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：

年级	15352272	专业（方向）	软件工程（移动信息工程）
学号	15352272	姓名	彭国栋
电话	15626066058	Email	578291308@qq.com
开始日期		完成日期	

一、 实验题目

学习 SQL 数据库的使用

学习 ContentProvider 的使用

复习 Android 界面编程

二、 实现内容

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。

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：

① 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。

② 长按条目：

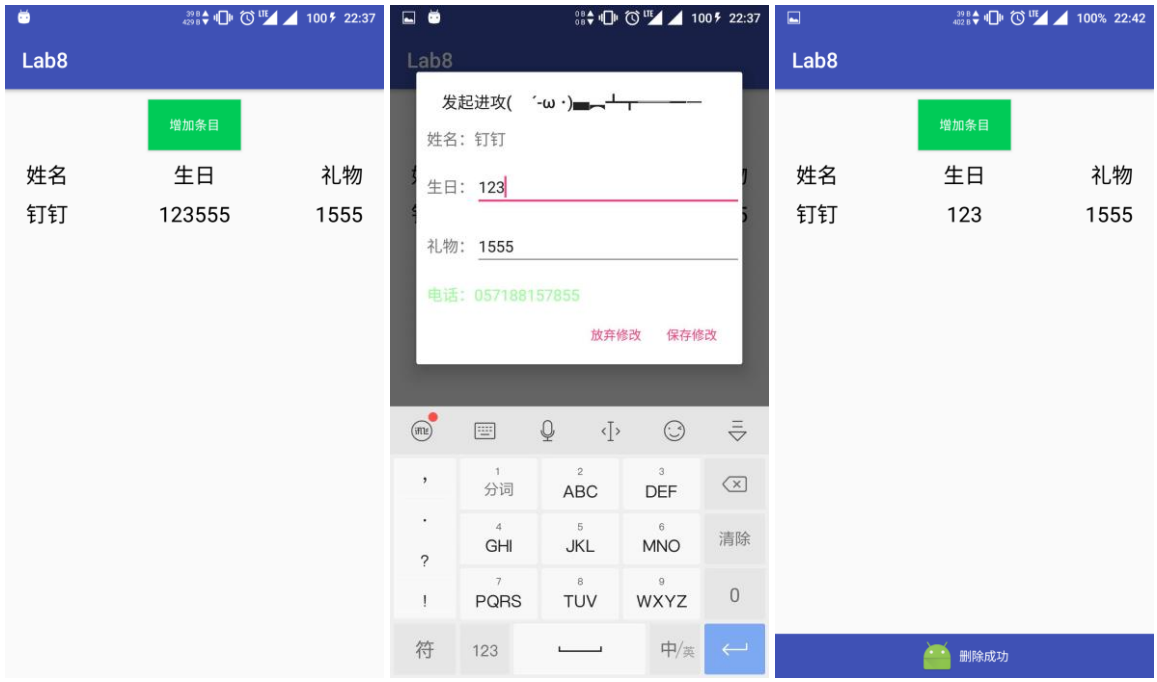
弹出对话框显示是否删除条目；

点击“是” 按钮，删除该条目，并更新主界面生日列表。

