Sqlite Visualiser

A look inside Sqlite.

By:
Paul Batty

Supervisor:
Andrew Scott

March 2016

The dissertation is submitted to
Lancaster University
As partial fulfilment of the requirements for the degree of
Integrated Masters of Science in Computer Science

Abstract

The Abstract.

1 Introduction

Sqlite unlike many other databases is a small, single file, self-contained database engine often used in embedded systems, storage or as an application file. Sqlite is used in many applications such as Firefox, Android and Windows 10. In addition to its wide adaptation Sqlite is server less, and has zero configuration putting it in a unique place among the other alternative systems. Despite the extensive research and testing performed on Sqlite none have attempted to visualise this data in real time.

This paper will help provide a way to see the Sqlite database in action, providing a useful tool for developers and researchers alike in understanding and debugging the internal structure of their own databases. In order to accomplish this paper will:

- Explore in depth the how the file format is put together (section 2). And how to traverse the file (section 2).
- Look at the design and development (section 3) including testing (section 6) of the tool. And how it takes this data and visualises it (section 4). Including the user experience (section 5).
- Evaluation of the tool (section 7) and where this research could be taken beyond this paper (section 8).

2 Background

2.1 The Problem

Throughout Sqlite's history many tests, papers, and tools have been developed in order to understand, and modify the future direction of Sqlite. However, when a user wants to understand at a deeper level how Sqlite is working or finding obscure bugs, they are stuck with manually trawling through a Hex editor. This paper aims to solve this by providing a visualisation of the internal structure, as well as a update log that is updated in real time, when the database is modified.

2.2 Sqlite

2.2.1 What is Sqlite

Sqlite is a single self-contained, serverless SQL database engine. Started on 29 May 2000 by D. Richard Hipp (Hipp, 2000) from gathered inspiration while working on software for guided missiles on a battleship where they needed a self-contained portable database. (Owens, 2006) He joined up with Joe Mistachkin followed by Dan Kennedy in 2002. Version 1.0 was released in August 2000, then in just over year on the 28 November 2001 2.0 which introduced, BTrees and many of the features seen in 3.0. Which came a lot later containing a full rewrite and improvement over 2.0, with the first public release on 18 June 2004. At the time of writing this paper we are currently sitting at version 3.10.4 (Hipp, 2000).

Sqlite is open source within the public domain making it accessible to everyone. The entire library size can be 350Kib, with some option features omitted it could be reduced to around 300Kib making it incredibly small compared to what it does. In addition to this the runtime usage is minimal with 4Kib stack space and 100Kib heap, allowing it to run on almost anything. Sqlite's main strength is that the entire database in encoded into a single portable file, that can be read, on any system whether 32 or 64 bit, big or small endian. It is often seen as a replacement for storage files rather then a database system (Hipp, 2000).

2.2.2 Where is Sqlite used

As Sqlite is a minimal portable database engine it has primarily two uses, The first as a relation database like any other, the second as a application file format. In the former case Sqlite would be set up the same way a tradition database system would be, with the primary purpose to hold the back end storage information. The latter case is more unique to Sqlite and is what sets it apart from the traditional database engines (Hipp, 2000). Because of this Sqlite is used everywhere, a few of

the big names include, apple, android, adobe even a special version was produced specifically for windows 10. In fact Sqlite might be the single most deployed software currently (Hipp, 2000, 2015).

2.2.3 How is Sqlite works

Sqlite consist of three main parts, the backend, core and the SQL compiler. Figure 1 shows the architectural digram of Sqlite.

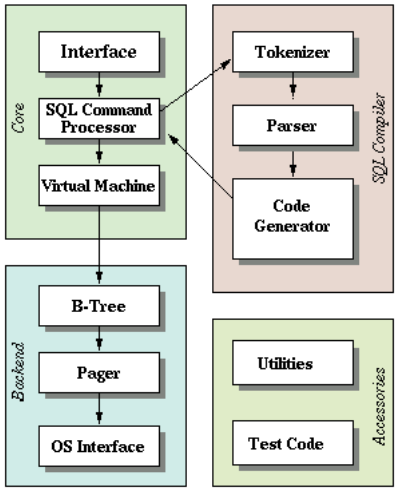

Figure 1: Sqlite architectural diagram (Hipp, 2000)

The SQL compiler is designed to take SQL strings and turn them into valid SQL commands, it is made up of three parts. Firstly the tokenizer takes a string containing SQL statements and turns it into tokens passing them one bay one to the parser. The parser takes the tokens and assign meaning to them. Lastly the code generator takes the tokens from the parser and assembles complete SQL statements that are to be ran on the database file.

The Core itself is actually a virtual machine implementing a specifically designed computing engine to manipulate database files. The interface module defines the interface between the virtual machine and the SQL library including the external API. Knowing this we can see that the virtual machine takes the code output from the code generator to manipulate the backend / database.

The final main module is the backend which is the main focus of this paper, controlling the file format. The OS interface contains an abstraction layer to write the files to disk or memory. The Pager takes the B-Trees and is responsible for

reading, writing and caching them. This involves locking, rollback and atomic commits of the database. Sqlite uses a B-Tree system to navigate and store the data on disk, this module contains the implementation of the B-tree and as such the defines the file format.

The last module, accessories is made up of two parts. Utilities contain functions that are used all around Sqlite, such as memory allocation, string comparison, random number generator and symbol tables. The test section contains all the test scripts and only exist for testing purposes, of which contains over 811 times more code then the actual project.

2.3 The Sqlite file format

2.3.1 The page system

Sqlite is made up of pages..

2.3.2 The Trees and Cells

The Trees and cells...

2.3.3 Encoding of the data

The Data is...

2.4 Similar Programs

2.4.1 Sqlite browser

One Similar program...

3 Design

3.1 System architecture

3.1.1 High level Overview

The Overall design...

3.1.2 Module Overview

The first module..

4 Implementation

4.1 The tools

I used..

4.2 The Modules

4.2.1 Database parser

The Database parser...

4.2.2 Log

The Log...

4.2.3 Live Updater

The Live updater...

5 System Operation

6 Testing

6.1 Code Tests

6.1.1 Unit tests

Unit testing...

6.1.2 Integration tests

Integration tests...

7 Evaluation

7.1 System Performance

The system was...

7.2 Design principles

I followed..

8 Conclusion

9 References

Owens M. (2006). The Definitive Guide to SQLite, Berkeley, California, Apress.

Hipp R. (2000) Sqlite. On line publication, Wyrick Company, Inc, https://www.sqlite.org/. Last Accessed 17th January 2016.

Hipp R. (2015) SQLite: The Database at the Edge of the Network. On line Video, Skookum, https://www.youtube.com/watch?v=Jib2AmRb_rk. Last Accessed 17th January 2016.

10 Appendix

11 Code

11.1 More code

11.1.1 Even more code

This is some very important code. This is a very long sentance in order to see hoow latex copes with very very long lines of non stop text.

```
1    // main
2    public static void main(String args[]) {
3        System.out.println("Hello World");
4    }
```

And so on..