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

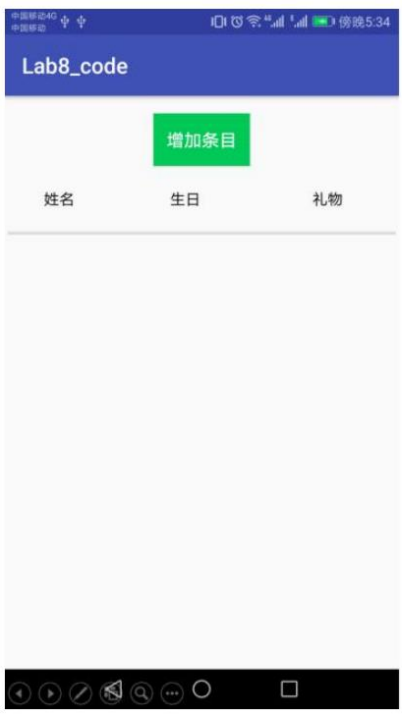
年级	15 级	专业（方向）	软件工程(移动)
学号	15352133	姓名	黄少豪
电话	13727024545	Email	328730316@qq.com
开始日期	2017.12.14	完成日期	2017.12.15

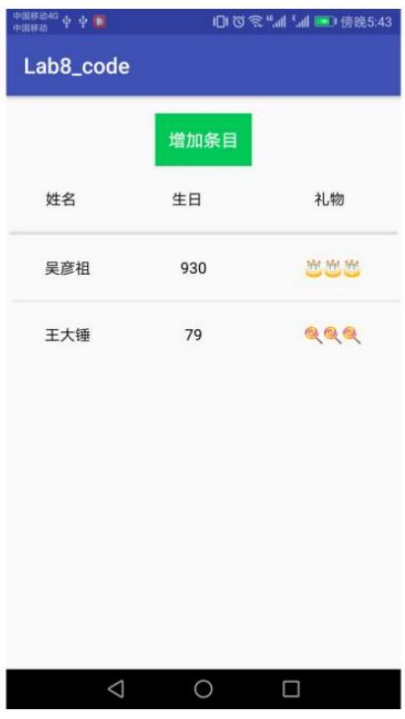
一、 实验题目

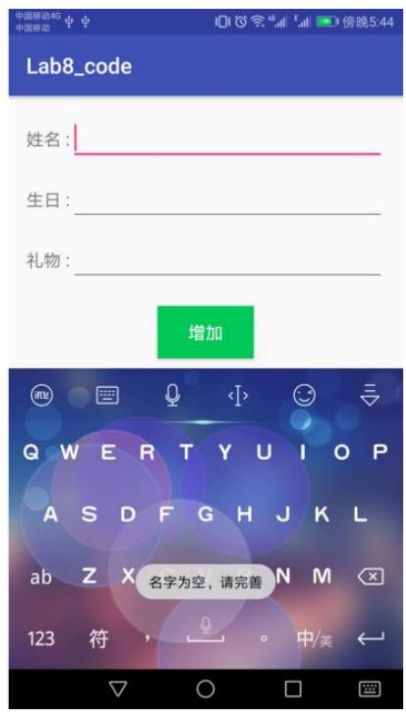
数据存储（二）

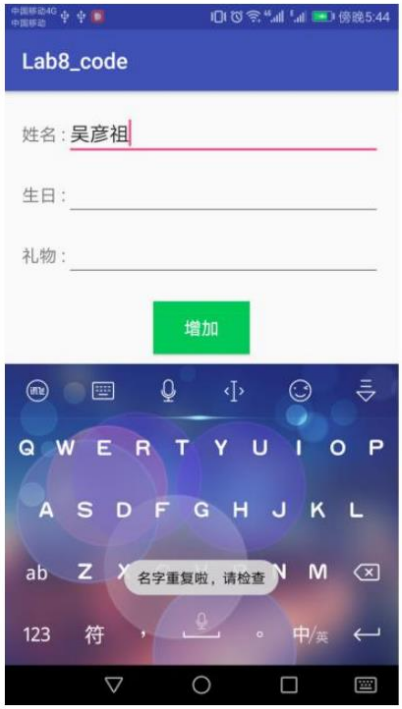
二、 实现内容

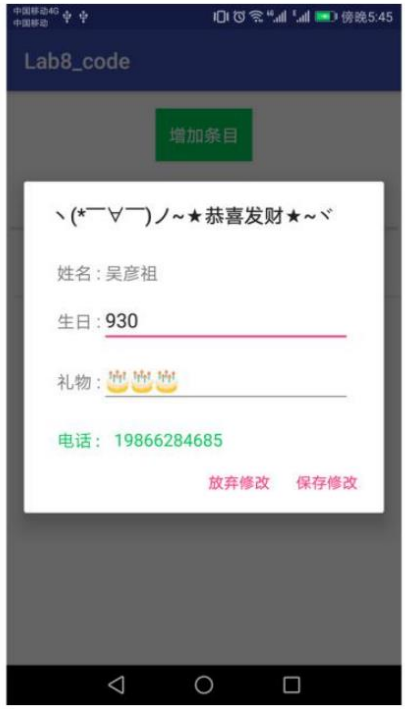
1. 实现一个生日备忘录，要求使用 SQLite 数据库存储数据，使得每一次打开应用都能看到存储在数据库里的内容
2. 使用 ContentProvider 获取手机通讯录中的电话号码
3. 点击主界面上方的添加按钮可进入信息界面编辑添加条目信息
4. 点击列表每一个条目都会弹出一个信息编辑窗口供修改信息
5. 列表项的姓名属性值不能为空，不能重复
6. 长按列表项可删除条目信息

