Introduction to Database Systems: CS312 Building Larger Database systems A discussion of code

Oliver Bonham-Carter

3 March 2022

Overview of The Entity-Relationship Model Design

Consider these!

- What is the *data* to store in the database?
- What are the *relationships* between the *entities* of information?
- What is the conceptual *design* of a system to link all this information together?

ER Model Basics

Keys

- **Primary keys**: Unique identifiers for the row of information sharing a relation (n-tuple).
- **Super keys**: A superkey is a set of attributes within a table whose values can be used to uniquely identify a n-tuple.
- **Candidate keys**: is a minimal set of attributes necessary to identify a n-tuple.
- **SuperKeys**: a set of attributes within a table whose values can be used to uniquely identify a tuple (each row is unique from the other rows)

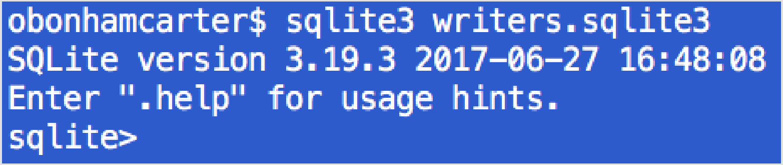
Keys

You will note the importance of keys once you start storing your data in your own databases!

Create the file!

The terminal command to open a new database

```
sqlite3 writersAndBooksDB.sqlite3
```



```
obonhamcarter$ sqlite3 writers.sqlite3
SQLite version 3.19.3 2017-06-27 16:48:08
Enter ".help" for usage hints.
sqlite>
```

The Writers Table

From last time...

Create Table command

```
DROP TABLE Writers;
CREATE TABLE Writers (
    id INTEGER NOT NULL PRIMARY KEY,
    firstName VARCHAR NOT NULL,
    middleName VARCHAR,
    lastName  VARCHAR NOT NULL,
    birthDate VARCHAR NOT NULL,
    deathDate VARCHAR,
    countryOfOrigin VARCHAR NOT NULL);
```

- Drop Table: if available, remove old table during update
- .schema `Writers`
- Note: id INTEGER NOT NULL PRIMARY KEY,
    - This attribute will be used to ensure all rows are unique
    - Used to connect to other tables

Adding Data to Writers Table

From last time...

Insert Commands

```
INSERT INTO Writers VALUES(1, "Francis","Scott",
   "Fitzgerald", "24Sept1896", "21Dec1940", "USA");

INSERT INTO Writers VALUES(2, "Arthur", "Conan",
   "Doyle", "22May1859", "7July1930", "UK");

INSERT INTO Writers VALUES(3, "Ernest", "Miller",
   "Hemingway", "21July1899", "2July1961", "USA");

INSERT INTO Writers VALUES(4, "John", "Edward",
   "Williams", "29Aug1922", "3Mar1994", "USA");
```

The Books Table

Create Table command

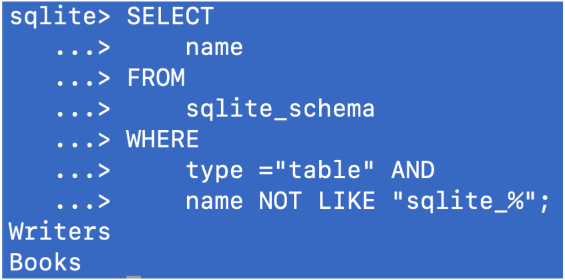
```
DROP TABLE Books;
CREATE TABLE Books(
    id INTEGER NOT NULL,
    title VARCHAR NOT NULL,
    year  VARCHAR NOT NULL,
    catagory VARCHAR NOT NULL,
    price NUMERIC NOT NULL );
```

- .schema Books
- All attributes must be present since “NOT NULL” is the *rule*

New Way To List SQL Tables

List Tables command

```
SELECT
    name
FROM
    sqlite_schema
WHERE
    type ="table" AND
    name NOT LIKE "sqlite_%";
```



```
sqlite> SELECT
   ...>     name
   ...> FROM
   ...>     sqlite_schema
   ...> WHERE
   ...>     type ="table" AND
   ...>     name NOT LIKE "sqlite_%";
Writers
Books
```

