification*** **[,** ***specification*****]** **…;**

```
sqlite> ALTER TABLE books ADD COLUMN price INTEGER;
sqlite> .schema books
CREATE TABLE books
(isbn TEXT, title TEXT, quantity INTEGER, price INTEGER);
sqlite> SELECT * FROM books;
123|The Practice of Programming|500|
234|The C Programming Language|800|
345|Algorithms in C|650|
sqlite>
```

SQLite Client: Altering Schema

```
sqlite> ALTER TABLE books DELETE COLUMN price;
Error: near "DELETE": syntax error
```

```
sqlite> ALTER TABLE books RENAME TO books2;
sqlite> CREATE TABLE books (isbn TEXT, title TEXT, quantity
INTEGER);
sqlite> INSERT INTO books (isbn, title, quantity) SELECT isbn,
title, quantity from books2;
sqlite> DROP TABLE books2;
sqlite> .schema books
CREATE TABLE books
(isbn TEXT, title TEXT, quantity INTEGER);
sqlite>
```



SQL and SQLite

[see slide]

Index implementation:

Typically, B-tree

Adding an index to field F of table T:

Improves performance of **searches** for rows in T by F
Degrades performance of **inserts** of rows into T
Degrades performance of **deletes** of rows from T

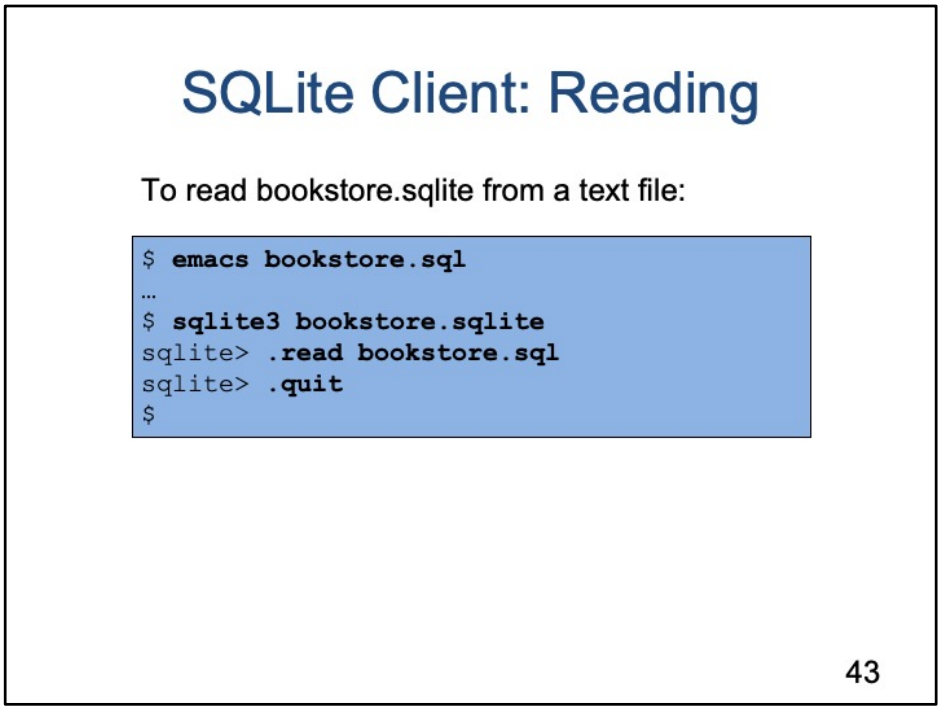
Indices matter for large tables

E.g. In the DB that you'll use for your assignments, I added appropriate indices
Without them, your programs would run noticeably slowly

