中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

课程名称：移动应用开发　　任课教师： 郑贵锋

年级	大三	专业（方向）	移动互联网
学号	15352408	姓名	张镓伟
电话	13531810182	Email	709075442@qq.com
开始日期	2017.12.9	完成日期	2017.12.10

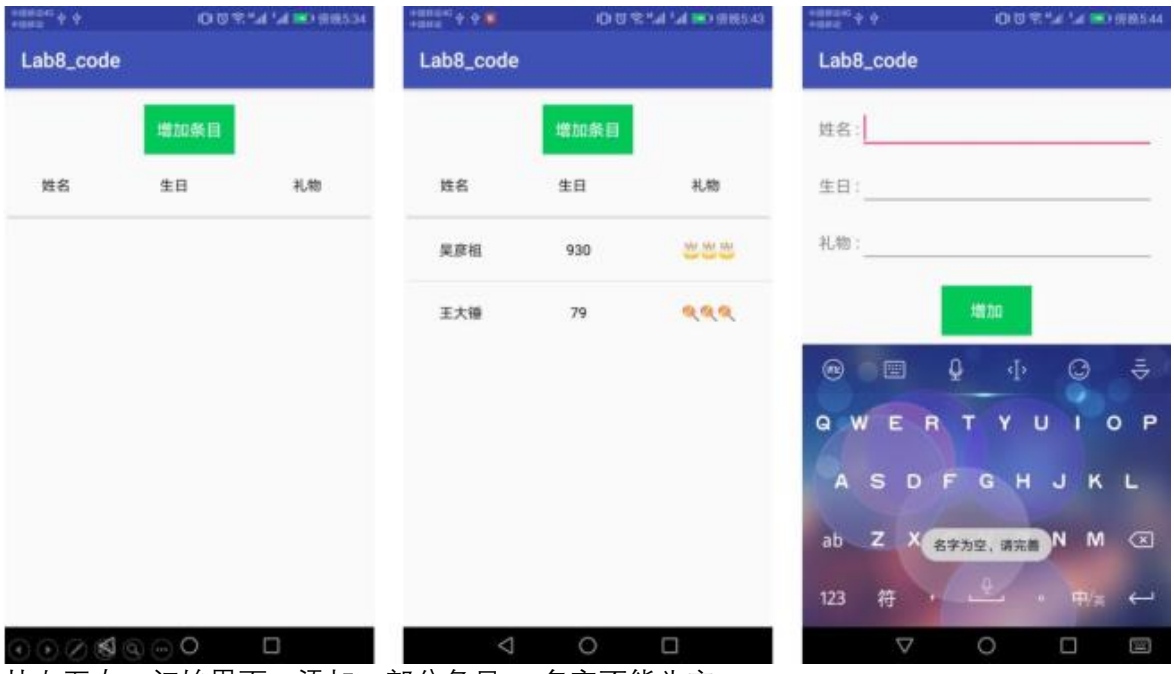
一、 实验题目

数据存储（二）

二、 实验目的

1. 学习 SQL 数据库的使用。
2. 学习 ContentProvider 的使用。
3. 复习 Android 界面编程。

三、 实验内容



从左至右：初始界面，添加一部分条目， 名字不能为空

从左至右：名字不能重复，点击条目显示信息（可修改）， 长按删除条目。

实现一个生日备忘录， 要求实现：

使用 SQLite 数据库保存生日的相关信息，并使得每一次运行程序都可以显示出已经存储在数据库里的内容；

使用 ContentProvider 来获取手机通讯录中的电话号码。

功能要求：

A. 主界面包含增加生日条目按钮和生日信息列表；

B. 点击“增加条目” 按钮，跳转到下一个 Activity 界面， 界面中包含三个信息输入框（姓名、生日、礼物） 和一个“增加” 按钮， **姓名字段不能为空且不能重复**；

C. 在跳转到的界面中，输入生日的相关信息后， 点击“增加” 按钮返回到主界面， **此时，主界面中应更新列表，增加相应的生日信息**；

D. 主界面列表点击事件：

- 点击条目：

  弹出对话框，对话框中显示该条目的信息，并允许修改；

  对话框下方显示该寿星电话号码**（如果手机通讯录中有的话，如果没有就显示“无”）**

  点击“保存修改” 按钮， 更新主界面生日信息列表。

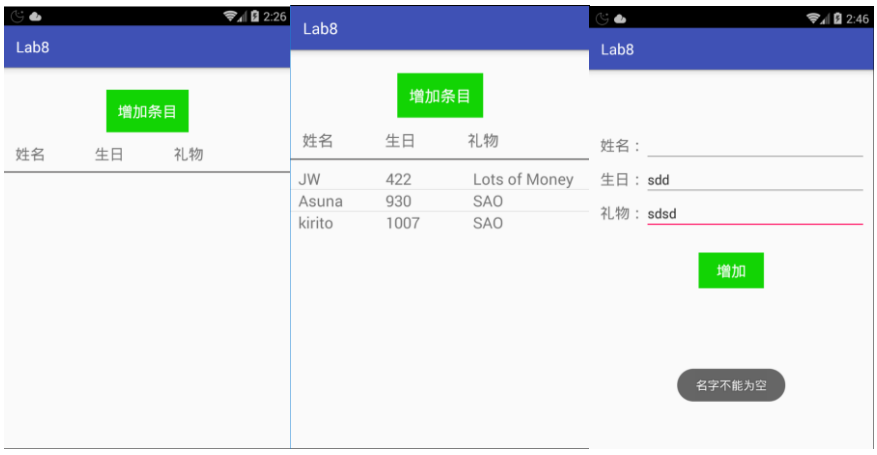
- 长按条目：

  弹出对话框显示是否删除条目；

  点击“是” 按钮，删除该条目，并更新主界面生日列表。

四、 课堂实验结果

（1） 实验截图



从左至右：初始界面，添加一部分条目， 名字不能为空。

从左至右：名字不能重复，点击条目显示信息（可修改）， 长按删除条目。

（2） 实验步骤以及关键代码

1.XML 布局代码

起始界面布局：

布局比较简单，一开始一个居中的增加条目的按钮，然后用一个水平的 LinearLayout 设置姓名、生日、礼物，然后是一个 listview 用于之后展示生日信息。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:orientation="vertical"
    tools:context="com.example.freedom.lab8.MainActivity">

    <Button
        android:id="@+id/add"
        android:layout_marginTop="30dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:padding="15dp"
        android:background="@color/green"
        android:text="增加条目"
        android:textColor="@color/white"
        android:textSize="20sp"/>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="15sp"
        android:layout_marginLeft="15sp"
        android:layout_marginRight="15sp">
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1.5"
            android:layout_height="wrap_content"
            android:text="姓名"
            android:textSize="20sp"/>
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1.5"
            android:layout_height="wrap_content"
            android:text="生日"
            android:textSize="20sp"/>
