《手机平台应用开发（与 Google 共建）实验》

实验八：数据存储（二）

实验报告

学 院 名 称 ：	数据科学与计算机学院
专　　业	软件工程（计应）
学 生 姓 名 ：	张凯鑫
学　　号 ：	14331362
班　　级 ：	周三上午 4-5 节、周五下午 7-8 节

【实验目的】

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

【实验内容】

实现一个生日备忘簿

技术要求：

1、 使用 SQLite 数据库保存生日相关信息，使得每次运行程序都可以显示出已经存储在数据库里的内容；

2、 使用 ContentProvider 来获取对应寿星的电话号码；

功能要求：

1、 主界面包含增加生日条目按钮和生日信息列表； （见图 1，图 2）

2、 点击<增加条目>按钮跳转到次界面；

3、 次界面输入生日相关信息后点击<增加>按钮会返回主界面（同时更新主界面的生日信息列表），且姓名字段不能为空，姓名字段不能重复 （见图 3，图 4）

4、 主界面中的列表点击事项处理：

a) 单击（查看并可修改该生日条目）： （见图 5）

i. 弹出对话框，显示该条目的相关信息，并提供修改。

ii. 同时，显示该生日条目寿星的电话号码；

iii. 点击<保存修改>按钮，更新主界面的生日信息列表

b) 长按（可删除该生日条目）： （见图 6）

i. 弹出对话框，显示是否删除；

ii. 点击<是>按钮，删除该生日条目，并更新主界面的生日信息列表

图 1 首次启动

图 2 增加一些条目后

图 3 名字不能重复

图 4 名字不能为空

图 5 点击处理

图 6 长按处理

【实验过程】

1. 创建新的 Android Studio 项目，命名为：ExperimentEight。
2. 按要求创建布局文件：activity_main.xml，添加控件 Button，和 TableLayout 布局的三个 TextView（姓名、生日、礼物），以及一个 ListView，并设计 ListView 的布局文件 Item.xml：TableLayout 布局的三个 TextView，且每个 TextView 设计大小一致。
3. 按要求创建布局文件：new_info.xml，一个 3*2 的表格用于表示姓名、生日、礼物 TextView，以及各自的输入 EditText，一个 Button，用于确认添加，返回主界面。
4. 设计自定义的 dialoglayout.xml，标题（TextView），内容（TableLayout 显示寿星的相关信息，姓名、生日、礼物、电话），以及两个 Button（用于修改或不修改）。
5. 创建 SQLite 数据库，写在 myDB.java 中，必须包含 myDB 的的构造函数，以及两个重载函数 onCreate()，onUpgrade()。
6. 在 java 文件 MainActivity.java 中，实现控件的基本功能：Button 跳转到次界面，ListView 读取 myDB 数据库中表格 myTable，并显示在 ListView 中，实现 ListView 的点击显示寿星相关信息 dialog 功能（显示联系人电话功能暂不完善）和长按显示 dialog 询问是否删除功能。
7. 在 java 文件 AddActivity.java 中，实现控件的基本功能，EditText 写入信息，Button 将写入的信息插入到 myDB 数据库中 myTable 表格中，并返回到主界面刷新列表。
8. 在配置文件 AndroidManifest.xml 中添加读取通讯录的权限如下：

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

   在 MainActivity.java 中使用 contentProvider 实现点击 ListView 显示联系人电话的功能。
9. 运行并调整实验代码直到实验完成。

【实验结果】

实验效果图及部分代码分析如下（实验代码详见 lab8_code 文件夹）：

主界面

次界面

姓名不能为空

添加“zhang”

跳转并更新列表和数据库信息

姓名不能重复

【数据库的创建及 listview 列表的显示】关键代码如下：

```
private myDB dbHelper;
dbHelper = new myDB(MainActivity.this, "birthday.db", null, 2);
public void showDB() {
    listItems = new ArrayList<HashMap<String, String>>();
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    Cursor cursor = db.query("myTable", null, null, null, null, null, null);
    if (cursor.moveToFirst()) {
        do {
            HashMap<String, String> map = new HashMap<String, String>();
            map.put("name", cursor.getString(cursor.getColumnIndex("name")));
            map.put("birthday", cursor.getString(cursor.getColumnIndex("birthday")));
            map.put("gift", cursor.getString(cursor.getColumnIndex("gift")));
            listItems.add(map);
        } while (cursor.moveToNext());
    }
    cursor.close();
    db.close();
    simpleAdapter = new SimpleAdapter(this, listItems, R.layout.item,
            new String[]{"name", "birthday", "gift"}, new int[] {R.id.name, R.id.birthday,
R.id.gift});
    lv.setAdapter(simpleAdapter);
}
```

【数据库信息的插入】关键代码如下：

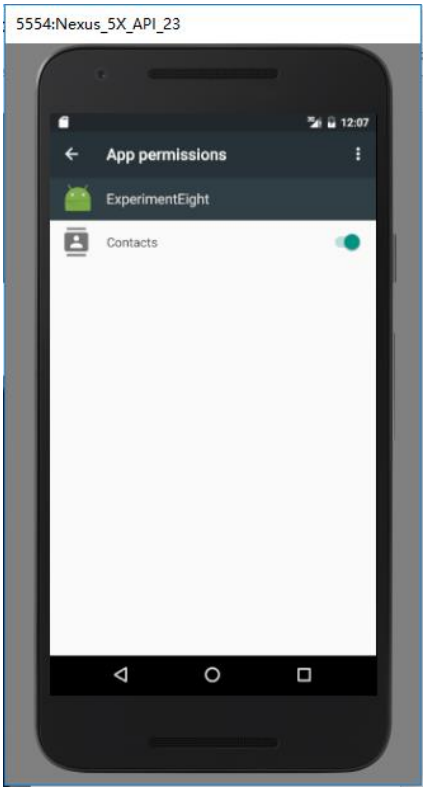
（添加信息时，以姓名为主键，不能为空，从数据库中查找是否已有此姓名，不能重复，正确输入则插入到数据库中）

```
btn_add.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (TextUtils.isEmpty(add_name.getText().toString())) {
            Toast.makeText(AddActivity.this, "名字为空，请完善",
Toast.LENGTH_SHORT).show();
        } else {
            SQLiteDatabase db = dbHelper.getWritableDatabase();
            Cursor cursor = db.rawQuery("select * from myTable where name = ?", new
String[] {add_name.getText().toString()});
            if (cursor.moveToFirst()) {
                Toast.makeText(AddActivity.this, "名字重复，请检查",
Toast.LENGTH_SHORT).show();
            } else {
                ContentValues values = new ContentValues();
                values.put("name", add_name.getText().toString());
                values.put("birthday", add_birthday.getText().toString());
                values.put("gift", add_gift.getText().toString());
                db.insert("myTable", null, values);/*
                Intent intent = new Intent(AddActivity.this, MainActivity.class);
                startActivity(intent);*/
                AddActivity.this.finish();
            }
            cursor.close();
            db.close();
        }
    }
});
```

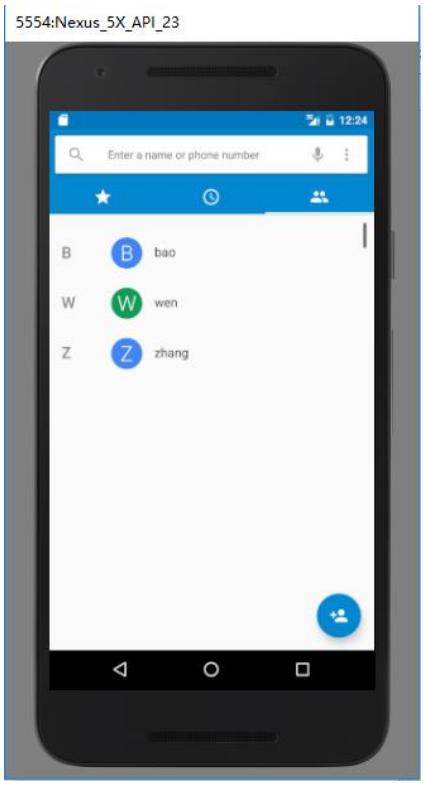
【插入数据后返回主界面，并重新加载主界面】关键代码如下：

```
@Override
protected void onResume() {
    super.onResume();
    showDB();
}
```

