## 1、FMDB源码主要有以下几个文件组成：

### FMResultSet : 表示FMDatabase执行查询之后的结果集。

### FMDatabase : 表示一个单独的SQLite数据库操作实例，通过它可以对数据库进行增删改查等等操作。

### FMDatabaseAdditions : 扩展FMDatabase类，新增对查询结果只返回单个值的方法进行简化，对表、列是否存在，版本号，校验SQL等等功能。

### FMDatabaseQueue : 使用串行队列 ，对多线程的操作进行了支持。

### FMDatabasePool : 使用任务池的形式，对多线程的操作提供支持。（不过官方对这种方式并不推荐使用，优先选择FMDatabaseQueue的方式：ONLY\_USE\_THE\_POOL\_IF\_YOU\_ARE\_DOING\_READS\_OTHERWISE\_YOULL\_DEADLOCK\_USE\_FMDATABASEQUEUE\_INSTEAD）

## 2、FMDatabase

### 打开数据库连接

### 查询数据库 executeQuery

4. GCD的快速迭代方法 dispatch\_apply

通常我们会用for循环遍历，但是GCD给我们提供了快速迭代的方法dispatch\_apply，使我们可以同时遍历。比如说遍历0~5这6个数字，for循环的做法是每次取出一个元素，逐个遍历。dispatch\_apply可以同时遍历多个数字。

4. GCD的快速迭代方法 dispatch\_apply

通常我们会用for循环遍历，但是GCD给我们提供了快速迭代的方法dispatch\_apply，使我们可以同时遍历。比如说遍历0~5这6个数字，for循环的做法是每次取出一个元素，逐个遍历。dispatch\_apply可以同时遍历多个数字。

dispatch\_queue\_t queue = dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0);

dispatch\_apply(6, queue, ^(size\_t index) {

NSLog(@"%zd------%@",index, [NSThread currentThread]);

});函数- (void)bindObject:(id)obj toColumn:(int)idx inStatement:(sqlite3\_stmt\*)pStmt进行参数绑定，把通配符形式的参数改成obj实际参数。

将通配符？:age按照索引 赋值为obj。比如：SELECT \* FROM t\_student WHERE age > :age –> SELECT \* FROM t\_student WHERE age > obj 或者 SELECT \* FROM t\_student WHERE age > ? –> SELECT \* FROM t\_student WHERE age > obj

### 更新数据库操作

解释：这里的更新，并不只是单单的更新数据。而是对数据库有更改的操作，增删改都算。FMDB调用的都是executeUpdate系列函数。这个函数基本上跟executeQuery系列函数的实现基本差不多。只是它生成statement对象后，直接调用rc = sqlite3\_step(pStmt);更新执行，而没有像executeQuery延迟到FMResultSet中的next函数中执行

### WithFormat形式的数据查询和更新

### 将format的sql语句 改为 可用的sql语句

SELECT \* FROM t\_student WHERE age > %d --> SELECT \* FROM t\_student WHERE age > ?

(void)extractSQL:(NSString \*)sql argumentsList:(va\_list)args intoString:(NSMutableString \*)cleanedSQL arguments:(NSMutableArray \*)arguments;

### 一次性执行多条sql语句。

(BOOL)executeStatements:(NSString \*)sql withResultBlock:(FMDBExecuteStatementsCallbackBlock)block ；

### FMDB的加解密

解释：FMDB中使用- [FMDatabase setKey:]和- [FMDatabase setKeyWithData:]输入数据库密码以求验证用户身份，使用- [FMDatabase rekey:]和- [FMDatabase rekeyWithData:]来给数据库设置密码或者清除密码。这两类函数分别对sqlite3\_key和sqlite3\_rekey函数进行了封装。

## 3、FMResultSet

实现思路：

1. 初始化对象
2. 遍历取得所有的结果集合
3. 列名与该列的列数的一一对应关系获得每一行中每一个列字段的值。
4. 获得每一行中每一个列字段的值
5. 获取每一行中所有的结果集合
6. 对KVC的支持

## 4、FMDatabaseAdditions

扩展FMDatabase类，新增对查询结果只返回单个值的方法进行简化，对表、列是否存在，版本号，校验SQL等等功能。

### :XXXForQuery系列函数

对查询结果只有一个值得情况进行优化，有多个值也只取第一个值。用法实例：

数据库的一些概要信息

-(BOOL)tableExists:(NSString\*)tableName;数据库表是否存在。

-(BOOL)columnExists:(NSString\*)columnName inTableWithName:(NSString\*)tableName;在tableName表中columnName是否存在。

-(FMResultSet\*)getSchema;数据库的一些概要信息。实例如下

### 校验sql语句是否合法

-(BOOL)validateSQL:(NSString\*)sql error:(NSError\*\*)error;

## 5、FMDatabaseQueue

### 1、初始化Queue。生成一个串行队列

之于为什么要用dispatch\_queue\_set\_specific和dispatch\_get\_specific判断是不是当前queue，是因为为了防止多线程操作时候出现死锁。可以参考[告诉你dispatch\_queue\_set\_specific和dispatch\_get\_specific是个什么鬼](http://blog.csdn.net/mr_yong/article/details/50697656)和[被废弃的dispatch\_get\_current\_queue](http://blog.csdn.net/yiyaaixuexi/article/details/17752925)。

### 2、串行执行数据库操作

### /\*\*FMDatabaseQueue虽然看似一个队列，实际上它本身并不是，它通过内部创建一个Serial的dispatch\_queue\_t来处理通过inDatabase和inTransaction传入的Blocks，所以当我们在主线程（或者后台）调用inDatabase或者inTransaction时，代码实际上是同步的。FMDatabaseQueue这么设计的目的是让我们避免发生并发访问数据库的问题，因为对数据库的访问可能是随机的（在任何时候）、不同线程间（不同的网络回调等）的请求。内置一个Serial队列后，FMDatabaseQueue就变成线程安全了，所有的数据库访问都是同步执行，而且这比使用@synchronized或NSLock要高效得多。

### 3、事务的实现

**数据库中的事务 也是保证数据库安全的一种手段。**

**关于延时性事务和独占性事务的区别如下：**

在SQLite 3.0.8或更高版本中，事务可以是延迟的，即时的或者独占的。“延迟的”即是说在数据库第一次被访问之前不获得锁。 这样就会延迟事务，BEGIN语句本身不做任何事情。直到初次读取或访问数据库时才获取锁。对数据库的初次读取创建一个SHARED锁 ，初次写入创建一个RESERVED锁。由于锁的获取被延迟到第一次需要时，别的线程或进程可以在当前线程执行BEGIN语句之后创建另外的事务 写入数据库。若事务是即时的，则执行BEGIN命令后立即获取RESERVED锁，而不等数据库被使用。在执行BEGIN IMMEDIATE之后， 你可以确保其它的线程或进程不能写入数据库或执行BEGIN IMMEDIATE或BEGIN EXCLUSIVE. 但其它进程可以读取数据库。 独占事务在所有的数据库获取EXCLUSIVE锁，在执行BEGIN EXCLUSIVE之后，你可以确保在当前事务结束前没有任何其它线程或进程 能够读写数据库。

### 4、存档与回滚

## 6、三、FMDatabasePool

### 使用任务池，对多线程的操作提供支持

不过官方对这种方式并不推荐使用（ONLY\_USE\_THE\_POOL\_IF\_YOU\_ARE\_DOING\_READS\_OTHERWISE\_YOULL\_DEADLOCK\_USE\_FMDATABASEQUEUE\_INSTEAD），优先选择FMDatabaseQueue的方式。