# 中山大学移动信息工程学院本科生实验报告

课程名称：移动应用开发　　任课教师：郑贵锋

## 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

实现一个简单的数据存储界面如下：

初始界面

添加一部分条目

名字不能为空


名字不能重复


点击条目显示信息


长按删除条目

**实现一个生日备忘录，具体要求如下：**

- 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。

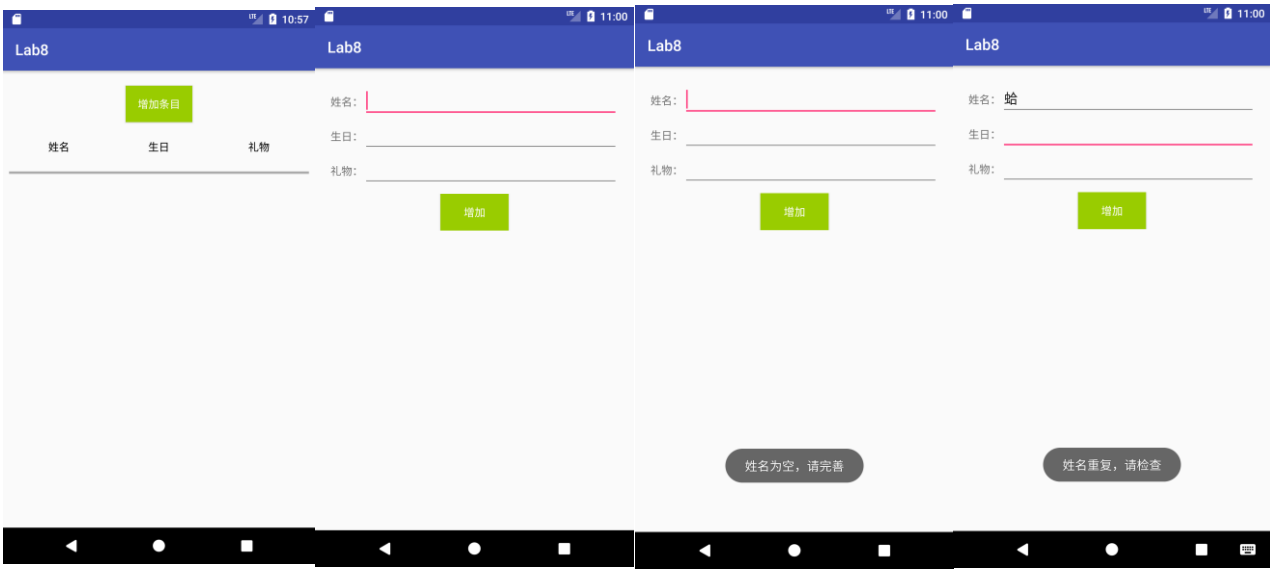
**功能要求：**

1. 主界面包含增加生日条目按钮和生日信息列表；
2. 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；
3. 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；
4. 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。

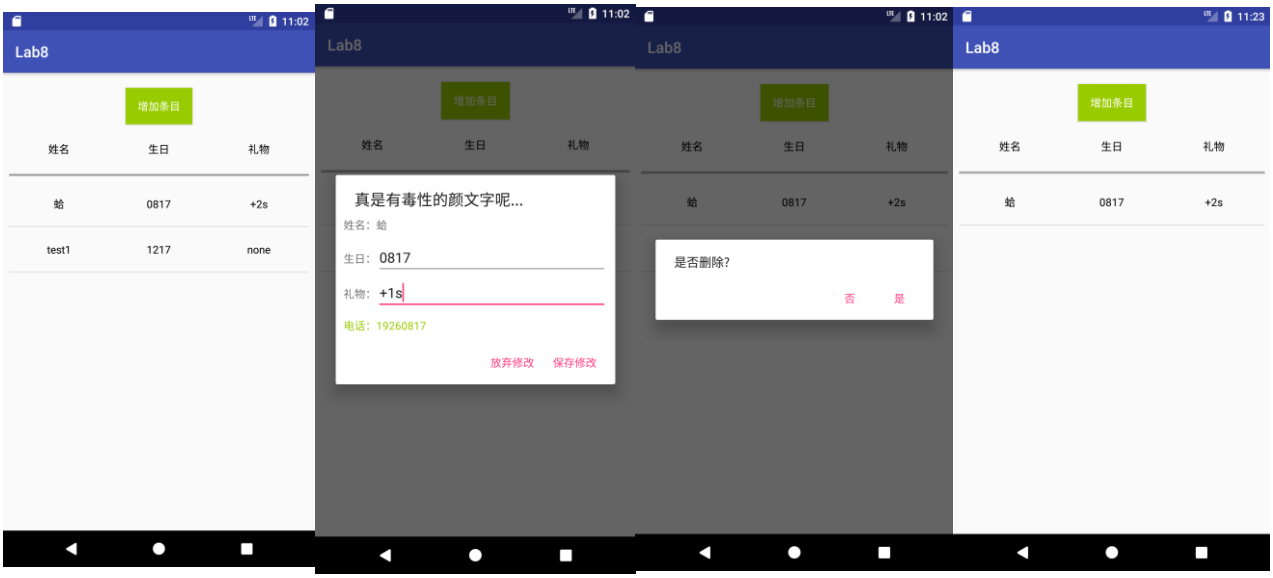
# 四、 课堂实验结果

## （一） 实验截图：

1. 虚拟机上效果如下：




初始界面　信息增加界面　姓名为空　姓名重复




信息添加　信息修改　信息删除　最终效果

## （二） 关键步骤：

1. SQlite 数据库的使用

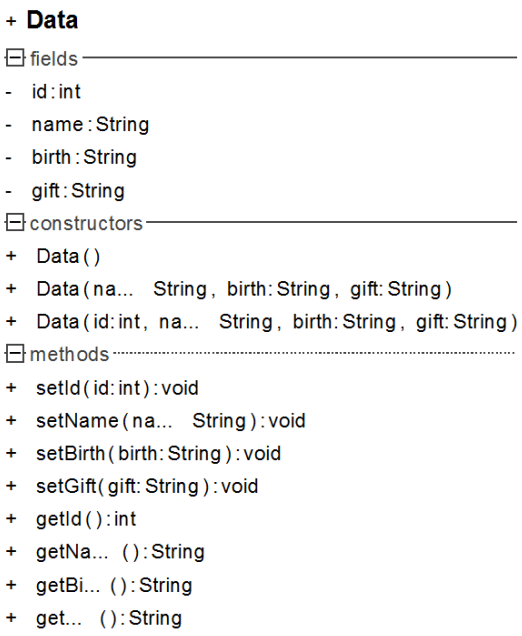
i. 创建数据的类，使用 SQLiteOpenHelper 的子类

此处需要重载两个函数 onCreate 和 onUpgrade，我们需要在 onCreate 中初始化表，此处因为我们不需要升级数据库，所以 onUpgrade 放空

```
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String NAME = "LAB8.db";
    private static final String TABLE_NAME="Birth_DATA";
    private static final int version = 1; //数据库版本

    public DatabaseHelper(Context context) {
        super(context, NAME , null, version);
    }

    @Override
```

```
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS "+TABLE_NAME
                +" (id integer primary key autoincrement, "
                + "name varchar(50) UNIQUE NOT NULL, "
                + "birth varchar(50), "
                + "gift varchar(1000)) "
                );
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

ii. 首先实现一个数据类用于存放数据




iii. 在数据库类中实现 insert、select、update、delete 的接口，此处直接定义在数据库类中需要的时候直接调用

- Insert：此处额外判断是否存在名字相同的数据

```
/**
 * 插入单个数据
 * @param data 数据类
 * @return 是否插入成功
 */
private boolean Insert(Data data){
    ContentValues cv=new ContentValues();
    cv.put("name",data.getName());
    cv.put("birth",data.getBirth());
    cv.put("gift",data.getGift());
    if(Select_id(data.getName())==-1) {//判断是否存在
        getWritableDatabase().insert(TABLE_NAME, null, cv);
        return true;
    }
    else return false;
}

/**
 * 插入上数据并返回 id
 * @param data 数据类
 * @return 插入成功返回 id  失败-1
 */
public int Insert_ReID(Data data){
    if(Insert(data)){
        return Select_id(data.getName());
    }
    return -1;
}
```

- Select：此处实现的是全搜索和按照名字搜索，全搜索中有一项一次查询最多 2M 数据返回，这个是在期中 pro 中发现的问题，所以全搜索使用先搜出全部 id 然后在通过 id 搜索单个数据

