International Journal of Computer Science and Mobile Computing SQLite: Light Database System

Article · April 2015

CITATIONS
0

READS
100

3 authors, including:

Satish Tanaji Bhosale
VPIMSR, Sangli
23 PUBLICATIONS **11** CITATIONS

SEE PROFILE

Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 4, April 2015, pg.882 – 885

RESEARCH ARTICLE

SQLite: Light Database System

S.T. Bhosale[1], Miss. Tejaswini Patil[2], Miss. Pooja Patil[3]

[1]Computer Department, Shivaji University, Kolhapur, India
[2]Computer Deptt. Student (MCA), VPIMSR, Sangli
[3]Computer Deptt. Student (MCA), VPIMSR, Sangli
[1] stbhosale@yahoo.co.in

Abstract- SQLite is advance database system useful for storing data. It is simple to use operate and supported by multiplatform. Basically it is more applied in android based application systems. Because of its flexibility and platform independency it is widely used. It contains simple data types, having dynamic querying capability.

Keywords: SQLite, database, multiplatform, android, platform independent.

I. INTRODUCTION

SQLite is an in-process library that implements a self-contained, zero-configuration, serverless, transactional SQL database engine. The source code for SQLite exists in the public domain and is free for both private and commercial purposes. SQLite has bindings to several programming languages such as C, C++, BASIC, C#, Python, Java and Delphi. The COM (ActiveX) wrapper which makes SQLite more accessible to scripted languages on Windows such as VB Script and JavaScript, thus adding capabilities to HTML applications. It is also available in embedded operating systems such as iOS, Android, Symbian OS, Maemo, Blackberry and WebOS because of its small size and ease of use.

SQLite is an in-process library that implements a SQL database engine. The code for SQLite is the public domain and therefore thus free to use for any purpose, commercial or private. SQLite is currently found in more applications, including several high-profile projects.

SQLite is an embedded SQL database engine and it does not have a separate server process like most other SQL databases. SQLite reads and writes directly to ordinary disk files. The database file format is cross-platform. These features make SQLite a popular choice as an Application File Format. SQLite is a compact library, the library size can be less than 500KiB, depending on the target platform and compiler optimization settings. If optional features are omitted, the size of the SQLite library can be reduced below 300KiB. SQLite can also be made to run in minimal stack space (4KiB). SQLite a popular database engine choice on memory constrained gadgets such as cellphones, PDAs, and MP3 players. SQLite generally runs faster. Nevertheless, performance is usually quite good even in low-memory environments. SQLite is very carefully tested and

prior to every release also it is very reliable. Most of the SQLite source code is devoted purely to testing and verification. SQLite responds gracefully to memory allocation failures and disk I/O errors. Of course, even with all this testing, there are still bugs. But SQLite is open and honest about all bugs and provides bugs lists including lists of critical bugs and minute-by-minute chronologies of bug reports and code changes.

II. ARCHITECTURE OF SQLITE

SQLite's architecture is relatively simple, and a brief description is necessary for understanding.

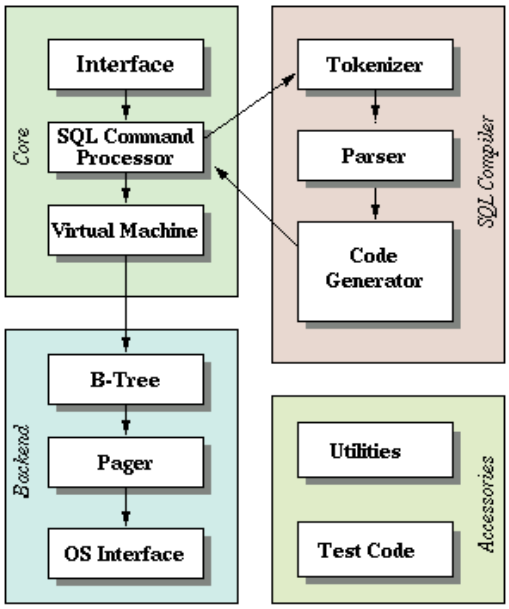

Fig.-1 Architecture of SQLITE
Source: Reference [5]-[7]

The SQLite infrastructure contains four main parts as:

1. Core 2. SQL compiler 3.Backend 4.Accessories

1. Core part contains user interface, the SQL command processor, and the virtual machine.
   The **user interface** consists of a library of C functions and structures to handle operations such as initializing databases, executing queries, and looking at results. Function calls that execute SQL queries use the **SQL command processor**. The command processor functions exactly like a compiler. When executing a program, the **virtual machine** directs control flow through a large switch statement, which jumps to a block of code based on the current opcode.
2. SQL compiler contains a tokenizer, a parser, and a code generator. When a string containing SQL statements is to be executed, the interface passes that string to the **tokenizer**. The job of the tokenizer is to break the original string up into tokens and pass those tokens one by one to the parser. The **parser** is the piece that assigns meaning to tokens based on their context. The parser for SQLite is generated using the Lemon LALR[3] parser generator. After the parser assembles tokens into complete SQL statements, it calls the code generator to produce virtual machine code that will do the work that the SQL statements request.
3. Backend contains B-Tree, Page Cache, OS Interface: An SQLite database is maintained on disk using a B-tree implementation found in the **btree.c** source file. A separate B-tree is used for each table and index in the database. All B-trees are stored in the same disk file. The B-tree module requests information from the disk in fixed-size chunks. The page cache is responsible for reading, writing, and caching these chunks. In order to provide portability between POSIX and Win32 operating systems, SQLite uses an abstraction layer to interface with the operating system.

4. Accessories contains Utilities and Test code: SQLite provides some utility related functionality such as memory allocation and case less string comparison routines are located in **util.c** more than half the total code base of SQLite is devoted to testing.

III. WHEN TO USE SQLITE

- **Embedded applications:** All applications that need portability, that do not require expansion, e.g. single-user local applications, mobile applications or games.
- **Disk access replacement:** In many cases, applications that need to read/write files to disk directly can benefit from switching to SQLite for additional functionality and simplicity that comes from using the Structured Query Language (SQL).
- **Testing:** It is an overkill for a large portion of applications to use an additional process for testing the business-logic (i.e. the application's main purpose: functionality).
- **Multi-user applications:** If you are working on an application whereby multiple clients need to access and use the same database, a fully-featured RDBM (e.g. MySQL) is probably better to choose over SQLite.

IV. FEATURES OF SQLITE

- SQLIT is atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- No setup or administration needed.
- A complete database is stored. Great for use as an application file format
- Supports terabyte-sized databases and gigabyte-sized strings and blobs.
- It require to writ very Small code less than 500KiB fully configured or much less with optional features omitted.
- Simple, easy to use Application Program Interface (API)
- Having a great feature of binding other languages.
- Available as a ANSI-C code and easy to add into a larger project.
- SQLite is self contained and therefore no external dependencies.
- Supports for cross-platform as Android, iOS, Linux, Mac, Solaris, and Windows (Win32, WinCE, WinRT)

V. ADVANTAGES OF SQLITE COMPARED WITH SQL & MYSQL

- SQL Server and MySQL are server-based but the SQLite is file-based,
- SQLlite is embeddable relational database management system whereas SQL is query language, MySQL is client-server relational database management system.
- Because of it light and support to relational database management system it is used mobile applications.
- SQL is based on relational algebra. SQLite, MySQL, MSSQL and etc are part of SQL. They use all SQL syntax but each database (sqlite, MySQL, MSSQL) have their own query convention.
- As SQLite is light database it is direct file system engine that uses SQL syntax. SQLite also doesn't require a special database server or anything.
- It is important to note that SQL Server and MySQL support stored procedures but SQLite does not.

VI. CONCLUSION

With the popularity of intelligent appliances, the formation of mobile computing environment, and the rise of mobile commerce, embedded database has become the focus of study currently, SQLite has small core, open source, and database is a file, it is very easy to realize the copy, move and cross platform sharing of database files, and can adapt to the needs of embedded system, it is very convenient to construct an embedded database system. At present, SQLite has become the stream database of embedded system rapidly with its unique advantages. In this paper, concept of SQLite, datatypes, characteristics, advantages, limitations, difference between SQLite, SQL, MySQL and applications of SQLite with its features, the architecture has been discussed.

SQLite will be more widely used in the embedded field, such as the remote control, intelligent mobile terminal, information appliances control, home medical equipment, mobile devices etc. In a multiplatform architecture SQLite having wide scope.

References:

[1] Research and Application of SQLite Embedded Database Technology, CHUNYUE BI, School of Computer Science and Information Technology, Zhejiang Wanli University, South Qian Hu Road Ningbo, Zhejiang P.R.CHINA, WSEAS TRANSACTIONS on COMPUTERS, ISSN: 1109-2750 , Issue 1, Volume 8, January 2009
[2] Michael Owens, The Definitive Guide to SQLite, USA: Apress, 2006, 341-362.
[3] SQLite homepage [EB/OL] , http://www.sqlite.org.
[4] Wang Jinqin, Wan Lixin, The comparison of Embedded Database Berkeley DB and SQLite,The application of SCM and Embedded System, 2005, 28(2). 5-7.
[5] http://en.wikipedia.org/wiki/SQLite
[6] www.tutorialspoint.com/sqlite
[7] www.sqlite.org/about.html
[8] www.techopedia.com/definition/24610/sqlite
[9] www.vogella.com/tutorials/AndroidSQLite/article.html
[10] http://stackoverflow.com/questions/711048/what-is-sqlite-used-for

