

Android 连接数据库

Android 采用关系型数据库 **SQLite3**, 它是一个支持 **SQL** 轻量级的嵌入式数据库, 在嵌入式操作上有很广泛的, **WM** 采用的也是 **SQLite3**

关于过于、原理方面的东西在这篇文章里不会提到, 但是如果你想能够快速的学习操作 **SQLite3**, 那这就是你要找的文章!

首先, 我们看一下 **api**, 所有数据库相关的接口、类都在 **.database** 和 **android.database.sqlite** 两个包下, 虽然只有两个包, 但是如果你英文不好或是太懒的话也要迷茫一段时间, 其实, 我们真正用的到的没有几个!

1、SQLiteOpenHelper (android.database.sqlite.SQLiteOpenHelper)

这是一个抽象类, 关于抽象类我们都知道, 如果要使用它, 一定是继承它!

这个类的方法很少, 有一个构造方法

```
SQLiteOpenHelper(android.content.Context context, java.lang.String  
name, android.database.sqlite.SQLiteDatabase.CursorFactory factory, int  
version);
```

参数不做过多的解释, **CursorFactory** 一般直接传 **null** 就可以

```
public void onCreate(SQLiteDatabase db)
```

此方法在创建数据库是被调用, 所以, 应该把创建表的操作放到这个方法里面, 一会儿在后面我们会再详细的说如何创建表

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
```

从方法名上我们就能知道这个方法是执行更新的, 没错, 当 **version** 改变是系统会调用这个方法, 所以在这个方法里应该执行删除现有表, 然后手动调用 **onCreate** 的操作

```
SQLiteDatabase getReadableDatabase()
```

可读的 **SQLiteDatabase** 对象

```
SQLiteDatabase getWritableDatabase()
```

获取可写的 **SQLiteDatabase** 对象

2、SQLiteDatabase(android.database.sqlite.SQLiteDatabase)

关于操作数据库的工作(增、删、查、改)都在这个类里

`execSQL(sql)`

执行 SQL 语句，用这个方法+SQL 语句可以非常方便的执行增、删、查、改

除此之外，Android 还提供了功过方法实现增、删、查、改

`long insert(TABLE_NAME, null, contentValues)`添加记录

`int delete(TABLE_NAME, where, whereValue)`删除记录

`int update(TABLE_NAME, contentValues, where, whereValue)` 更新记录

`Cursor query(TABLE_NAME, null, null, null, null, null, null)` 查询记录

除此之外，还有很多方法，如：`beginTransaction()`开始事务、`endTransaction()`结束事务...有兴趣的可以自己看 api，这里就不多赘述了

3、Cursor(android.database.Cursor)

游标(接口)，这个很熟悉了吧，Cursor 里的方法非常多，常用的有：

`boolean moveToPosition(position)`将指针移动到某记录

`getColumnIndex(Contacts.People.NAME)`按列名获取 id

`int getCount()`获取记录总数

`boolean requery()`重新查询

`boolean isAfterLast()`指针是否在末尾

`boolean isBeforeFirst()`时候是开始位置

`boolean isFirst()`是否是第一条记录

`boolean isLast()`是否是最后一条记录

`boolean moveToFirst()`、 `boolean moveToLast()`、 `boolean moveToNext()`同
`moveToPosition(position)`

4、SimpleCursorAdapter(android.widget.SimpleCursorAdapter)

也许你会奇怪了，之前我还说过关于数据库的操作都在 `database` 和 `database.sqlite` 包下，为什么把一个 `Adapter` 放到这里，如果你用过 `Android` 的 `SQLite3`，你一定会知道

，这是因为我们对数据库的操作会经常跟列表联系起来

经常有朋友会在这出错，但其实也很简单

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(
```

```
this,
```

```
R.layout.list,
```

```
myCursor,
```

```
new String[] {DB.TEXT1,DB. TEXT2},
```

```
new int[]{ R.id.list1,R.id.listText2 });
```

```
my.setAdapter(adapter);
```

一共 5 个参数，具体如下：

参数 1:Content

参数 2:布局

参数 3:Cursor 游标对象

参数 4:显示的字段，传入 `String[]`

参数 5:显示字段使用的组件，传入 `int[]`，该数组中是 `TextView` 组件的 `id`

到这里，关于数据库的操作就结束了，但是到目前为止我只做了翻译的工作，有些同学可能还是没有掌握，放心，下面我们一起顺着正常开发的思路理清一下头绪！

前面的只是帮没做过的朋友做下普及，下面才是你真正需要的!