SQLite Client: Exiting

.quit

```
sqlite> .quit
$
```

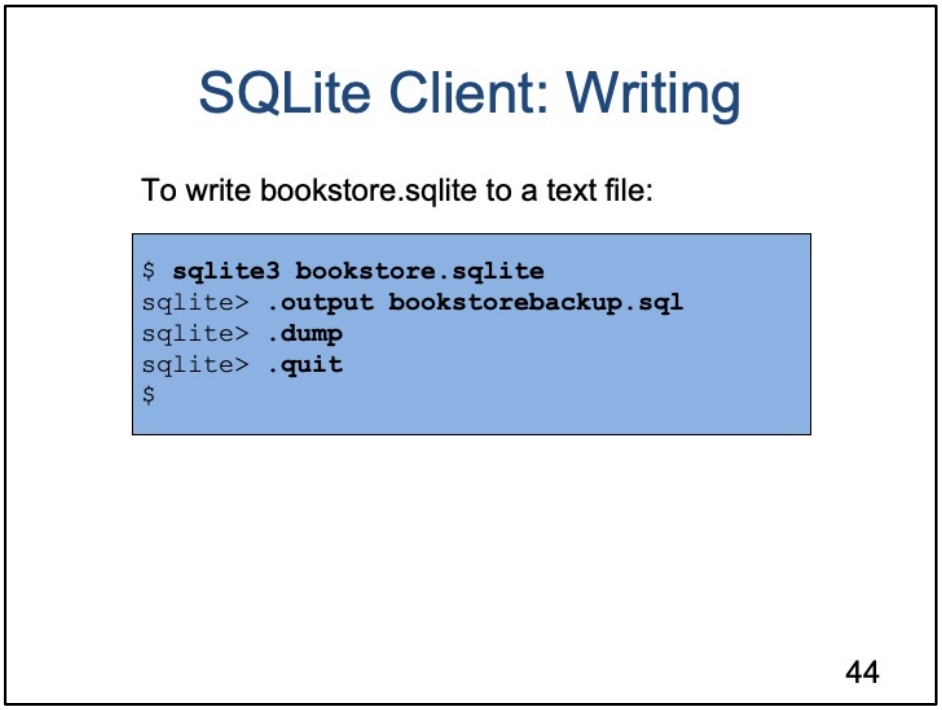


SQL and SQLite

The .read statement commands SQLite to read/interpret SQL/SQLite statements from a text file – just as if they were entered at the sqlite> prompt

That's the mechanism that I used to create the bookstore.sqlite database

[see slide]



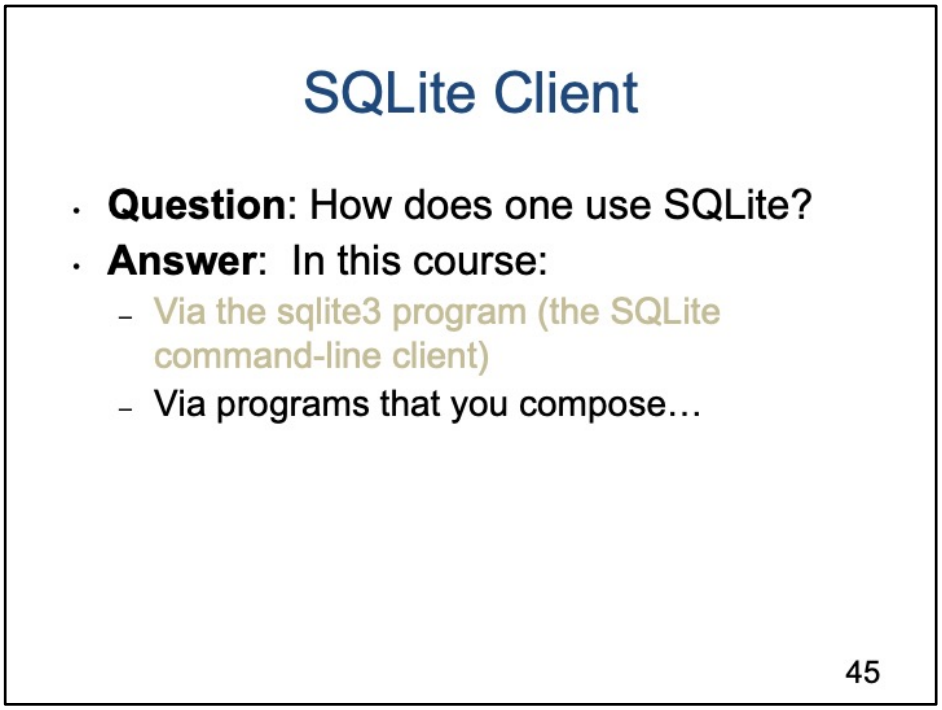
SQL and SQLite

The .dump statement commands SQLite to dump the CREATE TABLE and INSERT statement to a file – such that those statements create the database

[see slide]

That mechanism provides a good way to

- Backup a SQLite database as a text file
- Port a SQLite database to some other kind of database (e.g., PostgreSQL)



SQL and SQLite

[see slide]

