SQLite 的 SQL 语句高速缓存技术①

戴 昱，黄德才

(浙江工业大学 计算机科学与技术学院，杭州 310023)

摘 要：随着嵌入式应用领域的发展，系统功能的日趋复杂，嵌入式设备中使用嵌入式数据库越来越成为一种趋势。SQLite 因其性能和功能上的优势被广泛的使用于嵌入式应用中。但是嵌入式设备 CPU 处理能力相对较低，存储器容量有限，制约了 SQLite 的性能。针对上述问题，根据高速缓存原理，简化 SQL 语句执行过程中的词法分析、语法分析过程，以减少运行过程中的时间消耗。实验表明，本方法可有效提高 SQL 语句执行效率，在保持可用性与可靠性的前提下，提升 SQLite 的整体性能。

关键词：嵌入式数据库；SQLite；高速缓存；性能优化

SQLite's SQL Statement Cache Technology

DAI Yu, HUANG De-Cai

(Institute of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: With the development of embedded applications, and the growing complexity of system functions, use of embedded database is increasingly becoming a trend in embedded devices applications. SQLite for its performance and functionality advantages will be widely used in embedded applications. But embedded devices is relatively low CPU processing power, memory capacity is limited, which restricts the performance of SQLite. For these problems, this paper according to the high-speed cache principle to simplify the process of SQL statement execution lexical analysis and syntax analysis, to reduce the time consumption during operation. Experiments show that this method can effectively improve the efficiency of SQL statements implementation, maintaining the availability and reliability of system, to enhance the overall performance of SQLite.

Key words: embedded database; SQLite; cache; performance optimization

1 概述

嵌入式数据库因使用简单，通过事务来进行调度与并发控制，具有安全性、完整性检验等多种安全措施，跨平台间移植方便等优势被广泛的应用在嵌入式应用开发的数据存储和管理中。其中 SQLite 又因其在大小和功能之间找到了一个理想的平衡点，受到开发者的青睐。

随着嵌入式领域的发展，用户对嵌入式的数据库的性能要求越来越高，比如工业上的一些实时监控系统数，数据采集系统，远程控制系统等都要求嵌入式系统具有更高的数据存储效率和数据管理能力。这对 SQLite 的性能也提出了新的要求。

在一般数据库中，评价性能的好坏主要参考 SQL 操作执行的时间消耗，SQL 的执行时间消耗又由如下几个因数决定[1]：SQL 执行时间消耗=网络传输消耗+CPU 执行消耗+存储器读写消耗

网络传输消耗主要指 SQL 操作从客户端经过网络传输到服务端的所消耗的时间，它一般取决于外部网络条件，与 DBMS 自身无关。存储器读写消耗是指数据经过处理后存储到外存，以及从外存中读取数据过程所消耗的时间。减少 SQLite 存储器读写消耗的方法一般把大批量操作合成为一个事务进行处理以便大大减少 I/O 的次数[2,3]，提高总体读写性能。由于磁盘的读写速度往往比 CPU 的速度慢很多，所以存储器读写

① 收稿时间:2011-05-05;收到修改稿时间:2011-05-31

消耗一般比 SQL 执行消耗大很多。CPU 执行消耗是指获得 SQL 操作语句后，DBMS 对其进行分析处理的过程，许多大型数据库通过 SQL 语句高速缓存的方法实现，而 SQLite 在设计中没有这个过程。如何修改SQLite代码使它具有SQL语句高速缓存功能是本文的研究重点。

2　原理与设计

2.1 SQLite 的工作原理

传统的数据库大多由软件驱动来实现数据库的调用，在使用数据时，需要有一个数据库服务器在运行，并且在系统上配置数据源或调用 ODBC 等驱动程序，应用程序才能对数据进行读写操作。与传统的数据库不同，嵌入式数据库一般采用程序驱动，所谓程序驱动就是把嵌入式数据库程序直接整合到应用程序中，应用程序只要调用相应的 API 就能实现数据的存取。SQLite 也是如此（过程如图 1 所示），程序员可以把 SQLite 代码整合到应用软件的工程目录下一同编译，然后调用它的 API 进行数据存取操作，也可以把 SQLite 编译成动态连接库，应用程序通过调用这个动态连接库来使用 SQLite 的 API。

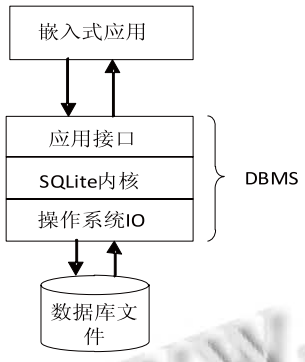

图 1　SQLite 工作过程

SQLite 提供给应用程序调用的 API 主要有：

int sqlite3_open(const char* filename，//参数 1：数据库名

sqlite3**ppDb//参数 2：数据句柄) ;

这个函数是用来打开数据库的，它的功能是打开名字为 filename（参数 1）的数据库文件，并返回数据库句柄，保存到 ppDb（参数 2）中。

int sqlite3_exec(sqlite3*,//参数 1：要操作的数据库的句柄

const char *sql, //参数 2：执行的 SQL 语句

sqlite_callback, //参数 3：回调函数，用来操作返回结果集

void *,//参数 4：回调函数的第一个参数

char**errmsg//参数 5：返回错误信息) ;

这个函数通过执行 SQL 语句来对数据库进行操作，要执行的语句保存在 sql（参数 2）中，sqlite_callback（参数 3）则用来取得返回集信息，如果 SQL 操作完后没有返回集，那么，需要把 sqlite_callback（参数 3）和 void *（参数 4）设置为 NULL。

int sqlite3_close(sqlite3*); //参数 1：打开的数据库句柄

这个函数用来关闭打开了的数据库。

table1:

ID	CLASS	NAME	GRADE
1	3	DY	50
2	2	LL	80

图 2　一张例表

而这 3 个之中，sqlite3_exec 是最重要的接口函数，它负责了绝大部分的数据库操作。比如有这样一张数据库表（如图 2）。需要把 ID 为 1 的成绩（GRAGE）改为 60，那么，就应该向接口函数 sqlite3_exec 的参数 sql（参数 2）传入字符串：

“update table1 set GRADE=60 where ID=1”;

之后，SQLite 内部进行了一系列操作，如图 3 所示[4]，先对字符串进行词法分析，语法分析，然后根据分析的结果产生虚拟机代码，并用虚拟机（SQLite 的核心）执行，虚拟机根据命令操作 B-tree（SQLite 存储数据在内存中的数据结构），最后调用操作系统 IO 接口把内存中修改的数据以文件的形式存储于外存，并返回结果集。

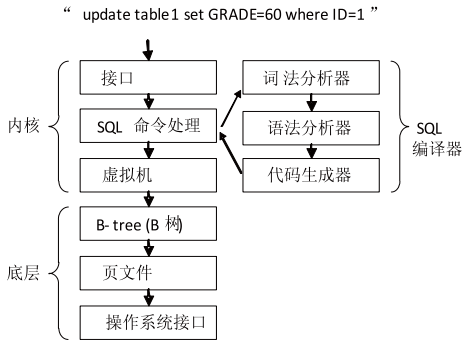

图 3　SQLite 中一个 SQL 语句处理过程

2.2 改进设计

SQLite 支持了大部分的 SQL92 语法。由于 SQLite 采用程序驱动，由应用程序直接调用 API 来进行数据存储，所以它不像传统数据库那样有用户和权限，所以除去操作权限的 SQL 语法外，根据 SQL 语句在一个应用中的使用频率，可以很容易把 SQL 分成 2 类：

A 类，数据定义语言(DDL)，包括 CREATE，ALTER，DROP。在一个数据库应用系统中用得较少，一般只在设计数据库时候用到的。表、表约束、字段、触发器、试图、索引等这些信息在完成数据库设计后，在整个应用系统需求不改变的情况下，不需要多次修改。

B 类，数据操作语言(DML)、 数据查询语言(DQL)、数据控制语言(DCL)，在应用系统的执行中经常要用的到有 SELECT，UPDATE，DELETE，INSERT 语句等。

由于嵌入式数据库应用系统中的数据库结构往往不是很复杂，表比较单一，需要处理的数据种类不是很多，所以常常会发现这些经常执行的语句都比较相似。比如，在一个实时温度监控的嵌入式应用系统中，它隔一个时间段就采集一次温度数据并把它保存到数据库中，它所执行的 SQL 语句都与下面语句相似：

“insert into record (cid, time, value)

values (参数 1, 参数 2 , 参数 3)”

从上节的图 3 得知，每执行一次 SQL 语句，SQLite 都对语句进行一次语法分析、词法分析，这个实时温度监控系统也不会例外。不难发现，SQLite 将重复的分析“insert into record (cid,time,value) values”部分，如果我们将这些雷同的部分预编译成虚拟机代码并保存到内存中的缓冲区，每次执行时直接调用这部分虚拟机代码，那么就减少了花费在 SQL 命令分析处理的执行时间，这将明显的提高 SQLite 的执行效率。这就是本文讨论的 SQLite 中 SQL 语句高速缓存功能实现的原理。

3 实现

通过对 SQLite 进行单步调试后发现，当它取得 SQL 命令字符串后，首先对语句进行字符串匹配，每当确定输入一个有效的词后，它就把这个词包装成一个结构，如果某两个词逻辑上有所联系，那么它就用某个特定的函数将两个由词包装好的小结构包装成一个更大结构，大结构再合并，最后，整个 SQL 语句被包装成一个结构体。虚拟机代码生成函数以这个结构体为参数生成输入 SQL 对应的虚拟机代码。然后虚拟机执行虚拟机代码，完成操作。

以“update table1 set GRADE=60 where ID=1”为例，它生成虚拟机器代码的具体过程如图 4 所示，它展示了 SQL 语句通过大量的字符匹配，上下文判断找出句中的关键字，并用函数将它层层包装，将所要操作的表名填充到 SrcList*结构中，将需要修改的内容填充到 ExprList*中，将 WHERE 条件信息填充到 Expr*中，最后汇总编码，形成虚拟机代码。

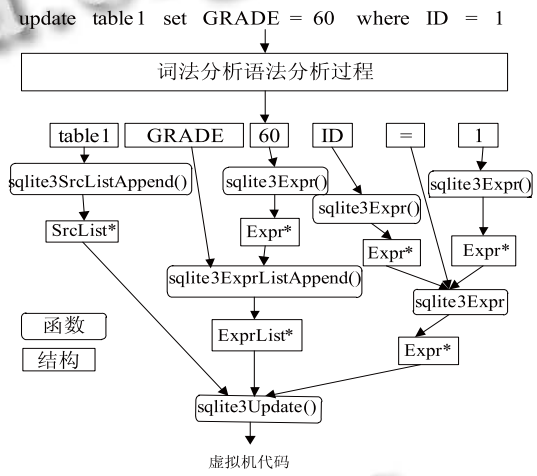

图 4 UPDATE 编译成虚拟机器代码过程

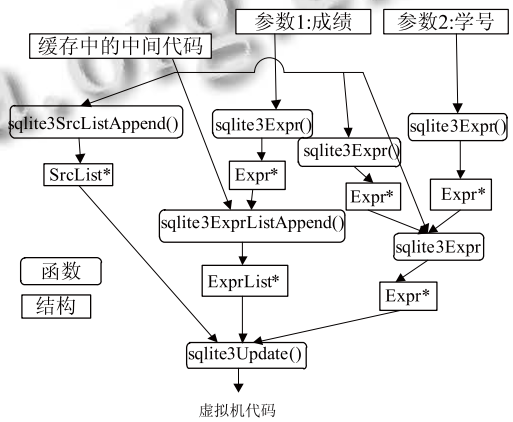

图 5 改进后 UPDATE 编译成虚拟机器代码过程

这样一种跟据学号修改成绩的 SQL 操作在一个应用系统中可能多次用到，但是并不总是把 1 号的成绩改成 60 分。不过可以把它们总结成：

“update table1 set GRADE = 参数 1

where ID = 参数 2 ”

我们可以先把那些重复部分进行词法语法分析并预编译生产一个中间虚拟机代码，并以缓存的形式在内存中保存，当执行某个具体操作时，接收变量绕过词法分析语法分析过程生成完整的虚拟机代码。修改后的 UPDATE 如图 5 所示。从图中可以看出，我们把 UPDATE 操作直接编写成为一个接口函数，在执行时，它首先从缓冲区中调用生成好的中间虚拟机代码，结合用户传递来的参数，生成完整的虚拟机代码并执行 SQL 操作。

SQL 语句重复部分预编译生成好的中间虚拟机代码以文件的形式在磁盘中保存，当 SQLite 读取到有类似操作时，从磁盘中加载中间虚拟机代码文件，并组织成一定的数据结构保存在内存缓存区的树状结构中，并结合变量形成完整的虚拟机代码比便执行。当再次需要同类中间虚拟机代码时，不必反复加载，在缓冲区中查找即可命中，就可以从缓冲区内直接调用，以提升总体性能。

4 测试结果

通过对多表多字段查询(案例 A)；查询并使用 avg（案例 B）、count 等计算函数（案例 C）；两表连接查询（案例 D）；嵌套查询（案例 E）；查询并排序（案例 F）这 6 种不同类型的查询多次操作取均值进行对比，结果如图 6 所示，不难发现使用 SQL 缓冲的系统明显比旧系统的性能提高了 10%以上，说明通过 SQL 语句高速缓冲的实现对 SQLite 的性能有显著效果。

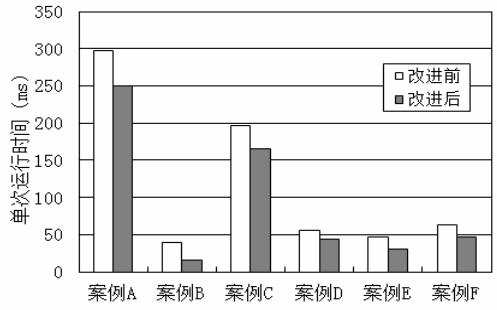

图 6　改进前后性能对比

5 总结与展望

本文通过对 SQLite 添加高速缓冲处理，减少 SQL 语句执行时的 CPU 消耗来提升 SQLite 的性能，取得了明显效果。为了进一步的提升 SQLite 的性能，在未来的研究中可以把 SQLite 高速缓冲技术与 SQLite 的事务处理过程结合，同时减少 CPU 消耗和 I/O 读写消耗，从而达到性能上更大的提升，完善整个系统，使之简单、易用、高效，能在工业领域有所应用。

参考文献

1 黄德才.数据库原理及其应用.第 3 版.北京:科学出版社,2010.

2 SQLite 官方网站. Database Speed Comparison. http:// www.sqlite.org/speed.html.

3 刘婷婷,黄林春.一种降低 SQLite 擦写 Flash 频率的方法.计算机工程,2010,36(15).

4 Owens M. The Definitive Guide to SQLite. APRESS, 2006.

(上接第 197 页)

2 邓铁六,白泰礼,马俊亭,等.单线圈电流型振弦式传感器.传感器技术,2000,19(4):22−25.

3 黄贤武,郑筱霞.传感器原理与应用.成都:电子科技大学出版社,1999.233−236.

4 高友.振弦式传感器测量过程中干扰问题的解决.仪器仪表与传感器,2007,(2):55.

5 白泽生.一种基于振弦式传感器测频方法实现.传感器与微系统,2007,26(8):81−83.

6 江修,经亚枝,詹焕春.基于扫频激励技术的振弦式传感器.传感器技术,2001,20(5):22−24.

7 白秦礼,邓铁六,谢军,等.振孩式传感器的精确数学模型及其应用岩石力学与工程学报,2005,24(2):693−697.

8 白秦礼,何羚,王彩云,等.基于振弦式传感器的多功能智能检测仪.传感器技术,2004,23(3):60−62.