```
/**
 * 通过 name 搜索 id
 * @param name 名字
 * @return 存在返回 id 否则-1
 */
private int Select_id(String name){
    int id=-1;
    StringBuffer whereBuffer = new StringBuffer();
    whereBuffer.append("name").append(" = ").append("'").append(name).append("'");
    String[] columns = {"_id"};
    Cursor cursor = getReadableDatabase().query(TABLE_NAME, columns, whereBuffer.toString(), null, null,
null, null);
    if (cursor.moveToNext()){
        id=cursor.getInt(cursor.getColumnIndex("_id"));
    }
    if (cursor != null) {
        cursor.close();
    }
    return id;
}

/**
 * 通过 id 搜索数据
 * @param id
 * @return 对应数据
 */
private Data Select(int id){
    Data data=new Data();
    data.setId(id);
    StringBuffer whereBuffer = new StringBuffer();
    whereBuffer.append("_id").append(" = ").append("'").append(id).append("'");
    String[] columns = {"name","birth","gift"};
    Cursor cursor = getReadableDatabase().query(TABLE_NAME, columns, whereBuffer.toString(), null, null,
null, null);
    if (cursor.moveToNext()) {
        data.setName(cursor.getString(cursor.getColumnIndex("name")));
        data.setBirth(cursor.getString(cursor.getColumnIndex("birth")));
        data.setGift(cursor.getString(cursor.getColumnIndex("gift")));
    }
    if (cursor != null) {
        cursor.close();
    }
    return data;
}

/**
 * 全数据搜索
 * @return 全数据
 */
public ArrayList<Data> Select_ALL(){
    ArrayList<Data>datas=new ArrayList<>();
    ArrayList<Integer>index=new ArrayList<>();
    Cursor cursor = getReadableDatabase().query(TABLE_NAME, new String[]{"_id"}, null, null, null,
null,"_id");
    while (cursor.moveToNext()) {//先搜索 ID 再通过 ID 索引全部数据，因为有 2M 的上限
        index.add(cursor.getInt(cursor.getColumnIndex("_id")));
    }
    cursor.close();
    for(int i=0;i<index.size();i++){
        datas.add(Select(index.get(i)));
    }
    return datas;
}
```

- Update：传入数据类根据其 id 删除

```
/**
 * 更新数据
 * @param data 源数据
 */
public void Update(Data data){
    StringBuffer whereBuffer = new StringBuffer();
    whereBuffer.append("_id").append(" = ").append("'").append(data.getId()).append("'");
    ContentValues cv=new ContentValues();
    cv.put("birth",data.getBirth());
    cv.put("gift",data.getGift());
    getWritableDatabase().update(TABLE_NAME,cv,whereBuffer.toString(),null);
}
```

- Delete：和 update 类似传入数据根据其 id 更新

```
/**
 * 删除数据
 * @param data 源数据
 */
public void Delete(Data data){
    StringBuffer whereBuffer = new StringBuffer();
    whereBuffer.append("_id").append(" = ").append("'").append(data.getId()).append("'");
    getWritableDatabase().delete(TABLE_NAME,whereBuffer.toString(),null);
}
```

2. ContentProvider 的使用

i. 在 AndroidManifest 中声明权限：

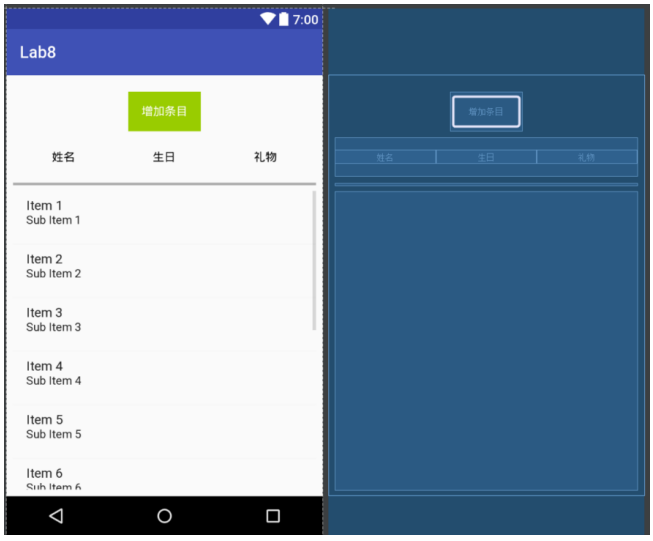
```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

