

Database Programming (Part 1)

Copyright © 2022 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
 - **Databases (DBs)** and **database management systems (DBMSs)**...
 - With a focus on **relational** DBs and DBMSs...
 - With a focus on the **SQLite** DBMS...
 - With a focus on **programming** with SQLite

2

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
- **Database management system (DBMS)**
 - Software that maintains a database
- **Database administrator (DBA)**
 - A person who administers DBs and DBMSs

4

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

- A good DBMS used by good DBAs can:
 - Reduce redundancy
 - Avoid inconsistencies
 - **Facilitate data sharing**
 - Enforce standards
 - Apply security restrictions
 - **Maintain integrity**
 - Balance conflicting requirements
 - Insure safety (backups)

*An Introduction to Database
Systems, C. J. Date*

5

Relational DBs and DBMSs

Question: Why not simply use files?

Answer: Centralized control

Database administrators (DBAs) control the data

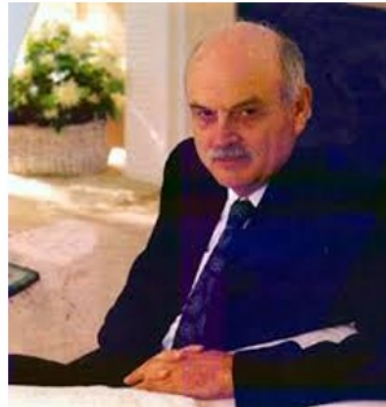
[see slide]

Note:

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

Relational DBs and DBMSs

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



Relational DBs and DBMSs

Relational DB structure:

Formally	Informally
Relations	Tables
Tuples	Rows
Attributes	Fields

7

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: Example

BOOKS

isbn	title	quantity
123	The Practice of Programming	500
234	The C Programming Language	800
345	Algorithms in C	650

ORDERS

isbn	custid	quantity
123	222	20
345	222	100
123	111	30

CUSTOMERS

custid	custname	street	zipcode
111	Princeton	114 Nassau St	08540
222	Harvard	1256 Mass Ave	02138
333	MIT	292 Main St	02142

AUTHORS

isbn	author
123	Kernighan
123	Pike
234	Kernighan
234	Ritchie
345	Sedgewick

ZIPCODES

zipcode	city	state
08540	Princeton	NJ
02138	Cambridge	MA
02142	Cambridge	MA

8

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 and SQLite

- **SQL**
 - Structured Query Language
 - Has been standardized
 - ISO/IEC 9075-1:2008

11

SQL

Structured Query Language

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?



SQL and SQLite

- **SQLite**
 - A popular free relational DBMS
 - Uses SQL
 - Lightweight

13

SQLite

A popular free relational DBMS

Uses **SQL**

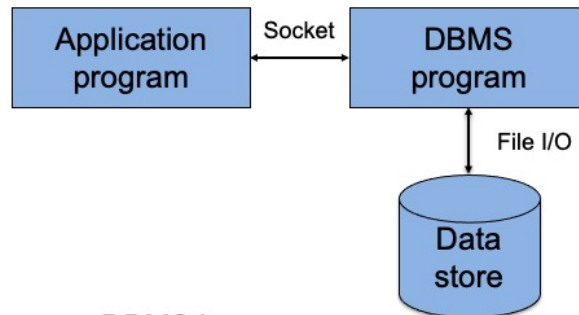
Extends SQL with additional statements
... As DBMSs typically do

Lightweight

Good choice for learning SQL

SQL and SQLite

Typical architecture when using a DBMS:



DBMS is a **program**

14

SQL and SQLite

[see slide]

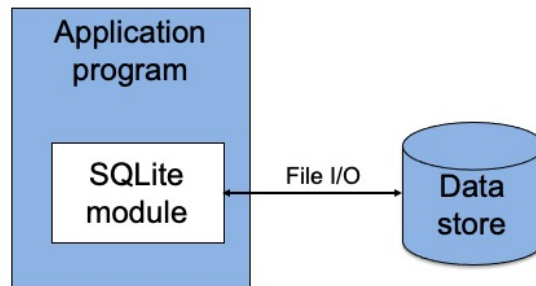
DBMS is a **program**

App pgm & DBMS pgm run in **distinct** processes

App process & DBMS process communicate via a **socket** (see upcoming *Networks* lecture)

SQL and SQLite

Typical architecture when using SQLite:



DBMS is a **module**

15

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**

SQLite Client

- **Question:** How does one use SQLite?
- **Answer:** In this course:
 - Via the `sqlite3` program (the SQLite command-line client)
 - Via programs that you compose

17

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

SQLite Client

- The `sqlite3` program

- From a shell prompt:

```
$ sqlite3 bookstore.sqlite
SQLite version 3.24.0 2018-06-04
14:10:15
Enter ".help" for usage hints.
sqlite>
```

- Type SQL/SQLite statements at `sqlite>` prompt

18

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

SQLite Client

Standard SQL Statements	SQLite Statements
Do not begin with a period	Begin with a period
Keywords are case insensitive	Keywords are case sensitive
Must end with a semicolon	Need not end with a semicolon

