ch_soft的专栏

我，只专注于移动互联网,技术交流群：10912272 博客：www.9idev.com

FMDatabase 的使用方法

分类： 【开发技术】数据库 2012-03-21 17:29 6269人阅读 评论(2) 收藏 举报

insert database methods sqlite dictionary parameters

FMDatabase 的使用方法

```
- (NSString*) getPath {
   NSArray* paths =NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,NSUserDomainMask, YES) ;
    return [[pathsobjectAtIndex:0]stringByAppendingPathComponent:@"MyTable"] ;
}


1.创建数据库
-(void)CreateTable;
{
dataBase = [FMDatabasedatabaseWithPath:[selfgetPath]];
    if (![dataBase open])
       NSLog(@"OPEN FAIL");
    }

    [dataBaseexecuteUpdate:@"CREATE TABLE IF NOT EXISTS MyTable(aa float,bb text,cc integer,dd integer,ee
text)"];
    [dataBaseclose];
}



2.查询数据
-(void)QueryData
{
//获取数据
   recordArray = [[NSMutableArrayalloc]init];

   db = [[FMDatabasealloc]initWithPath:[selfgetPath]];

    if ([db open]) {
       FMResultSet *rs = [dbexecuteQuery:@"SELECT * FROM MyTable"];
        while ([rs next]){
            OneRecord = [[OneRecord alloc]init];
            OneRecord.aa = [NSNumbernumberWithFloat:[rs doubleForColumn:@"aa"]];
            OneRecord.bb = [rs stringForColumn:@"bb"];
            OneRecord.cc = [NSNumbernumberWithInt:[rs intForColumn:@"cc"]];
            OneRecord.dd = [NSNumbernumberWithInt:[rs intForColumn:@"dd"]];
            OneRecord.ee = [rs stringForColumn:@"ee"];
            [recordArrayaddObject: OneRecord];
            [OneRecordrelease];
        }
        [rs close];
        [dbclose];
}

3.更新数据
-(void)UpdateData
{
if ([dbopen]) {
         [dbbeginTransaction];
         [dbexecuteUpdate:@"UPDATE MyTable SET aa = ? WHERE date = ?",aa1,aa2];
         [dbexecuteUpdate:@"UPDATE MyTable SET bb = ? WHERE date = ?",bb1,bb2];
         [dbexecuteUpdate:@"UPDATE MyTable SET cc = ? WHERE date = ?",cc1,cc2];
         [dbexecuteUpdate:@"UPDATE MyTable SET dd = ? WHERE date = ?",dd1,dd2];
         [dbcommit];
         db close];
   }

}

4。插入数据
-(void)insertData
{
//插入数据库
    [dbbeginTransaction];
    [dbexecuteUpdate:@"INSERT INTO MyTable (aa,bb,cc,dd,ee) VALUES (?,?,?,?,?)",
   NSNumber numberWithFloat:aa],bb,cc,dd,ee];
    db commit];
    db close];

}
```

声音类 (10)
微博分享 (2)
声音识别 (1)
【iOS7 专辑】 (2)

阅读排行

第二、UIScrollView的使 (50918)
IOS Socket使用大全 (23488)
第一、UITableView的使 (16909)
iOS 录音功能的实现 (10374)
IOS要:使用开源代码IOS (9902)
iOS-实现文件上传下载 (9712)
NSMutableAttributedStri (6287)
FMDatabase 的使用方 (6268)
新浪开发者平台(Sina Ap (6224)
Xcode GDB 调试 (5887)

评论排行

IOS Socket使用大全 (12)
AudioToolbox 详解 (9)
第二、UIScrollView的使 (8)
ios 人脸识别 资源 (8)
IOS要:使用开源代码IOS (6)
自己开发的Grid组件 针对 (6)
第一、UITableView的使 (5)
CATransform3D 矩阵变换 (4)
NSMutableAttributedStri (4)
@synthesize window=_w (4)

推荐文章

最新评论

淘宝账号基于OAuth2.0的登录验
孤好梦中杀人: 不详细，求demo
AudioToolbox使用方法总结
abamon: mark！正准备做ios音频的实时流传输。
IOS Socket使用大全 -将持续
mebay: 有代码示例可以提供下下载嘛？527519721@qq.com
第二、UIScrollView的使用大全
怪盗绅士L:
http://www.csdn.net/tag
【修复】【兼容iOS7】cocos2d-
ch_soft: @zszeng:iOS7是64位的
【修复】【兼容iOS7】cocos2d-
zszeng: 没有增加CGColorSpaceRelease(colorSpace似乎，我增加这个也不报错
【修复】【兼容iOS7】cocos2d-
zszeng: 谢谢，但是我没明白原理？
IOS要:使用开源代码IOS进行soc
beichen2015: 对于Socket了解的不多，楼主把流程写的很明白啊
第二、UIScrollView的使用大全
Ahao_plus: 很不错的讲解，Very nice！thanks！
第二、UIScrollView的使用大全

5。官方的使用方法为：

Usage

There are three main classes in FMDB:

1. `FMDatabase` - Represents a single SQLite database. Used for executing SQL statements.
2. `FMResultSet` - Represents the results of executing a query on an`FMDatabase`.
3. `FMDatabaseQueue` - If you're wanting to perform queries and updates on multiple threads, you'll want to use this class. It's described in the "Thread Safety" section below.