        <TextView
            android:layout_width="0dp"
            android:layout_weight="2"
            android:layout_height="wrap_content"
            android:text="礼物"
            android:textSize="20sp"/>
    </LinearLayout>

    <TextView
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:layout_width="match_parent"
        android:layout_height="2.5sp"
        android:background="@color/grey"/>

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </ListView>

</LinearLayout>
```

增加条目界面布局：

每一行信息是一个 LinearLayout，每个 LinearLayout 包含一个 TextView 和一个 EditText

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:orientation="vertical"
    tools:context="com.example.freedom.lab8.Additem">
    <LinearLayout
        android:layout_marginTop="80sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_marginLeft="15sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="姓名："
            android:textSize="20sp"/>
        <EditText
            android:id="@+id/add_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="10sp"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_marginLeft="15sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="生日："
            android:textSize="20sp"/>
        <EditText
            android:id="@+id/add_birth"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="10sp"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_marginLeft="15sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="礼物："
            android:textSize="20sp"/>
        <EditText
            android:id="@+id/add_gift"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="10sp"/>
    </LinearLayout>
    <Button
        android:id="@+id/add2"
        android:layout_marginTop="30dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:paddingTop="10dp"
        android:paddingBottom="10dp"
        android:paddingLeft="20dp"
        android:paddingRight="20dp"
        android:textSize="20sp"
        android:background="@color/green"
        android:textColor="@color/white"
        android:text="增加"/>
</LinearLayout>
```