At this point we've studied how to use SQLite via the sqlite3 program

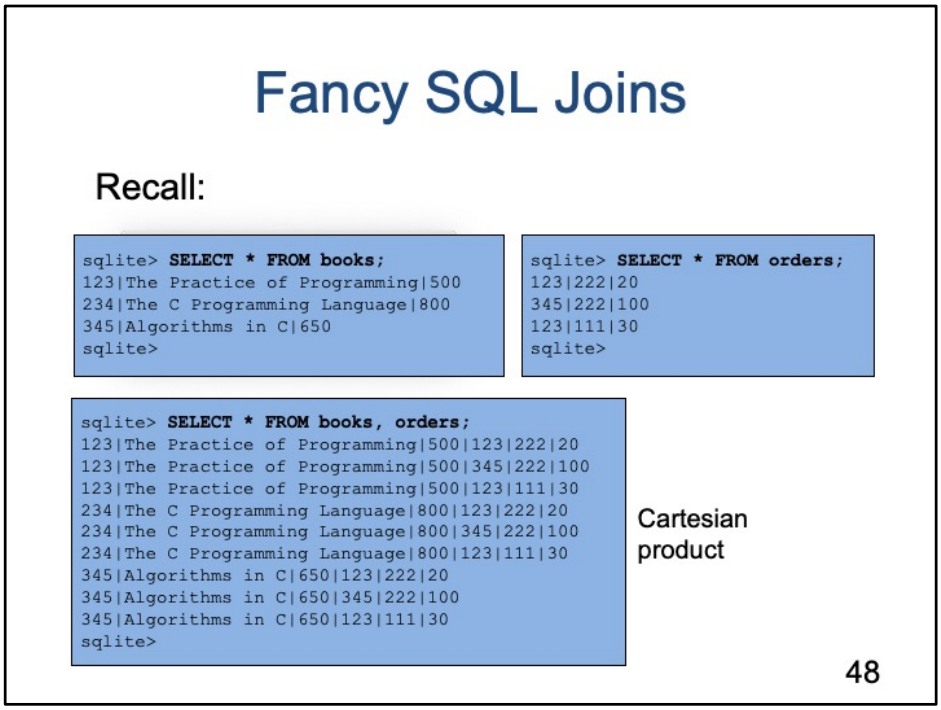
In the next lecture we'll study how to use SQLite via Python programs that you compose

Later we'll study how to use SQLite via Java, PHP, and JavaScript programs too

Summary

- We have covered:
  - Relational DBs and DBMSs
  - SQL and SQLite
  - The SQLite command-line client
- See also:
  - **Appendix 1**: Fancy SQL Joins

Appendix 1:
Fancy SQL Joins

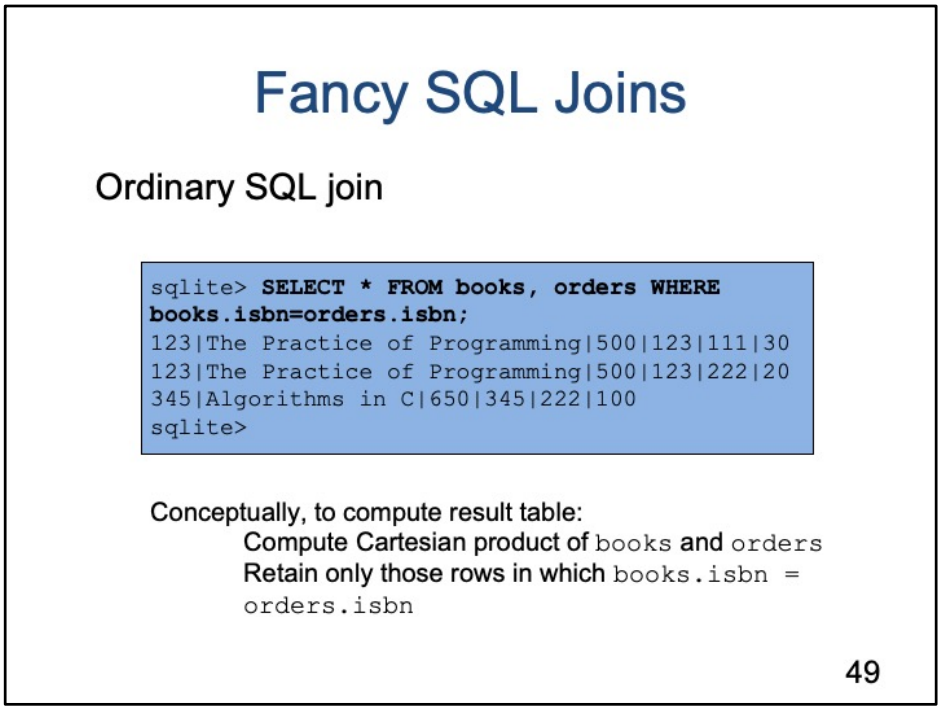


Fancy SQL Joins

Recall ordinary SQL joins, as described earlier in this lecture

Recall that, in the absence of a WHERE clause, the result of doing an ordinary SQL join of two tables is the Cartesian product of those two tables

[see slide]