Database Creation

An `FMDatabase` is created with a path to a SQLite database file. This path can be one of these three:

1. A file system path. The file does not have to exist on disk. If it does not exist, it is created for you.
2. An empty string (`@""`). An empty database is created at a temporary location. This database is deleted with the`FMDatabase` connection is closed.
3. `NULL`. An in-memory database is created. This database will be destroyed with the`FMDatabase` connection is closed.

(For more information on temporary and in-memory databases, read the sqlite documentation on the subject:http://www.sqlite.org/inmemorydb.html)

```
FMDatabase *db = [FMDatabase databaseWithPath:@"/tmp/tmp.db"];
```

Opening

Before you can interact with the database, it must be opened. Opening fails if there are insufficient resources or permissions to open and/or create the database.

```
if (![db open]) {
    [db release];
    return;
}
```

Executing Updates

Any sort of SQL statement which is not a `SELECT` statement qualifies as an update. This includes `CREATE`, `PRAGMA`,`UPDATE`, `INSERT`, `ALTER`,`COMMIT`, `BEGIN`, `DETACH`,`DELETE`, `DROP`, `END`,`EXPLAIN`, `VACUUM`, and `REPLACE` statements (plus many more). Basically, if your SQL statement does not begin with`SELECT`, it is an update statement.

Executing updates returns a single value, a `BOOL`. A return value of `YES` means the update was successfully executed, and a return value of `NO` means that some error was encountered. If you use the `-[FMDatabase executeUpdate:error:withArgumentsInArray:orVAList:]` method to execute an update, you may supply an`NSError **` that will be filled in if execution fails. Otherwise you may invoke the`-lastErrorMessage` and`-lastErrorCode` methods to retrieve more information.

Executing Queries

A `SELECT` statement is a query and is executed via one of the`-executeQuery...` methods.

Executing queries returns an `FMResultSet` object if successful, and `nil` upon failure. Like executing updates, there is a variant that accepts an`NSError **` parameter. Otherwise you should use the`-lastErrorMessage` and`-lastErrorCode` methods to determine why a query failed.

In order to iterate through the results of your query, you use a `while()` loop. You also need to "step" from one record to the other. With FMDB, the easiest way to do that is like this:

```
FMResultSet *s = [db executeQuery:@"SELECT * FROM myTable"];
while ([s next]) {
    //retrieve values for each record
}
```

You must always invoke `-[FMResultSet next]` before attempting to access the values returned in a query, even if you're only expecting one:

```
FMResultSet *s = [db executeQuery:@"SELECT COUNT(*) FROM myTable"];
if ([s next]) {
    int totalCount = [s intForColumnIndex:0];
}
```

`FMResultSet` has many methods to retrieve data in an appropriate format:

- `intForColumn:`
- `longForColumn:`
- `longLongIntForColumn:`
- `boolForColumn:`
- `doubleForColumn:`
- `stringForColumn:`
- `dateForColumn:`
- `dataForColumn:`
- `dataNoCopyForColumn:`
- `UTF8StringForColumnIndex:`
- `objectForColumn:`

WeiLi_X: 好！收了。。

Each of these methods also has a `{type}ForColumnIndex:` variant that is used to retrieve the data based on the position of the column in the results, as opposed to the column's name.

Typically, there's no need to `-close` an `FMResultSet` yourself, since that happens when either the result set is deallocated, or the parent database is closed.

Closing

When you have finished executing queries and updates on the database, you should `-close` the `FMDatabase` connection so that SQLite will relinquish any resources it has acquired during the course of its operation.

```
[db close];
```

Transactions

`FMDatabase` can begin and commit a transaction by invoking one of the appropriate methods or executing a begin/end transaction statement.

Data Sanitization

When providing a SQL statement to FMDB, you should not attempt to "sanitize" any values before insertion. Instead, you should use the standard SQLite binding syntax:

```
INSERT INTO myTable VALUES (?, ?, ?)
```

The `?` character is recognized by SQLite as a placeholder for a value to be inserted. The execution methods all accept a variable number of arguments (or a representation of those arguments, such as an `NSArray`,`NSDictionary`, or a`va_list`), which are properly escaped for you.

