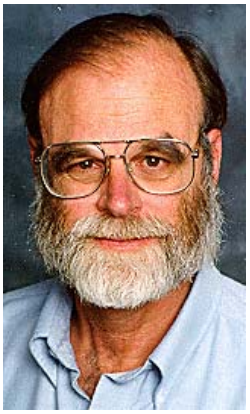


个人简介
专业打杂程序员
联系方式
新浪微博 腾讯微博

IT新闻:
苹果新Retina MacBook Pro（2014年中）开箱图+SSD简单测试 7分钟前
网吧里玩出的世界冠军 打场游戏赚了400万 10分钟前
Twitter收购深度学习创业公司Madbits 35分钟前

昵称：YY哥
园龄：7年2个月
粉丝：342
关注：2
+加关注

< 2009年2月 >

日	一	二	三	四	五	六
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
1	2	3	4	5	6	7

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

随笔分类
c/c++(9)
Linux相关(24)
MySQL(11)
Others(2)
Web技术(12)
数据结构与算法(15)
数据库技术(30)
系统相关(3)
云计算与虚拟化(3)

随笔档案
2014年7月 (4)

SQLite入门与分析(二)---设计与概念

写在前面:谢谢各位的关注,没想到会有这么多人关注。高兴的同时，也感到压力，因为我接触SQLite也就几天，也没在实际开发中用过，只是最近项目的需求才来研究它，所以我很担心自己的文章是否会有错误，误导别人。但是我很想把自己的学习成果与大家分享，所以如果大家觉得我有不对的地方，望不吝赐教。
我原打算直接从VDBE入手的，因为它起着承上启下的作用，是整个SQLite的核心，并分析源码，但考虑到这是一个系列的文章，我希望能把问题说全，所以还是从基本概念入手，对于初学者，如果没有这些概念，是很继续下去的。好了，下面开始第二章，由于这一章内容很多，我将分两部分讨论，下面开始第一部分。

1、 API

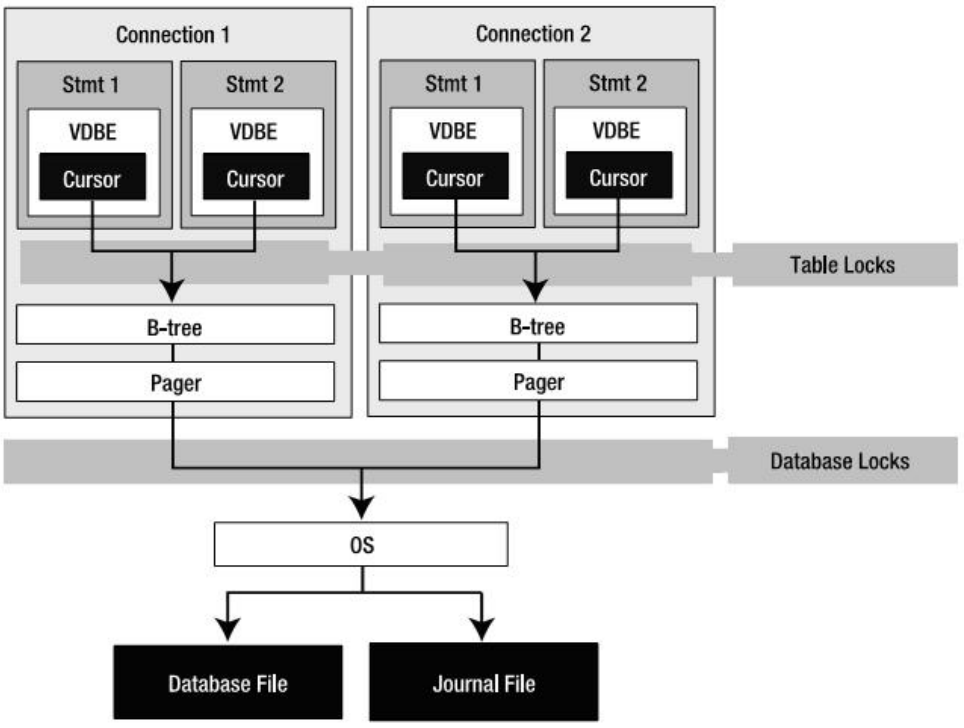
由两部分组成: 核心API(core API) 和扩展API（extension API）
核心API的函数实现基本的数据库操作：连接数据库，处理SQL，遍历结果集。它也包括一些实用函数，比如字符串转换，操作控制，调试和错误处理。
扩展API通过创建你自定义的SQL函数去扩展SQLite。

