移动应用开发实验报告

lab8：数据存储（二）

姓名	学号	年级	专业	电话	邮箱	日期
王若曦	15352319	15级	软件工程	15354222552	1511330303@qq.com	2017.12.10-2017.12.11

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

实验题目

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。

实验过程

整体思路分析：

> 首先本次实验使用SQLite轻量级数据库进行数据的存储，这个数据库之前的期中项目我们已经用到了，所以实现起来比较简单。对于该数据库的实现来说，首先定义DBHelper类进行数据库的定义和表的创建，以及一些方法的定义和实现，如查询，修改。所以java文件方面一共需要三个文件，两个Activity和一个DBHelper类；然后界面方面一共有两个Activity，一个主界面用于列表显示，一个增加界面用于数据的增加。然后对于点击列表中每个item要弹出一个对话框，写法之前实验用过，不过这次要自定义对话框的布局，所以布局xml方面一共需要定义实现四个xml文件，分别为两个Activity，一个item和一个对话框。本次实验的新功能是访问手机的联系人列表，并读取出来，所以这里需要特殊函数和权限。在后面实现中具体介绍。

SQLite数据库的定义和实现

使用SQLite数据库首先要定义Helper类用于数据库的创建，以及库中表的创建。同时定义一些以后会用到的函数。

SQLiteOpenHelper类是Android提供的一个辅助工具，用于指导我们进行SQLite的合理使用。具体使用步骤：

1. 新建一个继承自SQLiteOpenHelper的数据库操作类，重写onCreate和onUpgrade两个方法。其中onCreate方法只在第一次打开数据库时执行，在此可进行表结构的创建和操作；onUpgrade方法在数据库版本升高时执行，这次实验没有用到，所以不需要具体实现。
2. 封装保证数据库安全的必要方法，包括获取单例对象，打开数据库连接，关闭数据库连接。这里不需要我们具体实现，如关闭数据库直接调用close方法即可。
3. 提供对表记录进行增加，删除，修改，查询的操作方法。

```
    public MyDBHelper(Context context) {
        super(context, "contacts.db", null, 2);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        db.execSQL("create table Contacts (_id integer primary key autoincrement, name text,
birth text, gift text);");

    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
```

在MyDBHelper中创建数据库，即定义数据库的名字contacts.db。参数2为数据库的版本号，这里我么用不到，随便定义一个即可。表的定义根据要求，按照SQL语句规范进行定义即可。

对于数据库的增删改查操作，要用到ContentValues类，可以用键值对的方式存储要操作的数据并插入到数据库或其他操作。

```
    public void insert(String name, String birth, String gift)
    {
        SQLiteDatabase localSQLiteDatabase = getWritableDatabase();
        ContentValues localContentValues = new ContentValues();
        localContentValues.put("name", name);
        localContentValues.put("birth", birth);
        localContentValues.put("gift", gift);
        localSQLiteDatabase.insert("Contacts", null, localContentValues);
        localSQLiteDatabase.close();
    }

    public void update(String name, String birth, String gift)
    {
        SQLiteDatabase localSQLiteDatabase = getWritableDatabase();
        String[] arrayOfString = { name };
        ContentValues localContentValues = new ContentValues();
        localContentValues.put("name", name);
        localContentValues.put("birth", birth);
        localContentValues.put("gift", gift);
        localSQLiteDatabase.update("Contacts", localContentValues, "name = ?", arrayOfString);
        localSQLiteDatabase.close();
    }

    public void delete(String name)
    {
        SQLiteDatabase localSQLiteDatabase = getWritableDatabase();
        localSQLiteDatabase.delete("Contacts", "name = ?", new String[] { name });
        localSQLiteDatabase.close();
    }


    public Cursor query()
    {
        return getReadableDatabase().rawQuery("select * from Contacts", null);
    }
```

