SQL Basics

Joseph Hallett

February 1, 2023

What's all this about?

We've got a database for storing data...

- It'd be nice to be able to acutally use it and make queries!

For that we need SQL:

- Structured Query Language

SQL

Query language for asking questions about databases from 1974

- Standardized in 1986 in the US and 1987 everywhere else
- Still the dominant language for queries today

Not a general purpose programming language

- Not Turing complete
- Weird English-like syntax

Standardized?

You would be so lucky!

- *In theory*, yes
- *In practice*, absolutely not

Every database engine has *small* differences…

- Some have quite big ones too!

Lots have differences in performance

- SQLite is good with strings, most others prefer numbers

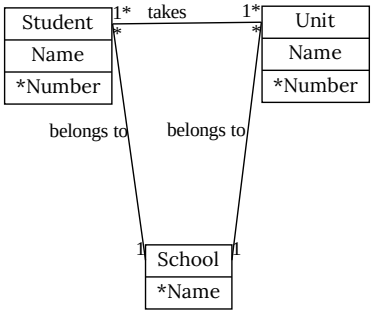
Managing these differences used to be an entire degree/job in its own right!

- Now we just manage databases badly!

I'll try and stick to SQLite's syntax…

CREATE TABLE

In the last lecture we had the following Entity relationship diagram:




```
CREATE TABLE IF NOT EXISTS student (
  name TEXT NOT NULL,
  number TEXT NOT NULL,
  PRIMARY KEY (number));

CREATE TABLE IF NOT EXISTS unit (
  name TEXT NOT NULL,
  number TEXT NOT NULL,
  PRIMARY KEY (number));

CREATE TABLE IF NOT EXISTS school (
  name TEXT NOT NULL,
  PRIMARY KEY (name));

CREATE TABLE IF NOT EXISTS class_register (
  student TEXT NOT NULL,
  unit TEXT NOT NULL,
  FOREIGN KEY (student) REFERENCES student(number),
  FOREIGN KEY (unit) REFERENCES unit(name),
  PRIMARY KEY (student, unit));
```

Lets build it in SQL

DROP TABLE

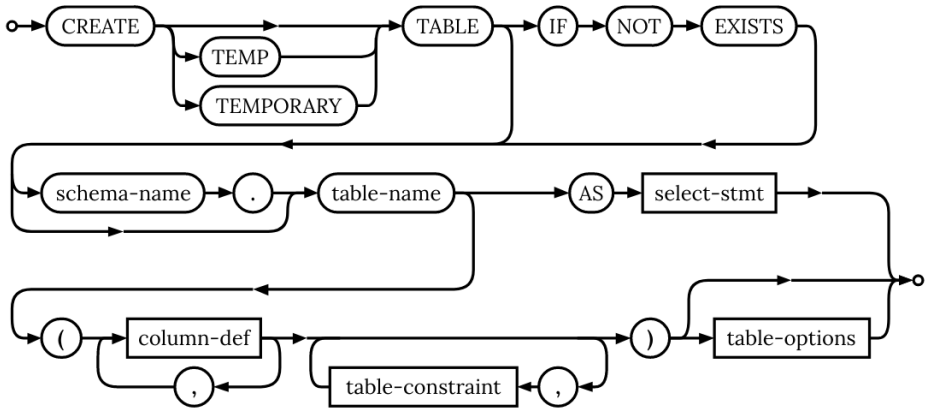
What about if we want to delete them?

```
DROP TABLE IF EXISTS class_register;
DROP TABLE IF EXISTS student;
DROP TABLE IF EXISTS unit;
DROP TABLE IF EXISTS school;
```

Syntax, syntax, syntax

If you go on the *SQLite* documentation page...

- ...you can find syntax diagrams for all of SQL!
- `https://www.sqlite.org/lang_createtable.html`



Types

When creating the fields in our database we made them all of type `TEXT`...

- What other types exist?

`INTEGER`	whole numbers
`REAL`	lossy decimals
`BLOB`	binary data (images/audio/files...)
`VARCHAR(10)`	a string of 10 characters
`TEXT`	any old text
`BOOLEAN`	True or false
`DATE`	Today
`DATETIME`	Today at 2pm

But really types

Databases sometimes *simplify* these types

- SQLite makes the following tweaks…

`INTEGER`	whole numbers
`REAL`	lossy decimals
`BLOB`	binary data (images/audio/files…)
`VARCHAR(10)`	*actually* `TEXT`
`TEXT`	any old text
`BOOLEAN`	*actually* `INTEGER`
`DATE`	*actually* `TEXT`
`DATETIME`	*actually* `TEXT`

(others may exist… *read the manual*!)

Table constraints

In the earlier examples we marked some columns as `NOT NULL`

- Others as `PRIMARY KEY` and others as `FOREIGN KEY`...
- ...what other constraints have we got

...but SQLite won't actually enforce any of these types or constraints unless you ask it to `:-(`

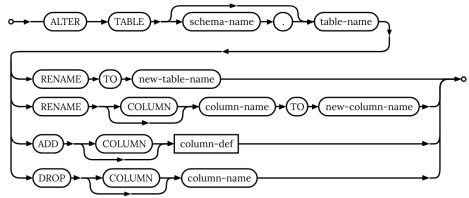
- Check out the `STRICT` keyword when creating the table.