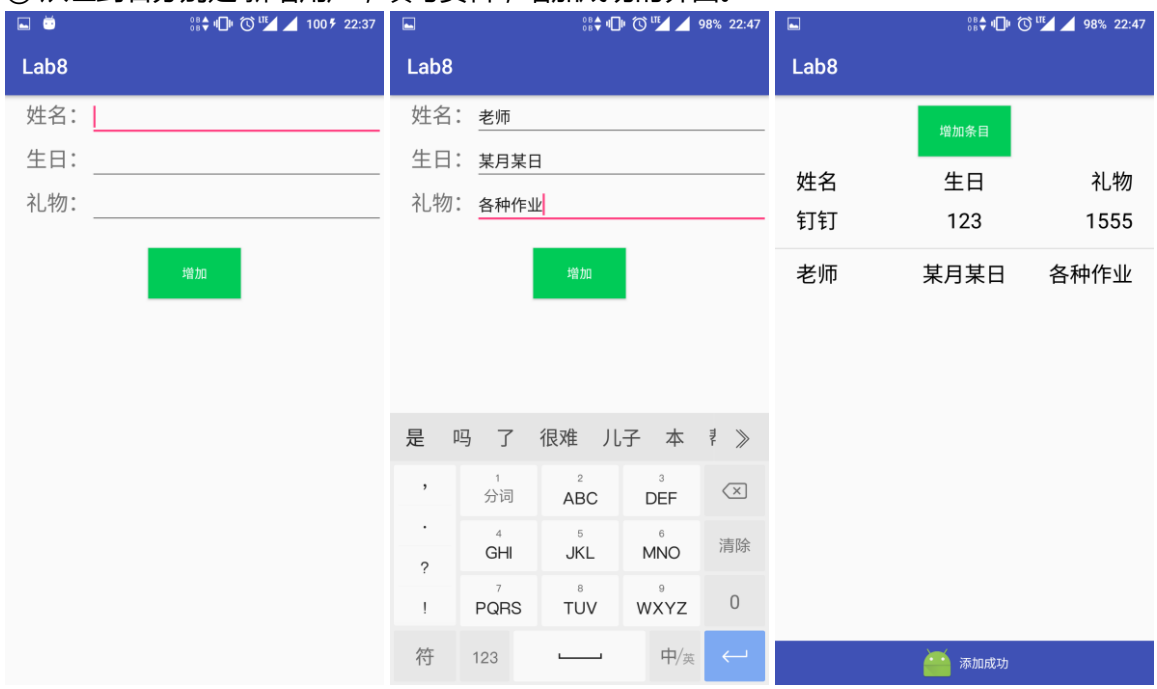
三、 课堂实验结果

（1） 实验截图

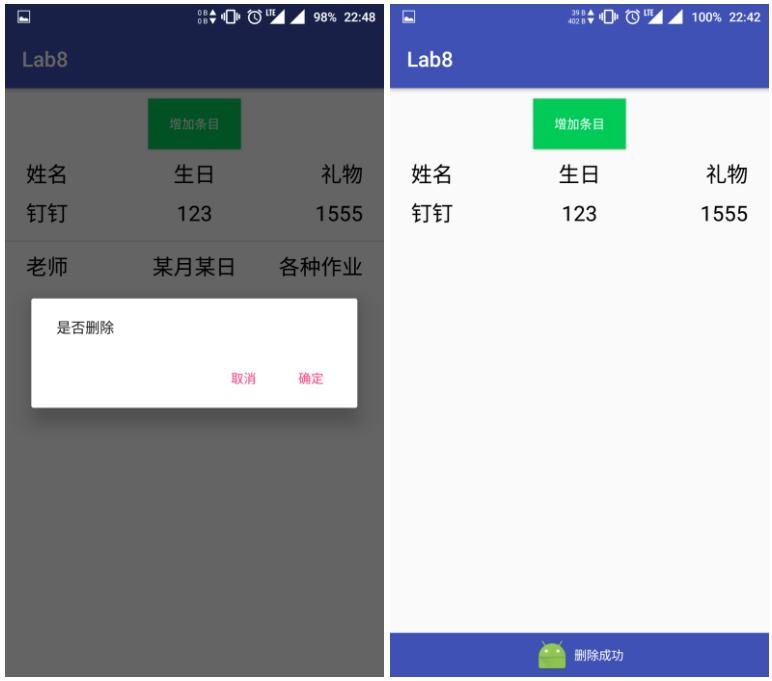
① 从左到右分别是进入界面，点击人物并修改资料，修改成功的步骤。



② 从左到右分别是 新增用户，填写资料，增加成功的界面。



③ 从左到右分别是 长按弹出对话框，点击删除后列表更新的步骤。



（2） 实验步骤以及关键代码

1、首先是界面设计，在界面设计的时候，需要为对话框单独设计一个界面，然后借助 LayoutInflater 来实现在对话框中添加布局的操作。这个是比起以往较为不同的地方。

2、数据库设计：

首先是定义数据库表，根据要求，我们可以建立一个含有 id（主键），姓名，生日，和礼物 4 个数据的表。然后根据功能，这次实验主要涉及到 3 个数据库的操作，首先是增加人物的生日备忘录，对应的数据库操作是 insert；然后是修改人物的生日备忘录，对应的数据库操作是 update；然后是长按删除人物，对应的数据库操作是 delete。