Alternatively, you may use named parameters syntax:

```
INSERT INTO myTable VALUES (:id, :name, :value)
```

The parameters *must* start with a colon. SQLite itself supports other characters, but internally the Dictionary keys are prefixed with a colon, do**not** include the colon in your dictionary keys.

```
NSDictionary *argsDict = [NSDictionary dictionaryWithObjectsAndKeys:@"My Name", @"name", nil];
[db executeUpdate:@"INSERT INTO myTable (name) VALUES (:name)" withParameterDictionary:argsDict];
```

Thus, you SHOULD NOT do this (or anything like this):

```
[db executeUpdate:[NSString stringWithFormat:@"INSERT INTO myTable VALUES (%@)", @"this has \"
lots of ' bizarre \" quotes '"]];
```

Instead, you SHOULD do:

```
[db executeUpdate:@"INSERT INTO myTable VALUES (?)", @"this has \" lots of ' bizarre \" quotes
'"];
```

All arguments provided to the `-executeUpdate:` method (or any of the variants that accept a `va_list` as a parameter) must be objects. The following will not work (and will result in a crash):

```
[db executeUpdate:@"INSERT INTO myTable VALUES (?)", 42];
```

The proper way to insert a number is to box it in an `NSNumber` object:

```
[db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:42]];
```

Alternatively, you can use the `-execute*WithFormat:` variant to use `NSString`-style substitution:

```
[db executeUpdateWithFormat:@"INSERT INTO myTable VALUES (%d)", 42];
```

Internally, the `-execute*WithFormat:` methods are properly boxing things for you. The following percent modifiers are recognized:`%@`, `%c`, `%s`,`%d`, `%D`, `%i`,`%u`, `%U`, `%hi`,`%hu`, `%qi`, `%qu`,`%f`, `%g`, `%ld`,`%lu`, `%lld`, and `%llu`. Using a modifier other than those will have unpredictable results. If, for some reason, you need the`%` character to appear in your SQL statement, you should use`%%`.

Using FMDatabaseQueue and Thread Safety.

Using a single instance of FMDatabase from multiple threads at once is a bad idea. It has always been OK to make a FMDatabase object*per thread*. Just don't share a single instance across threads, and definitely not across multiple threads at the same time. Bad things will eventually happen and you'll eventually get something to crash, or maybe get an exception, or maybe meteorites will fall out of the sky and hit your Mac Pro. *This would suck*.

So don't instantiate a single FMDatabase object and use it across multiple threads.

Instead, use FMDatabaseQueue. It's your friend and it's here to help. Here's how to use it:

First, make your queue.

```
FMDatabaseQueue *queue = [FMDatabaseQueue databaseQueueWithPath:aPath];
```

Then use it like so:

```
[queue inDatabase:^(FMDatabase *db) {
    [db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:1]];
    [db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:2]];
    [db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:3]];

    FMResultSet *rs = [db executeQuery:@"select * from foo"];
    while ([rs next]) {
        …
    }
}];
```

An easy way to wrap things up in a transaction can be done like this:

```
[queue inTransaction:^(FMDatabase *db, BOOL *rollback) {
    [db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:1]];
    [db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:2]];
    [db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:3]];

    if (whoopsSomethingWrongHappened) {
        *rollback = YES;
        return;
    }
    // etc…
```

```
    [db executeUpdate:@"INSERT INTO myTable VALUES (?)", [NSNumber numberWithInt:4]];
}];
```

FMDatabaseQueue will make a serialized GCD queue in the background and execute the blocks you pass to the GCD queue. This means if you call your FMDatabaseQueue's methods from multiple threads at the same time GDC will execute them in the order they are received. This means queries and updates won't step on each other's toes, and every one is happy.

更多 1

上一篇 IOS 下使用AdMob广告

下一篇 App Icons on iPad and iPhone

主题推荐 nsmutablearray transactions dictionary character 数据库

猜你在找

AFNetworking速成教程
iOS 的目录操作基础
Core Data数据持久性存储基础教程
支付宝 报错 rsa_private read error : private key
iOS设备的硬件适配（关于armv6, armv7, armv7s 个人
ios webview清除缓存
IOS百度地图开发系列-百度地图不能正常显示
iPhone开发笔记（5）scrollView和pageControl的搭配
sqlite3用法详解草稿
CGContextRef用法

查看评论

1楼 谁抢名字啊 2013-07-16 09:33发表

新人来学习的，看完以后还是不怎么懂，楼主能不能给一个小Demo让我亲身体验下？谢谢！

Re: ch_soft 2013-07-17 11:38发表

回复u010986295：没记错的话，我csdn资源里面有个小例子

核心技术类目

全部主题 Java VPN Android iOS ERP IE10 Eclipse CRM JavaScript Ubuntu NFC
WAP jQuery 数据库 BI HTML5 Spring Apache Hadoop .NET API HTML SDK IIS
Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE
Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
Spark HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap