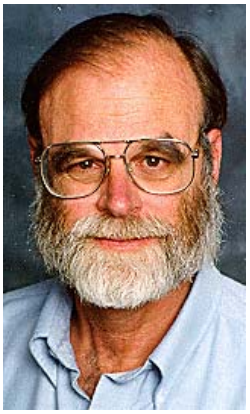


个人简介
专业打杂程序员
联系方式
新浪微博 腾讯微博

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 9分钟前
Twitter收购深度学习创业公司Madbits 34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(三)---内核概述(2)

写在前面:本节是前一节内容的后续部分，这两节都是从全局的角度SQLite内核各个模块的设计和功能。只有从全局上把握SQLite，才会更容易的理解SQLite的实现。SQLite采用了层次化，模块化的设计，而这些使得它的可扩展性和可移植性非常强。而且SQLite的架构与通用DBMS的结构差别不是很大，所以它对于理解通用DBMS具有重要意义。好了，下面我们开始讨论SQLite剩余的两部分：Back-end(后端)和compiler(编译器)。

2、B-tree和Pager

B-Tree使得VDBE可以在O(logN)下查询，插入和删除数据，以及O(1)下双向遍历结果集。B-Tree不会直接读写磁盘，它仅仅维护着页面(pages)之间的关系。当B-TREE需要页面或者修改页面时，它就会调用Pager。当修改页面时，pager保证原始页面首先写入日志文件，当它完成写操作时，pager根据事务状态决定如何做。B-tree不直接读写文件，而是通过page cache这个缓冲模块读写文件对于性能是有重要意义的（注：这和操作系统读写文件类似，在Linux中,操作系统的上层模块并不直接调用设备驱动读写设备，而是通过一个高速缓冲模块调用设备驱动读写文件,并将结果存到高速缓冲区）。

2.1、数据库文件格式（Database File Format）

数据库中所有的页面都按从1开始顺序标记。一个数据库由许多B-tree构成——每一个表和索引都有一个B-tree（注：索引采用B-tree，而表采用B+tree，这主要是表和索引的需求不同以及B-tree和B+tree的结构不同决定的：B+tree的所有叶子节点包含了全部关键字信息，而且可以有两种顺序查找——具体参见《数据结构》，严蔚敏。而B-tree更适合用来作索引）。所有表和索引的根页面都存储在sqlite_master表中。
数据库中第一个页面（page 1）有点特殊，page 1的前100个字节包含一个描述数据库文件的特殊的文件头。它包括库的版本，模式的版本，页面大小，编码等所有创建数据库时设置的参数。这个特殊的文件头的内容在btree.c中定义，page 1也是sqlite_master表的根页面。

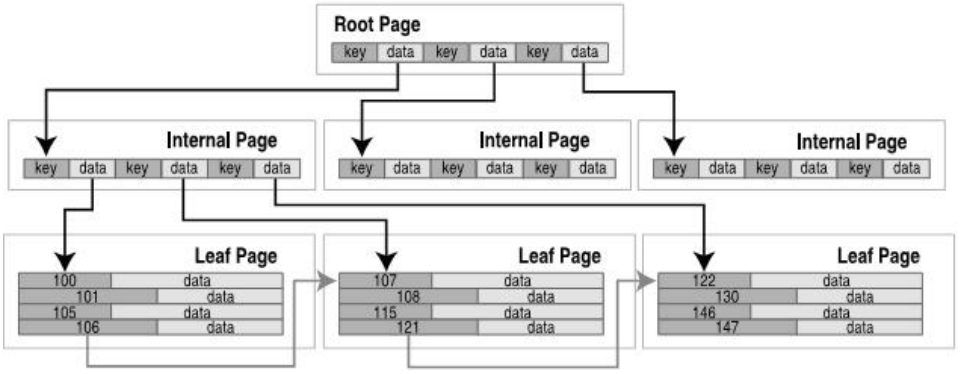
2.1、页面重用及回收(Page Reuse and Vacuum)