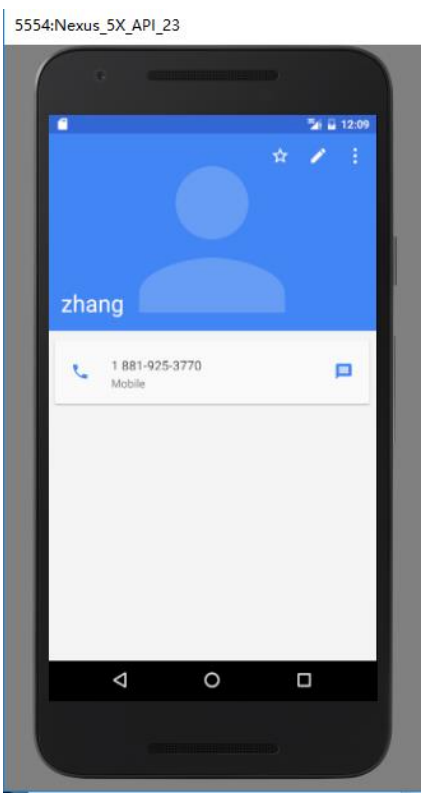
当前数据库列表



手动开启读取通讯录的权限



当前手机通讯录联系人列表（含有“zhang”，而不含有“kaixin”）



点击列表中姓名为“zhang”的信息，弹出对话框，显示的电话是从通讯录中读取到的

点击列表中姓名为“kaixin”的信息，弹出对话框，显示的电话为为空，因为通讯录中无此联系人电话

【使用 contentProvider 进行访问通讯录信息】关键代码如下：

```
public void getPhoneNum() {
    ContentResolver cr = MainActivity.this.getContentResolver();
    Cursor cursor = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null,
null);
    while (cursor.moveToNext()) {
        String contactId =
cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts._ID));
        String contactName =
cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME)
);
        if (et_name.getText().toString().equals(contactName)) {
            Cursor phoneNumbers =
cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                    ContactsContract.CommonDataKinds.Phone.CONTACT_ID + "=" + contactId,
null, null);
            if (phoneNumbers.moveToFirst()) {
                String number =
phoneNumbers.getString(phoneNumbers.getColumnIndex(ContactsContract.CommonDataKinds.Phone.N
UMBER));
                et_number.setText(number);
            }
            phoneNumbers.close();
            break;
        }
    }
    cursor.close();
}
```

一个程序访问其他程序时，需要给予这个程序权限，在配置文件 AndroidManifest.xml 中说明：

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

SDK2.3 除了这样说明权限外，还需要手动开启这个程序的权限，否则仍不能成功访问通讯录。

【自定义对话框的实现】关键代码如下：

```
LayoutInflater factory = LayoutInflater.from(MainActivity.this);
final View vi = factory.inflate(R.layout.dialoglayout, null);
final AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
builder.setView(vi);
final Dialog dialog = builder.create();
dialog.show();
```

如此之后，我们就可以将一个 layout 的内容（dialoglayout）全部显示在对话框（builder）里了。之后便可以对对 layout 里面的元素进行操作，对控件进行监听。

支持修改列表中某一项数据，并更新数据库及主界面列表

支持修改列表中某一项的的所有可修改数据，并全部更新到数据库及主界面列表

【数据库信息的更新包括 ListView 列表的更新】关键代码如下：
（点击“保存修改”按键时，当某个或多个 EditText 有输入时，则更新要修改的对应信息，数据库的更新使用 update()，ListView 的更新使用 notifyDataSetChanged()方法）

```
public void updateDBandList(int position) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    ContentValues values = new ContentValues();
    if (!TextUtils.isEmpty(et_birthday.getText().toString())) {
        values.put("birthday", et_birthday.getText().toString());
        listItems.get(position).put("birthday", et_birthday.getText().toString());
    }
    if (!TextUtils.isEmpty(et_gift.getText().toString())) {
        values.put("gift", et_gift.getText().toString());
        listItems.get(position).put("gift", et_gift.getText().toString());
    }
    db.update("myTable", values, "name = ?", new String[]
{listItems.get(position).get("name")});
    db.close();
    simpleAdapter.notifyDataSetChanged();
}
```

