表格的好处&局限性

储存大量数据，便于排序...

但有上限——规模问题

提前决定数据的布局

csv——flat file database；其他，比如tsv；但只能保存静态值，不能像excle一样去求和

csv 读写

```python
import csv
with open("favorite TV shows - From Respones 1.csv","r"):
        reader=csv.reader(file)
        next(reader)
        for row in reader:
                print(row[1])

```

改进：DictReader，这个函数天生把第一行看成键-值。如果不存在第一行头文件的话，不能正确使用这个功能

```python
import csv
with open("favorite TV shows - From Responses 1.csv","r"):
        reader=csv.DictReader(file)
        for row in reader:
                print(row["title"])
```

求和，排序，标准化

```python
import csv
titles=set()
with open("favorite TV shows - From Responses 1.csv","r"):
        reader=DictReader(file)
        for row in reader:
                titles.add(row["title"].strip().upper())#大写，除去开头结尾

for title in sorted(titles):#排序已经封装好了
        print(title)
```

```
        print(title)
```

用字典进一步改良

```python
import csv
titles={}
with open("favorite TV shows - From Responses 1.csv","r"):
        reader=DictReader(file)
        for row in reader:
                title=(row["title"].strip().upper())#大写，除去开头结尾的无关
                if title in titles:
                        titles[title]=+1
                else:
                        titles[title]=1#第一个出现，之前自己想出来了欸，开心
def f(title):
        return titles[title]
#定义一个函数，输入键，返回值
for title in sorted(titles,key=f,reverse=True):#排序已经封装好了
        print(title)
#f而不写f()是为了防止不断调用f，使程序变慢
```

lambda fuction

key=f改为：

```python
key=lambda title:titles[title]
```

“匿名函数”，一次性使用，冒号前面是传入值，冒号后面是返回值

lambda：其他语言中也存在，并且有复杂的词源

计数

```python
import csv

title = input("Title: ").strip().upper()

with open("Favorite TV Shows - Form Responses 1.csv", "r") as file:
    reader = csv.DictReader(file)

    counter = 0
```

```
    for row in reader:
        if row["title"].strip().upper() == title:
            counter += 1

print(counter)


```

上述都是flat file database，而不是ralational database。后者和电子表格比较相近，在硬数据和查找之间存在运行的软件

SQLite

SQL wiki，结构化查询语言，可以和其他语言一起用

sqlite3，一种供查询的命令行语言

CRUD

create，read，update，delete，(insert) 四种基本功能

```sql
CREATE TABLE table (column type,...)
```

其封装好，自动化读取第一行的指令是

```sql
.import 'filename.csv'
```

选择单个的行/全部的行

按照惯例，会将sql命令大写，表格名之类的小写

```sql
SELETE columns FROM table
SELETE * FROM table
```

一些常见的更改所检索的数据的函数：

从title中找含有office，但是前后可能有空格，不精确的状况

```sql
SELECT title FROM shows WHERE title LIKE "%Office%"
```

计数，和之前python实现的相似

```sql
SELECT UPPER(title),COUNT(title) FROM shows GROUP BY UPPER(title)
```

加上排序

```sql
SELECT UPPER(title),COUNT(title) FROM shows GROUP BY UPPER(title) ORDER
```

降序，前十

```sql
SELECT UPPER(title),COUNT(title) FROM shows GROUP BY UPPER(title) ORDER
```

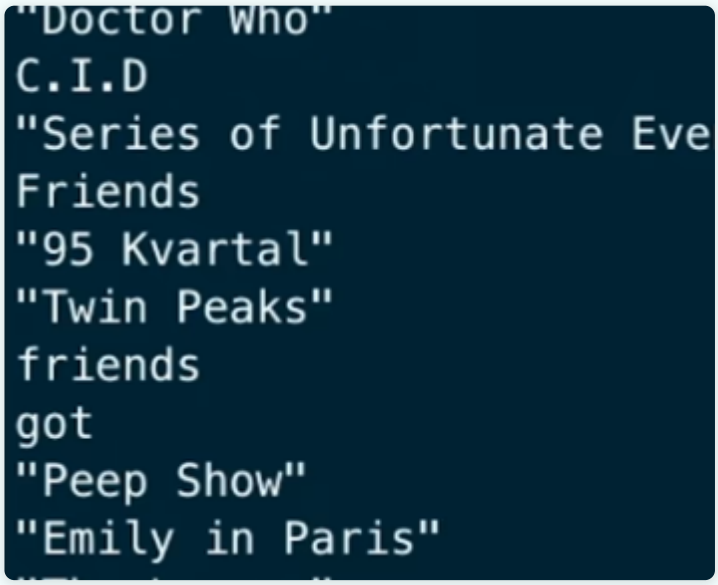
储存

```sql
.save filename.db
```

Q运行时间是什么？ O（n），但后面可以优化

查找某一类别

```sql
SELECT title FROM shows WHERE genres LIKE "%Comedy%"
```

> 诶，哈佛大学的学生怎么也会和我一样沉迷瓦夏啊（恼
>
> 
>

但是无法区分近似，比如音乐和音乐剧

```sql
SELECT title FROM shows WHERE genres LIKE "%Music%"
```

可以暴力解，比如

```sql
SELECT title FROM shows WHERE genres LIKE "%Music,%" OR "%Music%"
```

于是需要用不同的方式导入csv...

插入新的内容

```sql
INSERT INTO table(column,...) VALUES(value,...);
INSERT INTO shows(Timestamp,title,genres) VALUES("now","The Muppet Show"
```

更新

```sql
```

```sql
UPDATE shows SET genres="Comedy,Drama,Musical" WHERE title ="The Muppet S
```

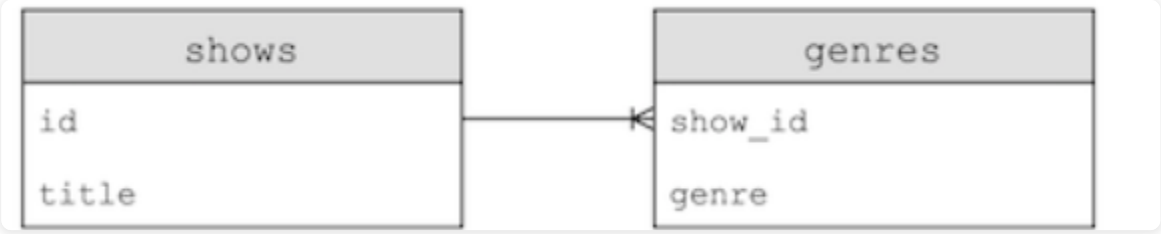
删除

```sql
DELETE FROM shows WHERE title LIKE "friends";
```

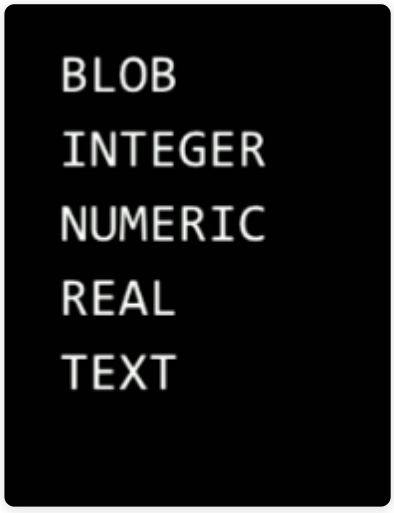
(太繁琐了，一方面就在作业里慢慢学，然后后面就下官方的note作为索引吧)

relation database

一个例子，用一个id串联两个表



数据类型



real是浮点数；blob是二进制；numeric是数字样数据，比如年月日

和python的联用

```python
import csv

from cs50 import SQL

open("shows.db", "w").close()#创建了一个可以交互的空文件
db = SQL("sqlite:///shows.db")#cs50内置的函数，打开上述的空文件
```