一、写一个类继承 SQLiteOpenHelper

```
public class DatabaseHelper extends SQLiteOpenHelper
```

构造方法:

```
DatabaseHelper(Context context) {  
  
    super(context, DATABASE_NAME, null, DATABASE_VERSION);  
  
}
```

在 onCreate 方法里写建表的操作

```
public void onCreate(SQLiteDatabase db) {  
  
    String sql = "CREATE TABLE tb_test (_id INTEGER DEFAULT '1' NOT NULL  
    PRIMARY KEY AUTOINCREMENT,class_jb TEXT NOT NULL,class_ysbj  
    TEXT NOT NULL,title TEXT NOT NULL,content_ysbj TEXT NOT NULL)";  
  
    db.execSQL(sql);//需要异常捕获  
  
}
```

在 onUpgrade 方法里删除现有表，然后手动调用 onCreate 创建表

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
    String sql = "drop table "+tbname;  
  
    db.execSQL(sql);  
  
    onCreate(db);  
  
}
```

对表增、删、查、改的方法，这里用的是 SQLiteOpenHelper 提供的方法，也可以用 sql 语句实现，都是一样的

关于获取可读/可写 SQLiteDatabase，我不说大家也应该会想到，只有查找才会用到可读的 SQLiteDatabase

```
/**
 * 添加数据
 */

public long insert(String tname, int tage, String ttel){

    SQLiteDatabase db= getWritableDatabase();//获取可写 SQLiteDatabase 对象

    //ContentValues 类似 map，存入的是键值对

    ContentValues contentValues = new ContentValues();

    contentValues.put("tname", tname);

    contentValues.put("tage", tage);

    contentValues.put("ttel", ttel);

    return db.insert(tbname, null, contentValues);

}

/**
 * 删除记录
 * @param _id
 */

public void delete(String _id){

    SQLiteDatabase db= getWritableDatabase();

    db.delete(tbname,

        "_id=?",
```

```
new String[]{_id});

}

/**

* 更新记录的，跟插入的很像

*/

public void update(String _id,String tname, int tage, String ttel){

    SQLiteDatabase db= getWritableDatabase();

    ContentValues contentValues = new ContentValues();

    contentValues.put("tname", tname);

    contentValues.put("tage", tage);

    contentValues.put("ttel", ttel);

    db.update(tbname, contentValues,

    "_id=?",

    new String[]{_id});

}

/**

* 查询所有数据

* @return Cursor

*/

public Cursor select(){

    SQLiteDatabase db = getReadableDatabase();
```

```
return db.query(  
  
    tbname,  
  
    new String[]{"_id","tname","tage","ttel","taddr"},  
  
    null,  
  
    null, null, null, "_id desc");  
  
}
```

关于 `db.query` 方法的参数，有很多，为了防止大家弄乱，我简单说一下

参数 1：表名

参数 2：返回数据包含的列信息，`String` 数组里放的都是列名

参数 3：相当于 `sql` 里的 `where`，`sql` 里 `where` 后写的内容放到这就行了，例如：`tage>?`

参数 4：如果你在参数 3 里写了?(知道我为什么写 `tage>?` 了吧)，那个这里就是代替?的值 接上例：`new String[]{"30"}`

参数 5：分组，不解释了，不想分组就传 `null`

参数 6：having，想不起来的看看 `SQL`

参数 7：orderBy 排序

到这里，你已经完成了最多的第一步!我们来看看都用到了那些类：

`SQLiteOpenHelper` 我们继承使用的

`SQLiteDatabase` 增删查改都离不开它，即使你直接用 `sql` 语句，也要用到 `execSQL(sql)`

二、这里无非是对 `DatabaseHelper` 类定义方法的调用，没什么可说的，不过我还是对查询再唠叨几句吧

Android 查询出来的结果一 `Cursor` 形式返回

