# **Android ：SQLlite数据库 使用手册**

[](https://www.jianshu.com/u/383970bef0a0)

[Carson\_Ho](https://www.jianshu.com/u/383970bef0a0)  关注

 1.4 2016.11.18 19:58\* 字数 851 阅读 21386评论 11喜欢 151赞赏 1



# **前言**

* 在Android开发中，主要的数据存储有5种，具体如下

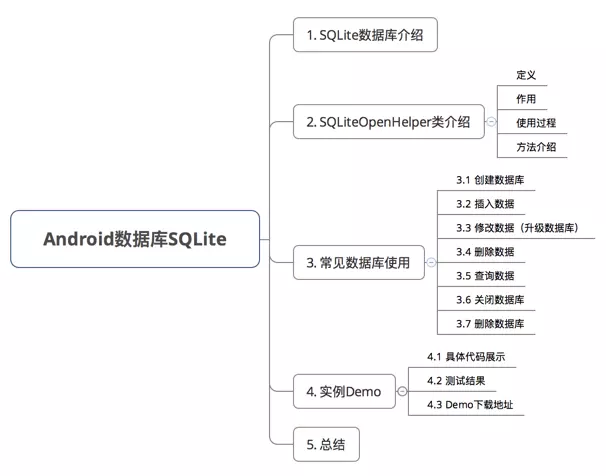


示意图

其中，SQLite数据库的存储方式 非常常用

* 今天我将带大家全面了解关于Android SQLite数据库的操作（增、删、查、改）

# **目录**



目录

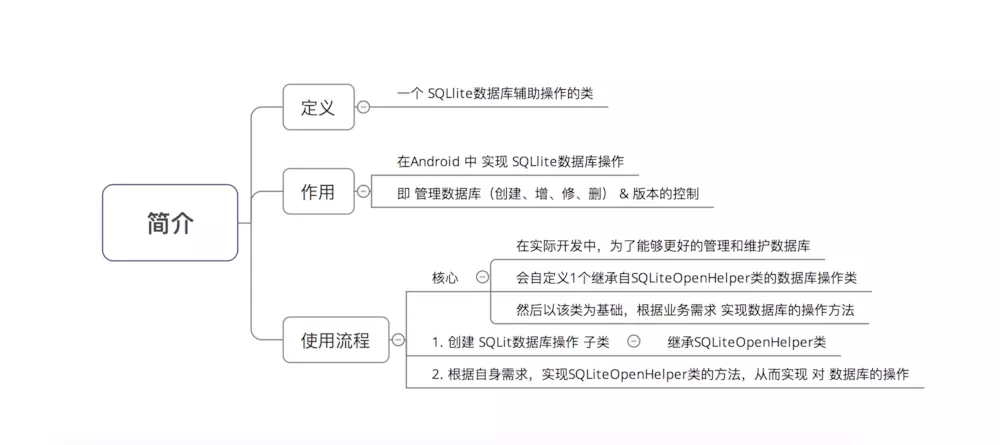
# **1. SQLlite数据库 简介**



示意图

# **2. SQLiteOpenHelper类**

### **2.1 简介**



示意图

### **2.2 SQLiteOpenHelper类 常用方法**

/\*\*

\* 创建数据库

\*/

// 1. 创建 or 打开 可读/写的数据库（通过 返回的SQLiteDatabase对象 进行操作）

getWritableDatabase（）

// 2. 创建 or 打开 可读的数据库（通过 返回的SQLiteDatabase对象 进行操作）

getReadableDatabase（）

// 3. 数据库第1次创建时 则会调用，即 第1次调用 getWritableDatabase（） / getReadableDatabase（）时调用

// 在继承SQLiteOpenHelper类的子类中复写

onCreate(SQLiteDatabase db)

// 4. 数据库升级时自动调用

// 在继承SQLiteOpenHelper类的子类中复写

onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)

// 5. 关闭数据库

close（）

/\*\*

\* 数据库操作（增、删、减、查）

\*/

// 1. 查询数据

(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 = 防止SQL注入）

// 2. 删除数据行

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

// 3. 添加数据行

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

// 4. 更新数据行

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

// 5. 执行一个SQL语句，可以是一个select or 其他sql语句

// 即 直接使用String类型传入sql语句 & 执行

(void) execSQL(String sql)

