# SQLite Internals

by Abdur-Rahmaan Janhangeer

## Introduction

SQLite is a file-based database which is extremely reliable and stable. It is the world's most used database. The codebase and mechanisms it used is extremely complex. The seemingly simple nature of it and adoption makes a good case for deep diving into in a fascinating piece of software.

# Chapter 1: The Story Behind

SQLite was written by Dwayne Richard Hipp. It is not uncommon to see it being abbreviated to D. Richard Hipp or DRH for short. The story of how the database came around is fascinating. It sheds light on the author's mindset and SQLite general coding culture.

DRH holds a computer science doctorate. Since his early days he was very dedicated. He dropped out of academia as the race was full of candidates. He turned to consulting. During that time, he was signed a software contract with shipyard *Bath Iron Works*. His work involved finding the solution to pipe burst failure by controlling valves on a warship: the *DDG-79 Oscar Austin*.

Richard had a problem. The software often did not work as the database server was down all the time. The ship was using *Informix*. So, he thought of spinning his own database.

> one of the guys I was working with says, “Richard, why don’t you just write one?” “Okay, I’ll give it a try.
>
> …
>
> all government contracts got shut down, so I was out of work for a few months, and I thought, “Well, I’ll just write that database engine now.”

Contrary to many popular projects, Richard thought of a bytecode-driven engine since the begining. This shows his previous exposure to compiler crafstmanship.

> so I wrote a byte code engine that would actually run a query and then I wrote a compiler that would translate SQL into that byte code and voila, SQLite was born.

## How SQLite picked up speed

SQLite was not an overnight success though people did realise it’s potential since the early days. This is a list of some milestones which led to SQLite what it is today.

**2000 - The Internet:** Since the shipyard was adamant on *Informix*, SQLite was not used on the warship. Robert put the code out in the wild on the internet. One great moment was a personal initiative from a user running it on his *Palm Pilot*.

**2001 - Motorola OS:** Motorola was a phone manufacturing company. The operating system they were using had SQLite on it. They wanted some help. During the whole time, Richard has been working on the project as an OpenSource one. So, they proposed an $80k contract to Richard for support and enhancements. It was the first time that the author realized that OpenSource can bring in money. He rounded his OSS team and shipped the project. This would be the first in a series of long-lasting relationship with phone companies.

**200x - America Online:** The next serious company to reach out was America Online. They wanted the database on CDs they were mailing to customers. Richard enthusiastically accepted the offer and midway realized the solution he had in mind would not work. These types of challenges helped SQLite grow into a robust product.

**200x - Symbian OS:** Symbian flew Richard to their office in London. Among many databases they evaluated, both OSS and closed-source, SQLite was chosen. Symbian was a great company but they had a problem. They wanted to ensure that the project lives on even if Richard is no longer around. They wanted to increase the bus factor by having a SQLite consorsium.

**200x - SQLite Consortium:** Richard liked the idea of a consorsium. He started devising a plan of his own. Luckily someone from the Mozilla foundation reached out to him. They did not like the way he was setting up the framework around the consorsium by giving members voting rights. They proposed keeping the direction of the project in developers hand. The friend from Mozilla being a lawyer was adamant on this point and saw through the implementation of the current setup.

**200x - Google & Android:** Google was a complete outsider to the phone game. Soon, they approached Richard for a daring project. Having a phone connected to the internet with a robust software lifecycle was something extraordinary. They wanted SQLite to behave perfectly on this innovation. Richard's experience with the phone industry knew that Android was going to be a huge hit.

> We were going around boasting to everybody naively that SQLite didn’t have any bugs in it, or no serious bugs, but Android definitely proved us wrong. … It’s amazing how many bugs will crop up when your software suddenly gets shipped on millions of devices.

**200x - Rockwell Collins:** Rockwell Collins was a multinational corporation providing avionics and information technology systems and services to government agencies and aircraft manufacturers. They wanted the *DO-178B* aviation quality standard for SQLite. It meant 100% MCDC test coverage. This helped shaped SQLite test-backed approach to development.

SQLite tests are better than even postgres which relies on peer reviews [3]. This allows the developers to experiment and change code fearlessly.

## A from scratch principle

SQLite is notorious for implenting a bunch of functionalities from scratch. It’s a daring, amazing, bold and crazy spirit which requires confidence and professionalism. People also call it the *From First Principles* approach. With no internet at the tips of the fingers and no wikipedia to consult, the author deserves massive respect. His teachers must have been proud to have their student be the living embodiment of what computer science and software engineering should be about.

DRH does look for alternatives. He does try out libraries. But, at the end of the day he ends up coding from scratch.

First, he needed a database engine, he looked around, was not satisfied and went on to pull off his own implementation.

The same goes for the b-tree layer. Much like a hero from a movie, he pulled Donald Knuth’s algorithm book from the shelf and coded the b-tree he needed. He also completed the book’s exercise about deleting elements.

He doesn’t understand the use of YACC, Bison and Lex when anybody can code their own parsers.

He was using Git, but some functionalities were scratching his itch to build his own Control Version System. So, as usual, he wrote *Fossil*. It’s the CVS you would download and configure if you download the source as is from the website.

> … And it’s GPL, and so SQLite Version 1 was GPL, it had to be because it was linking against the GPL library. But GDBM is only key-value, I can’t do range queries with it. Then I said, “I’m gonna write my own B-tree layer

The from scratch spirit is much preferred as it enables the developers to have the freedom they want. They can choose what they want or how they implement things. Just wrapping over another library might be a problem waiting to happen.

We can expect the library to be fairly complex as there are several components present which require knoledge of their own.

> never understood lex because it’s so easy to write a bunch of C codes faster then Lex [1]

# Chapter 2: Overview

A rough overview of SQLite is as follows

```
--------------     ------------
| SQLite lib |  ⇐  | SQL code |
--------------     ------------
      ⇑ ⇓
---------------
| Binary file |
---------------
```