然后就可以借助代码来进行实现了，这里主要使用的是借助 ContentValues 和 Where 语句来实现操作：

① insert：

```
/**
 *
 * @param name
 * @param birth
 * @param gift
 */
public void insert(String name,String birth,String gift)
{
    SQLiteDatabase db=getWritableDatabase();
    ContentValues values=new ContentValues();
    values.put("name",name);
    values.put("birth",birth);
    values.put("gift",gift);
    db.insert(TABLE_NAME,null,values);
    db.close();
}
```

② update：根据名字进行更新

```
/**
 *
 * @param name
 * @param birth
 * @param gift
 */
public void update(String name,String birth,String gift)
{
    SQLiteDatabase db=getWritableDatabase();
    ContentValues values=new ContentValues();
    String whereClause="name=? ";
    String[] whereArgs={name};
    values.put("name",name);
    values.put("birth",birth);
    values.put("gift",gift);
    db.update(TABLE_NAME,values,whereClause,whereArgs);
    db.close();
}
```

③ 根据名字直接删除

```
/**
 *
 * @param name
 */
public void delete(String name)
{
    SQLiteDatabase db=getWritableDatabase();
    String whereClause="name=? ";
    String[] whereArgs={name};
    db.delete(TABLE_NAME,whereClause,whereArgs);
    db.close();
}
```

3、主界面设计：

① list_update 列表更新，在这里使用了较为暴力的方式，直接每次更新的时候先将存有的 List 清空，然后在数据库里面查找内容，再填充进列表里进行更新。

```
private void list_update()
{
    data.clear();//列表显示的内容
    SQLiteDatabase sqLiteDatabase=db.getWritableDatabase();
    Cursor cursor=sqLiteDatabase.rawQuery("select * from "+TABLE_NAME,null);
    //读取数据库中的值
    if(cursor!=null)
    {
        while(cursor.moveToNext())
        {
            int name_index=cursor.getColumnIndex("name");
            int birth_index=cursor.getColumnIndex("birth");
            int gift_index=cursor.getColumnIndex("gift");
            Map<String,Object> map=new LinkedHashMap<>();
            String name=cursor.getString(name_index);
            String birth=cursor.getString(birth_index);
            String gift=cursor.getString(gift_index);
            map.put("name",name);
            map.put("birth",birth);
            map.put("gift",gift);
            data.add(map);
        }
    }
```

② list_delete 删除项目，主要是在删除的时候，需要顺便将数据库表更新，并且移除当前位置的数据，再更新列表。

```
private void list_delete()
{
    item_list.setOnItemLongClickListener((parent, view, position, id) -> {
            AlertDialog.Builder builder=new AlertDialog.Builder(MainActivity.this);
            builder.setMessage("是否删除")
                    .setPositiveButton("确定", (dialog, which) -> {
                            db.delete(data.get(position).get("name").toString());
                            data.remove(position);
                            simpleAdapter.notifyDataSetChanged();
                            Toast.makeText(MainActivity.this,"删除成功",Toast.LENGTH_SHORT).show();
                    })
                    .setNegativeButton("取消", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {

                        }
                    })
                    .create().show();

            return true;//如果这个长按事件被消耗掉了，那么就返回true,如果没有被消耗掉，那么返回false.
    });
}
```