SQLite利用一个空闲列表(free list)进行页面回收。当一个页面的所有记录都被删除时，就被插入到该列表。当运行VACUUM命令时，会清除free list，所以数据库会缩小，本质上它是在新的文件重新建立数据库，而所有使用的页在都被拷贝过去，而free list却不会，结果就是一个新的，变小的数据库。当数据库的autovacuum开启时，SQLite不会使用free list，而且在每一次commit时自动压缩数据库。

2.2、B-Tree记录

B-tree中页面由B-tree记录组成，也叫做payloads。每一个B-tree记录，或者payload有两个域：关键字域(key field)和数据域(data field)。Key field就是ROWID的值，或者数据库中表的关键字的值。从B-tree的角度，data field可以是任何无结构的数据。数据库的记录就保存在这些data fields中。B-tree的任务就是排序和遍历，它最需要就是关键字。Payloads的大小是不定的，这与内部的关键字和数据域有关，当一个payload太大不能存在一个页面内进便保存到多个页面。

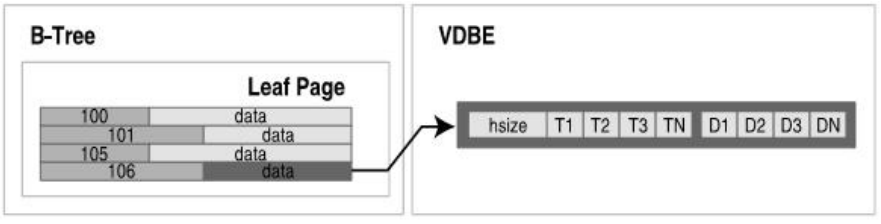
B+Tree按关键字排序，所有的关键字必须唯一。表采用B+tree，内部页面不包含数据，如下：



B+tree中根页面(root page)和内部页面(internal pages)都是用来导航的，这些页面的数据域都是指向下级页面的指针，仅仅包含关键字。所有的数据库记录都存储在叶子页面(leaf pages)内。在叶节点一级，记录和页面都是按照关键字的顺序的，所以B-tree可以水平方向遍历，时间复杂度为O(1)。

2.3、记录和域（Records and Fields）

位于叶节点页面的数据域的记录由VDBE管理，数据库记录以二进制的形式存储，但有一定的数据格式。记录格式包括一个逻辑头（logical header）和一个数据区(data segment)，header segment包括header的大小和一个数据类型数组，数据类型用来在data segment的数据的类型，如下：



2.4、层次数据组织(Hierarchical Data Organization)

2014年3月 (1)
2013年9月 (1)
2013年8月 (1)
2013年2月 (1)
2012年11月 (4)
2012年1月 (1)
2011年12月 (1)
2011年10月 (1)
2011年3月 (1)
2010年9月 (1)
2010年8月 (1)
2010年7月 (3)
2010年6月 (2)
2010年5月 (7)
2010年4月 (1)
2010年3月 (1)
2010年1月 (1)
2009年12月 (2)
2009年10月 (2)
2009年9月 (14)
2009年8月 (4)
2009年6月 (14)
2009年5月 (3)
2009年4月 (1)
2009年3月 (3)
2009年2月 (11)
2008年10月 (7)
2008年8月 (5)
2008年7月 (1)
2008年6月 (2)
2008年5月 (2)
2008年4月 (5)