19

SQL and SQLite

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

There are two kinds of statements that you can enter

[see slide]

SQLite Client: Fundamentals

.help

```
sqlite> .help
.dump ?TABLE? ...      Dump the database in an SQL text format
                        If TABLE specified, only dump tables matching
                        LIKE pattern TABLE.

.help                  Show this message
.output ?FILE?         Send output to FILE or stdout
.quit                  Exit this program
.read FILENAME          Execute SQL in FILENAME
.schema ?PATTERN?      Show the CREATE statements matching PATTERN
                        Add --indent for pretty-printing
.tables ?TABLE?         List names of tables
                        If TABLE specified, only list tables matching
                        LIKE pattern TABLE.
...
sqlite>
```

20

SQL and SQLite

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

[see slide]

.help

Writes a description of each SQLite statement

SQLite Client: Fundamentals

.tables

```
sqlite> .tables  
authors      books      customers  orders      zipcodes  
sqlite>
```

.schema [table]

```
sqlite> .schema  
CREATE TABLE books (isbn TEXT, title TEXT, quantity  
INTEGER);  
CREATE TABLE authors (isbn TEXT, author TEXT);  
CREATE TABLE customers (custid TEXT, custname TEXT,  
street TEXT, zipcode TEXT);  
CREATE TABLE zipcodes (zipcode TEXT, city TEXT, state  
TEXT);  
CREATE TABLE orders (isbn TEXT, custid TEXT, quantity  
INTEGER);  
sqlite> .schema books  
CREATE TABLE books (isbn TEXT, title TEXT, quantity  
INTEGER);  
sqlite>
```

21

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

SQLite Client: Data Types

SQLite Data Type	Python Data Type
INTEGER	int
REAL	float
TEXT	str
BLOB	bytes

SQLite: NULL
Python: None

22

SQL and SQLite

Speaking of data types...

[see slide]

BLOB is an abbreviation for “binary large object”

SQLite Client: Selecting Data

SELECT *expr*, ... **FROM** *table*, ... [**WHERE** *condition*]
[**ORDER BY** *column* [**ASC** | **DESC**]];

```
sqlite> SELECT * FROM books;
123|The Practice of Programming|500
234|The C Programming Language|800
345|Algorithms in C|650
sqlite> SELECT isbn, title FROM books;
123|The Practice of Programming
234|The C Programming Language
345|Algorithms in C
sqlite> SELECT * FROM books ORDER BY quantity DESC;
234|The C Programming Language|800
345|Algorithms in C|650
123|The Practice of Programming|500
sqlite>
```

Note: The result is a table

23

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

SQLite Client: Selecting Data

WHERE clauses:

```
sqlite> SELECT * FROM books WHERE quantity=650;
345|Algorithms in C|650
sqlite> SELECT * FROM books WHERE quantity>=650;
234|The C Programming Language|800
345|Algorithms in C|650
sqlite> SELECT * FROM orders WHERE isbn=123 AND custid=222;
123|222|20
sqlite> SELECT * FROM orders WHERE isbn=123 OR custid=222;
123|222|20
345|222|100
123|111|30
sqlite>
```

24

SQL and SQLite

[see slide]

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

SQLite Client: Selecting Data

The `LIKE` operator and wildcards:

```
sqlite> SELECT * FROM books WHERE title LIKE 'The%';  
123|The Practice of Programming|500  
234|The C Programming Language|800  
sqlite> SELECT * FROM books WHERE title LIKE '%of%';  
123|The Practice of Programming|500  
sqlite> SELECT * FROM books WHERE title LIKE 'T_e%';  
123|The Practice of Programming|500  
234|The C Programming Language|800  
sqlite>
```

25

SQL and SQLite

You can use wildcards:

- % matches any 0 or more characters

- _ matches any one character

[see slide]

SQLite Client: Selecting Data

Case (in)sensitivity:

```
sqlite> SELECT * FROM books WHERE title LIKE 't_e%';  
123|The Practice of Programming|500  
234|The C Programming Language|800  
sqlite> PRAGMA case_sensitive_like=ON;  
sqlite> SELECT * FROM books WHERE title LIKE 't_e%';  
sqlite>
```

26

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...

SQLite Client: Selecting Data

The ESCAPE clause for the LIKE operator

```
sqlite> SELECT * FROM books WHERE title LIKE 'The%'
123|The Practice of Programming|500
234|The C Programming Language|800
sqlite> SELECT * FROM books WHERE title LIKE 'The\%' ESCAPE '\';
sqlite>
```

28

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]

SQLite Client: Joining Tables

Full Cartesian product of 2 tables:

```
sqlite> SELECT * from books, authors;  
123|The Practice of Programming|500|123|Kernighan  
123|The Practice of Programming|500|123|Pike  
123|The Practice of Programming|500|234|Kernighan  
123|The Practice of Programming|500|234|Ritchie  
123|The Practice of Programming|500|345|Sedgewick  
234|The C Programming Language|800|123|Kernighan  
234|The C Programming Language|800|123|Pike  
234|The C Programming Language|800|234|Kernighan  
234|The C Programming Language|800|234|Ritchie  
234|The C Programming Language|800|345|Sedgewick  
345|Algorithms in C|650|123|Kernighan  
345|Algorithms in C|650|123|Pike  
345|Algorithms in C|650|234|Kernighan  
345|Algorithms in C|650|234|Ritchie  
345|Algorithms in C|650|345|Sedgewick
```

29

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

SQLite Client: Joining Tables

More reasonable join of 2 tables:

```
sqlite> SELECT * from books, authors WHERE  
books.isbn=authors.isbn;  
123|The Practice of Programming|500|123|Kernighan  
123|The Practice of Programming|500|123|Pike  
234|The C Programming Language|800|234|Kernighan  
234|The C Programming Language|800|234|Ritchie  
345|Algorithms in C|650|345|Sedgewick  
sqlite>
```

30

SQL and SQLite

So it's common to specify an appropriate WHERE clause

[see slide]

SQLite Client: Joining Tables

Qualifying fields:

```
sqlite> SELECT title, quantity FROM books,  
orders WHERE books.isbn=orders.isbn;  
Error: ambiguous column name: quantity  
sqlite> SELECT title, orders.quantity FROM  
books, orders WHERE books.isbn=orders.isbn;  
The Practice of Programming|20  
The Practice of Programming|30  
Algorithms in C|100  
sqlite>
```

31

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.

SQLite Client: Joining Tables

Joining more than 2 tables:

```
sqlite> SELECT custname, title, orders.quantity
FROM books, customers, orders WHERE
books.isbn=orders.isbn AND
orders.custid=customers.custid;
Harvard|The Practice of Programming|20
Harvard|Algorithms in C|100
Princeton|The Practice of Programming|30
sqlite>
```

32

SQL and SQLite

[see slide]

You can join more than two tables

Aside: Joining Tables Warning

Beware:

In bookstore.sqlite some books have no orders

```
sqlite> SELECT * FROM books, orders WHERE  
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>
```

No row for isbn 234 is in result table

Beware (Assignment 1):

In reg.sqlite some courses have no professors

33

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>
```


SQLite Client: Adding Indices

```
CREATE INDEX index ON table (field);
```

```
sqlite> CREATE INDEX books_index ON books (isbn);  
sqlite> .schema books  
CREATE TABLE books  
(isbn TEXT, title TEXT, quantity INTEGER);  
CREATE INDEX books_index ON books (isbn);  
sqlite>
```

Adds an **index** on the `isbn` field of the `books` table
Allows fast search on the `isbn` field

41

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  
$
```

SQLite Client: Reading

To read bookstore.sqlite from a text file:

```
$ emacs bookstore.sql
...
$ sqlite3 bookstore.sqlite
sqlite> .read bookstore.sql
sqlite> .quit
$
```

43

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]

SQLite Client: Writing

To write bookstore.sqlite to a text file:

```
$ sqlite3 bookstore.sqlite
sqlite> .output bookstorebackup.sql
sqlite> .dump
sqlite> .quit
$
```

44

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)

SQLite Client

- **Question:** How does one use SQLite?
- **Answer:** In this course:
 - Via the `sqlite3` program (the SQLite command-line client)
 - Via programs that you compose...

45

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:

```
sqlite> SELECT * FROM books;  
123|The Practice of Programming|500  
234|The C Programming Language|800  
345|Algorithms in C|650  
sqlite>
```

```
sqlite> SELECT * FROM orders;  
123|222|20  
345|222|100  
123|111|30  
sqlite>
```

```
sqlite> SELECT * FROM books, orders;  
123|The Practice of Programming|500|123|222|20  
123|The Practice of Programming|500|345|222|100  
123|The Practice of Programming|500|123|111|30  
234|The C Programming Language|800|123|222|20  
234|The C Programming Language|800|345|222|100  
234|The C Programming Language|800|123|111|30  
345|Algorithms in C|650|123|222|20  
345|Algorithms in C|650|345|222|100  
345|Algorithms in C|650|123|111|30  
sqlite>
```

Cartesian
product

48

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

Ordinary SQL join

```
sqlite> SELECT * FROM books, orders WHERE  
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>
```

Conceptually, to compute result table:

Compute Cartesian product of `books` and `orders`
Retain only those rows in which `books.isbn =`
`orders.isbn`

49

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

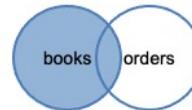
```
sqlite> SELECT * FROM books INNER JOIN orders
ON books.isbn=orders.isbn;
123|The Practice of Programming|500|123|111|30
123|The Practice of Programming|500|123|222|20
345|Algorithms in C|650|345|222|100
sqlite>
```

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:

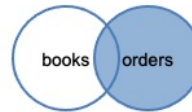
- Compute inner join

- 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

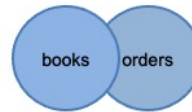
Not supported by SQLite

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
- 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

54

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