对话框布局：

每一行信息是一个 LinearLayout，每个 LinearLayout 包含一个 TextView 和一个 EditText，所但是名字那一栏是两个 TextView，因为名字不应被修改。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_marginTop="25sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_marginLeft="20sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="姓名："/>
        <TextView
            android:id="@+id/D_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="123"/>
    </LinearLayout>
    <LinearLayout
        android:layout_marginTop="25sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_marginLeft="20sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="生日："/>
        <EditText
            android:id="@+id/D_birth"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="123"/>
    </LinearLayout>
    <LinearLayout
        android:layout_marginTop="25sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_marginLeft="20sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="礼物："/>
        <EditText
            android:id="@+id/D_gift"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="123"/>
    </LinearLayout>
    <TextView
        android:layout_marginTop="25sp"
        android:layout_marginLeft="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/phone"
        android:textSize="20sp"
        android:textColor="@color/green"
        android:text=""/>
</LinearLayout>
```

ListView 中每一项信息的布局：

每一行一个水平 LinearLayout，里面用三个 TextView 显示相关信息。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="15sp"
    android:layout_marginLeft="15sp"
    android:layout_marginRight="15sp">
    <TextView
        android:id="@+id/name"
        android:layout_marginLeft="10dp"
        android:layout_width="0dp"
        android:layout_weight="1.5"
        android:layout_height="wrap_content"
        android:textSize="20sp"/>
    <TextView
        android:id="@+id/birthday"
        android:layout_width="0dp"
        android:layout_weight="1.5"
        android:layout_height="wrap_content"
        android:textSize="20sp"/>
    <TextView
        android:id="@+id/gift"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="wrap_content"
        android:textSize="20sp"/>
</LinearLayout>
```

2.JAVA 代码

登录界面部分：

将数据库表的所有内容放进 ListView 中显示，这里可以使用 SimpleCursorAdapter 直接将查询数据库得到的游标内容放进 ListAdapter 中，省去了自己写转化的步骤。

```
final ListView list = (ListView) findViewById(R.id.list);
Cursor cursor = db.getAll();
if (cursor != null) { //查询数据并通过SimpleCursorAdapter绑定到ListView上
    ListAdapter adapter = new SimpleCursorAdapter(
            this, R.layout.item,
            cursor,
            new String[]{"name", "birth", "gift"},
            new int[]{R.id.name, R.id.birthday, R.id.gift},
            0
    );
    list.setAdapter(adapter);
    cursor.moveToFirst();
    for (int i = 0; i < cursor.getCount(); i++) {
        name.add(cursor.getString(1));
        birth.add(cursor.getString(2));
        gift.add(cursor.getString(3));
        cursor.moveToNext();
    }
}
```

点击增加条目按钮，跳转到增加条目的 activity

```
final Button add_item = (Button) findViewById(R.id.add);
add_item.setOnClickListener((view) -> {
        Intent intent = new Intent(MainActivity.this, Additem.class);
        startActivity(intent);
});
```

设置 listview 的长按点击事件，弹出一个 AlertDialog 询问是否删除数据，点击否则不动作，点击是则删除该条信息。

```
final AlertDialog.Builder dialog2 = new AlertDialog.Builder(this);
list.setOnItemLongClickListener((parent, view, position, id) -> {
        dialog2.setTitle("是否删除？").setMessage("").setNegativeButton("否",
```

```
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    }
                }).setPositiveButton("是",
                (dialog, which) -> {
                        db.delete(name.get(position));
                        onResume();
                }).create();
        dialog2.show();
        return true;
});
```

