# SQLite的工作模式

## —— 文件数据库、内存数据库及临时数据库

## 一、SQLite

SQLite的体型小巧而功能强大，是开源的内存数据库（内嵌式数据库）。SQLite应用极广，手机、mp3，机顶盒也可能存在SQLite，Apple的Mac OS，Linux，或者Windows在安装第三方软件时也可以应用SQLite。

### 1.1SQLite技术优点

1.  SQLite轻量级、跨平台的关系型开源内存数据库，使用SQLite只需带上动态库，就可使用SQLite全部功能（动态库Windows下487KB，Linux下347KB）；

2.   核心引擎不依赖第三方软件，也不需要安装；

3.   数据库中所有的信息（比如表、视图、触发器、等）都包含在一个文件内。这个文件可以copy到其它目录或其它机器上，也照用不误。如果使用内存方式，可以没有该文件；

4.   除了主流操作系统，SQLite还支持了很多冷门的操作系统。它对很多嵌入式系统（比如Android、Windows Mobile、Symbin、Palm、VxWorks等）也支持；

5.   SQLite的API不区分当前操作的数据库是在内存还是在文件（对于存储介质是透明的）。

### 1.2缺点

1. 并发访问的锁机制

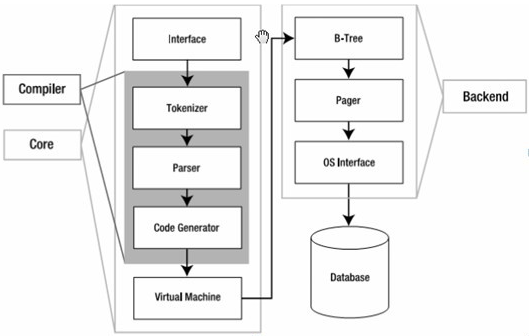
SQLite在并发（包括多进程和多线程）读写方面的性能不太理想。数据库可能会被写操作独占，从而导致其它读写操作阻塞或出错。

2. SQL标准支持不全

参见文档2

## 二、体系机构

SQLite模块将查询过程分为几个不连续的任务，在结构栈的顶部编译查询语句中不执行，在底部处理操作系统的存储和接口。



SQLite的体系结构（注：结构图转载于《SQLite权威指南》）

## 三、SQLite文件数据库、内存数据库建立及导入导出

SQLite官网（[www.SQLite.org](http://www.sqlite.org/)）同时提供已编译版本和源程序。同时适用于Windows和Linux。

### 3.1文件数据库、内存数据库的建立

#### 3.1.1文件数据库的建立

Windows：

1）下载SQLite最新版本SQLite3.exe；

2）dos进入到执行程序目录下；

3）输入 sqlite3.exe  d:\test.db（如果后面执行路径存在test.db 则打开数据库；如果执行路径下不存在test.db则新建test.db）。

Linux：

1）下载SQLite最新版本SQLite3；

2）Shell进入到可执行文件目录下；

3）输入./sqlite3 /home/test.db（如果后面执行路径存在test.db 则打开数据库；如果执行路径下不存在test.db则新建test.db）。

#### 3.1.2内存数据库的建立

在SQLite中，数据库通常是存储在磁盘文件中的。然而在有些情况下，我们可以让数据库始终驻留在内存中。最常用的一种方式是在调用sqlite3\_open()的时候，数据库文件名参数传递":memory:"，如：

rc = sqlite3\_open(":memory:", &db);

在调用完以上函数后，不会有任何磁盘文件被生成，一个新的数据库在内存中被成功创建了。由于没有持久化，该数据库在当前数据库连接被关闭后就会立刻消失。需要注意的是，尽管多个数据库连接都可以通过上面的方法创建内存数据库，然而它们却是不同的数据库，相互之间没有任何关系。事实上，我们也可以通过Attach命令将内存数据库像其他普通数据库一样，附加到当前的连接中，如：

ATTACH DATABASE ':memory:' AS aux1;

#### 3.1.3内存数据库的建立及内存文件模式双向转换

1. C代码示例：
3. SQLite3 \*db;
5. rc = SQLite3\_open(":memory:",&db);    // 内存中创建一个数据库
7. 3.2 文件数据库、内存数据库的导入导出
9. C代码示例：
11. ////////////////////////////////////////////////////////////////////////////////////////////
13. //参数说明:
15. //pInMemory: 指向内存数据库指针
17. //zFilename: 指向文件数据库目录的字符串指针
19. //isSave  0: 从文件数据库载入到内存数据库 1：从内存数据库备份到文件数据库
21. ////////////////////////////////////////////////////////////////////////////////////////////
23. **int** loadOrSaveDb(SQLite3 \*pInMemeory, **const** **char** \*zFilename, **int** isSave)
25. {
27. **int** rc;
29. SQLite3 \*pFile;
31. SQLite3\_backup \*pBackup;
33. SQLite3 \*pTo;
35. SQLite3 \*pFrom;
37. rc = SQLite3\_open(zFilename, &pFile);
39. **if**(rc == SQLITE\_OK)
41. {
43. pFrom = (isSave?pInMemeory:pFile);
44. pTo = (isSave?pFile:pInMemeory);
46. pBackup = SQLite3\_backup\_init(pTo,"main",pFrom,"main");
48. **if**(pBackup)
49. {
50. (**void**)SQLite3\_backup\_step(pBackup,-1);
51. (**void**)SQLite3\_backup\_finish(pBackup);
52. }
54. rc = SQLite3\_errcode(pTo);
56. }
58. (**void**)SQLite3\_close(pFile);
60. **return** rc;
61. }

 调用实例：

1. **int** ret = 0;
3. **char** \*filename = “d:\\test.db”;
4. SQLite3 \*memoryDb;
6. ret = SQLite3\_open(“memory:”, &memoryDb);
7. ret = loadOrSaveDb(memoryDb, filename, 0) //文件数据库导入到内存数据库

### 3.2文件数据库命令格式的导入导出

#### 3.2.1 文件数据库*命令格式*数据导出、备份

（SQLite shell 命令行）

（1）指定输出目标（输出重定向）

SQLite>.output ./test.sql

（2）输出

SQLite>.dump

（3）恢复输出定向

SQLite>.output stdout

#### 3.2.2文件数据库*命令格式*数据导入

$ sqlite3 test.db < test.sql

## 四、SQLite临时数据库

### 4.1命令行创建临时表

[liangxx@localhost test]$ sqlite3 test.db

SQLite version 3.7.13 2012-06-11 02:05:22

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite> .q

[liangxx@localhost test]$ ls

[liangxx@localhost test]$ sqlite3 test.db

SQLite version 3.7.13 2012-06-11 02:05:22

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite> create **temp** table testtable(ID integer primary key, GOAL real);

sqlite> .q

[liangxx@localhost test]$ ls

test.db

[liangxx@localhost test]$ sqlite3 test.db

SQLite version 3.7.13 2012-06-11 02:05:22

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite> .table

sqlite> create temp table testtable(ID integer primary key, GOAL real);

sqlite> .table

**temp.testtable**

sqlite> .q

[liangxx@localhost test]$ sqlite3 test.db

SQLite version 3.7.13 2012-06-11 02:05:22

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite> .tables

sqlite> create temp table testtable(ID integer primary key, GOAL real);

sqlite> .tables

temp.testtable

sqlite>

### 4.2临时数据库

在调用sqlite3\_open()函数或执行ATTACH命令时，如果数据库文件参数传的是空字符串，那么一个新的临时文件将被创建作为临时数据库的底层文件，如：

rc = sqlite3\_open("", &db);

或

ATTACH DATABASE '' AS aux2;

和内存数据库非常相似，两个数据库连接创建的临时数据库也是各自独立的，在连接关闭后，临时数据库将自动消失，其底层文件也将被自动删除。

尽管磁盘文件被创建用于存储临时数据库中的数据信息，但是实际上临时数据库也会和内存数据库一样通常驻留在内存中，唯一不同的是，当临时数据库中数据量过大时，SQLite为了保证有更多的内存可用于其它操作，因此会将临时数据库中的部分数据写到磁盘文件中，而内存数据库则始终会将数据存放在内存中。