Add Data to the Books Table

Insert Commands

```
/* Populate the table */

/* "Francis","Scott", "Fitzgerald" */
INSERT INTO Books VALUES(1,"The Great Gatsby","1925","F",5);
INSERT INTO Books VALUES(1,"This Side of Paradise","1920","F",8);
INSERT INTO Books VALUES(1,"Tender is the Night","1934","F",9.50);
INSERT INTO Books VALUES(1,"A Life in Letters","1975","nF",15);
/* "nF" not written by this person */


/* "Arthur", "Conan", "Doyle" */
INSERT INTO Books VALUES(2,"The Hound of the Baskervilles","1902","D",6.50);
INSERT INTO Books VALUES(2,"The Adventures of Sherlock Holmes","1892","D",10);
INSERT INTO Books VALUES(2,"The Lost World","1912","D",13);
INSERT INTO Books VALUES(2,"The Valley of Fear","1915","D",6);
```

Add Data to the Books Table

Insert Commands

```
/* Populate the table */

/* "Ernest", "Miller", "Hemingway" */
INSERT INTO Books VALUES(3,"The Old Man and the Sea","1951","H",10);
INSERT INTO Books VALUES(3,"Men Without Women","1927","H",12);
INSERT INTO Books VALUES(3,"A Moveable Feast: The Restored Edition","2009","nH",15);
INSERT INTO Books VALUES(3,"Green Hills of Africa","1935","H",15);

/* "John", "Edward", "Williams" */
INSERT INTO Books VALUES(4,"Stoner","1965","W",27);
INSERT INTO Books VALUES(4,"Nothing but the Night","1948","W",14);
INSERT INTO Books VALUES(4,"Butcher’s Crossing","1960","W",20);
INSERT INTO Books VALUES(4,"The Broken Landscape: Poems","1949","W",20);
```

Queries!

Drawing from a single table

Query in English

Show me all rows in the Books table

SQL programming

```
SELECT * FROM Books;
```

Show me all rows in the Writers table

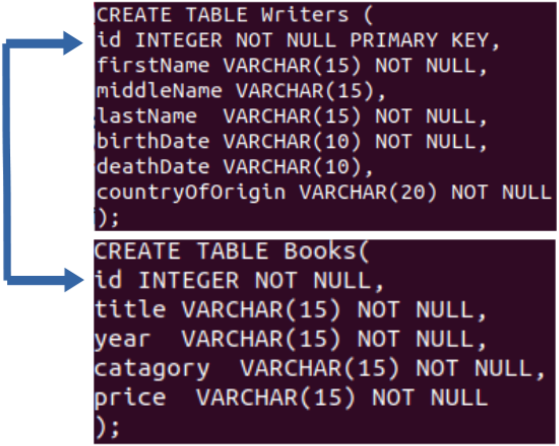
```
SELECT * FROM Writers;
```

Show me only rows for writers in the table who are from USA

```
SELECT * FROM Writers WHERE countryOfOrigin ==
"USA";
```

Table Relationship

- Relationship between the two tables by `id` (a primary key).
- This relationship will be used to connect tables when writing queries that takes information from **both** tables at the same time.



```
CREATE TABLE Writers (
id INTEGER NOT NULL PRIMARY KEY,
firstName VARCHAR(15) NOT NULL,
middleName VARCHAR(15),
lastName  VARCHAR(15) NOT NULL,
birthDate VARCHAR(10) NOT NULL,
deathDate VARCHAR(10),
countryOfOrigin VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE Books(
id INTEGER NOT NULL,
title VARCHAR(15) NOT NULL,
year  VARCHAR(15) NOT NULL,
catagory  VARCHAR(15) NOT NULL,
price  VARCHAR(15) NOT NULL
);
```

Queries!

Drawing from Both tables together

Query question

Show me all books written by each of the writers.

General Form

```
SELECT
   Writers.lastName, Books.title
FROM
   Writers, Books
WHERE
   Writers.ID == Books.ID;
```

Written on one line

SELECT Writers.lastName, Books.title FROM Writers, Books WHERE Writers.ID == Books.ID;

Queries!