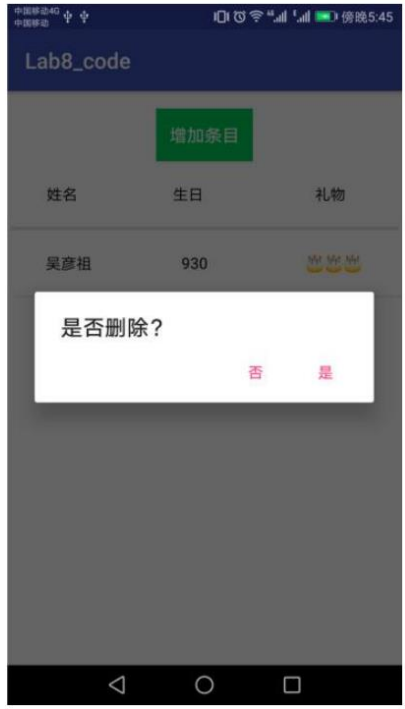






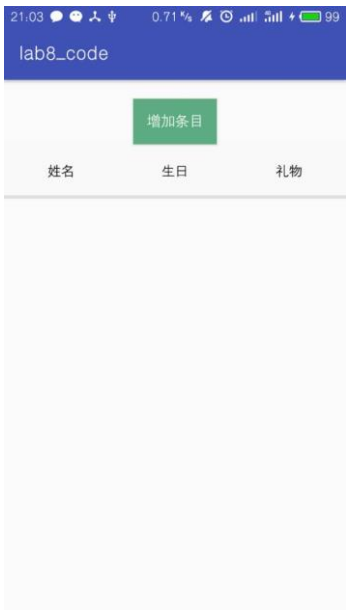


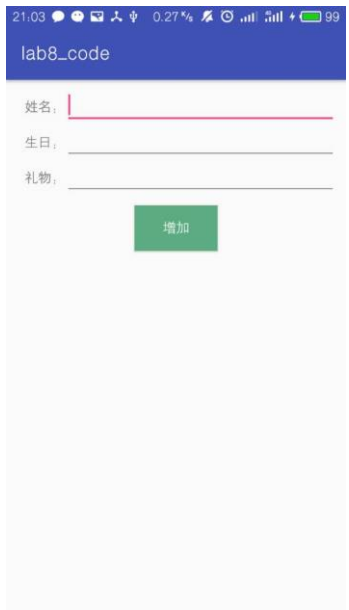


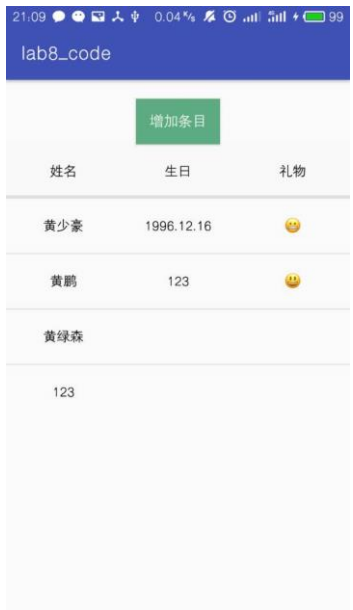


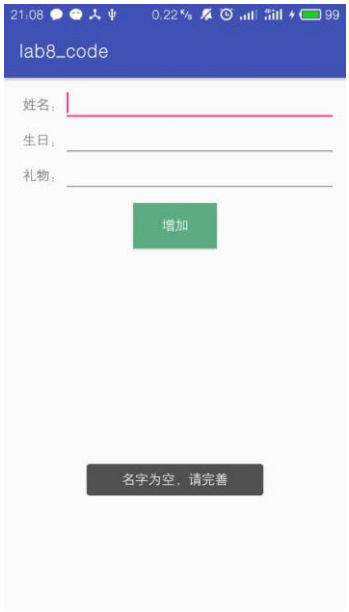
三、 课堂实验结果

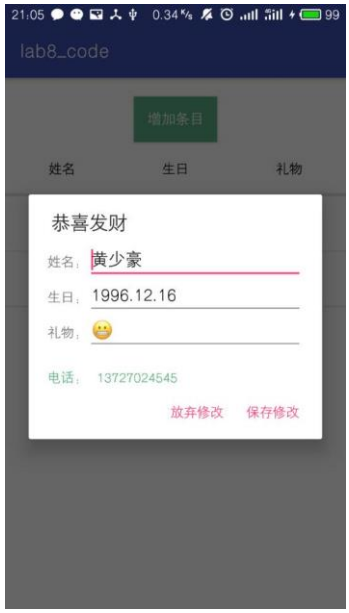
（1） 实验截图

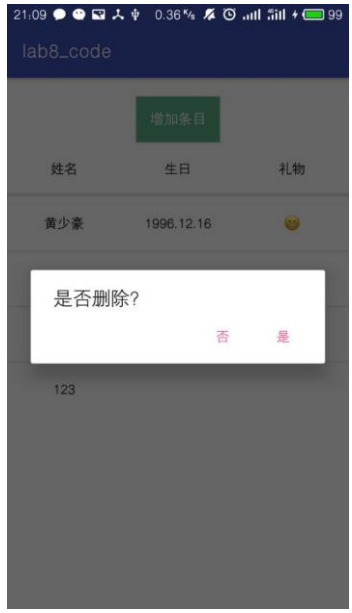


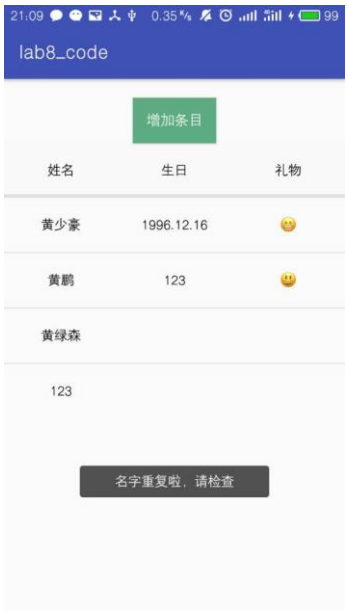












（2） 实验步骤以及关键代码

1. 主界面 UI 搭建

主界面布局主要是一个 Button，三个 TextView 和一条背景下划线以及一个 ListView。ListView 使用 SimpleAdapter 作为适配器。进入应用时，需要先给适配器配置一个列表，列表数据来自于 SQLite 数据库

```
public void getDatas() {
    List<Map<String, String> >data = mydb.query();
    list.clear();
    nameSet = new HashSet<>();
    for (int i=0;i<data.size();++i) {
        Map<String, String> map = data.get(i);
        map.put("phone", findPhone(map.get("name").trim()));
        list.add(map);
        nameSet.add(map.get("name").trim());
    }
}
list = new ArrayList<>();
nameSet = new HashSet<>();
getDatas();
listView = (ListView) findViewById(R.id.activity_main_list);
adapter = new SimpleAdapter(this, list, R.layout.list_item,
        new String[] {"name", "birthday", "gift"},
        new int[] {R.id.list_item_name, R.id.list_item_birthday, R.id.list_item_gift});
listView.setAdapter(adapter);
```