这里要注意的是，对数据库进行访问操作前要调用getWritableDatabase()方法来获取操作权限，之后才能进行操作。

不同的数据库操作的方法需要的参数也不一样，如insert方法，第一个参数为表名，第二个参数为防止insert空值时的方法，具体为：

> **nullColumnHack**：当values参数为空或者里面没有内容的时候，我们insert是会失败的（底层数据库不允许插入一个空行），为了防止这种情况，我们要在这里指定一个列名，到时候如果发现将要插入的行为空行时，就会将你指定的这个列名的值设为null，然后再向数据库中插入。

第三个参数为values，即要插入的值，类型为ContentValues。

特别的对于查询方法query，这里直接调用了rawQuery方法，该方法返回的是每一行的元组，在后面利用游标查出后调用getString和getColumnIndex方法获取对应列的值即可。

修改和删除方法还要传入参数“where”和其值。即对指定的数据进行修改和删除，符合SQL语句规范。

这里要保证contentvalues的键要和表中的列名一致。

对数据库中数据的列表显示

在MainActivity中要把数据库中的数据查询出来显示在listview上，需要用到游标cursor进行查询。这里cursor查询出来的是以每个元组的形式存储，所以每次movetonext即移动到下一行，然后再根据键取出对应数据即可。

```
Cursor localCursor = this.myDatabase.query();
while (localCursor.moveToNext())
{
    LinkedHashMap temp = new LinkedHashMap();
    temp.put("name",localCursor.getString(localCursor.getColumnIndex("name")));
 temp.put("birth",localCursor.getString(localCursor.getColumnIndex("birth")));
temp.put("gift",localCursor.getString(localCursor.getColumnIndex("gift")));
    this.listitem.add(temp);
}
```

将每次cursor的对应的键值利用temp存储到listitem数组中，用于后面对listview进行setadpter。

这里adpter用的是simpleadpter，然后利用刚才保存的list数组listitem进行值的绑定。这里没有使用simplecursoradpter直接绑定是因为后面要对listview进行实时更新，即每次修改或新增之后都要及时更新listview，所以要把listitem清空然后再重新setadpter。用simplecursoradpter的话不知道如何把查出的cursor清空，所以直接用原来的方法了。

关于cursor的具体用法：

> 调用query方法查询出来后，cursor相当于一个数组，通过movetonext方法进行遍历。
>
> 每个当前的cursor是以一个元组的形式存在，所以可以通过getString（列）方法获取对应列的值，而列可以通过调用getColumnIndex（键）来获取。这里的键即为定义表时的列名。

点击listview事件触发

这里设置listview中item的onclick事件触发。点击每个item会弹出一个对话框显示当前item信息，并可以进行修改，所以要自定义对话框的布局并事件按钮的触发事件；长按item会弹出对话框询问是否删除，这个对话框不用自定义布局，直接使用默认布局，然后定义两个按钮“是”和“否”即可。

```
LayoutInflater factor = LayoutInflater.from(MainActivity.this);
View view_in = factor.inflate(R.layout.dialog,null);
final AlertDialog.Builder alertDialog1 = new AlertDialog.Builder(MainActivity.this);
alertDialog1.setView(view_in);
final TextView name = (TextView) view_in.findViewById(R.id.show_name);
final EditText birth = (EditText) view_in.findViewById(R.id.show_birth);
final EditText gift = (EditText) view_in.findViewById(R.id.show_gift);
final TextView phone = (TextView) view_in.findViewById(R.id.show_phone);
name.setText(((Map)MainActivity.this.listitem.get(i)).get("name").toString());
birth.setText(((Map)MainActivity.this.listitem.get(i)).get("birth").toString());
gift.setText(((Map)MainActivity.this.listitem.get(i)).get("gift").toString());
```

自定义对话框使用LayoutInflater类，将dialog布局中的内容全部显示在弹出的对话框中，并对其中的的元素进行操作。如这里的birth和gift为edittext控件，所以可以在对话框中进行修改即对数据库进行修改。