设置 listview 的点击事件，此时应该弹出一个 AlertDialog 对话框显示详细信息，但是这个对话框 使用了其他 XML 布局文件，这里可以通过 LayoutInflater.inflate 方法加载这个布局文件，并通过 AlertDialog.setView 方法设置这个布局。然后是设置好要显示的信息，以及放弃修改和保存修改两个按钮的动作，前者不动作，后者使用 update 更新数据库。

```
list.setOnItemClickListener((adapterView, view, i, l) -> {
        //通过LayoutInflater 加载其他xml布局
        LayoutInflater factory = LayoutInflater.from(MainActivity.this);
        View myview = factory.inflate(R.layout.dialoglayout, null);
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setView(myview);
        builder.setTitle("ヽ(•‾̀ω‾́)ﾉ～★恭喜发财★～ﾍﾍ");

        final TextView diaName = (TextView) myview.findViewById(R.id.D_name);
        final EditText diaBirth = (EditText) myview.findViewById(R.id.D_birth);
        final EditText diaGift = (EditText) myview.findViewById(R.id.D_gift);
        final TextView diaPhone = (TextView) myview.findViewById(R.id.phone);
        //设置好显示的信息
        diaName.setText(name.get(i));
        diaBirth.setText(birth.get(i));
        diaGift.setText(gift.get(i));
        //通过getPhoneNumber方法获得通讯录中的电话号码
        String phone = getPhoneNumber(name.get(i));
        if (phone.equals("")) diaPhone.setText("电话：无");
        else diaPhone.setText("电话：" + phone);

        builder.setNegativeButton("放弃修改",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    }
                }).setPositiveButton("保存修改",
                (dialog, which) -> {
                        db.update(diaName.getText().toString(), diaBirth.getText().toString(), diaGift.getText().toString());
                        onResume();
                });
        builder.create();
        builder.show();
});
```

getPhoneNumber 方法功能是获取指定名字的联系人电话，这个通过 ContentProvider 的 getContentResolver 方法去读取联系人列表，然后返回的是一个 Cursor 类型，就相当于查询了一个数据库。

```
public String getPhoneNumber(String Name) {
    String number = new String();
    //读取联系人列表
    Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
    while (cursor.moveToNext()) {
        String id = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts._ID));
        String name = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
        //找到联系人中名字与生日信息中名字一样的联系人
        if (name.equals(Name)) {
            //判断该联系人信息中是否有号码
            int isHas = Integer.parseInt(cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER)));
            if (isHas > 0) { //有号码则取出该号码
                Cursor c = getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                        ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " + id, null, null);
                while (c.moveToNext()) {
                    number += c.getString(c.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER)) + " ";
                }
                c.close();
            }
        }
    }
    return number;
}
```

这个时候我们的 ListView 列表是不会更新的，我们可以重写 onResume 方法，在这个方法里去更新 ListView 的内容，这个方法会在从其他 activity 切换回来或者结束与当前 activity 的用户交互之后调用。

```
@Override
protected void onResume() {
    super.onResume();
    final ListView list = (ListView) findViewById(R.id.list);
    name.clear();
    birth.clear();
    gift.clear();
    Cursor cursor = db.getAll();
    if (cursor != null) {
        ListAdapter adapter = new SimpleCursorAdapter(this, R.layout.item, cursor,
                new String[]{"name", "birth", "gift"},
                new int[]{R.id.name, R.id.birthday, R.id.gift}, 0);
        list.setAdapter(adapter);
        cursor.moveToFirst();
        for (int i = 0; i < cursor.getCount(); i++) {
            name.add(cursor.getString(1));
            birth.add(cursor.getString(2));
            gift.add(cursor.getString(3));
            cursor.moveToNext();
        }
    }
}
```