kernel

kernel中文社区
LDN
The Linux Document Project
The Linux Kernel Archives

manual

cppreference
gcc manual
mysql manual

sites

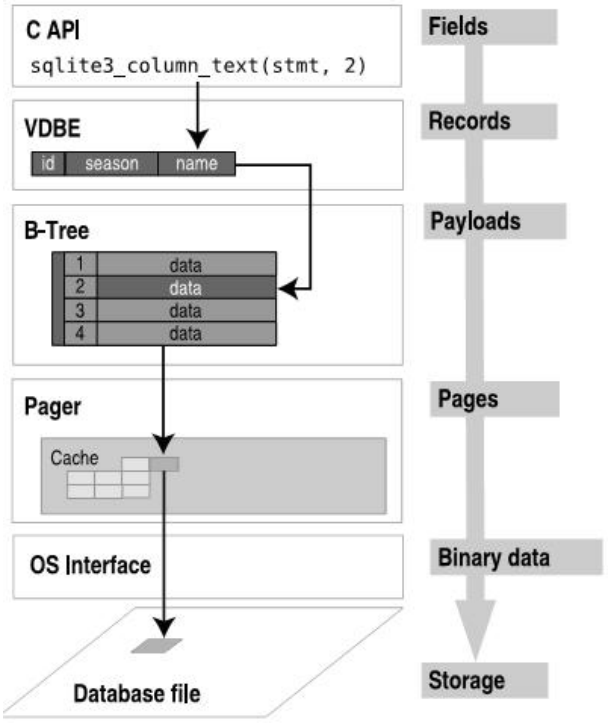
Database Journal
Fedora镜象
highscalability
KFUPM ePrints
Linux docs
Linux Journal
NoSQL
SQLite

技术社区

apache
CSDN
IBM-developerworks
lucene中国
nutch中国
oldlinux
oracle's forum

最新评论

1. Re:理解MySQL——架构与概念
我试验了下.数据 5 9 10 13 18begin;select * from asf_execution where num> 5 and num 5 and INSTANCE_ID_<18 lock in share mode;会有 1.行锁 2.间隙所 [5 18)插



从上往下，数据越来越无序，从下向上，数据越来越结构化.

2.5、B-Tree API

B-Tree模块有它自己的API，它可以独立于C API使用。另一个特点就是它支持事务。由pager处理的事务，锁和日志都是为B-tree服务的。根据功能可以分为以下几类：

2.5.1、访问和事务函数

sqlite3BtreeOpen: Opens a new database file. Returns a B-tree object.
sqlite3BtreeClose: Closes a database.
sqlite3BtreeBeginTrans: Starts a new transaction.
sqlite3BtreeCommit: Commits the current transaction.
sqlite3BtreeRollback: Rolls back the current transaction.
sqlite3BtreeBeginStmt: Starts a statement transaction.
sqlite3BtreeCommitStmt: Commits a statement transaction.
sqlite3BtreeRollbackStmt: Rolls back a statement transaction.

2.5.2、表函数

sqlite3BtreeCreateTable: Creates a new, empty B-tree in a database file.
sqlite3BtreeDropTable: Destroys a B-tree in a database file.
sqlite3BtreeClearTable: Removes all data from a B-tree, but keeps the B-tree intact.

2.5.3、游标函数(Cursor Functions)

sqlite3BtreeCursor: Creates a new cursor pointing to a particular B-tree.
sqlite3BtreeCloseCursor: Closes the B-tree cursor.
sqlite3BtreeFirst: Moves the cursor to the first element in a B-tree.
sqlite3BtreeLast: Moves the cursor to the last element in a B-tree.
sqlite3BtreeNext: Moves the cursor to the next element after the one it is currently pointing to.
sqlite3BtreePrevious: Moves the cursor to the previous element before the one it is currently pointing to.

sqlite3BtreeMoveto: Moves the cursor to an element that matches the key value passed in as a parameter.

2.5.4、记录函数(Record Functions)

sqlite3BtreeDelete: Deletes the record that the cursor is pointing to.
sqlite3BtreeInsert: Inserts a new element in the appropriate place of the B-tree.
sqlite3BtreeKeySize: Returns the number of bytes in the key of the record that the cursor is pointing to.
sqlite3BtreeKey: Returns the key of the record the cursor is currently pointing to.
sqlite3BtreeDataSize: Returns the number of bytes in the data record that the cursor is currently pointing to.
sqlite3BtreeData: Returns the data in the record the cursor is currently pointing to.

