概述

Android 也提供了几种方法用来保存数据，使得这些数据即使在程序结束以后依然不会丢失。这些方法有：

- **文本文件**：可以保存在应用程序自己的目录下，安装的每个app都会在/data/data/目录下创建个文件夹，名字和应用程序中AndroidManifest.xml文件中的package一样。
- **SDcard保存**：
- **Preferences保存**：这也是一种经常使用的数据存储方法，因为它们对于用户而言是透明的，并且从应用安装的时候就存在了。
- **Assets保存**：用来存储一些只读数据，Assets是指那些在assets目录下的文件，这些文件在你将你的应用编译打包之前就要存在，并且可以在应用程序运行的时候被访问到。但有时候我们需要对保存的数据进行一些复杂的操作，或者数据量很大，超出了文本文件和Preference的性能能的范围，所以需要一些更加高效的方法来管理，从Android1.5开始，Android就自带SQLite数据库了。SQLite它是一个独立的，无需服务进程，支持事务处理，可以使用SQL语言的数据库。

SQLite的特性

1、 ACID事务

> ACID：
> 指数据库事务正确执行的四个基本要素的缩写。包含：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）。一个支持事务（Transaction）的数据库，必需要具有这四种特性，否则在事务过程（Transaction processing）当中无法保证数据的正确性，交易过程极可能达不到交易方的要求。

2、 零配置 – 无需安装和管理配置

3、储存在单一磁盘文件中的一个完整的数据库

4、数据库文件可以在不同字节顺序的机器间自由的共享

5、支持数据库大小至2TB

6、 足够小, 大致3万行C代码, 250K

7、比一些流行的数据库在大部分普通数据库操作要快

8、简单, 轻松的API

9、 包含TCL绑定, 同时通过Wrapper支持其他语言的绑定

10、良好注释的源代码, 并且有着90%以上的测试覆盖率

11、 独立: 没有额外依赖

12、 Source完全的Open, 你可以用于任何用途, 包括出售它

13、支持多种开发语言，C，PHP，Perl，Java，ASP.NET，Python

Android 中使用 SQLite

Activites 可以通过 Content Provider 或者 Service 访问一个数据库。

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 类根据开发应用程序的需要，封装创建和更新数据库使用的逻辑就行了。

SQLiteOpenHelper 的子类，至少需要实现三个方法：

```
public class DatabaseHelper extends SQLiteOpenHelper {

    /**
     * @param context  上下文环境（例如，一个 Activity）
     * @param name   数据库名字
     * @param factory  一个可选的游标工厂（通常是 Null）
     * @param version  数据库模型版本的整数
     *
     * 会调用父类 SQLiteOpenHelper的构造函数
     */
    public DatabaseHelper(Context context, String name, CursorFactory factory,
int version) {
        super(context, name, factory, version);

    }

    /**
     *  在数据库第一次创建的时候会调用这个方法
     *
     *根据需要对传入的SQLiteDatabase 对象填充表和初始化数据。
     */
    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    /**
     * 当数据库需要修改的时候（两个数据库版本不同），Android系统会主动的调用这个方法。
     * 一般我们在这个方法里边删除数据库表，并建立新的数据库表.
     */
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        //三个参数，一个 SQLiteDatabase 对象，一个旧的版本号和一个新的版本号

    }

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

继承SQLiteOpenHelper之后就拥有了以下两个方法：

- getReadableDatabase()　创建或者打开一个查询数据库
- getWritableDatabase()　创建或者打开一个可写数据库

```
DatabaseHelper database = new DatabaseHelper(context);//传入一个上下文参数
SQLiteDatabase db = null;
db = database.getWritableDatabase();
```

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。

SQLiteDatabase类为我们提供了很多种方法，而较常用的方法如下：

> (int) delete(String table,String whereClause,String[] whereArgs)

删除数据行

> (long) insert(String table,String nullColumnHack,ContentValues values)

添加数据行

> (int) update(String table, ContentValues values, String whereClause, String[] whereArgs)

更新数据行

> (void) execSQL(String sql)

执行一个SQL语句，可以是一个select或其他的sql语句

> (void) close()

关闭数据库

> (Cursor) query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)

查询指定的数据表返回一个带游标的数据集。

各参数说明：

- table：表名称
- colums：列名称数组
- selection：条件子句，相当于where
- selectionArgs：条件语句的参数数组
- groupBy：分组
- having：分组条件
- orderBy：排序类
- limit：分页查询的限制
- Cursor：返回值，相当于结果集ResultSet

> (Cursor) rawQuery(String sql, String[] selectionArgs)

运行一个预置的SQL语句，返回带游标的数据集（与上面的语句最大的区别就是防止SQL注入）

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 user(_id INTEGER PRIMARY KEY
       AUTOINCREMENT, username TEXT, password TEXT);");
```

这条语句会创建一个名为 user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。另外还有两列：username(字符) 和 password(字符)。 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。要 删除表和索引，需要使用 execSQL() 方法调用 DROP INDEX 和 DROP TABLE 语句。

添加数据

有两种方法可以给表添加数据。

①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：

```
String sql = "insert into user(username,password) values ('finch','123456');//插入操作的SQL语句
db.execSQL(sql);//执行SQL语句
```

②使用 SQLiteDatabase 对象的 insert()。

```
ContentValues cv = new ContentValues();
cv.put("username","finch");//添加用户名
cv.put("password","123456"); //添加密码
db.insert("user",null,cv);//执行插入操作
```

更新数据（修改）

①使用SQLiteDatabase 对象的 update()方法。

```
ContentValues cv = new ContentValues();
cv.put("password","654321");//添加要更改的字段及内容
String whereClause = "username=?";//修改条件
String[] whereArgs = {"finch"};//修改条件的参数
db.update("user",cv,whereClause,whereArgs);//执行修改
```

该方法有四个参数：

表名；

列名和值的 ContentValues 对象；

可选的 WHERE 条件；

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.

②使用execSQL方式的实现

```
String sql = "update [user] set password = '654321' where username="finch";//修改的SQL语句
db.execSQL(sql);//执行修改
```

删除数据

①使用SQLiteDatabase 对象的delete()方法。

```
String whereClause = "username=?";//删除的条件
String[] whereArgs = {"finch"};//删除的条件参数
db.delete("user",whereClause,whereArgs);//执行删除
```

②使用execSQL方式的实现

```
String sql = "delete from user where username="finch";//删除操作的SQL语句
db.execSQL(sql);//执行删除操作
```

查询数据

①使用 rawQuery() 直接调用 SELECT 语句

```
Cursor c = db.rawQuery("select * from user where username=?",new Stirng[]
{"finch"});

if(cursor.moveToFirst()) {
    String password = c.getString(c.getColumnIndex("password"));
}
```

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。

②通过query实现查询

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。所以代码可写成：

```
Cursor c = db.query("user",null,null,null,null,null,null);//查询并获得游标
if(c.moveToFirst()){//判断游标是否为空
    for(int i=0;i<c.getCount();i++){
c.move(i);//移动到指定记录
String username = c.getString(c.getColumnIndex("username");
String password = c.getString(c.getColumnIndex("password"));
    }
}
```

使用游标

不管你如何执行查询，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 user表:

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