# **3. 具体使用**

* 使用步骤 = 自定义数据库子类（继承SQLiteOpenHelper类）、创建数据库 & 操作数据库（增、删、查、改）
* 下面，我将逐一介绍每个步骤

### **3.1 自定义数据库子类（继承 SQLiteOpenHelper 类）**

/\*\*

\* 创建数据库子类，继承自SQLiteOpenHelper类

\* 需 复写 onCreat（）、onUpgrade（）

\*/ public class DatabaseHelper extends SQLiteOpenHelper {

// 数据库版本号

private static Integer Version = 1;

/\*\*

\* 构造函数

\* 在SQLiteOpenHelper的子类中，必须有该构造函数

\*/

public DatabaseHelper(Context context, String name, SQLiteDatabase.CursorFactory factory,

int version) {

// 参数说明

// context：上下文对象

// name：数据库名称

// param：一个可选的游标工厂（通常是 Null）

// version：当前数据库的版本，值必须是整数并且是递增的状态

// 必须通过super调用父类的构造函数

super(context, name, factory, version);

}

/\*\*

\* 复写onCreate（）

\* 调用时刻：当数据库第1次创建时调用

\* 作用：创建数据库 表 & 初始化数据

\* SQLite数据库创建支持的数据类型： 整型数据、字符串类型、日期类型、二进制

\*/

@Override

public void onCreate(SQLiteDatabase db) {

// 创建数据库1张表

// 通过execSQL（）执行SQL语句（此处创建了1个名为person的表）

String sql = "create table person(id integer primary key autoincrement,name varchar(64),address varchar(64))";

db.execSQL(sql);

// 注：数据库实际上是没被创建 / 打开的（因该方法还没调用）

// 直到getWritableDatabase() / getReadableDatabase() 第一次被调用时才会进行创建 / 打开

}

/\*\*

\* 复写onUpgrade（）

\* 调用时刻：当数据库升级时则自动调用（即 数据库版本 发生变化时）

\* 作用：更新数据库表结构

\* 注：创建SQLiteOpenHelper子类对象时,必须传入一个version参数，该参数 = 当前数据库版本, 若该版本高于之前版本, 就调用onUpgrade()

\*/

@Override

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

// 参数说明：

// db ： 数据库

// oldVersion ： 旧版本数据库

// newVersion ： 新版本数据库

// 使用 SQL的ALTER语句

String sql = "alter table person add sex varchar(8)";

db.execSQL(sql);

}

}

### **3.2 创建数据库：getWritableDatabase（）、getReadableDatabase（）**

// 步骤1：创建DatabaseHelper对象

// 注：此时还未创建数据库

SQLiteOpenHelper dbHelper = new DatabaseHelper(SQLiteActivity.this,"test\_carson");

// 步骤2：真正创建 / 打开数据库

SQLiteDatabase sqliteDatabase = dbHelper.getWritableDatabase(); // 创建 or 打开 可读/写的数据库

SQLiteDatabase sqliteDatabase = dbHelper.getReadableDatabase(); // 创建 or 打开 可读的数据库

****注：当需操作数据库时，都必须先创建数据库对象 & 创建 / 打开数据库。****

1. 对于操作 = “增、删、改（更新）”，需获得 可"读 / 写"的权限：getWritableDatabase()
2. 对于操作 = “查询”，需获得 可"读 "的权限getReadableDatabase()

### **3.3 操作数据库（增、删、查、改）**

/\*\*

\* 1. 创建 & 打开数据库

\*/

// a. 创建DatabaseHelper对象

// 注：一定要传入最新的数据库版本号

SQLiteOpenHelper dbHelper = new DatabaseHelper(SQLiteActivity.this,"test\_carson"，2);

// b.创建 or 打开 可读/写的数据库

SQLiteDatabase sqliteDatabase = dbHelper.getWritableDatabase();

/\*\*

\* 操作1：插入数据 = insert()

\*/

// a. 创建ContentValues对象

ContentValues values = new ContentValues();

// b. 向该对象中插入键值对

values.put("id", 1);

values.put("name", "carson");

//其中，key = 列名，value = 插入的值

//注：ContentValues内部实现 = HashMap，区别在于：ContenValues Key只能是String类型，Value可存储基本类型数据 & String类型

// c. 插入数据到数据库当中：insert()

sqliteDatabase.insert("user", null, values);

// 参数1：要操作的表名称

// 参数2：SQl不允许一个空列，若ContentValues是空，那么这一列被明确的指明为NULL值

// 参数3：ContentValues对象

// 注：也可采用SQL语句插入

String sql = "insert into user (id,name) values (1,'carson')";

db.execSQL(sql) ；

/\*\*

\* 操作2：修改数据 = update（）

\*/

// a. 创建一个ContentValues对象

ContentValues values = new ContentValues();

values.put("name", "zhangsan");