2.5.5、配置函数(Configuration Functions)

sqlite3BtreeSetCacheSize: Controls the page cache size as well as the synchronous writes (as defined in the synchronous pragma).
sqlite3BtreeSetSafetyLevel: Changes the way data is synced to disk in order to increase or decrease how well the database resists damage due to OS crashes and power failures. Level 1 is the same as asynchronous (no syncs() occur and there is a high probability of damage). This is the equivalent to pragma synchronous=OFF. Level 2 is the default. There is a very low but non-zero probability of damage. This is the equivalent to pragma synchronous=NORMAL. Level 3 reduces the probability of damage to near zero but with a write performance reduction. This is the equivalent to pragma synchronous=FULL.

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)

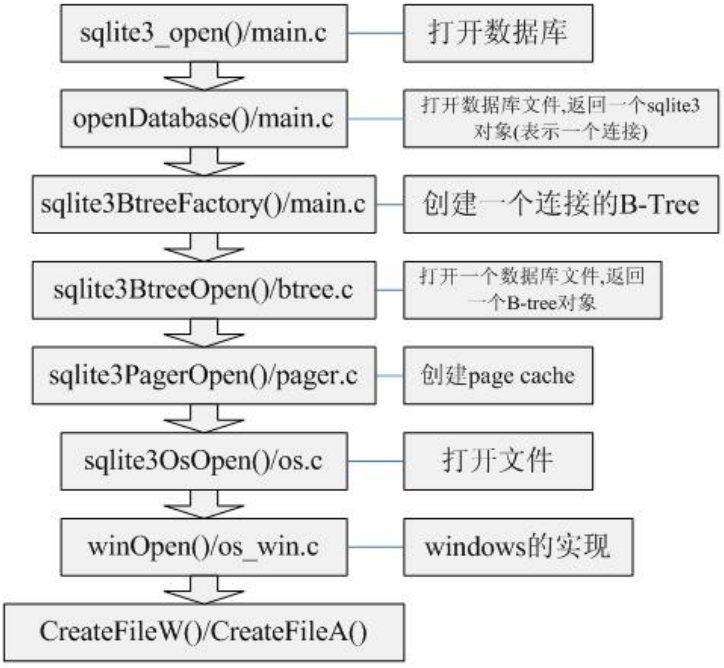
@arrowcat
这类语句本身没什么意义，但是楼主思考的角度让我豁然开朗。

--HJWAJ

sqlite3BtreeSetPageSize: Sets the database page size.
sqlite3BtreeGetPageSize: Returns the database page size.
sqlite3BtreeSetAutoVacuum: Sets the autovacuum property of the database.
sqlite3BtreeGetAutoVacuum: Returns whether the database uses autovacuum.
sqlite3BtreeSetBusyHandler: Sets the busy handler

2.6、实例分析

最后以sqlite3_open的具体实现结束本节的讨论(参见Version 3.6.10的源码)：



由上图可以知道，SQLite的所有IO操作，最终都转化为操作系统的系统调用(一名话：DBMS建立在痛苦的OS之上)。同时也可以看到SQLite的实现非常的层次化，模块化，使得SQLite更易扩展，可移植性非常强。

3、编译器（Compiler）

3.1、分词器(Tokenizer)

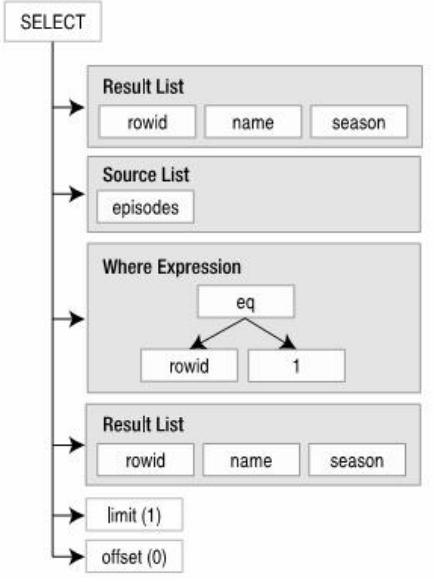
接口把要执行的SQL语句传递给Tokenizer,Tokenizer按照SQL的词法定义把它切分一个一个的词，并传递给分析器(Parser)进行语法分析。分词器是手工写的，主要在Tokenizer.c中实现。

3.2、分析器(Parser)

SQLite的语法分析器是用Lemon——一个开源的LALR(1)语法分析器的生成器，生成的文件为parser.c。

一个简单的语法树：

SELECT rowid, name, season FROM episodes WHERE rowid=1 LIMIT 1



3.3、代码生成器（Code Generator）

代码生成器是SQLite中取庞大，最复杂的部分。它与Parser关系紧密，根据语法分析树生成VDBE程序执行SQL语句的功能。由诸多文件构成：select.c,update.c,insert.c,delete.c,trigger.c,where.c等文件。这些文件生成相应的VDBE程序指令，比如SELECT语句就由select.c生成。下面是一个读操作中打开表的代码的生成实现：

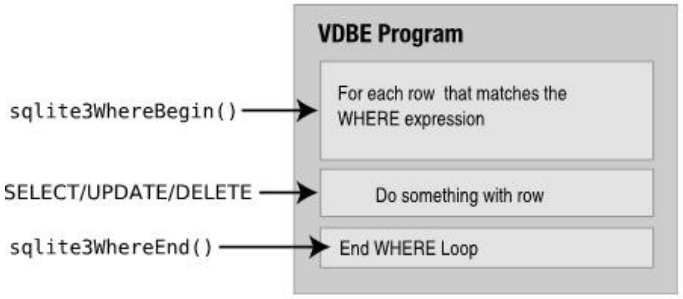
```
/* Generate code that will open a table for reading.
*/
void sqlite3OpenTableForReading(
  Vdbe *v,       /* Generate code into this VDBE */
```

```
  int iCur,      /* The cursor number of the table */
  Table *pTab    /* The table to be opened */
){
  sqlite3VdbeAddOp(v, OP_Integer, pTab->iDb, 0);
  sqlite3VdbeAddOp(v, OP_OpenRead, iCur, pTab->tnum);
  VdbeComment((v, "# %s", pTab->zName));
  sqlite3VdbeAddOp(v, OP_SetNumColumns, iCur, pTab->nCol);
}
```

Sqlite3vdbeAddOp函数有三个参数：（1）VDBE实例（它将添加指令），（2）操作码（一条指令），（3）两个操作数。

3.4、查询优化

代码生成器不仅负责生成代码，也负责进行查询优化。主要的实现位于where.c中，生成的WHERE语句块通常被其它模块共享，比如select.c，update.c以及delete.c。这些模块调用sqlite3WhereBegin()开始WHERE语句块的指令生成，然后加入它们自己的VDBE代码返回，最后调用sqlite3WhereEnd()结束指令生成，如下：



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(三)---内核概述(1)
» 下一篇：SQLite入门与分析(四)---Page Cache之事务处理(1)

posted @ 2009-02-17 21:12 YY哥 阅读(9244) 评论(15) 编辑 收藏

评论列表

#1楼 2009-02-17 23:07 魔都浪子~

很好，继续关注中，
顺便问一下，LZ的图是哪里来的？

支持(0) 反对(0)

#2楼 2009-02-18 00:15 梁逸晨

楼主，问个很现实的问题：有个网站，因为某种绝对原因限制不能使用MSSQL，也不能使用MYSQL和ORACLE等“真正”的数据库，最大并发量不会超过30，如果按照ACCESS的计算方法，估计1年后数据量会在300M左右，2年后会在700M左右，不知道使用SQLITE能不能应付这个需要？

支持(0) 反对(0)

#3楼[楼主] 2009-02-18 08:51 YY哥

@梁逸晨
据说SQLite可支持2TB的数据，30的并发量是不成问题的。

支持(0) 反对(0)

#4楼[楼主] 2009-02-18 08:52 YY哥

@WCF技术联盟
呵呵，有些图是从相关文献上截的，有些图是自己画的。不知道你是指哪个图?

支持(0) 反对(0)

#5楼 2009-02-18 09:12 魔都浪子~

第一副和最后一个图片， 没猜错的话2.6、实例分析那个图是LZ画的吧

支持(0) 反对(0)

#6楼[楼主] 2009-02-18 09:28 YY哥

@WCF技术联盟
嗯，那个实例分析是自己画的。画得太丑了，想尽力把它画好，但是实在是能力有限，没办法。呵呵

支持(0) 反对(0)

#7楼[楼主] 2009-02-18 09:30 YY哥

@WCF技术联盟
其余的图是从 The Definitive Guide to SQLite这本书上取的。

支持(0) 反对(0)

#8楼 2009-02-18 11:33 chanzuo[未注册用户]

我想知道sqlite中的虚拟机是怎么实现并且怎么工作的。希望楼主能详细描一下。谢谢。

#9楼 2009-02-18 12:11 Kingthy

@arrowcat
理论上是这样.但我在一个项目中知道,如果当数据库文件达到1GB以上时.如果要进行一个更新操作(Update)时,那时速度就会受不了...

支持(0) 反对(0)

#10楼[楼主] 2009-02-18 13:00 YY哥

@Kingthy
我没有用SQLite做过实际的项目，所以不好说。但是性能不好，我想不仅仅只是数据太大的原因，比如和硬件，应用程序的设计，数据库的设计，以及访问连接数量都有关系。
SQLite的目的也不是要取代通用DBMS应用场景。

支持(0) 反对(0)

#11楼 2009-02-18 13:04 true[未注册用户]

希望博主多侧重于自己的分析，尤其是代码的分析，比如“2.6、实例分析
”。其他的在Definitive Guide中有介绍的，可以简单些，呵呵，一点自己的看法。另外，看懂Guide，其实离读懂sqlite的内部构造，差很远，很远，使用sqlite和阅读他的代码，难度上不是一个数量级.期待下文

#12楼[楼主] 2009-02-18 13:45 YY哥

@true
谢谢你的建议，以后会在这上面下功夫的。

支持(0) 反对(0)

#13楼 2009-02-18 17:25 魔都浪子~

@arrowcat
我觉得你已经画的不错了，谦虚了。
以后跟你学习啦：）

支持(0) 反对(0)

#14楼 2009-03-05 08:44 Soli

Not bad!

支持(0) 反对(0)

#15楼 2013-07-29 10:34 hark.perfe

非常欣赏楼主几篇对sqlite的研究总结,文章写的很精彩,层次结构清晰,配合图案讲解通俗易懂.
有一点非常想请教楼主!您对sqlite3这几篇文章目录把握这么清晰,怎么做到的呢?非常想听到您的指教!谢谢!!盼复!

支持(0) 反对(0)

最新IT新闻:
· Twitter收购深度学习创业公司Madbits
· 这两个前亚马逊员工要把亚马逊赶出印度
· Twitter财报中你不能错过的6个数据
· 甲骨文对CEO拉里森每年股票奖励削减过半
· Facebook关闭Gifts礼品商店：探索电商新路
» 更多新闻...

最新知识库文章:
· 如何在网页中使用留白
· SQL/NoSQL两大阵营激辩：谁更适合大数据
· 如何获取（GET）一杯咖啡——星巴克REST案例分析
· 为什么程序员的工作效率跟他们的工资不成比例
· 我眼里的DBA
» 更多知识库文章...