增加条目界面部分：

这个界面的代码比较简单，就是将输入的内容，插入到数据库中，其中还要判断名字是否为空和名字是否重复。

```
public class Additem  extends AppCompatActivity {
    final myDB db = new myDB(this);
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_additem);
        final EditText add_name = (EditText) findViewById(R.id.add_name);
        final EditText add_birth = (EditText) findViewById(R.id.add_birth);
        final EditText add_gift = (EditText) findViewById(R.id.add_gift);

        Button button = (Button) findViewById(R.id.add2);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(TextUtils.isEmpty(add_name.getText().toString())){
                    Toast.makeText(Additem.this, "名字不能为空", Toast.LENGTH_SHORT).show();
                }
                else if (db.query(add_name.getText().toString()) == false) {
                    db.insert(add_name.getText().toString(), add_birth.getText().toString(), add_gift.getText().toString());
                    finish();
                }
                else{
                    Toast.makeText(Additem .this, "已存在重复的名字", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

数据库类部分：

创建数据库：

```
public myDB(Context context) { super(context, DB_NAME, null, DB_VERSION); }

@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) { //创建数据库
    String CREATE_TABLE = "CREATE TABLE if not exists "
            + TABLE_NAME
            + "(_id INTEGER PRIMARY KEY,name TEXT,birth TEXT,gift TEXT)";
    sqLiteDatabase.execSQL(CREATE_TABLE);
}
```

实现 insert、update 和 delete 方法，借助 ContentValues 来实现

```
public void insert(String name, String birth, String gift){//插入数据
    SQLiteDatabase db = getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put("name", name);
    cv.put("birth", birth);
    cv.put("gift", gift);
    db.insert(TABLE_NAME, null, cv);
    db.close();
}

public void update(String name, String birth, String gift){//更新数据
    SQLiteDatabase db = getWritableDatabase();
    ContentValues updateValues = new ContentValues();
    updateValues.put("birth",birth);
    updateValues.put("gift",gift);
    db.update(TABLE_NAME, updateValues,"name="+"'"+name+"'",null);
    db.close();
}

public void delete(String name){
    SQLiteDatabase db = getWritableDatabase();
    db.delete(TABLE_NAME,"name="+"'"+name+"'",null);
    db.close();
}
```

查询部分，有两种，一种是查询表中所有信息，一种是查询表中是否存在指定名字的信息，其中第二种通过遍历这个数据表来比较判断。

```
public Cursor getAll(){//查询表中所有信息
    SQLiteDatabase db = getWritableDatabase();
    Cursor cursor = db.query(TABLE_NAME, new String[]{"_id", "name", "birth", "gift"}, null, null, null, null, null);
    return cursor;
}