Show me all books written by each of the writers.

```
SELECT Writers.lastName, Books.title FROM Writers, Books WHERE
Writers.ID == Books.ID;
```

Show me all writers' last names, their book titles and the year their book was written.

```
SELECT Writers.lastName, Books.title, Books.year FROM Writers, Books
WHERE Writers.ID == Books.ID;
```

Show me the last name and book title of any work by someone whose first name is “Ernest.”

```
SELECT Writers.lastName, Books.title FROM Writers, Books WHERE
writers.ID == Books.ID AND Writers.firstName == ”Ernest”;
```

Queries!

Show me all writers' lastnames, birthdays, their book titles, the year the book was written and the price of their book.

```
SELECT
  Writers.lastName, Writers.birthDate,
Books.title, Books.year, Books.price
FROM
  Writers, Books
WHERE
  Writers.ID == Books.ID;
```

Queries!

Show me the above information, but only for books less than 12 dollars.

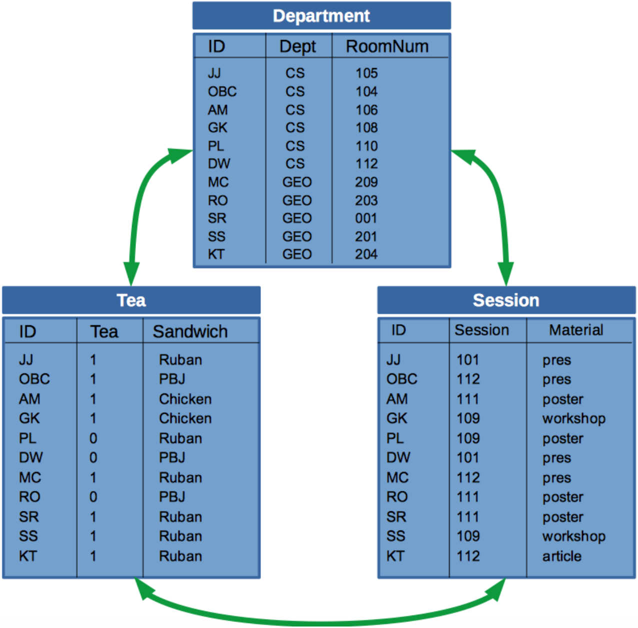
```
SELECT
  Writers.lastName, Writers.birthDate,
  Books.title, Books.year, Books.price
FROM
  Writers, Books
WHERE
  Writers.ID == Books.ID and price $<$ 12;
```

Consider this …

- Can you design and populate a database?
- Can you run queries to access particular attributes?

Try This: Create and Link Tables

You have previous example code to guide you



Department

ID	Dept	RoomNum
JJ	CS	105
OBC	CS	104
AM	CS	106
GK	CS	108
PL	CS	110
DW	CS	112
MC	GEO	209
RO	GEO	203
SR	GEO	001
SS	GEO	201
KT	GEO	204

Tea

ID	Tea	Sandwich
JJ	1	Ruban
OBC	1	PBJ
AM	1	Chicken
GK	1	Chicken
PL	0	Ruban
DW	0	PBJ
MC	1	Ruban
RO	0	PBJ
SR	1	Ruban
SS	1	Ruban
KT	1	Ruban

Session

ID	Session	Material
JJ	101	pres
OBC	112	pres
AM	111	poster
GK	109	workshop
PL	109	poster
DW	101	pres
MC	112	pres
RO	111	poster
SR	111	poster
SS	109	workshop
KT	112	article

Example code

Taken from last time...

Create

```
 CREATE TABLE department(
    ID varchar,
    Dept varchar,
    RoomNum varchar
);
```

Populate

```
INSERT INTO department VALUES ("0BC","CP","104" );
```

Query each table

Single table

Show me all rows from each of the tables, individually.

Two tables

Show me the name, dept and whether the person will have tea.

Show me the name and dept of each person who will have a Ruban.

Three tables

Show me the sandwich type and the session room number of each person.

Can you think of other interesting queries here?