1.1、SQLite Version 3的一些新特点：

(1)SQLite的API全部重新设计，由第二版的15个函数增加到88个函数。这些函数包括支持UTF-8和UTF-16编码的功能函数。
(2)改进并发性能。加锁子系统引进一种锁升级模型(lock escalation model)，解决了第二版的写进程饿死的问题(该问题是任何一个DBMS必须面对的问题)。这种模型保证写进程按照先来先服务的算法得到排斥锁(Exclusive Lock)。甚至，写进程通过把结果写入临时缓冲区(Temporary Buffer)，可以在得到排斥锁之前就能开始工作。这对于写要求较高的应用，性能可提高400%（引自参考文献）。
(3)改进的B-树。对于表采用B+树，大大提高查询效率。
(4)SQLite 3最重要的改变是它的存储模型。由第二版只支持文本模型，扩展到支持5种本地数据类型。
总之，SQLite Version 3与SQLite Vertion 2有很大的不同，在灵活性，特点和性能方面有很大的改进。

1.2、主要的数据结构(The Principal Data Structures)

SQLite由很多部分组成－parser,tokenize,virtual machine等等。但是从程序员的角度，最需要知道的是:connection, statements, B-tree和pager。它们之间的关系如下：



上图告诉我们在编程需要知道的三个主要方面：API,事务(Transaction)和锁(Locks)。从技术上来说，B-tree和pager不是API的一部分。但是它们却在事务和锁上起着关键作用（稍后将讨论）。

1.3、Connections和Statements

Connection和statement是执行SQL命令涉及的两个主要数据结构，几乎所有通过API进行的操作都要用到它们。一个连接(Connection)代表在一个独立的事务环境下的一个连接A (connection represents a single connection to a database as well as a single transaction context)。每一个statement都和一个connection关联，它通常表示一个编译过的SQL语句，在内部，它以VDBE字节码表示。Statement包括执行一个命令所需要一切，包括保存VDBE程序执行状态所需的资源，指向硬盘记录的B-树游标，以及参数等等。

1.4、B-tree和pager

一个connection可以有多个database对象---一个主要的数据库以及附加的数据库，每一个数据库对象有一个B-tree对象，一个B-tree有一个pager对象(这里的对象不是面向对象的"对象"，只是为了说清楚问题)。

Statement最终都是通过connection的B-tree和pager从数据库读或者写数据，通过B-tree的游标(cursor)遍历存储在页面(page)中的记录。游标在访问页面之前要把数所从disk加载到内存，而这就是pager的任务。任何时候，如果B-tree需要页面，它都会请求pager从disk读取数据，然后把页面(page)加载到页面缓冲区(page cache)，之后，B-tree和与之关联的游标就可以访问位于page中的记录了。
如果cursor改变了page，为了防止事务回滚，pager必须采取特殊的方式保存原来的page。总的来说，pager负责读写数据库，管理内存缓存和页面（page），以及管理事务，锁和崩溃恢复(这些在事务一节会详细介绍)。
总之，关于connection和transaction，你必须知道两件事：
(1) 对数据库的任何操作，一个连接存在于一个事务下。
(2) 一个连接决不会同时存在多个事务下。
whenever a connection does anything with a database, it always operates under exactly one transaction, no more, no less.

1.5、核心API

核心API 主要与执行SQL命令有关，本质上有两种方法执行SQL语句：prepared query 和wrapped query。Prepared query由三个阶段构成：preparation，execution和finalization。其实wrapped query只是对prepared query的三个过程包装而已，最终也会转化为prepared query的执行。

1.5.1、连接的生命周期(The Connection Lifecycle)

和大多数据库连接相同，由三个过程构成：
（1）连接数据库(Connect to the database)：
每一个SQLite数据库都存储在单独的操作系统文件中，连接，打开数据库的C API为：sqlite3_open()，它的实现位于main.c文件中，如下：

```
int sqlite3_open(const char *zFilename, sqlite3 **ppDb)
{
 return openDatabase(zFilename, ppDb, SQLITE_OPEN_READWRITE | SQLITE_OPEN_CREATE, 0);
}
```

当连接一个在磁盘上的数据库，如果数据库文件存在，SQLite打开一个文件；如果不存在，SQLite会假定你想创建一个新的数据库。在这种情况下，SQLite不会立即在磁盘上创建一个文件，只有当你向数据库写入数据时才会创建文件，比如：创建表、视图或者其它数据库对象。如果你打开一个数据，不做任何事，然后关闭它，SQLite会创建一个文件，只是一个空文件而已。
另外一个不立即创建一个新文件的原因是，一些数据库的参数，比如：编码，页面大小等，只在在数据库创建前设置。默认情况下，页面大小为1024字节，但是你可以选择512-32768字节之间为 2幂数的数字。有些时候，较大的页面能更有效的处理大量的数据。
（2）执行事务(Perform transactions)：
all commands are executed within transactions。默认情况下，事务自动提交，也就是每一个SQL语句都在一个独立的事务下运行。当然也可以通过使用BEGIN..COMMIT手动提交事务。
（3）断开连接(Disconnect from the database)：
主要是关闭数据库的文件。