Fancy SQL Joins

The Cartesian product contains many rows that are nonsensical

So it's common to use a WHERE clause

[see slide]

To compute the result table:

> In reality, a typical DBMS would use a reasonably efficient algorithm
>
> Conceptually, we could think of the DBMS as computing the full Cartesian product, and retaining only those rows that satisfy the WHERE clause

Fancy SQL Joins

Inner join

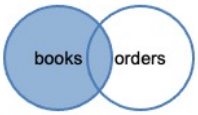
```
sqlite> SELECT * FROM books INNER JOIN orders
ON books.isbn=orders.isbn;
123|The Practice of Programming|500|123|111|30
123|The Practice of Programming|500|123|222|20
345|Algorithms in C|650|345|222|100
sqlite>
```

Same as ordinary join

Note: No row for book with `isbn` 234 is present

Fancy SQL Joins

Left outer join




```
sqlite> SELECT * FROM books LEFT OUTER JOIN
orders ON books.isbn=orders.isbn;
123|The Practice of Programming|500|123|111|30
123|The Practice of Programming|500|123|222|20
234|The C Programming Language|800|||
345|Algorithms in C|650|345|222|100
sqlite>
```

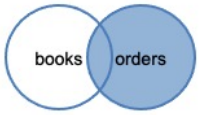
Conceptually, to compute result table:

- Compute inner join
- Add each `book` row that is missing, padded with `NULL` fields

Fancy SQL Joins

Right outer join

```
SELECT * from books
   RIGHT OUTER JOIN orders
   ON books.isbn = orders.isbn;
```



Conceptually, to compute result table:

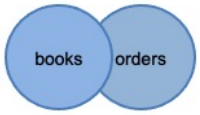
Compute inner join
Add each `orders` row that is missing,
padded with `NULL` fields

Not supported by SQLite
But could use left outer join with tables switched!

Fancy SQL Joins

Full outer join

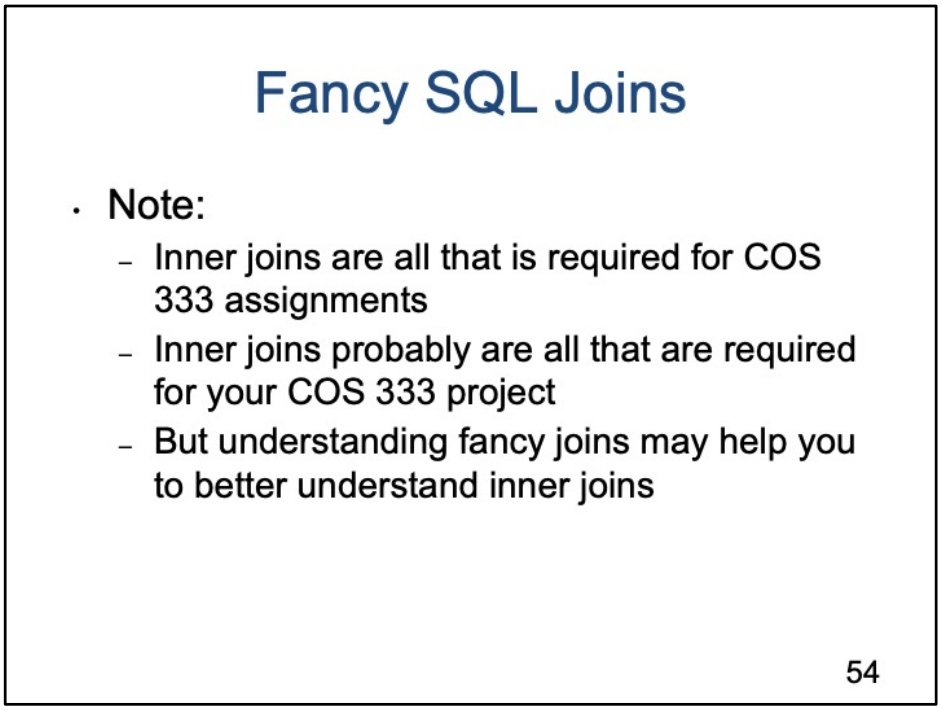
```
SELECT * from books
   FULL OUTER JOIN orders
   ON books.isbn = orders.isbn;
```



Conceptually, to compute result table:

Compute inner join
Add each `book` row that is missing,
padded with NULL fields
Add each `orders` row that is missing,
padded with NULL fields

Not supported by SQLite



Fancy SQL Joins

Note:

Inner joins are all that is required for COS 333 assignments

(If you think you need fancier joins, then you're thinking about the problem incorrectly)

Inner joins probably are all that are required for your COS 333 project

But understanding fancy joins may help you to better understand inner joins