长按询问是否删除列表中某一项，“是”则删除数据库中和列表中的对应信息，“否”则不处理

【数据库信息的的删除与 ListView 列表某项的删除】关键代码如下：
（点击简单对话框的“是”确认删除，数据库信息的删除使用 delete()方法，ListView 列表随着使用 remove()方法删除，并用用 notifyDataSetChanged()更新当前列表）

```
builder.setPositiveButton("是", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        db.delete("myTable", "name = ?", new String[]
{listItems.get(position).get("name")});
        db.close();
        listItems.remove(position);
        simpleAdapter.notifyDataSetChanged();
        dialog.dismiss();
    }
});
```

【遇到的问题与解决方法】

1. 实验中 ListView 列表中每一项的空间难以调整，设置了 layout_weight 属性，但还是不能保证各列各行的 TextView 占据的空间相同，改用 RelativeLayout 用相对属性去约束也不能实现，最后使用了 TableLayout 布局，android:stretchColumns=“*”和 android:layout_width=“1dip”，这两行代码的结合使用，使得每一 TableRow 中各个控件自行平分整行，成功使得 ListView 中信息的显示位置在基本相同。
2. SQLite 数据库创建表时，一开始使用实验文档中的表头：“_id, name, birth, gift”，实际在插入数据时却忽略了主键_id，使用数据插入一直不成功，之后修改表头为“name, birthday, gift”，其中 name 为主键，使得插入成功。
3. 使用 contentProvider 访问通讯录时，在配置文件中声明权限后，还是不能成功读取，按实验文档查看是否开启了权限，手动开启，（setting—Applications—ExperimentEight），果然没有开启权限，手动开启之；但还是不能成功读取，后来发现通讯录联系人列表和通话录电话号码不相同，分别是 ContactsContract.Contacts.CONTENT_URI，ContactsContract.CommonDataKinds.Phone.CONTENT_URI。
4. Activity 的生命周期问题，在次界面增加数据库信息后，次界面 finish()，主界面需要更新信息，最初采用 startActivity()使主界面 activity 重新入栈，但这会使 stack 中有多个主界面 activity，若在主界面跳转到次界面前调用 finish()，则在次界面时按返回键无法返回到主界面。最后了解了一下 activity 的生命周期，在 activity 启动后，它会先后调用 onCreate()——onStart()——onResume()，而在跳转到其他界面时，会先后调用 onPause()——onStop()，在从其他界面切换到主界面时，会调用 onResume()，因此重写函数 onResume()，在这一阶段实现主界面的刷新。实验证明，在长按出现的自定义对话框，到自定义对话框消失的过程中，activity 没有调用 onResume()，因此，在对话框更新和删除数据库信息后，界面的更新是使用 ListView 的 remove()、delete()、notifyDataSetChanged()，而不是重新加载。具体代码在前面关键代码中已说明。

【实验心得与体会】

1. 本次实验学习 SQLiter 数据库的使用和 ContentProvider 的使用，同时复习了 Android 界面编程，如 ListView、Adapter、自定义 dialog、TableLayout 等。
2. 实验过程中由于对 Android 界面编程不够熟悉，在 ListView、dialog 以及一些控件的布局上花了不少

少时间，但也因此对这些元素的使用更加熟悉了，这对 Android 的开发是很有用的。

3. 本次实验还涉及到程度之间的数据访问，初次涉及 ContentProvider，对这个也有了一定的了解，最重要的一点就是要声明权限，但有时候时候声明权限后并不起作用，还是需要手动开启权限。
4. SQLite 是 Android 系统内置的一款轻量轻的关系型数据库，占用资源很少，不仅支持标准的 SQL 语法，还遵循了数据库的 ACID 事务。可以进行 CRID 操作，即创建、查找、插入（更新）、删除。文件存储和 SharedPreferences 存储只适用于去保存一些简单的数据和键值对，SQLite 数据库可以用于存储数据量大、结构性复杂的的数据。
5. 本次实验对 activity 的生命周期和各阶段调用的方法有了更深入的理解，主要有 onCreate()、onStart()、onResume()、onRestart()、onPause()、onStop()、onDestory()等，通过重写这些方法有时可以收到奇效。