1.5.2、执行Prepared Query

前面提到，预处理查询(Prepared Query)是SQLite执行所有SQL命令的方式，包括以下三个过程：
(1) Prepared Query：
分析器（parser），分词器(tokenizer)和代码生成器(code generator)把SQL Statement编译成VDBE字节码，编译器会创建一个statement句柄(sqlite3_stmt)，它包括字节码以及其它执行命令和遍历结果集的所有资源。
相应的C API为sqlite3_prepare()，位于prepare.c文件中，如下：

```
int sqlite3_prepare(
  sqlite3 *db,            /* Database handle. */
  const char *zSql,       /* UTF-8 encoded SQL statement. */
  int nBytes,             /* Length of zSql in bytes. */
  sqlite3_stmt **ppStmt,  /* OUT: A pointer to the prepared statement */
  const char **pzTail     /* OUT: End of parsed string */
){
  int rc;
  rc = sqlite3LockAndPrepare(db,zSql,nBytes,0,ppStmt,pzTail);
  assert( rc==SQLITE_OK || ppStmt==0 || *ppStmt==0 );  /* VERIFY: F13021 */
  return rc;
}
```

(2) Execution：
虚拟机执行字节码，执行过程是一个步进(stepwise)的过程，每一步(step)由sqlite3_step()启动，并由VDBE执行一段字节码。由sqlite3_prepare编译字节代码，并由sqlite3_step()启动虚拟机执行。在遍历结果集的过程中，它返回SQLITE_ROW，当到达结果末尾时，返回SQLITE_DONE。
(3) Finalization：
VDBE关闭statement，释放资源。相应的C API为sqlite3_finalize()。

通过下图可以更容易理解该过程：

入INSERT I.....
--麒麟飞
2. Re:理解MySQL——架构与概念
例1-5
insert into t(i) values(1);
这句话应该是可以插入的.
不会被阻塞
--麒麟飞
3. Re:理解MySQL——架构与概念
注：SELECT ... FOR UPDATE仅在自动提交关闭(即手动提交)时才会对元组加锁，而在自动提交时，符合条件的元组不会被加锁。

这个是错误的.自动提交的,也会尝试获取排它锁.
你可以试验下.
--麒麟飞
4. Re:浅谈mysql的两阶段提交协议
YY哥 偶像啊!细腻文笔 配有说服力的代码和图 我崇拜你 !!!
之前sqlite的深入分析帮了我大忙..现在做mysql相关 有来你的博客找东西 哈哈哈!!
--hark.perfe
5. Re:(i++)+(i++)与(++i)+(++i)
@arrowcat
这类语句本身没什么意义，但是楼主思考的角度让我豁然开朗。
--HJWAJ

阅读排行榜

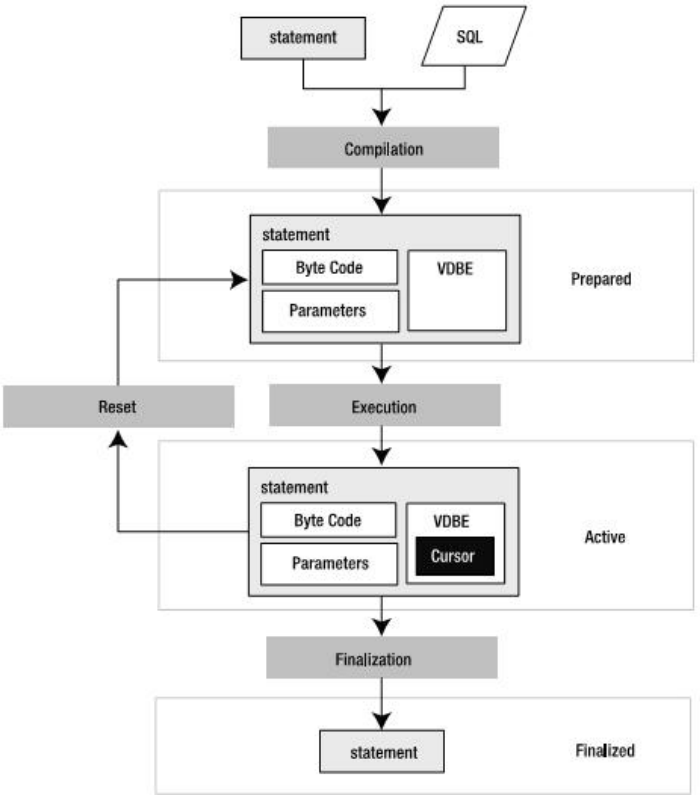
1. 理解MySQL——索引与优化(77627)
2. SQLite入门与分析(一)---简介(48610)
3. 理解MySQL——复制(Replication)(26209)
4. libevent源码分析(19048)
5. SQLite入门与分析(二)---设计与概念(16977)