ii. 实现读取联系人电话，使用方式和数据库查询类似，此处不需要判断有无只需要使用 moveToNext 判断有没有数据被查询到。

```
/**
 * 获取联系人列表
 * @param name 指定联系人名字
 * @return 联系人电话
 */
private String getPhone(String name) {
    //读取联系人列表
    String Phone = "无";
    Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
    while (cursor.moveToNext()) {//获取全部联系人
        int id = cursor.getInt(cursor.getColumnIndex("_id"));
        String name_c = cursor.getString(cursor.getColumnIndex("display_name"));
        if (name.equals(name_c)) { //判断名字是否是需要的
            Cursor phone = getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
ContactsContract.CommonDataKinds.Phone.CONTACT_ID + "=" + id, null, null);
            while (phone.moveToNext()) {//查询获取电话
                Phone = phone.getString(phone.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER)) + "
";
            }
            if (phone != null) phone.close();
        }
    }
    if (cursor != null) cursor.close();
    return Phone;
}
```

3. 主界面的设计

i. 主界面使用 ListView，适配器使用我们之前 lab3 中完成的通用适配器，界面设计如下：




ii. 主界面控件获取以及数据库内容的拉取填充

```
/**
 * 控件获取
 * 数据库内容拉取填充
 */
private void Init() {
    Add = (Button) findViewById(R.id.button);
    listView = (ListView) findViewById(R.id.listview);
```

```
    try {
        if(!databaseHelper.tabIsExist("Birth_DATA")) {//主界面数据库
            databaseHelper.Create(); //不存在数据表则创建
        }
        datas = databaseHelper.Select_ALL(); //获取全部数据
    }catch (SQLException e){

    }
    myAdapter = new MyAdapter(this, datas);
    listView.setAdapter(myAdapter);
}
```

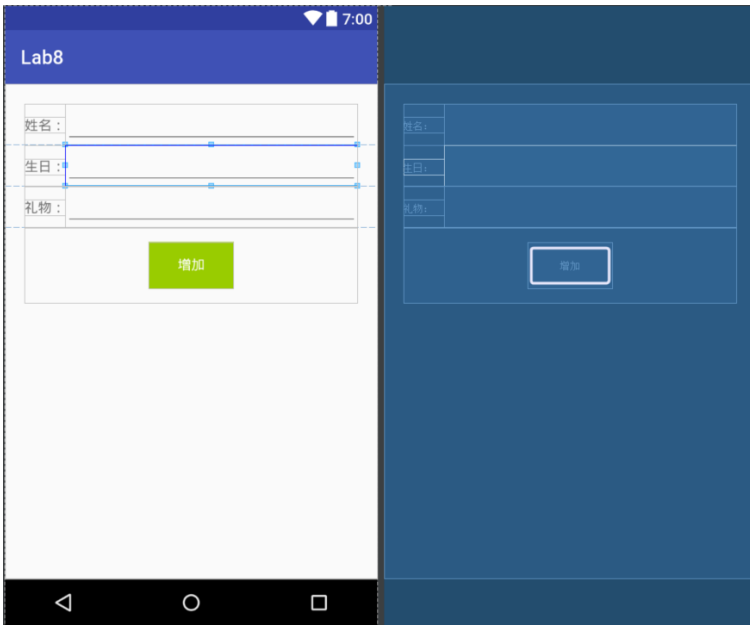
iii. 实现对应的单击弹出 dialog 修改界面，长按删除，此处直接调用之前数据库接口

```
/**
 * 点击事件
 */
private void Init_listener() {
    Add.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent(MainActivity.this, Additem.class);
            startActivity(i);
        }
    });
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view,final int i, long l) {//单击修改
            LayoutInflater inflater = LayoutInflater.from(MainActivity.this);//自定义dialog
            final View view1 = inflater.inflate(R.layout.dialoglayout, null);
            final TextView name = (TextView) view1.findViewById(R.id.dialog_name);
            final EditText birth = (EditText) view1.findViewById(R.id.dialog_birth);
            final EditText gift = (EditText) view1.findViewById(R.id.dialog_gift);
            final TextView phone = (TextView) view1.findViewById(R.id.dialog_phone);
            final AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            //填充一波数据
            name.setText("姓名："+datas.get(i).getName());
            phone.setText("电话："+getPhone(datas.get(i).getName()));
            birth.setText(datas.get(i).getBirth());
            gift.setText(datas.get(i).getGift());
            builder.setView(view1);
            builder.setTitle("真是有毒性的颜文字呢...");
            builder.setPositiveButton("保存修改", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int j) {//确定保存
                    Data data=new
Data(datas.get(i).getId(),datas.get(i).getName(),birth.getText().toString(),gift.getText().toString());
                    databaseHelper.Update(data); //数据库修改
                    datas.remove(i);
                    myAdapter.notifyDataSetChanged();//更新列表
                    datas.add(i,data);
                    myAdapter.notifyDataSetChanged();
                }
            });
            builder.setNegativeButton("放弃修改", new DialogInterface.OnClickListener() { //放弃无视
                @Override
                public void onClick(DialogInterface dialogInterface, int j) {

                }
            });
            builder.setCancelable(true);//误点击可取消
            builder.create().show();

        }
    });
    listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
        @Override
        public boolean onItemLongClick(AdapterView<?> adapterView, View view, final int i, long l) {//长按删除
            final AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setMessage("是否删除?");
            builder.setPositiveButton("是", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int j) {
                    databaseHelper.Delete(datas.get(i)); //修改数据库
                    datas.remove(i);
                    myAdapter.notifyDataSetChanged();
                }
```

```
            });
            builder.setNegativeButton("否", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {

                }
            });
            builder.setCancelable(true);//误点击可取消
            builder.create().show();
            return true;
        }
    });
}
```