public boolean query(String name){//通过名字查询是否存在相关记录
    boolean In = false;
    SQLiteDatabase db = getReadableDatabase();
    Cursor cursor = db.query(TABLE_NAME, new String[]{"_id", "name", "birth", "gift"}, null, null, null, null, null);
    cursor.moveToFirst();
    if (cursor != null && cursor.getCount() >= 0) {
        for (int i = 0; i < cursor.getCount(); i++) {
            if (cursor.getString(1).equals(name)) {
                In = true;
                break;
            }
            cursor.moveToNext();
        }
    }
    db.close();
    return In;
}
```

（3） 实验遇到困难以及解决思路

1.如何比较方便地将数据库表中的数据显示在 ListView 的容器中？

可以直接用 Adapter 对象处理，但是工作量比较大。Android SDK 提供了一个专用于数据绑定的 Adapter 类----SimpleCursorAdapter。SimpleCursorAdapter 与 SimpleAdapter 用法相近。只是将 List 对象换成了 Cursor 对象。而且 SimpleCursorAdapter 类构造方法的第四个参数 from 表示 Cursor 对象中的字段，而 SimpleAdapter 类构造方法的第四个参数 from 表示 Map 对象中的 key。除此之外，这两个 Adapter 类在使用方法完全相同。下面是 SimpleCursorAdapter 类构造方法定义。

public SimpleCursorAdapter(Context context,int layout,Cursor c,String[] from,int[] to, int flags);

可以看到第一个参数就是当前上下文，第二个是你要使用的 ListView 中每一项内容的布局，第三个就是查询数据库得到的 cursor，第 4 个是 cursor 字段名，第 5 个是与 from 数据对应数据的视图 ID。最后一个 flags 有两个值，分别是 CursorAdapter.FLAG_AUTO_REQUERY 和 CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER，前者可以自动查询如果数据库中的数据发生了变化以实时反映到界面上来，第二个参数是会在回调代码里更新数据。

2.如何让增加条目后或者修改信息后，ListView 的内容可以跟着更新？

A.第一种方法可以重写 MainActivity 的 onResume 方法，在这个方法中重新查询数据库并重新绑定到 ListView 中去显示。OnResume 一般适用于以下三种场景：

a.一个 Activity 启动另一个 Activity: onPause()->onStop(),
再返回：onRestart()->onStart()->onResume()

b. 程序按 back 退出： onPause()->onStop()->onDestory(),
再进入：onCreate()->onStart()->onResume();

c. 程序按 home 退出：onPause()->onStop(),
再进入：onRestart()->onStart()->onResume();

B.第二种方法是在 SimpleCursorAdapter 的第 6 个参数中使用 CursorAdapter.FLAG_AUTO_REQUERY 来自动更新，但是这样使用有一个问题，查询数据是发生在 UI 线程中，会造成卡顿的现象，在 android3.0 及以上已经弃用。

3.数据库的 insert 方法第二个参数是什么（db.insert(TABLE_NAME, null, cv)）？

不管第三个参数是否包含数据，执行 Insert()方法必然会添加一条记录，如果第三个参数为空，会添加一条除主键之外其他字段值为 Null 的记录。Insert()方法内部实际上通过构造 insert SQL 语句完成数据的添加，Insert()方法的第二个参数用于指定空值字段的名称。

如果第三个参数 values 为 Null 或者元素个数为 0， 由于 Insert()方法要求必须添加一条除了主键之外其它字段为 Null 值的记录，为了满足 SQL 语法的需要， insert 语句必须给定一个字段名，如：insert into TABLE_NAME (name) values(NULL)，倘若不给定字段名， insert 语句就成了这样： insert into TABLE_NAME () values()，显然这不满足标准 SQL 的语法。对于字段名，建议使用主键之外的字段，如果使用了 INTEGER 类型的主键字段，执行类似 insert into TABLE_NAME (id) values(NULL)的 insert 语句后，该主键字段值也不会为 NULL。如果第三个参数 values 不为 Null 并且元素的个数大于 0 ，可以把第二个参数设置为 null。

4.query 各参数含义？

query 参数如下：

```
Cursor query (boolean distinct,
            String table,
            String[] columns,
            String selection,
            String[] selectionArgs,
            String groupBy,
            String having,
            String orderBy,
            String limit,
            CancellationSignal cancellationSignal
```

distinct，设置为 true,每一行的数据必须唯一。反之亦然。

table，query 函数要操作的表名。

columns，要返回的列的名字的数组。如果设置为 null，返回所有列。

selection，一个决定返回哪一行的过滤器，相当于 SQL 语句中的 WHERE 关键字。传递 null 则会返回给定表的所有行。

selectionArgs，用于替换上一个参数中的？ ,顺序对应 selection 中？的顺序。格式限制为 String 格式。

groupBy，用于设定返回行的分组方式，相当于 SQL 语句中的 GROUP BY 关键字。传递 null 表示返回的行不会被分组。

having，决定哪一行被放到 Cursor 中的过滤器。如果使用了行分组，相当于 SQL 语句中的 HAVING 关键字。传递 null 会导致所有的行都包含在内，前提是 groupBy 属性也设置为 null。

orderBy，行的排列方式，相当于 SQL 语句中的“ORDER BY”关键字，传递 null 表示使用默认的排序方式，可能是无序排列。

limit，设置 query 语句返回行的数量，相当于 SQL 语句中的“LIMIT”关键字，传递 null 表示没有设置 limit 语句。**注意**格式为 String,传递的时候需要传递数字字符串，例如“12”

cancellationSignal，取消程序操作的信号，如果没有则设置为 null。如果操作取消了，query 语句运行时会抛出 OperationCanceledException 异常。

五、 课后实验结果

在增加条目信息的界面增加右滑返回。

如果我们要使得整个 Activity 的界面滑动，就需要使得根布局 decorView 滑动

通过在 Activity 中用如下代码可以获得 decorView：