`NOT NULL`	can't be `NULL`
`UNIQUE`	can't be the same as another row
`CHECK`	arbitrary checking (including it conforms to a regular expression)
`PRIMARY KEY`	unique, not `NULL` and (potentially) autogenerated
`FOREIGN KEY`	(IGNORED BY MARIADB) other key must exist

Can I add constraints later?

Yes with the `ALTER TABLE` statement

- But often easiest just to save the table somewhere else
- Drop the table
- Reimport it



INSERT INTO

What about if we want to add data to a table?

```
INSERT INTO unit(name, number)
VALUES ("Software Tools", "COMS100012");
```

So far

We've introduced how to:

- `CREATE TABLE`
- `DROP TABLE`
- `INSERT INTO`

Next step: querying data!

I'm going to use a database from an old iTunes library for demo purposes

- Chinook database

SELECT

Basic command for selecting rows from a table is `SELECT`

```
SELECT * FROM album
LIMIT 5;
```

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2
3	Restless and Wild	2
4	Let There Be Rock	1
5	Big Ones	3

```
SELECT * FROM artist
LIMIT 5;
```

ArtistId	Name
1	AC/DC
2	Accept
3	Aerosmith
4	Alanis Morissette
5	Alice In Chains

JOIN

Ideally we'd like those two tables combined into one...

```
SELECT *
FROM album
JOIN artist
ON album.artistid = artist.artistid
LIMIT 5;
```

AlbumId	Title	ArtistId	ArtistId	Name
1	For Those About To Rock We Salute You	1	1	AC/DC
2	Balls to the Wall	2	2	Accept
3	Restless and Wild	2	2	Accept
4	Let There Be Rock	1	1	AC/DC
5	Big Ones	3	3	Aerosmith

Reducing the columns...

Clearly there are too many columns here... lets only select the ones we need

```
SELECT album.title, artist.name
FROM album
JOIN artist
ON album.artistid = artist.artistid
LIMIT 5;
```

Title	Name
For Those About To Rock We Salute You	AC/DC
Balls to the Wall	Accept
Restless and Wild	Accept
Let There Be Rock	AC/DC
Big Ones	Aerosmith

Renaming columns

Title and *Name* aren't particularly meaningful without context

- Lets name them something sensible

```
SELECT album.title AS album,
       artist.name AS artist
FROM album
JOIN artist
ON album.artistid = artist.artistid
LIMIT 5;
```

album	artist
For Those About To Rock We Salute You	AC/DC
Balls to the Wall	Accept
Restless and Wild	Accept
Let There Be Rock	AC/DC
Big Ones	Aerosmith

I'm feeling rocky

I want to listen to something a bit rocky...

- Lets filter all the albums to the ones that have *Rock* in the title

```
SELECT album.title AS album,
       artist.name AS artist
FROM album
JOIN artist
ON album.artistid = artist.artistid
WHERE album LIKE '%Rock%'
LIMIT 5;
```

album	artist
For Those About To Rock We Salute You	AC/DC
Let There Be Rock	AC/DC
Deep Purple In Rock	Deep Purple
Rock In Rio [CD1]	Iron Maiden
Rock In Rio [CD2]	Iron Maiden

Who rocks?

So who has put out an album with *Rock* in it?

```
SELECT artist.name AS artist
FROM album
JOIN artist
ON album.artistid = artist.artistid
WHERE album.title LIKE '%Rock%'
LIMIT 5;
```

artist
AC/DC
AC/DC
Deep Purple
Iron Maiden
Iron Maiden

```
SELECT DISTINCT artist.name AS artist
FROM album
JOIN artist
ON album.artistid = artist.artistid
WHERE album.title LIKE '%Rock%'
LIMIT 5;
```

artist
AC/DC
Deep Purple
Iron Maiden
The Cult
The Rolling Stones

How many *rock* albums has each artist put out?

Lets `group` by artist and `count` the albums!

```
SELECT artist.name AS artist,
       COUNT(album.title) as albums
FROM album
JOIN artist
ON album.artistid = artist.artistid
WHERE album.title LIKE '%Rock%'
GROUP BY artist
LIMIT 5;
```

artist	albums
AC/DC	2
Deep Purple	1
Iron Maiden	2
The Cult	1
The Rolling Stones	1

Really we want this list ordered...

Lets `group` by artist and `count` the albums...

- And order it by album count!

```
SELECT artist.name AS artist,
       COUNT(album.title) as albums
FROM album
JOIN artist
ON album.artistid = artist.artistid
WHERE album.title LIKE '%Rock%'
GROUP BY artist
ORDER BY albums DESC
LIMIT 5;
```

artist	albums
Iron Maiden	2
AC/DC	2
The Rolling Stones	1
The Cult	1
Deep Purple	1

Conclusions

So thats the basics of SQL!

- You can do a *bunch* more things with SQL `SELECT` statements...
- ...you can pick them up as you write queries.
- ...most SQL engines have a bunch more counting and query functions too

Go read the documentation!