③ item_update，列表更新，首先是判断是否有读取联系人的权限，然后声明一个 LayoutInflater，接着再绑定对话框布局，以及绑定对话框布局内的部件，并给他们设置内容，包括名字，生日，礼物三个内容。而对于电话的操作，则需要先查询电话本里是否有联系人，当有联系人的前提下，再去查询是否有这个名字的联系人，当查询到之后，显示查询到的第一个人在电话本上。最后就是设置保存修改和放弃修改的操作。保存修改的时候需要更新数据库，并调用 list_update 来显示列表。

```
item_list.setOnItemClickListener((parent, view, position, id) -> {

        if (ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.READ_CONTACTS)!= PackageManager.PERMISSION_GRANTED)
        {
            ActivityCompat.requestPermissions(MainActivity.this, new String[]{Manifest.permission.READ_CONTACTS}, 0);
        }//检查权限

        LayoutInflater factor = LayoutInflater.from(MainActivity.this);//创建LayoutInflater从MainActivity
        View view_in = factor.inflate(R.layout.dialog_layout, null);//绑定设定的对话框Layout

        //绑定相应的控件
        dialog_name=(TextView)view_in.findViewById(R.id.dialog_name);
        dialog_birth=(EditText)view_in.findViewById(R.id.dialog_birth);
        dialog_gift=(EditText)view_in.findViewById(R.id.dialog_gift);
        dialog_number=(TextView)view_in.findViewById(R.id.dialog_number);
        //设置显示内容
        dialog_name.setText((CharSequence)data.get(position).get("name"));
        dialog_birth.setText((CharSequence)data.get(position).get("birth"));
        dialog_gift.setText((CharSequence)data.get(position).get("gift"));
```

```
Cursor cursor=getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,null,null,null,null);
cursor.moveToFirst();//移动到第一行
int isHas=Integer.parseInt(cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER)));
String number="";
if(isHas>0)
{
    Cursor phone=getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
            ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME + "=?" , new String[] {data.get(position).get("name").toString()}, null);
    //查询联系人的名字
    number="";
    if(phone.getCount()>0) //有这个人的存在
    {
        phone.moveToFirst();//显示第一个人的名字
        number+=phone.getString(phone.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
    }
    else
    {
        number="无";
    }
}
else
{
    number="无";
}
```

```
dialog_number.setText((CharSequence)number);
final AlertDialog.Builder alertDialog1 = new AlertDialog.Builder(MainActivity.this);
alertDialog1.setTitle("发起进攻(  ´-ω ･)▄▅▆▇═══一")
        .setView(view_in)
        .setPositiveButton("保存修改", (dialog, which) → {
                String name=data.get(position).get("name").toString();
                String birth=dialog_birth.getText().toString();
                String gift=dialog_gift.getText().toString();
                db.update(name,birth,gift);
                list_update();
        })
        .setNegativeButton("放弃修改", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        })
        .create()
        .show();
```

④ setAdd_item，设置增加项目操作，直接跳转即可。

```
private void setAdd_item()
{
    add_item.setOnClickListener((v) → {
            Intent intent=new Intent(MainActivity.this,add_layout.class);
            startActivityForResult(intent,REQUEST_CODE);
    });
}
```

⑤ onActivityResult，对于增加的项目进行列表更新

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent intent)
{
    if(requestCode==REQUEST_CODE&&resultCode==RESULT_CODE)
    {
        list_update();
    }
}
```

4、增加项目

① setAdd_button，当添加按钮被点击的时候，需要注意的是，因为数据库底层在更新和插入的时候是根据名字来进行的，所以为了防止在列表中出现两个相同名字的人，需要在增加人物的时候手动判断一下，然后如果不存在则直接将填写的值加入数据库中。

```
SQLiteDatabase sqLiteDatabase=db.getWritableDatabase();
Cursor cursor=sqLiteDatabase.rawQuery("select * from "+TABLE_NAME+" where name =? ",new String[]{name_temp});
if(cursor.getCount()!=0)//表中存在
{
    Toast.makeText(getApplicationContext(),"表中已经存在该姓名的联系人",Toast.LENGTH_SHORT).show();

}
else
{
    String birth_temp=birth_edit.getText().toString();
    String gift_temp=gift_edit.getText().toString();
    db.insert(name_temp,birth_temp,gift_temp);
    Toast.makeText(getApplicationContext(),"添加成功",Toast.LENGTH_SHORT).show();
    Intent intent=new Intent();
    setResult(RESULT_CODE,intent);
    finish();
}
```

（3） 实验遇到困难以及解决思路

首先是弹框显示，发现布局绑定的时候总是失败，并且弹出类似未进行声明的时候出现的错误。经过测试，可以在绑定布局的时候，使用这样的方式进行处理。

```
View view_in = factor.inflate(R.layout.dialog_layout, null);//绑定设定的对话框Layout

//绑定相应的控件
dialog_name=(TextView)view_in.findViewById(R.id.dialog_name);
```

首先是设置了联系人之后，发现点击人物弹出对话框就程序崩溃；经过查询发现是在查询是否存在联系人之后，需要将光标移动到第一行，即 cursor.moveToFirst()，否则会出现空指针异常。

四、 实验思考及感想

这次实验涉及到的数据库和数据库操作都较为简单，所以并不会有太大的难度，感觉主要是帮助我们去了解数据库在 android 中的使用，我相信经过这次实验之后，对于大作业肯定很有帮助。