// b. 调用update方法修改数据库：将id=1 修改成 name = zhangsan

sqliteDatabase.update("user", values, "id=?", new String[] { "1" });

// 参数1：表名(String)

// 参数2：需修改的ContentValues对象

// 参数3：WHERE表达式（String），需数据更新的行； 若该参数为 null, 就会修改所有行；？号是占位符

// 参数4：WHERE选择语句的参数(String[]), 逐个替换 WHERE表达式中 的“？”占位符;

// 注：调用完upgrate（）后，则会回调 数据库子类的onUpgrade()

// 注：也可采用SQL语句修改

String sql = "update [user] set name = 'zhangsan' where id="1";

db.execSQL(sql);

/\*\*

\* 操作3：删除数据 = delete()

\*/

// 删除 id = 1的数据

sqliteDatabase.delete("user", "id=?", new String[]{"1"});

// 参数1：表名(String)

// 参数2：WHERE表达式（String），需删除数据的行； 若该参数为 null, 就会删除所有行；？号是占位符

// 参数3：WHERE选择语句的参数(String[]), 逐个替换 WHERE表达式中 的“？”占位符;

// 注：也可采用SQL语句修改

String sql = "delete from user where id="1"；

db.execSQL(sql);

/\*\*

\* 操作4：查询数据1 = rawQuery()

\* 直接调用 SELECT 语句

\*/

Cursor c = db.rawQuery("select \* from user where id=?",new Stirng[]{"1"});

// 返回值一个 cursor 对象

// 通过游标的方法可迭代查询结果

if(cursor.moveToFirst()) {

String password = c.getString(c.getColumnIndex("password"));

}

//Cursor对象常用方法如下：

c.move(int offset); //以当前位置为参考,移动到指定行

c.moveToFirst(); //移动到第一行

c.moveToLast(); //移动到最后一行

c.moveToPosition(int position); //移动到指定行

c.moveToPrevious(); //移动到前一行

c.moveToNext(); //移动到下一行

c.isFirst(); //是否指向第一条

c.isLast(); //是否指向最后一条

c.isBeforeFirst(); //是否指向第一条之前

c.isAfterLast(); //是否指向最后一条之后

c.isNull(int columnIndex); //指定列是否为空(列基数为0)

c.isClosed(); //游标是否已关闭

c.getCount(); //总数据项数

c.getPosition(); //返回当前游标所指向的行数

c.getColumnIndex(String columnName);//返回某列名对应的列索引值

c.getString(int columnIndex); //返回当前行指定列的值

// 通过游标遍历1个名为user的表

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();

// 若查询是动态的，使用该方法会复杂。此时使用 query() 会方便很多

// 注：无法使用SQL语句，即db.execSQL(sql);

/\*\*

\* 操作4：查询数据2 = query()

\* 直接调用 SELECT 语句

\*/

// 方法说明

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, String groupBy, String having, String orderBy, String limit);

// 参数说明

// table：要操作的表

// columns：查询的列所有名称集

// selection：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

// 所有方法将返回一个Cursor对象，代表数据集的游标

// 具体使用

Cursor cursor = sqliteDatabase.query("user", new String[] { "id","name" }, "id=?", new String[] { "1" }, null, null, null);

// 参数1：（String）表名

// 参数2：（String[]）要查询的列名

// 参数3：（String）查询条件

// 参数4：（String[]）查询条件的参数

// 参数5：（String）对查询的结果进行分组

// 参数6：（String）对分组的结果进行限制

// 参数7：（String）对查询的结果进行排序

// 注：无法使用SQL语句，即db.execSQL(sql);

/\*\*

\* 操作5：关闭数据库 = close()

\* 注：完成数据库操作后，记得调用close（）关闭数据库，从而释放数据库的连接

\*/

sqliteDatabase.close();

/\*\*

\* 操作6：删除数据库 = deleteDatabase（）

\*/

// 删除 名为person的数据库

deleteDatabase("test.db");