```
cursor = sqLiteHelper.select();//是不是很简单？
```

查询出来的 `cursor` 一般会显示在 `listView` 中，这就要用到刚才提到的 `SimpleCursorAdapter`

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(  
  
this,  
  
R.layout.list_row,  
  
cursor,  
  
new String[]{"tname","ttel"},  
  
new int[]{R.id.TextView01,R.id.TextView02}  
  
);
```

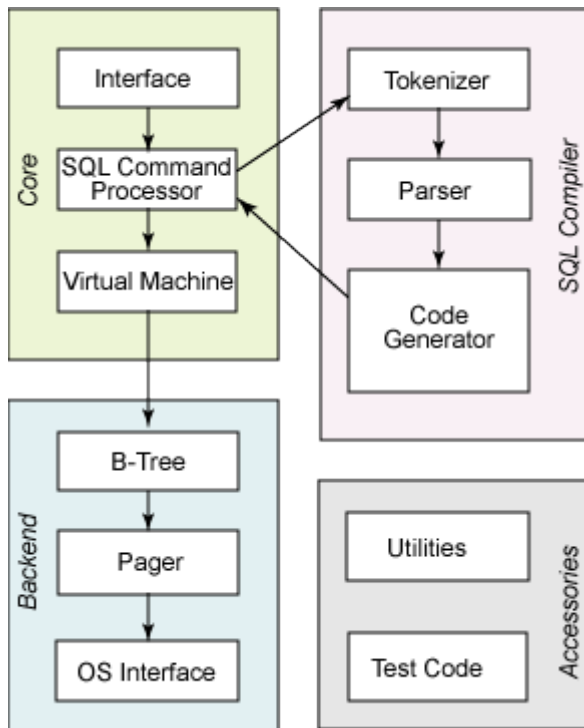
里面带有实例。自己好好学习吧！

SQLite 介绍

SQLite 一个非常流行的嵌入式数据库，它支持 SQL 语言，并且只利用很少的内存就有很好的性能。此外它还是开源的，任何人都可以使用它。许多开源项目（Mozilla, PHP, Python）都使用了 SQLite.

SQLite 由以下几个组件组成：SQL 编译器、内核、后端以及附件。SQLite 通过利用虚拟机和虚拟数据库引擎（VDBE），使调试、修改和扩展 SQLite 的内核变得更加方便。

图 1. SQLite 内部结构



SQLite 基本上符合 SQL-92 标准，和其他的主要 SQL 数据库没什么区别。它的优点就是高效，Android 运行时环境包含了完整的 SQLite。

SQLite 和其他数据库最大的不同就是对数据类型的支持，创建一个表时，可以在 CREATE TABLE 语句中指定某列的数据类型，但是你可以把任何数据类型放入任何列中。当某个值插入数据库时，SQLite 将检查它的类型。如果该类型与关联的列不匹配，则 SQLite 会尝试将该值转换成该列的类型。如果不能转换，则该值将作为其本身具有的类型存储。比如可以把一个字符串 (String) 放入 INTEGER 列。SQLite 称这为“弱类型” (manifest typing.)。

此外，SQLite 不支持一些标准的 SQL 功能，特别是外键约束 (FOREIGN KEY constraints)，嵌套 transaction 和 RIGHT OUTER JOIN 和 FULL OUTER JOIN，还有一些 ALTER TABLE 功能。

除了上述功能外，SQLite 是一个完整的 SQL 系统，拥有完整的触发器，交易等等。

[回页首](#)

Android 集成了 SQLite 数据库

Android 在运行时 (run-time) 集成了 SQLite, 所以每个 Android 应用程序都可以使用 SQLite 数据库。对于熟悉 SQL 的开发人员来时, 在 Android 开发中使用 SQLite 相当简单。但是, 由于 JDBC 会消耗太多的系统资源, 所以 JDBC 对于手机这种内存受限设备来说并不合适。因此, Android 提供了一些新的 API 来使用 SQLite 数据库, Android 开发中, 程序员需要学使用这些 API。

数据库存储在 data/< 项目文件夹 >/databases/ 下。

[回页首](#)

Android 开发中使用 SQLite 数据库

Activites 可以通过 Content Provider 或者 Service 访问一个数据库。下面会详细讲解如果创建数据库, 添加数据和查询数据库。

创建数据库

Android 不自动提供数据库。在 Android 应用程序中使用 SQLite, 必须自己创建数据库, 然后创建表、索引, 填充数据。Android 提供了 SQLiteOpenHelper 帮助你创建一个数据库, 你只要继承 SQLiteOpenHelper 类, 就可以轻松的创建数据库。SQLiteOpenHelper 类根据开发应用程序的需要, 封装了创建和更新数据库使用的逻辑。SQLiteOpenHelper 的子类, 至少需要实现三个方法:

- 构造函数, 调用父类 SQLiteOpenHelper 的构造函数。这个方法需要四个参数: 上下文环境 (例如, 一个 Activity), 数据库名字, 一个可选的游标工厂 (通常是 Null), 一个代表你正在使用的数据库模型版本的整数。
- onCreate() 方法, 它需要一个 SQLiteDatabase 对象作为参数, 根据需要对这个对象填充表和初始化数据。
- onUpgrade() 方法, 它需要三个参数, 一个 SQLiteDatabase 对象, 一个旧的版本号和一个新的版本号, 这样你就可以清楚如何把一个数据库从旧的模型转变到新的模型。

下面示例代码展示了如何继承 SQLiteOpenHelper 创建数据库:

```
public class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context, String name, CursorFactory
cursorFactory, int version)
    {
        super(context, name, cursorFactory, version);
    }

    @Override
```

```

    public void onCreate(SQLiteDatabase db) {
        // TODO 创建数据库后，对数据库的操作
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        // TODO 更改数据库版本的操作
    }

    @Override
    public void onOpen(SQLiteDatabase db) {
        super.onOpen(db);
        // TODO 每次成功打开数据库后首先被执行
    }
}

```

接下来讨论具体如何创建表、插入数据、删除表等等。调用 `getReadableDatabase()` 或 `getWritableDatabase()` 方法，你可以得到 `SQLiteDatabase` 实例，具体调用那个方法，取决于你是否需要改变数据库的内容：

```

db=(new DatabaseHelper(getContext())).getWritableDatabase();
return (db == null) ? false : true;

```

上面这段代码会返回一个 `SQLiteDatabase` 类的实例，使用这个对象，你就可以查询或者修改数据库。

当你完成了对数据库的操作（例如你的 `Activity` 已经关闭），需要调用 `SQLiteDatabase` 的 `close()` 方法来释放掉数据库连接。

创建表和索引

为了创建表和索引，需要调用 `SQLiteDatabase` 的 `execSQL()` 方法来执行 DDL 语句。如果没有异常，这个方法没有返回值。

例如，你可以执行如下代码：

```

db.execSQL("CREATE TABLE mytable (_id INTEGER PRIMARY KEY
        AUTOINCREMENT, title TEXT, value REAL);");

```

这条语句会创建一个名为 mytable 的表，表有一个列名为 _id，并且是主键，这列的值是会自动增长的整数（例如，当你插入一行时，SQLite 会给这列自动赋值），另外还有两列：title（字符）和 value（浮点数）。SQLite 会自动为主键列创建索引。

通常情况下，第一次创建数据库时创建了表和索引。如果你不需要改变表的 schema，不需要删除表和索引。删除表和索引，需要使用 execSQL() 方法调用 DROP INDEX 和 DROP TABLE 语句。

给表添加数据

上面的代码，已经创建了数据库和表，现在需要给表添加数据。有两种方法可以给表添加数据。

像上面创建表一样，你可以使用 execSQL() 方法执行 INSERT, UPDATE, DELETE 等语句来更新表的数据。execSQL() 方法适用于所有不返回结果的 SQL 语句。例如：

```
db.execSQL("INSERT INTO widgets (name, inventory)"+
"VALUES (' Sprocket', 5)");
```