4. 添加界面的实现

i. 界面布局如下：




ii. 根据事件判断，然后针对重名使用数据库接口进行判断

```
private void Init_Listener(){
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Data data=new Data(name.getText().toString(),birth.getText().toString(),gift.getText().toString());
            if(data.getName().length()==0){//姓名为空
                Toast.makeText(Additem.this,"姓名为空，请完善",Toast.LENGTH_SHORT).show();
            }
            else {
                if(databaseHelper.Insert_ReID(data)==-1) {//查询数据库
                    Toast.makeText(Additem.this, "姓名重复，请检查", Toast.LENGTH_SHORT).show();
                }else{
                    Intent intent=new Intent(Additem.this,MainActivity.class);
                    startActivity(intent);
                }
            }
        }
    });
}
```

实验中遇到的困难和解决思路：

1. 使用自定义 dialog 的时候一直崩溃，后来发现是因为 listview 中的点击事件的 i 和 dialog 取消 i 重复了，导致数据库崩溃，但是还有有一点发现数据库用 try_catch 这种好像无法捕捉到一些异常，导致 debug 困难。

# 五、 课后实验结果

1. 增加动态权限获取，不需要进入设置给予权限

```
private static final int REQUEST_READ_CONTACTS=0;
private static String[] PERMISSIONS_READ_CONTACTS = {
        Manifest.permission.READ_CONTACTS};
public static void verifyStoragePermissions(Activity activity) {
    int permission = ActivityCompat.checkSelfPermission(activity,
            Manifest.permission.READ_CONTACTS);//直接检测权限
    if (permission != PackageManager.PERMISSION_GRANTED) { //没有就去申请
        ActivityCompat.requestPermissions(activity, PERMISSIONS_READ_CONTACTS,REQUEST_READ_CONTACTS );
    }
}
```

# 六、 实验思考及感想

这次实验因为之前期中 pro 中做过数据库，所以很多数据库模块拿过来改改就可以直接使用，而 ContentProvider 的查询也和数据库一致，所以实现非常简单，主要就是数据库 debug 比较麻烦，之前也给期中 pro 数据库 debug，一开始载入数据库数据的时候如果崩溃连 debug 模式都看不到报错这也是最烦的，之前为了应对这种情况专门实现了判断数据库数据表是否存在的函数，这次算是也是派上用场，实验中给 name 定义了唯一性，想通过 try_catch 的方式在插入重复时捕获异常，但是发现数据库异常好像捕获不了，最后只能退而求其次先去查询一下是否存在。