应用程序可以使用temp\_store pragma和SQLITE\_TEMP\_STORE编译时参数强制临时数据库如内存数据库般工作。

### 4.3相关编译指令

**SQLITE\_DEFAULT\_TEMP\_CACHE\_SIZE**编译期参数可以用于指定**临时表**和索引在占用多少Cache Page时才需要被刷新到磁盘文件，该参数的缺省值为500页。

对于SQLite来说，回滚日志、主数据库日志和SQL语句日志文件在需要的时候SQLite都会将它们写入磁盘文件，但是对于其它类型的临时文件，SQLite是可以将它们存放在内存中以取代磁盘文件的，这样在执行的过程中就可以减少大量的IO操作了。要完成该优化主要依赖于以下三个因素：

1. 编译时参数**SQLITE\_TEMP\_STORE**

该参数是源代码中的宏定义(#define)，其取值范围是0到3(缺省值为1)，见如下说明：

1). 等于0时，临时文件总是存储在磁盘上，而不会考虑pragma temp\_store指令的设置。

2). 等于1时，临时文件缺省存储在磁盘上，但是该值可以被pragma temp\_store指令覆盖。

3). 等于2时，临时文件缺省存储在内存中，但是该值可以被pragma temp\_store指令覆盖。

4). 等于3时，临时文件总是存储在内存中，而不会考虑pragma temp\_store指令的设置。

2. 运行时指令**pragma temp\_store**

该指令的取值范围是0到2(缺省值为0)，在程序运行时该指令可以被动态的设置，见如下说明：

1). 等于0时，临时文件的存储行为完全由SQLITE\_TEMP\_STORE编译期参数确定。

2). 等于1时，如果编译期参数SQLITE\_TEMP\_STORE指定使用内存存储临时文件，那么该指令将覆盖这一行为，使用磁盘存储。

3). 等于2时，如果编译期参数SQLITE\_TEMP\_STORE指定使用磁盘存储临时文件，那么该指令将覆盖这一行为，使用内存存储。

\*pragma temp\_store\_directory用于更改磁盘日志位置。

\*当改变temp\_store设置，所有已存在的临时表、索引、触发器及视图将被立即删除。

如果将临时表存放于文件中，它用于指定文件的存放路径。该设置仅在当前的数据库连接中有效，当数据库重新在新的连接中打开时，将回到其默认值。当改变temp\_store\_directory设置时，所有已存在的临时表、索引、触发器以及视图都将立即删除。在实际应用中，打开数据库后应该立即设置该值。目录名directory-name应该用单引号引起来。如果想回到默认值，将directory-name设为空字符串，如PRAGMA temp\_store\_directory = '';如果路径找不到或目录directory-name无法写入，会提示错误。默认的临时文件存放的目录依赖于操作系统。 **对于Unix/Linux/OSX来说，默认可路径是/var/tmp, /usr/tmp, /tmp以及当前目录 current-directory中第一个可写的目录。**对于 Windows NT，默认路径由Windows决定，通常是C:\Documents and Settings\user-name\Local Settings\Temp\。**SQLite创建的临时文件在打开后会被立即删除（unlink）, 这样当SQLite进程退出时，操作系统就可以自动删除这些文件。所以正常状态下，使用ls或dir命令是无法看到这些临时文件的。**

3. 临时文件的大小

对于以上两个参数，都有参数值表示缺省情况是存储在内存中的，只有当临时文件的大小超过一定的阈值后才会根据一定的算法，将部分数据写入到磁盘中，以免临时文件占用过多的内存而影响其它程序的执行效率。

最后在重新赘述一遍，SQLITE\_TEMP\_STORE编译期参数和pragma temp\_store运行时指令只会影响除回滚日志和主数据库日志之外的其它临时文件的存储策略。换句话说，回滚日志和主数据库日志将总是将数据写入磁盘，而不会关注以上两个参数的值。