另一种方法是使用 SQLiteDatabase 对象的 insert(), update(), delete() 方法。这些方法把 SQL 语句的一部分作为参数。示例如下：

```
ContentValues cv=new ContentValues();
cv.put(Constants.TITLE, "example title");
cv.put(Constants.VALUE, SensorManager.GRAVITY_DEATH_STAR_I);
db.insert("mytable", getNullColumnHack(), cv);
```

update() 方法有四个参数，分别是表名，表示列名和值的 ContentValues 对象，可选的 WHERE 条件和可选的填充 WHERE 语句的字符串，这些字符串会替换 WHERE 条件中的“?”标记。update() 根据条件，更新指定列的值，所以用 execSQL() 方法可以达到同样的目的。

WHERE 条件和其参数和用过的其他 SQL APIs 类似。例如：

```
String[] parms=new String[] {"this is a string"};
db.update("widgets", replacements, "name=?", parms);
```

delete() 方法的使用和 update() 类似，使用表名，可选的 WHERE 条件和相应的填充 WHERE 条件的字符串。

查询数据库

类似 INSERT, UPDATE, DELETE, 有两种方法使用 SELECT 从 SQLite 数据库检索数据。

1 . 使用 `rawQuery()` 直接调用 SELECT 语句;

使用 `query()` 方法构建一个查询。

- **Raw Queries**

正如 API 名字, `rawQuery()` 是最简单的解决方法。通过这个方法你就可以调用 SQL SELECT 语句。例如:

```
Cursor c=db.rawQuery(
    "SELECT name FROM sqlite_master WHERE type='table' AND
    name='mytable'", null);
```

在上面例子中,我们查询 SQLite 系统表(`sqlite_master`)检查 `table` 表是否存在。返回值是一个 `cursor` 对象,这个对象的方法可以迭代查询结果。

如果查询是动态的,使用这个方法就会非常复杂。例如,当你需要查询的列在程序编译的时候不能确定,这时候使用 `query()` 方法会方便很多。

- **Regular Queries**

`query()` 方法用 SELECT 语句段构建查询。SELECT 语句内容作为 `query()` 方法的参数,比如:要查询的表名,要获取的字段名,WHERE 条件,包含可选的位置参数,去替代 WHERE 条件中位置参数的值,GROUP BY 条件,HAVING 条件。

除了表名,其他参数可以是 `null`。所以,以前的代码段可以可写成:

```
String[] columns={"ID", "inventory"};
String[] parms={"snicklefritz"};
Cursor result=db.query("widgets", columns, "name=?",parms, null, null,
null);
```

使用游标

不管你是否执行查询,都会返回一个 `Cursor`,这是 Android 的 SQLite 数据库游标,使用游标,你可以:

通过使用 `getCount()` 方法得到结果集中有多少记录；

通过 `moveToFirst()`, `moveToNext()`, 和 `isAfterLast()` 方法遍历所有记录；

通过 `getColumnNames()` 得到字段名；

通过 `getColumnIndex()` 转换成字段号；

通过 `getString()`, `getInt()` 等方法得到给定字段当前记录的值；

通过 `requery()` 方法重新执行查询得到游标；

通过 `close()` 方法释放游标资源；

例如，下面代码遍历 `mytable` 表

```
Cursor result=db.rawQuery("SELECT ID, name, inventory FROM mytable");
result.moveToFirst();
while (!result.isAfterLast()) {
    int id=result.getInt(0);
    String name=result.getString(1);
    int inventory=result.getInt(2);
    // do something useful with these
    result.moveToNext();
}
result.close();
```

[回页首](#)

在 Android 中使用 SQLite 数据库管理工具

在其他数据库上作开发，一般都使用工具来检查和处理数据库的内容，而不是仅仅使用数据库的 API。使用 Android 模拟器，有两种可供选择的方法来管理数据库。

首先，模拟器绑定了 `sqlite3` 控制台程序，可以使用 `adb shell` 命令来调用他。只要你进入了模拟器的 `shell`，在数据库的路径执行 `sqlite3` 命令就可以了。数据库文件一般存放在：

