
Sqlite 的基础知识

1. Sqlite 简介

Sqlite 是一种轻量级，零配置的，可嵌入的程序驱动型的二进制文件，同时也是一种关系型数据库。鉴于 Sqlite 数据库的这些优点，现在流行的操作系统 Android 和 ios 都选择使用 Sqlite 作为数据存储的主要方式。

2. Sqlite 的使用场景

现在的主流移动设备像 Android、iPhone 等都使用 SQLite 作为复杂数据的存储引擎，在我们为移动设备开发应用程序时，也许就要使用到 SQLite 来存储我们大量的数据，所以我们就需要掌握移动设备上的 SQLite 开发技巧。对于 Android 平台来说，系统内置了丰富的 API 来供开发人员操作 SQLite，我们可以轻松的完成对数据的存取

3. 数据库的创建:

数据库的创建是通过数据库的帮助类来实现的，当用户要创建一个数据库就要定义一个帮助类，让其继承 SqliteOpenHelper 这个类，然后子该类中实现一定的回调方法就可以创建数据库了。

```
public class DBHelper extends SQLiteOpenHelper {  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);
```

```
    }
```

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        db.execSQL("ALTER TABLE person ADD COLUMN other STRING");  
    }
```

```
}
```

4. Sqlite 数据库中获取数据库对象的两个方法：

- 1) 获取数据库实例时使用了 `getWritableDatabase()` 方法。
- 2) 在 `getReadableDatabase()` 方法中，首先判断是否已存在数据库实例并且是打开状态，如果是，则直接返回该实例，否则试图获取一个可读写模式的数据库实例，如果遇到磁盘空间已满等情况获取失败的话，再以只读模式打开数据库，获取数据库实例并返回，然后为数据库对象赋值为最新打开的数据库实例。
- 3) `getReadableDatabase()` 一般都会返回和 `getWritableDatabase()` 一样的数据库实例，所以我们在 `DBManager` 构造方法中使用 `getWritableDatabase()` 获取整个应用所使用的数据库实例是可行的
- 4) 当调用 `SQLiteOpenHelper` 的 `getWritableDatabase()` 或者 `getReadableDatabase()` 方法获取用于操作数据库的 `SQLiteDatabase` 实例的时候，如果数据库不存在，Android 系统会自动生成一个数据库，接着调用 `onCreate()` 方法。
- 5) `onCreate()` 方法在初次生成数据库时才会被调用，在 `onCreate()` 方法里可以生成数据库表结构及添加一些应用使用到的初始化数据。
- 6) `onUpgrade()` 方法在数据库的版本发生变化时会被调用，一般在软件升级时才需改变版本号，而数据库的版本是由程序员控制的。

5. Sqlite 数据库中简单的 Sql 语句：

* 增

```
insert into info (name,phone) values ('wuyudong','111')
```

* 删

```
delete from person where name = 'wuyudong'
```

* 改

```
update person set number='119' where name='wuyudong'
```

* 查

```
select * from person
```

```
select * from person where name='wuyudong'
```

6. Sql 语句的简单操作

对 SQLiteDatabase 的学习，我们应该重点掌握 execSQL() 和 rawQuery() 方法。

execSQL() 方法可以执行 insert、delete、update 和 CREATE TABLE 之类有更改行为的 SQL 语句；rawQuery() 方法用于执行 select 语句。

7. 使用 Api 封装方法：

Android 提供了一个名为 SQLiteDatabase 的类，该类封装了一些操作数据库的 API，使用该类可以完成对数据进行添加(Create)、查询(Retrieve)、更新(Update)和删除(Delete)操作（这些操作简称为 CRUD）。

1) 插入，修改和删除操作：

```
db.insert(String table, String nullColumnHack, ContentValues values);  
db.update(String table, ContentValues values, String whereClause, String whereArgs);  
db.delete(String table, String whereClause, String whereArgs);
```

insert 中的第二个参数表示如果插入的数据每一列都为空的话，需要指定此行中某一列的名称，系统将此列设置为 NULL，不至于出现错误；insert 中的第三个参数是 ContentValues 类型的变量，是键值对组成的 Map，key 代表列名，value 代表该列要

插入的值 ;update 的第二个参数也很类似 ,只不过它是更新该字段 key 为最新的 value 值 ,第三个参数 whereClause 表示 WHERE 表达式 ,比如 "age > ? and age < ?" 等 ,最后的 whereArgs 参数是占位符的实际参数值 ; delete 方法的参数也是一样。

下面来说说查询操作。

2) 查询操作 :

```
db.rawQuery(String sql, String[] selectionArgs);  
  
db.query(String table, String[] columns, String selection, String[]  
        selectionArgs, String groupBy, String having, String orderBy);  
  
db.query(String table, String[] columns, String selection, String[]  
        selectionArgs, String groupBy, String having, String orderBy,  
        String limit);  
  
db.query(String distinct, String table, String[] columns, String selection,  
        String[] selectionArgs);
```

上面几种都是常用的查询方法 ,第一种最为简单 ,将所有的 SQL 语句都组织到一个字符串中 ,使用占位符代替实际参数 ,selectionArgs 就是占位符实际参数集 ;下面的几种参数都很类似 ,columns 表示要查询的列所有名称集 ,selection 表示 WHERE 之后的条件语句 ,可以使用占位符 ,groupBy 指定分组的列名 ,having 指定分组条件 ,配合 groupBy 使用 ,orderBy 指定排序的列名 ,limit 指定分页参数 ,distinct 可以指定 "true" 或 "false" 表示要不要过滤重复值。需要注意的是 ,selection、groupBy、having、orderBy、limit 这几个参数中不包括 "WHERE"、"GROUP BY"、"HAVING"、"ORDER BY"、"LIMIT" 等 SQL 关键字。