Referencing data from linked tables

Normalised data - SQL JOIN

Why the need of more tables?

If we look at the structure of our previous example table books, we might notice that the table is self-contained and in a sense flat. All the data is contained in the same table.

This leads to duplication of data. For instance, the Publisher Bonnier is duplicated many times.

Books table

```
sqlite> select * from books;
author           title       isbn         publisher
---------------  ----------  -----------  ----------
John Smith       Life        0-0-0-0-0-1  Bonnier
James Woody      Love        0-0-0-0-0-2  Bonnier
Joan Carmen      Guns        0-0-0-0-0-3  Bonnier
Johnanna Boyd    Code        0-0-0-0-0-4  Bonnier
Eva Peron        Cars        0-0-0-0-0-5  Books R us
```

What's the problem with duplication of data?

One problem with duplication is obviously storage. Another one could be efficiency (it takes longer to search strings than e.g. integers). Another problem is to do with consistency.

What if we add a book but misspell Bonnier as Bonier?

Then we'd have inconsistency because we think of it as published by Bonnier but the database will consider it as published by "Bonier" instead (a completely different beast).

Fixing the duplication of Publisher

Let’s assume that there is a second table alongside books, which is called publishers:

```
sqlite> .schema publishers
CREATE TABLE publishers (publisher_id INTEGER PRIMARY KEY, name TEXT);
sqlite> SELECT * FROM publishers;
publisher_id     name
---------------  ----------
1                Bonnier
2                Books R us
```

Consistency achieved!

Now we have only one canonical publisher called Bonnier, which is good.

But how do we reference the `publishers` table from the `books` table?

Let’s change books to have it reference publisher

What if we, instead of storing the publishers’ names in the table, stored the ID of the publisher from the publishers table?

```
sqlite> select * from books;
author           title       isbn          publisherid
---------------  ----------  ------------  -----------
John Smith       Life        0-0-0-0-0-1   1
James Woody      Love        0-0-0-0-0-2   1
Joan Carmen      Guns        0-0-0-0-0-3   1
Johnanna Boyd    Code        0-0-0-0-0-4   1
Eva Peron        Cars        0-0-0-0-0-5   2
```

For reference, how did I create that table?

```
sqlite> CREATE TABLE books2(author TEXT, title TEXT,
  isbn TEXT PRIMARY KEY, publisherid INTEGER);
sqlite> INSERT INTO books2 (author, title, isbn)
  SELECT author, title, isbn FROM books;
sqlite> UPDATE books2 SET publisherid = 1 WHERE isbn < '0-0-0-0-0-5';
sqlite> UPDATE books2 SET publisherid = 2 WHERE isbn = '0-0-0-0-0-5';
sqlite> DROP TABLE books;
sqlite> ALTER TABLE books2 RENAME TO books;
```

How to list the publishers' names?

But if we now have this:

```
sqlite> select * from books;
author            title       isbn          publisherid
---------------   ----------  ------------  -----------
John Smith        Life        0-0-0-0-0-1   1
James Woody       Love        0-0-0-0-0-2   1
Joan Carmen       Guns        0-0-0-0-0-3   1
Johnanna Boyd     Code        0-0-0-0-0-4   1
Eva Peron         Cars        0-0-0-0-0-5   2
```

How do we list all books including the publisher names?

Making the connection

This is one way of listing all books with their corresponding publishers:

```
sqlite> SELECT books.author, books.title, books.isbn, publishers.name
  FROM books, publishers WHERE books.publisherid = publishers.publisher_id;
author           title       isbn          name
---------------  ----------  ------------  ----------
John Smith       Life        0-0-0-0-0-1   Bonnier
James Woody      Love        0-0-0-0-0-2   Bonnier
Joan Carmen      Guns        0-0-0-0-0-3   Bonnier
Johnanna Boyd    Code        0-0-0-0-0-4   Bonnier
Eva Peron        Cars        0-0-0-0-0-5   Books R us
sqlite>
```

But we are not pleased with the header “name”.

Fixing the column header

We can use aliases using the SQL operator AS:

```
sqlite> SELECT books.author, books.title, books.isbn,
               publishers.name  AS Publisher
          FROM books, publishers
         WHERE books.publisherid = publishers.publisher_id;
author           title       isbn          Publisher
---------------  ----------  ------------  ----------
John Smith       Life        0-0-0-0-0-1   Bonnier
James Woody      Love        0-0-0-0-0-2   Bonnier
Joan Carmen      Guns        0-0-0-0-0-3   Bonnier
Johnanna Boyd    Code        0-0-0-0-0-4   Bonnier
Eva Peron        Cars        0-0-0-0-0-5   Books R us
sqlite>
```

Using aliases

We can shorten the previous statement using aliases:

```
sqlite> SELECT b.author, b.title, b.isbn,
               p.name  AS Publisher
          FROM books b, publishers p
         WHERE b.publisherid = p.publisher_id;
author            title       isbn         publisher
---------------   ----------  -----------  ----------
John Smith        Life        0-0-0-0-0-1  Bonnier
James Woody       Love        0-0-0-0-0-2  Bonnier
Joan Carmen       Guns        0-0-0-0-0-3  Bonnier
Johnanna Boyd     Code        0-0-0-0-0-4  Bonnier
Eva Peron         Cars        0-0-0-0-0-5  Books R us
sqlite>
```

Using SQL JOIN

There is an alternative syntax to use, called JOIN. We think it is good if you have seen it:

```
sqlite> SELECT author, title, isbn, name AS Publisher
           FROM books
           JOIN publishers
             ON books.publisherid = publishers.publisher_id;
author           title       isbn          Publisher
---------------  ----------  ------------  ----------
John Smith       Life        0-0-0-0-0-1   Bonnier
James Woody      Love        0-0-0-0-0-2   Bonnier
Joan Carmen      Guns        0-0-0-0-0-3   Bonnier
Johnanna Boyd    Code        0-0-0-0-0-4   Bonnier
Eva Peron        Cars        0-0-0-0-0-5   Books R us
sqlite>
```

Summary

We moved the Publisher name to its own table consisting of `publisher_id` and `name`.

We changed the `books` table to include an id from the `publishers` table rather than the publisher name itself.

We saw how we can join two tables in a `SELECT` query so that we could see all books and their respective publisher name, connecting the `publisherid` of the `books` table with the corresponding `publisher_id` of the `publishers` table in order to get the name of the publisher.

Read

http://zetcode.com/db/sqlite/joins/

http://www.w3schools.com/sql/sql_join_left.asp

http://www.w3schools.com/sql/sql_alias.asp

http://www.w3schools.com/sql/sql_join.asp

https://en.wikipedia.org/wiki/Database_normalization