### **3.4 特别注意**

* 除了“查询”，所有的数据库操作都可使用 SQL 语句，即使用下列方法代替：

db.executeSQL(String sql);

db.executeSQL(String sql, Object[] bindArgs);// sql语句中使用占位符，参数2 = 实际的参数集

* 使用建议：都使用SQL语句，因为：SQL语句 通用 & 简单

原生的方法insert()、delete()等方法具备多个参数，使用复杂

# **4. 实例Demo**

* Demo描述：创建数据库 & 对数据库进行操作（增、删、查、改）
* 实现步骤：
  1. 实现子类继承SQLiteOpenHelper类（复写onCreat()、onUpgrade()）
  2. 对数据库进行各种操作（增、删、查、改）

### **4.1 具体代码如下：**

建议先下载Demo再进行阅读：[Carson的Github:DataBase\_Demo](https://links.jianshu.com/go?to=https://github.com/Carson-Ho/DataBase_Demo" \t "/Users/wjw/Documents\\x/_blank)

****步骤1：实现子类继承SQLiteOpenHelper类（复写onCreat()、onUpgrade()）****

MySQLiteOpenHelper.java

package scut.carson\_ho.database\_demo;

import android.content.Context;import android.database.sqlite.SQLiteDatabase;import android.database.sqlite.SQLiteOpenHelper;

/\*\*

\* Created by Carson\_Ho on 16/11/18.

\*/public class MySQLiteOpenHelper extends SQLiteOpenHelper {

//数据库版本号

private static Integer Version = 1;

//在SQLiteOpenHelper的子类当中，必须有该构造函数

public MySQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory,

int version) {

//必须通过super调用父类当中的构造函数

super(context, name, factory, version);

}

//参数说明

//context:上下文对象

//name:数据库名称

//param:factory

//version:当前数据库的版本，值必须是整数并且是递增的状态

public MySQLiteOpenHelper(Context context,String name,int version)

{

this(context,name,null,version);

}

public MySQLiteOpenHelper(Context context,String name)

{

this(context, name, Version);

}

//当数据库创建的时候被调用

@Override

public void onCreate(SQLiteDatabase db) {

System.out.println("创建数据库和表");

//创建了数据库并创建一个叫records的表

//SQLite数据创建支持的数据类型： 整型数据，字符串类型，日期类型，二进制的数据类型

String sql = "create table user(id int primary key,name varchar(200))";

//execSQL用于执行SQL语句

//完成数据库的创建

db.execSQL(sql);

//数据库实际上是没有被创建或者打开的，直到getWritableDatabase() 或者 getReadableDatabase() 方法中的一个被调用时才会进行创建或者打开

}

//数据库升级时调用

//如果DATABASE\_VERSION值被改为2,系统发现现有数据库版本不同,即会调用onUpgrade（）方法

@Override

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

System.out.println("更新数据库版本为:"+newVersion);

}

}

****步骤2：在MainActivity里实现需要进行的数据库操作（增、删、查、改）****

MainActivity.java

package scut.carson\_ho.database\_demo;

import android.content.ContentValues;import android.database.Cursor;import android.database.sqlite.SQLiteDatabase;import android.os.Bundle;import android.support.v7.app.AppCompatActivity;import android.view.View;import android.widget.Button;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

private Button instablish;

private Button insert;

private Button upgrade;

private Button modify;

private Button delete;

private Button query;

private Button delete\_database;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//绑定按钮

instablish = (Button) findViewById(R.id.instablish);

insert = (Button) findViewById(R.id.insert);

upgrade = (Button) findViewById(R.id.upgrade);

modify = (Button) findViewById(R.id.modify);

delete = (Button) findViewById(R.id.delete);

query = (Button) findViewById(R.id.query);

delete\_database = (Button) findViewById(R.id.delete\_database);

//设置监听器

instablish.setOnClickListener(this);

insert.setOnClickListener(this);

upgrade.setOnClickListener(this);

modify.setOnClickListener(this);

delete.setOnClickListener(this);

query.setOnClickListener(this);

delete\_database.setOnClickListener(this);

}

//设置每个按钮对数据库的操作进行控制

@Override

public void onClick(View v) {

switch (v.getId()) {

//点击创建数据库库

case R.id.instablish:

// 创建SQLiteOpenHelper子类对象

MySQLiteOpenHelper dbHelper = new MySQLiteOpenHelper(this,"test\_carson");

//数据库实际上是没有被创建或者打开的，直到getWritableDatabase() 或者 getReadableDatabase() 方法中的一个被调用时才会进行创建或者打开

SQLiteDatabase sqliteDatabase = dbHelper.getWritableDatabase();

// SQLiteDatabase sqliteDatabase = dbHelper.getReadbleDatabase();

break;

//点击更新数据

case R.id.upgrade:

// 创建SQLiteOpenHelper子类对象

MySQLiteOpenHelper dbHelper\_upgrade = new MySQLiteOpenHelper(this,"test\_carson",2);

// 调用getWritableDatabase()方法创建或打开一个可以读的数据库

SQLiteDatabase sqliteDatabase\_upgrade = dbHelper\_upgrade.getWritableDatabase();

// SQLiteDatabase sqliteDatabase = dbHelper.getReadbleDatabase();

break;

//点击插入数据到数据库

case R.id.insert:

System.out.println("插入数据");

// 创建SQLiteOpenHelper子类对象

////注意，一定要传入最新的数据库版本号

MySQLiteOpenHelper dbHelper1 = new MySQLiteOpenHelper(this,"test\_carson",2);

// 调用getWritableDatabase()方法创建或打开一个可以读的数据库

SQLiteDatabase sqliteDatabase1 = dbHelper1.getWritableDatabase();

// 创建ContentValues对象

ContentValues values1 = new ContentValues();

// 向该对象中插入键值对

values1.put("id", 1);

values1.put("name", "carson");

// 调用insert()方法将数据插入到数据库当中

sqliteDatabase1.insert("user", null, values1);

// sqliteDatabase.execSQL("insert into user (id,name) values (1,'carson')");

//关闭数据库

sqliteDatabase1.close();

break;

//点击查询数据库

case R.id.query:

System.out.println("查询数据");

// 创建DatabaseHelper对象

MySQLiteOpenHelper dbHelper4 = new MySQLiteOpenHelper(MainActivity.this,"test\_carson",2);

// 调用getWritableDatabase()方法创建或打开一个可以读的数据库

SQLiteDatabase sqliteDatabase4 = dbHelper4.getReadableDatabase();

// 调用SQLiteDatabase对象的query方法进行查询

// 返回一个Cursor对象：由数据库查询返回的结果集对象

Cursor cursor = sqliteDatabase4.query("user", new String[] { "id",

"name" }, "id=?", new String[] { "1" }, null, null, null);

String id = null;

String name = null;

//将光标移动到下一行，从而判断该结果集是否还有下一条数据

//如果有则返回true，没有则返回false

while (cursor.moveToNext()) {

id = cursor.getString(cursor.getColumnIndex("id"));

name = cursor.getString(cursor.getColumnIndex("name"));

//输出查询结果

System.out.println("查询到的数据是:"+"id: "+id+" "+"name: "+name);

}

//关闭数据库

sqliteDatabase4.close();

break;

//点击修改数据

case R.id.modify:

System.out.println("修改数据");

// 创建一个DatabaseHelper对象

// 将数据库的版本升级为2

// 传入版本号为2，大于旧版本（1），所以会调用onUpgrade()升级数据库

MySQLiteOpenHelper dbHelper2 = new MySQLiteOpenHelper(MainActivity.this,"test\_carson", 2);

// 调用getWritableDatabase()得到一个可写的SQLiteDatabase对象

SQLiteDatabase sqliteDatabase2 = dbHelper2.getWritableDatabase();

// 创建一个ContentValues对象

ContentValues values2 = new ContentValues();

values2.put("name", "zhangsan");

// 调用update方法修改数据库

sqliteDatabase2.update("user", values2, "id=?", new String[]{"1"});

//关闭数据库

sqliteDatabase2.close();

break;

//点击删除数据

case R.id.delete:

System.out.println("删除数据");

// 创建DatabaseHelper对象

MySQLiteOpenHelper dbHelper3 = new MySQLiteOpenHelper(MainActivity.this,"test\_carson",2);

// 调用getWritableDatabase()方法创建或打开一个可以读的数据库

SQLiteDatabase sqliteDatabase3 = dbHelper3.getWritableDatabase();

//删除数据

sqliteDatabase3.delete("user", "id=?", new String[]{"1"});

//关闭数据库

sqliteDatabase3.close();

break;

//点击删除数据库

case R.id.delete\_database:

System.out.println("删除数据库");

MySQLiteOpenHelper dbHelper5 = new MySQLiteOpenHelper(MainActivity.this,

"test\_carson",2);

// 调用getReadableDatabase()方法创建或打开一个可以读的数据库

SQLiteDatabase sqliteDatabase5 = dbHelper5.getReadableDatabase();

//删除名为test.db数据库

deleteDatabase("test\_carson");

break;

default:

break;