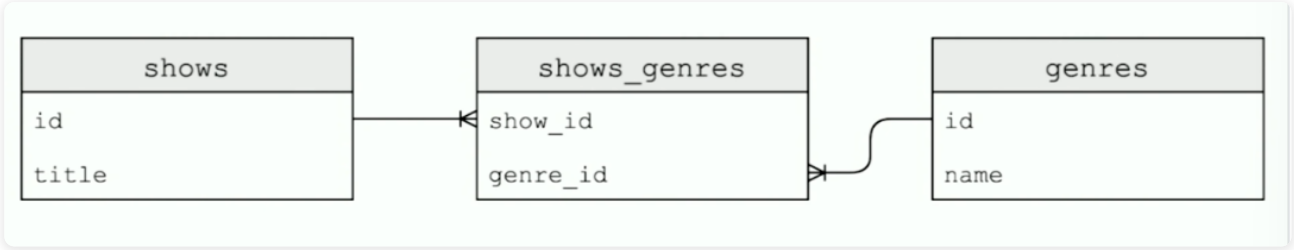
```
db.execute("CREATE TABLE shows (id INTEGER, title TEXT, PRIMARY KEY(id))"
db.execute("CREATE TABLE genres (show_id INTEGER, genre TEXT, FOREIGN KE
#原理就是把python代码传递到SQL的命令行中

with open("Favorite TV Shows - Form Responses 1.csv", "r") as file:
    reader = csv.DictReader(file)
    for row in reader:
        title = row["title"].strip().upper()

        id = db.execute("INSERT INTO shows (title) VALUES(?)", title)#INS
        #? 即SQL里的占位符，像c里的%i
        #KEY会因为SQL内置的功能自动增加
        for genre in row["genres"].split(", "):#python中的拆分功能，把用，分
            db.execute("INSERT INTO genres (show_id, genre) VALUES(?, ?)"
```

到现在，得到了干净的便于处理的数据，比如可以通过show名字，对应id，再在子表中查询其genre

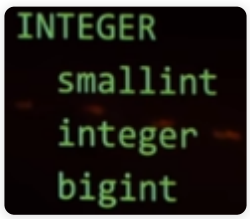
继续规范化



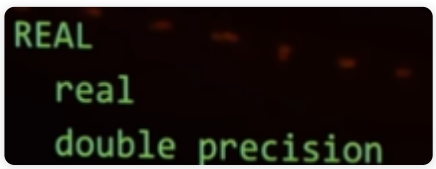
多对多关系，通过中间表达成

数据类型

除了上述数据类型，还有子类型。



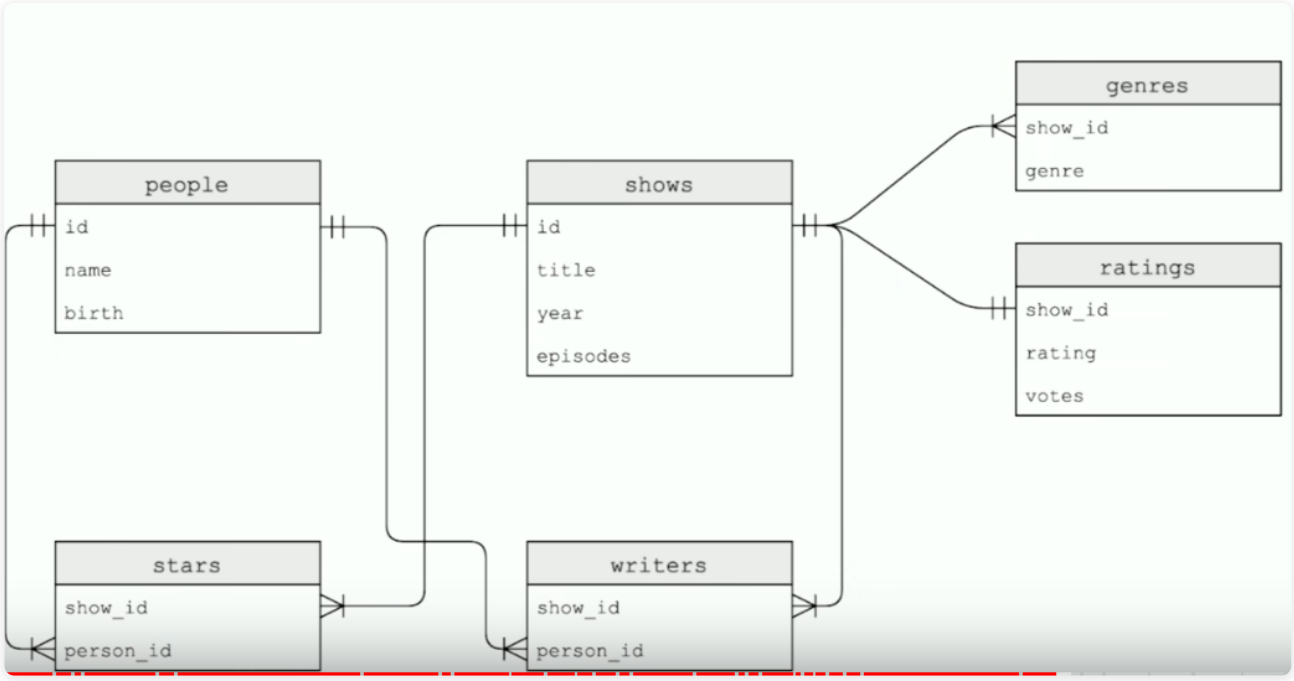
int的子类型



```
NUMERIC
  boolean
  date
  datetime
  numeric(scale,precision)
  time
  timestamp
```

```
TEXT
  char(n)
  varchar(n)
  text
```