```
//获得decorView
View decorView = getWindow().getDecorView()
```

如何滑动：

1.当手指按下时，获得按下时的坐标（xDown, yDown）

2.手指移动时，获得当前手指坐标（xCurrent, yCurrent）,可以得到水平移动距离

distanceX = xCurrent - xDown

3.获得水平移动距离之后，通过 setX()方法设置界面的 X 坐标。

获得手指位置：

在 Activity 的 onTouchEvent(MotionEvent event)方法中即可获得手指的位置。

event.getX() 为手指的 X 坐标；

event.getY() 为手指的 Y 坐标。

抬起手指后的行为：

1. 当滑动超过一半时，松开手后，界面继续向右侧滑动直到完全在屏幕上不可见，这时调用 finish()。

2. 当滑动没有超过一半时，松开手后，界面返回到初始状态，即 x 坐标为 0。

在 Additem 类中加入下面代码就可以实现简单的右滑返回：

全局变量：

```
//整个Activity视图的根视图
View decorView;
//手指按下时的坐标
float downX, downY;
//手机屏幕的宽度和高度
float screenWidth, screenHeight;
```

OnCreate 方法中：

```
//获得decorView
decorView = getWindow().getDecorView();

// 获得手机屏幕的宽度和高度，单位像素
DisplayMetrics metrics = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metrics);
screenWidth = metrics.widthPixels;
screenHeight = metrics.heightPixels;
```

重写 onTouchEvent 方法处理触摸事件：

```
//通过重写onTouchEvent方法，对触摸事件进行处理
@Override
public boolean onTouchEvent(MotionEvent event) {
    if(event.getAction() == MotionEvent.ACTION_DOWN){// 当按下时
        // 获得按下时的X坐标
        downX = event.getX();

    }else if(event.getAction() == MotionEvent.ACTION_MOVE){// 当手指滑动时
        // 获得滑过的距离
        float moveDistanceX = event.getX() - downX;
        if(moveDistanceX > 0){// 如果是向右滑动
            decorView.setX(moveDistanceX); // 设置界面的X到滑动到的位置
        }

    }else if(event.getAction() == MotionEvent.ACTION_UP){// 当抬起手指时
        // 获得滑过的距离
        float moveDistanceX = event.getX() - downX;
        if(moveDistanceX > screenWidth / 2){
            // 如果滑动的距离超过了手机屏幕的一半，结束当前Activity
            finish();
        }else{ // 如果滑动距离没有超过一半
            // 恢复初始状态
            decorView.setX(0);
        }
    }
    return super.onTouchEvent(event);
}
```

效果：

六、 课后思考及感想

这次实验我们学习了使用数据库 SQLite，这对于后续大作业的编写有很大帮助。我们也知道了在 java 中既可以直接通过写 sql 语句执行相关操作，也可以通过调用一些提供好的方法去执行相关操作，我觉得熟悉 sql 语句的还是直接写 sql 语句吧，比较快。另外我还知道了用 SimpleCursorAdapter 可以很方便直接将查询数据库得到的内容绑定给 ListView 去显示。希望后续能继续提高自己安卓的开发技巧。