然后对对话框的标题和按钮进行设置。分别调用setTitle,setPositiveButton和setNegativeButton方法。

这里positivebutton为保存修改，即将当前edittext的值update到数据库中。调用之前在Helper类中定义的update方法即可，将对应的参数传进去。这里要注意的是要对list数组listitem进行清空，即调用clear方法。因为此时数据库已经修改，所以之前查询出来绑定到listview上的数据要进行更新，所以这里我们清空之后重新利用游标cursor查询一次，并再全部存到listitem中，然后进行setadpter绑定。

```
.  setPositiveButton("保存修改", new DialogInterface.OnClickListener(){
public void onClick(DialogInterface face, int ind)
{
 MainActivity.this.myDatabase.update(((Map)MainActivity.this.listitem.get(i)).get("name").toStri
ng(), birth.getText().toString(), gift.getText().toString());
MainActivity.this.listitem.clear();
Cursor localCursor3 = MainActivity.this.myDatabase.query();
while(localCursor3.moveToNext())
{
    LinkedHashMap localLinkedHashMap = new LinkedHashMap();
                                  localLinkedHashMap.put("name",
localCursor3.getString(localCursor3.getColumnIndex("name")));
                                localLinkedHashMap.put("birth",
localCursor3.getString(localCursor3.getColumnIndex("birth")));
                                localLinkedHashMap.put("gift",
localCursor3.getString(localCursor3.getColumnIndex("gift")));
MainActivity.this.listitem.add(localLinkedHashMap);
}
final SimpleAdapter adpter = new SimpleAdapter(getApplicationContext(),
MainActivity.this.listitem , R.layout.item, new String[] { "name", "birth", "gift" }, new int[]
{ R.id.name, R.id.birth, R.id.gift });
list.setAdapter(adpter);
}
```

这里如果不清空直接setadpter的话会在原来的基础上增加，所以要先clear掉。

negativebutton为放弃修改，即直接弹出Toast信息即可。

```
.setNegativeButton("放弃修改", new DialogInterface.OnClickListener(){
                        public void onClick(DialogInterface face2, int ind2)
                        {
                            Toast.makeText(MainActivity.this.getApplicationContext(), "修改已放
弃", Toast.LENGTH_SHORT).show();
                        }
```

最后要记得调用create，然后调用show显示出来。

长按某个item会弹出对话框询问是否删除，positivebutton和negativebutton分别为是和否。这次不需要自定义布局，所以比较简单，builder定义好之后直接电泳setitle等方法即可。

如果确定删除的话，则调用delete方法将对应数据从数据库中删除，然后再将数组listitem清空，重新查询，add。最后setadpter。

否的话直接弹出Toast信息。

```
AlertDialog.Builder dialog2 = new AlertDialog.Builder(MainActivity.this);
                dialog2.setTitle("是否删除？ ").setPositiveButton("是", new
DialogInterface.OnClickListener(){
                public void onClick(DialogInterface face3, int ind3){
                    String str =
((Map)MainActivity.this.listitem.get(i1)).get("name").toString();
                    MainActivity.this.myDatabase.delete(str);
                    MainActivity.this.listitem.clear();
                    Cursor localCursor = MainActivity.this.myDatabase.query();
                    while (localCursor.moveToNext())
                    {
                        LinkedHashMap localLinkedHashMap = new LinkedHashMap();
                        localLinkedHashMap.put("name",
localCursor.getString(localCursor.getColumnIndex("name")));
                        localLinkedHashMap.put("birth",
localCursor.getString(localCursor.getColumnIndex("birth")));
                        localLinkedHashMap.put("gift",
localCursor.getString(localCursor.getColumnIndex("gift")));
                        MainActivity.this.listitem.add(localLinkedHashMap);
                    }
                    final SimpleAdapter adpter = new SimpleAdapter(getApplicationContext(),
MainActivity.this.listitem, R.layout.item, new String[] { "name", "birth", "gift" }, new int[] {
R.id.name, R.id.birth, R.id.gift });
                    list.setAdapter(adpter);
                }
            }).setNegativeButton("否",new DialogInterface.OnClickListener(){
                    public void onClick(DialogInterface face4, int in4){
                        Toast.makeText(MainActivity.this.getApplicationContext(), "不删除联系人",
Toast.LENGTH_SHORT).show();
                    }
                }).create();
                dialog2.show();
```

使用**ContentProvider**读取联系人信息

我们需要读取手机中联系人信息并显示，所以要用到ContentProvider。

首先要在AndroidManifest中声明权限，得以访问手机中的联系人信息。

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

然后通过游标cursor调用getContentResolver方法读取联系人列表。这里cursor查询出来的也是以每个元组的形式存储的。

```
Cursor localCursor1 =
MainActivity.this.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null, null,
null, null);
```

首先我们要判断通讯录中是否存在我们点击的item中的人的信息。所以首先获取当前cursor的姓名

```
String get_name =
localCursor1.getString(localCursor1.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLA
Y_NAME));
```

然后进行判断当前cursor的name是否为我们点击的item的name，如果是的话则进行读取手机号并settext显示。

```
 if(get_name.equals(((Map)MainActivity.this.listitem.get(i)).get("name").toString()))
                {
                    String str1 = localCursor1.getString(localCursor1.getColumnIndex("_id"));
                    Cursor localCursor2 =
MainActivity.this.getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
 null, "contact_id=" + str1, null, null);
                    str2 = "";
                    if
(Integer.parseInt(localCursor1.getString(localCursor1.getColumnIndex("has_phone_number"))) > 0)
                        while (localCursor2.moveToNext())
                        {
                            String str3 =
localCursor2.getString(localCursor2.getColumnIndex("data1"));
                            str2 = " " + str3;
                        }
                    phone.setText(str2);
                }
```

如果是的话，首先获取当前联系人的id，跟后根据该id查询电话信息。

然后判断该联系人信息中是否有电话号码，如果有的话则读取电话号码信息保存到string中，最后settext显示在phone的textview上。

如果没有的话则将phone这个textview的值set为无。

```
                if(str2.equals("")&&str2.length()==0){
                    phone.setText("无");
                }
```

MainActivity的及时刷新

每次增加条目后会返回MainActivity，此时需要及时更新listview中的数据，即要把刚刚新增的条目也显示出来，所以此时要重写onStart方法。

onStart方法在每次对应Activity显示时调用，即每次这个Activity被显示出来的时候都会调用这个方法。所以我们可以在这个方法里定义实时更新的方法。

实时更新即需要对数据库进行查询，增加完后返回MainActivity，此时再进行一次查询并setAdpter，即可实现实时更新。方法和前面的修改删除一样，将数组listitem清空，利用cursor重新查询，add，然后setadpter。

```
    protected void onStart()
    {
        super.onStart();
        this.listitem.clear();
        Cursor localCursor = this.myDatabase.query();
        while (localCursor.moveToNext())
        {
            LinkedHashMap localLinkedHashMap = new LinkedHashMap();
            localLinkedHashMap.put("name",
localCursor.getString(localCursor.getColumnIndex("name")));
            localLinkedHashMap.put("birth",
localCursor.getString(localCursor.getColumnIndex("birth")));
            localLinkedHashMap.put("gift",
localCursor.getString(localCursor.getColumnIndex("gift")));
            this.listitem.add(localLinkedHashMap);
        }
        SimpleAdapter localSimpleAdapter = new SimpleAdapter(getApplicationContext(),
this.listitem, R.layout.item, new String[] { "name", "birth", "gift" }, new int[] {
R.id.name,R.id.birth, R.id.gift });
        list.setAdapter(localSimpleAdapter);
    }
```

之前不知道有onstart这个方法，所以期中项目为了实现这个实时更新用的是onnewintent。但是还要根据传的intent进行更新，比这个方法麻烦很多。所以这个方法感觉很实用。

增加条目界面

增加条目界面比较简单，将edittext控件中输入的值get之后通过button的监听器，调用之前定义好的insert方法进行插入即可。

由于不可添加空值，所以要进行判断，当有某一项或几项输入为空时要提示不可为空；或者name不可重复，这里需要对数据库中数据利用cursor遍历查询，如果已经存在则不能重复定义。

```
        this.add_button.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View paramAnonymousView)
            {
                //Additem.access$002(Additem.this, false);
                if (TextUtils.isEmpty(Additem.this.Name.getText().toString()))
                    Toast.makeText(Additem.this.getApplicationContext(), "名字为空，请完善",
Toast.LENGTH_SHORT).show();
                else
                {
                    Cursor localCursor = Additem.this.myDatabase.query();
                    while (localCursor.moveToNext())
                    {
                        if
(localCursor.getString(localCursor.getColumnIndex("name")).equalsIgnoreCase(Additem.this.Name.ge
tText().toString()))
                        {
                            Toast.makeText(Additem.this.getApplicationContext(), "名字重复啦，请
检查", Toast.LENGTH_SHORT).show();
                            exit=1;
                        }
                    }
                    if(exit==0)
                    {
                        Additem.this.myDatabase.insert(Additem.this.Name.getText().toString(),
Additem.this.Birthday.getText().toString(), Additem.this.Gift.getText().toString());
                        Additem.this.finish();
                    }
                }
            }
        });
```

实验总结

实验过程中遇到的问题：

1. 忘记在AndroidManifest文件中增加Activity。每次增加界面后，都会忘记在AndroidManifest文件中增加对应的Activity，导致跳转时闪退。不过还好有报错信息，否则会debug好久。
2. 布局方面。写布局时感觉约束布局很麻烦，虽然可以在design里直接拖动设置，但是感觉很乱，而且又是没有约束好的话控件会乱跳，导致整个页面乱掉。所以一般我都用线性布局，然后嵌套起来用，感觉比较方便。尤其不需要手动对齐等操作。
3. 自定义对话框。自定义对话框时要自己写布局文件，和其他布局文件一样，只是把对话框的窗口当做整个窗口即可。第一次实验的时候我也想自定义一个对话框，但是当时调用LayoutInflater类一直报错，不知道为什么。可能是但是还不太会写，所以写的位置不对，或者没有添加依赖。
4. 手机联系人信息的读取。读取手机通讯录首先获取权限，和获取拍照，通知一样。这里我认为手机的通讯录也是以数据库中表格的形式存储在手机中，我们访问时也是对这个表进行查询，并获取对应的数据。只不过语句特殊一点，不清楚表格的列名是怎么定义的，上网查一下即可。
5. onStart的使用。这次实验知道了onStart这个方法的使用，之前不知道，都是用onnewintent来实现这个功能的，实现起来很麻烦。这次实验最大的收获就是这个了，在Activity显示时调用的方法，有很大的实际用处。
6. 点击事件没有处理。一开始写完运行时，点击某个item时没有反应，也没有报错信息。显示点击被搁置了好像（过了很久才写实验报告，忘了之前的报错是什么了）。然后上网查了一下那个信息有人说是布局里嵌套

太多了导致点击没有传递进去。但是我只写了一层嵌套，之前也这么写过没有出现这个问题。所以找了很久发现好像是控件要通过MainActivity.this，具体是什么原因记不清了，不好意思。

本次实验主要是SQLite数据库的使用，但由于之前使用过，所以实现起来比较简单，主要是对话框的自定义和手机通讯录的读取没做过，所以主要时间用在了这两个方面。感觉手机系统各功能的访问通过权限都有专门的方法调用，所以感觉比较方便。对话框的自定义布局只要将写好的布局setview进去即可。这次实验最大的收获感觉是onstart方法，因为之前确实不知道有这个方法，也没看到过，可能是没注意。所以之前使用onnewintent的坑感觉都能用onstart找回来了。