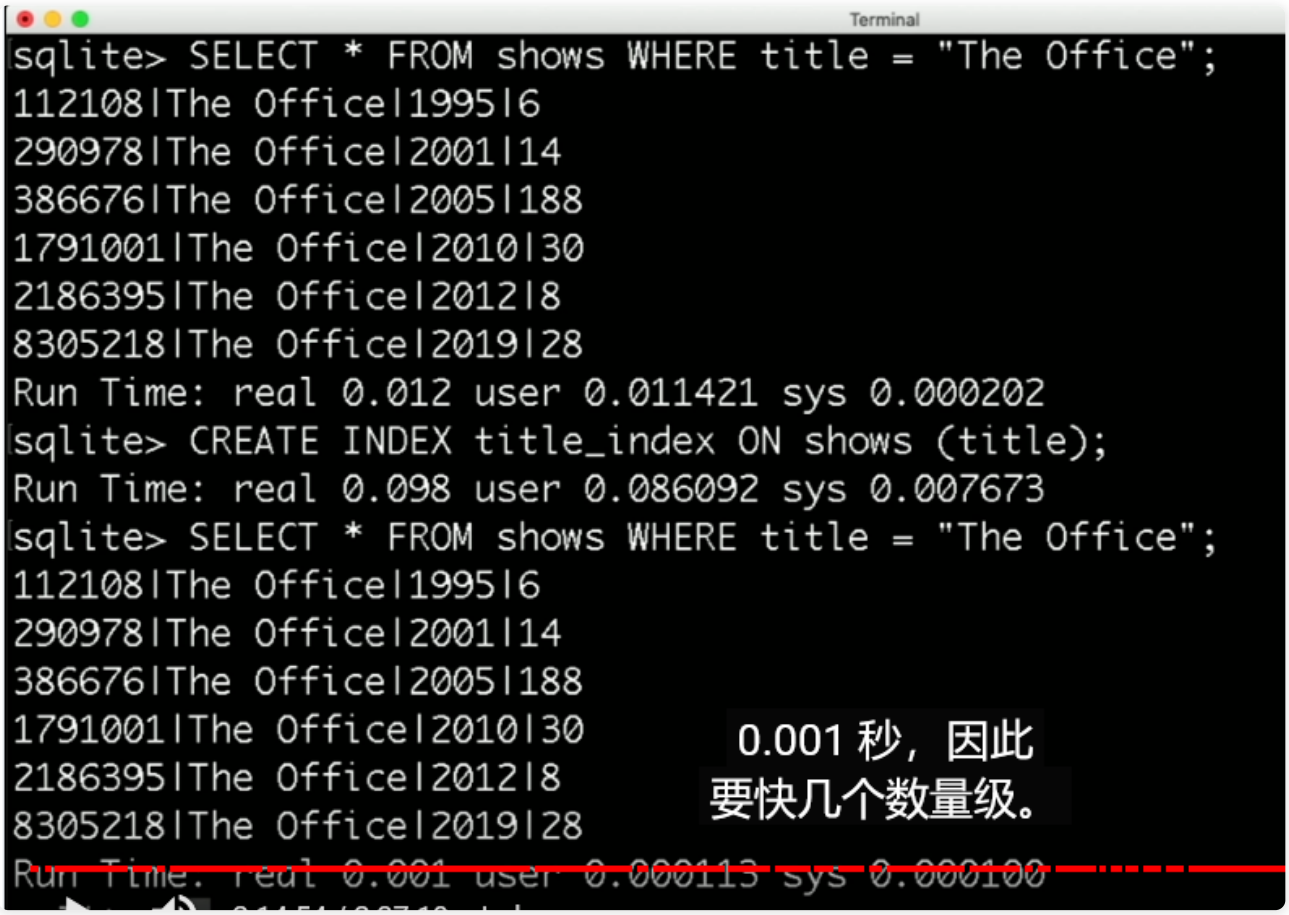
IMDb，复杂数据库的样子：



INDEX

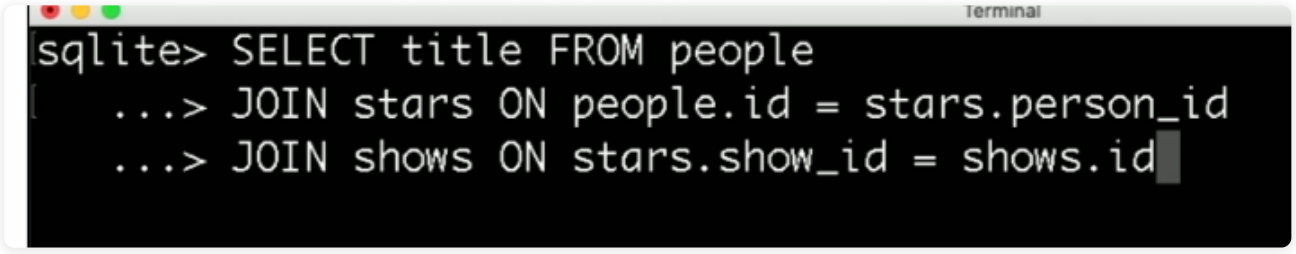
对数时间，通过数据结构B-tree的使用

```
CREATE INDEX title_index ON shows (title);
```



JOIN

嵌套查询的升级版

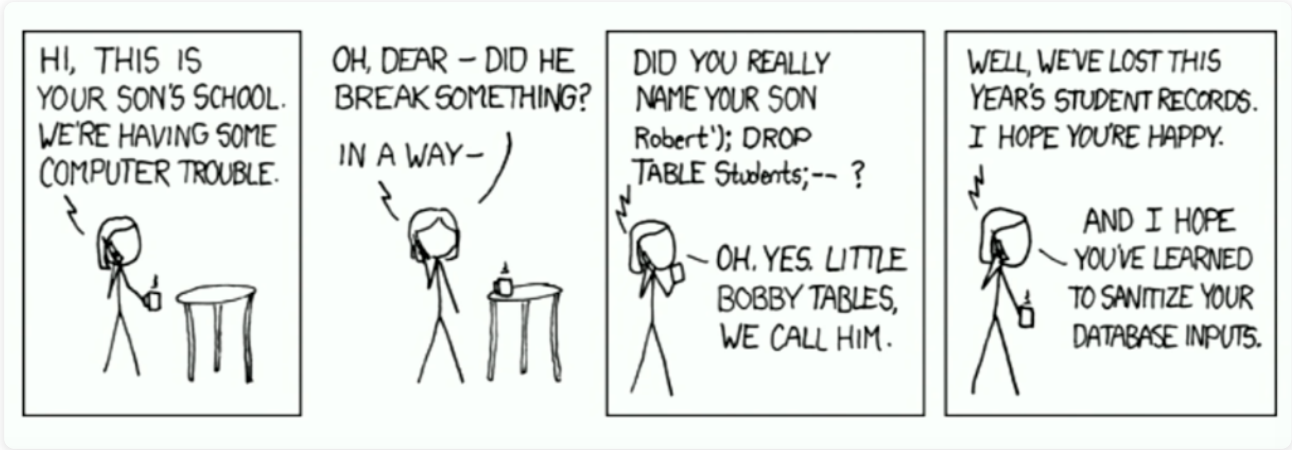


（再加个WHERE...）

SQL injection attacks

恶意插入delete update之类的有实际含义的字母。不用占位符？的话就会产生严重后果，就会真的解读成SQL代码。通过使用三方库可以预防

SQL注入攻击的meme



race condition竞争条件

点赞按照时间顺序混合在一起，并且混合在不同的服务器上。但碰巧时间数目一样，可能会被两个会被记作一个

（即多线程条件）

通过事务，锁定一个表，使其先后执行