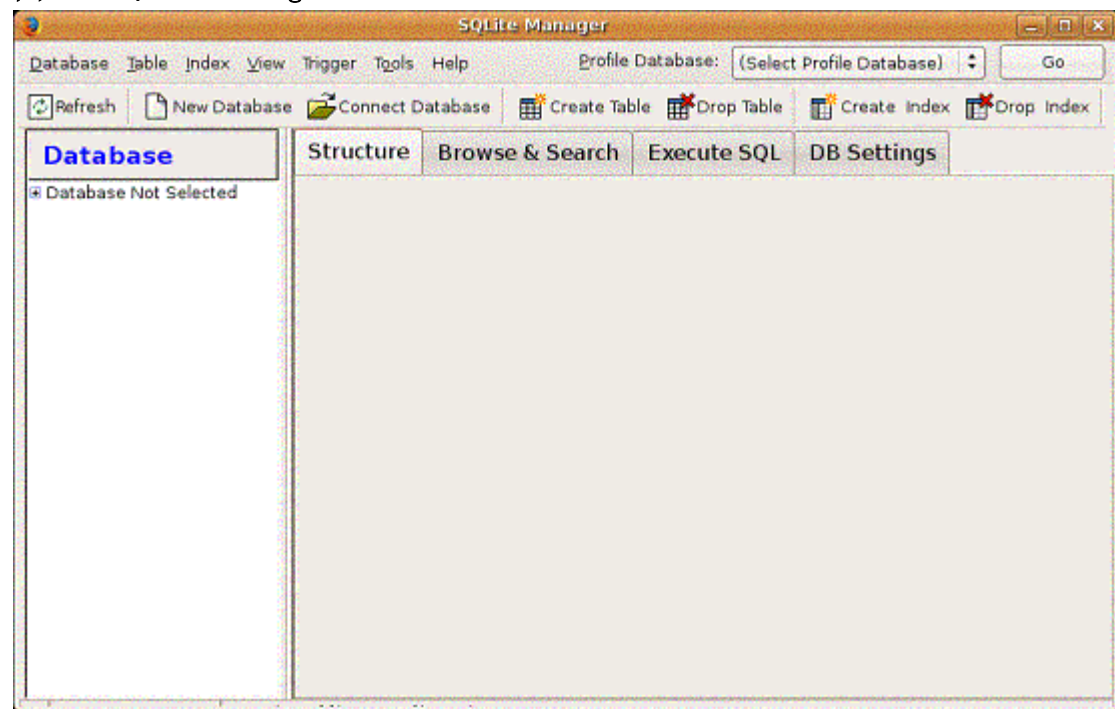
`/data/data/your.app.package/databases/your-db-name`

如果你喜欢使用更友好的工具，你可以把数据库拷贝到你的开发机上，使用 `SQLite-aware` 客户端来操作它。这样的话，你在一个数据库的拷贝上操作，如果你想要你的修改能反映到设备上，你需要把数据库备份回去。

把数据库从设备上考出来，你可以使用 `adb pull` 命令（或者在 IDE 上做相应操作）。存储一个修改过的数据库到设备上，使用 `adb push` 命令。

一个最方便的 SQLite 客户端是 Firefox SQLite Manager 扩展，它可以跨所有平台使用。

图 2. SQLite Manager



[回页首](#)

结束语

如果你想要开发 Android 应用程序，一定需要在 Android 上存储数据，使用 SQLite 数据库是一种非常好的选择。本文介绍了如何在 Android 应用程序中使用 SQLite 数据库，主要介绍了在 Android 应用程序中使用 SQLite 创建数据库和表、添加数据、更新和检索数据，还介绍了比较常用的 SQLite 管理工具，通过阅读本文，你可以在 Android 中轻松操作 SQLite 数据库。

参考资料

- 访问 developerWorks [Open source 专区](#) 获得丰富的 how-to 信息、工具和项目更新以及[最受欢迎的文章和教程](#)，帮助您用开放源码技术进行开发，并将它们与 IBM 产品结合使用。

- 随时关注 developerWorks [技术活动](#)和[网络广播](#)。