评论排行榜

1. (i++)+(i++)与(++i)+(++i)(40)
2. SQLite入门与分析(一)---简介(30)
3. 浅谈SQLite——实现与应用(20)
4. 一道算法题,求更好的解法(18)
5. 理解MySQL——索引与优化(16)

推荐排行榜

1. SQLite入门与分析(一)---简介(12)
2. 理解MySQL——索引与优化(12)
3. 浅谈SQLite——查询处理及优化(10)
4. 乱谈服务器编程(9)
5. libevent源码分析(6)



最后以一个具体的例子结束本节，下节讨论事务。

```c
#include <stdio.h>
#include <stdlib.h>
#include "sqlite3.h"

#include <string.h>

int main(int argc, char **argv)
{
    int rc, i, ncols;
    sqlite3 *db;
    sqlite3_stmt *stmt;
    char *sql;
    const char *tail;
    //打开数据
    rc = sqlite3_open("foods.db", &db);

    if(rc) {
        fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
        sqlite3_close(db);
        exit(1);
    }

    sql = "select * from episodes";
    //预处理
    rc = sqlite3_prepare(db, sql, (int)strlen(sql), &stmt, &tail);

    if(rc != SQLITE_OK) {
        fprintf(stderr, "SQL error: %s\n", sqlite3_errmsg(db));
    }

    rc = sqlite3_step(stmt);
    ncols = sqlite3_column_count(stmt);

    while(rc == SQLITE_ROW) {

        for(i=0; i < ncols; i++) {
            fprintf(stderr, "'%s' ", sqlite3_column_text(stmt, i));
        }

        fprintf(stderr, "\n");

        rc = sqlite3_step(stmt);
    }
```

```
    //释放statement
    sqlite3_finalize(stmt);
    //关闭数据库
    sqlite3_close(db);

    return 0;
}
```

/Files/hustcat/2008/Learn.rar

主要参考:The Definitive Guide to SQLite

分类: 数据库技术

YY哥
关注 - 2
粉丝 - 342
+加关注

« 上一篇：SQLite入门与分析(一)---简介
» 下一篇：SQLite入门与分析(二)---设计与概念(续)

posted @ 2009-02-13 22:43 YY哥 阅读(16977) 评论(12) 编辑 收藏

评论列表

#1楼 2009-02-13 22:54 醉春风

以前注意了一下sqllite，但因为忙其他的事就耽误继续学习sqllite。
博主写的不错。

支持(0) 反对(0)

#2楼 2009-02-13 22:59 Angel Lucifer

Woo...支持。
跟着楼主学习。

支持(0) 反对(0)

#3楼[楼主] 2009-02-13 23:00 YY哥

@醉春风
谢谢:)

支持(0) 反对(0)

#4楼 2009-02-13 23:02 魔都浪子~

很好很强大，
请问LZ，这些知识，你是哪里学来的啊
或者是哪里参考来的？

支持(0) 反对(0)

#5楼[楼主] 2009-02-13 23:04 YY哥

@WCF技术联盟
参考文献和www.sqlite.org

支持(0) 反对(0)

#6楼 2009-02-13 23:10 魔都浪子~

@arrowcat
谢谢，能对数据库的内部执行机制作个了解的话，
我想对提升自己数据库方面的功力是大有裨益的！

支持(0) 反对(0)

#7楼[楼主] 2009-02-13 23:20 YY哥

@WCF技术联盟
不客气。
我研究的目的也是想从内部理解DBMS,并希望能与大家一起分享。

支持(0) 反对(0)

#8楼 2009-02-14 09:51 Otis's Technology Space

楼主写得真好！ 学习了．并祝楼主情人节快乐！ ^_^

支持(0) 反对(0)

#9楼 2009-02-14 10:45 式工[未注册用户]

楼主写得真好！ 学习了．并祝楼主情人节快乐！ ^_^

#10楼 2009-02-14 10:46 式工[未注册用户]

楼主写得真好！ 学习了．并祝楼主情人节快乐！ ^_^
到此一游

#11楼 2009-08-17 09:27 ask0000000[未注册用户]

我要做个时间比较长的
sqlite3_prepare
while () {
sqlite3_step
}

这期间写数据库是否会失败或者 block？

#12楼[楼主] 2009-08-23 11:32 YY哥

@ask0000000
参见sqlite锁的部分，也许对你有帮助。

支持(0) 反对(0)