SQLiteDatabase

一、实验目的

1. 掌握 Android 数据存储的方法
2. 学习 SQliteDatabase
3. 掌握 Contract 类
4. 掌握 SQLiteOpenHelper

二、实验条件

1.PC 机

2.JDK（建议安装 JDK8 及其以上版本）、Android Studio

三、实验原理

Android 提供了三种数据存储方式，文件存储、SharedPreferences 存储和数据库存储。Android 内置了对开源数据库 SQLite 的支持。它提供了一个名为 SQLiteDatabase 的类，该类封装了一些操作 SQLite 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）。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()方法可以执行 select 语句。SQLiteDatabase 还专门提供了对应于添加、删除、更新、查询的操作方法： insert()、delete()、update()和 query() 。这些方法实际上是给那些不太了解 SQL 语法的菜鸟使用的，对于熟悉 SQL 语法的程序员而言，直接使用 execSQL()和 rawQuery()方法执行 SQL 语句就能完成数据的添加、删除、更新、查询操作。

SQL 语法

(1)创建/删除表

```
SQLitebase db;
String sql="Create table "+TABLE_NAME+"("+FIELD_ID+" integer primary key autoincrement," +FIELD_TITLE+" text );";
db.execSQL(sql);
String sql=" DROP TABLE IF EXISTS "+TABLE_NAME;
db.execSQL(sql);
```

(2)插入操作

```
db.execSQL("insert into person(name, age) values('林计钦', 24)");
```

SQLiteOpenHelper

如果应用使用到了 SQLite 数据库，在用户初次使用软件时，需要创建应用使用到的数据库表结构及添加一些初始化记录，另外在软件升级的时候，也需要对数据表结构进行更新。在 Android 系统，为我们提供了一个名为 SQLiteOpenHelper 的类，该类用于对数据库版本进行管理，该类是一个抽象类，必须继承它才能使用。使用它必须实现它的 onCreate(SQLiteDatabase)，onUpgrade(SQLiteDatabase, int, int)方法

onCreate：当数据库第一次被建立的时候被执行，例如创建表,初始化数据等。

onUpgrade：当数据库需要被更新的时候执行，例如删除久表，创建新表。

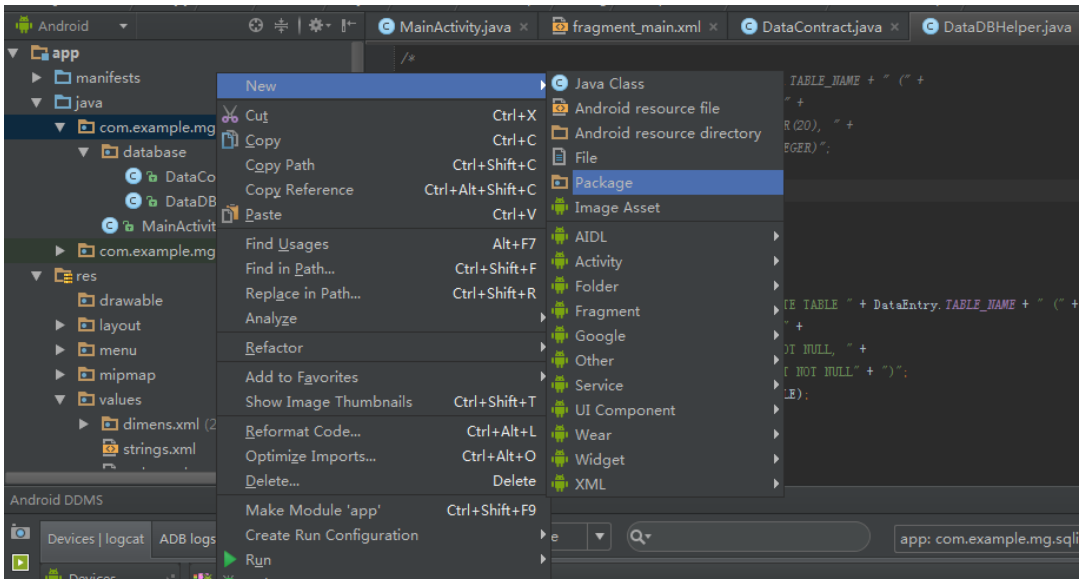
Contract:

SQL 数据库的主要原则之一是构架：一个关于这个数据库是如何组织的一个正式声明。构架构架反映在创建 SQL 数据库的语句中。你可能会发现创建一个同伴类（companion class）很有用，同伴类同时被称作合约类（contract class），其中明确规定了你的构架的布局，以一种系统且自说明的方式。一个合约类（contract class）是一个常量的容器，这些常量定义了 URI，表的名字，列的名字。合约类允许你在同一个包的其他类中使用这些名字常量。这就允许了你在一个地方改变列名，而同时把它传播到代码的其他地方去。

四、实验步骤

1. 新建工程（blank activity with fragment）。

2. 创建一个 package，命名为 database 来存储数据库的代码



3. 在 database 里，创建 DataContract.java

我们使用 contract 来定义数据库的结构。一种较好的组织方法是用一个类定义一些全局变量，再在这个类内，为每一个数据库的表创建内部类

```
// 注意，这边的变量值不能带有空格！比如 public static final String COULUMN_PHONE_NUMBER = "phone number";就是错误的。
public class DataContract {
    public static final class DataEntry implements BaseColumns{ //表的定义
        public static final String TABLE_NAME = "datademo";
        public static final String COULUMN_USER_NAME = "name";
        public static final String COULUMN_PHONE_NUMBER = "phonenumber";
    }
}
```

4. 创建工具类 DataDBHelper.java

```
import com.example.mg.sqlitedemo.database.DataContract.DataEntry;
```

```
/**
 * Created by MG on 4/8/2015.
 */
public class DataDBHelper extends SQLiteOpenHelper{
    private static final int DATABASE_VERSION = 1; //数据库的版本号（每次更新数据库结构时更改）
    private static final String DATABASE_NAME = "DataDemo";

    public DataDBHelper(Context context)
    {
        super(context,DATABASE_NAME,null,DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

        final String SQL_CREATE_DATABASE_TABLE = "CREATE TABLE " + DataEntry.TABLE_NAME + " (" +
                DataEntry._ID + " INTEGER PRIMARY KEY," +
                DataEntry.COULUMN_USER_NAME + " TEXT NOT NULL, " +
                DataEntry.COULUMN_PHONE_NUMBER + " TEXT NOT NULL" +
")";
        sqLiteDatabase.execSQL(SQL_CREATE_DATABASE_TABLE);


    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldversion, int newversion) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + DataEntry.TABLE_NAME);
    }
}
```

5．在 MainActivity.java 中

```
import com.example.mg.sqlitedemo.database.DataContract;
import com.example.mg.sqlitedemo.database.DataDBHelper;


public class MainActivity extends ActionBarActivity {
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (savedInstanceState == null) {
                getSupportFragmentManager().beginTransaction()
                                .add(R.id.container, new PlaceholderFragment())
                                .commit();
        }
}


@Override
public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
                return true;
        }

        return super.onOptionsItemSelected(item);
}

/**
 * A placeholder fragment containing a simple view.
 */
public static class PlaceholderFragment extends Fragment {


        public static DataDBHelper mDataDBHelper;
```

```
        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {

            View rootView = inflater.inflate(R.layout.fragment_main, container, false);
            mDataDBHelper = new DataDBHelper(getActivity());
            Button add_button = (Button)rootView.findViewById(R.id.sql_add);
            add_button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {

                    insertData("ZHANG SAN","W123456");


                }
            });

            return rootView;
        }

        public static void insertData(String name, String number){

            SQLiteDatabase sq = mDataDBHelper.getWritableDatabase();

/*
            sq.execSQL("INSERT INTO person1(name, age) VALUES(?,?)", new
Object[]

                    {
                            name, number
                    });
*/
            String sql = "INSERT INTO " + DataContract.DataEntry.TABLE_NAME +
"(" +
                    DataContract.DataEntry.COULUMN_USER_NAME + ", " +
                    DataContract.DataEntry.COULUMN_PHONE_NUMBER + ")
VALUES(?, ?)";
            sq.execSQL(sql, new Object[]{
                    name, number
            });
```

```
                Log.v(" INSERT",name + number + "insert successfully");

            }

        }


}
```

五、实验报告要求

实验报告中要包含以下几个部分：

1、实验目的

2、实验条件

3、实验原理

4、实验步骤分析

5、实验结果与总结

6、实验思考题

实验步骤要详细，关键步骤要有截图，运行结果也要有截图。

六、实验思考题

1. 增加数据库的查询、删除、更新等功能
2. 在哪些情况下需要使用数据库？

七、参考资料