}

}

}

****步骤3：在主布局文件设置按钮控制数据库的操作****  
activity\_main.xml

<?xml version="1.0" encoding="utf-8"?><RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context="scut.carson\_ho.database\_demo.MainActivity">

<Button

android:id="@+id/instablish"

android:layout\_centerHorizontal="true"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="创建数据库"/>

<Button

android:id="@+id/upgrade"

android:layout\_below="@+id/instablish"

android:layout\_centerHorizontal="true"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="更新数据库"/>

<Button

android:id="@+id/insert"

android:layout\_below="@+id/upgrade"

android:layout\_centerHorizontal="true"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="插入数据"/>

<Button

android:id="@+id/modify"

android:layout\_centerHorizontal="true"

android:layout\_below="@+id/insert"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="修改数据"/>

<Button

android:id="@+id/query"

android:layout\_centerHorizontal="true"

android:layout\_below="@+id/modify"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="查询数据"/>

<Button

android:id="@+id/delete"

android:layout\_centerHorizontal="true"

android:layout\_below="@+id/query"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="删除数据"/>

<Button

android:id="@+id/delete\_database"

android:layout\_centerHorizontal="true"

android:layout\_below="@+id/delete"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="删除数据库"/>

</RelativeLayout>

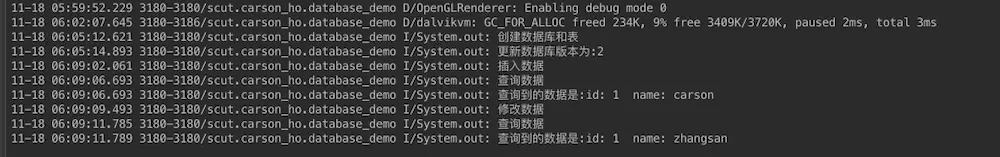
### **4.2 测试结果**

界面显示



界面

按顺序点击：创建 - 更新 - 插入 - 查询 - 修改 - 查询 - 删除数据



测试结果

### **4.3 Demo地址**

[Carson的Github地址：DataBase\_Demo](https://links.jianshu.com/go?to=https://github.com/Carson-Ho/DataBase_Demo" \t "/Users/wjw/Documents\\x/_blank)

# **5. 总结**

* 本文全面介绍了 Android SQLite数据库中所有知识
* 接下来，我会继续介绍Android中的常用数据库 的库，如Realm、GreenDao3.0 和 腾讯刚开源的WCDB，感兴趣的同学可以继续关注本人运营的Wechat Public Account：
* [我想给你们介绍一个与众不同的Android微信公众号（福利回赠）](https://www.jianshu.com/p/2e92908af6ec" \t "/Users/wjw/Documents\\x/_blank)
* [我想邀请您和我一起写Android（福利回赠）](https://www.jianshu.com/p/2c5d57fb054d" \t "/Users/wjw/Documents\\x/_blank)

#### **请点赞！因为你们的赞同/鼓励是我写作的最大动力！**

****相关文章阅读****  
[Android开发：最全面、最易懂的Android屏幕适配解决方案](https://www.jianshu.com/p/ec5a1a30694b" \t "/Users/wjw/Documents\\x/_blank)  
[Android事件分发机制详解：史上最全面、最易懂](https://www.jianshu.com/p/38015afcdb58" \t "/Users/wjw/Documents\\x/_blank)  
[Android开发：史上最全的Android消息推送解决方案](https://www.jianshu.com/p/b61a49e0279f" \t "/Users/wjw/Documents\\x/_blank)  
[Android开发：最全面、最易懂的Webview详解](https://www.jianshu.com/p/3c94ae673e2a" \t "/Users/wjw/Documents\\x/_blank)  
[Android开发：JSON简介及最全面解析方法!](https://www.jianshu.com/p/b87fee2f7a23" \t "/Users/wjw/Documents\\x/_blank)  
[Android四大组件：Service服务史上最全面解析](https://www.jianshu.com/p/d963c55c3ab9" \t "/Users/wjw/Documents\\x/_blank)  
[Android四大组件：BroadcastReceiver史上最全面解析](https://www.jianshu.com/p/ca3d87a4cdf3" \t "/Users/wjw/Documents\\x/_blank)

### **欢迎关注[Carson\_Ho](https://www.jianshu.com/users/383970bef0a0/latest_articles" \t "/Users/wjw/Documents\\x/_blank)的简书！**

不定期分享关于****安卓开发****的干货，追求****短、平、快****，但****却不缺深度****。