点击 Button 能进入添加数据界面。方法是给 Button 设置一个点击事件监听器，当发现有点击事件时，通过 Intent 进入数据添加界面。

```
button = (Button) findViewById(R.id.activity_main_add_item);
button.setOnClickListener((view) -> {
        Intent intent = new Intent(MainActivity.this, ItemActivity.class);
        startActivityForResult(intent, 1);
});
```

点击列表某一条记录，会出现一个对话框，供查看具体信息与修改信息。长按记录会弹出一个窗口，询问是否删除记录(nameSet 是一个 Set 对象，用来防止名字重复)。

```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
builder.setTitle("是否删除？");
builder.setNegativeButton("否", (dialog, which) -> {
        dialog.dismiss();
});
builder.setPositiveButton("是", (dialog, which) -> {
        dialog.dismiss();
        mydb.delete(list.get(position).get("name").toString().trim());
        nameSet.remove(list.get(position).get("name").toString().trim());
        list.remove(position);
        adapter.notifyDataSetChanged();
});
Dialog dialog = builder.create();
dialog.setOnDismissListener((dialog) -> { tag = true; });
dialog.show();
```

2. 添加信息界面搭建

add 函数是点击增加按钮后调用执行的函数。基本流程是：(1)获取三个文本编辑框的文本；(2)判断姓名字段是否为空；(3)判断姓名字段

是否重复；(4)将数据返回给 MainActivity 插入数据到数据库。

```
public void add(View view) {
    Bundle bundle = new Bundle();
    EditText name = (EditText) findViewById(R.id.activity_item_name_edit);
    //如果名字为空，弹出信息提示
    if (name.getText().toString().trim().length() == 0){
        Toast.makeText(this, "名字为空，请完善", Toast.LENGTH_SHORT).show();
        return;
    }
    //如果名字重复，弹出信息提示
    if (MainActivity.name_set.contains(name.getText().toString().trim())) {
        Toast.makeText(this, "名字重复啦，请检查", Toast.LENGTH_SHORT).show();
        return;
    }
    //将数据压入bundle返回给MainActivity将数据插入数据库
    bundle.putString("name", name.getText().toString());
    EditText birthday = (EditText) findViewById(R.id.activity_item_birthday_edit);
    bundle.putString("birthday", birthday.getText().toString());
    EditText gift = (EditText) findViewById(R.id.activity_item_gift_edit);
    bundle.putString("gift", gift.getText().toString());
    Intent intent = new Intent();
    intent.putExtras(bundle);
    setResult(RESULT_OK, intent);
    finish();
}
```

MainActivity 接受到返回数据后将数据插入数据库

```
String name = bundle.getString("name");
String birthday = bundle.getString("birthday");
String gift = bundle.getString("gift");
mydb.insert(name, birthday, gift);
```

更新列表数据

```
Map<String, String> map = new HashMap<>();
map.put("name", name);
map.put("birthday", birthday);
map.put("gift", gift);
map.put("phone", findPhone(name));
list.add(map);
adapter.notifyDataSetChanged();
```

3. 修改信息对话框布局

先定义一个布局文件 dialoglayout.xml 用来规定对话框的布局。

在 MainActivity 中给列表的记录设置一个点击监听事件，当出现点击事件时，弹出对话框。使用 LayoutInflater 对象的 inflate 方法动态将对话框布局对象引入 MainActicity。使用 setView 方法将引入的布局作用于对话框。

```
listView.setOnItemClickListener((parent, view, position, id) -> {
        if (tag == false) return;
        tag = false;
        Map<String, Object> map = (Map<String, Object>) adapter.getItem(position);
        LayoutInflater factor = LayoutInflater.from(MainActivity.this);
        final View view_in = factor.inflate(R.layout.dialoglayout, null);
        final AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setTitle("恭喜发财");
        builder.setView(view_in);
```

给对话框设置一个确定按钮，点击确定按钮后保存编辑信息在列表中并通过更新操作更新存储在数据库的数据。nameSet 存储数据库姓名字段的值，用来判断姓名字段是否有重复。

```
builder.setPositiveButton("保存修改", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        dialog.dismiss();
        Map<String, Object> map = (Map<String, Object>) adapter.getItem(position);
        EditText name = (EditText) view_in.findViewById(R.id.dialoglayout_name_edit);
        String s = map.get("name").toString().trim();
        //判断修改信息有没有出现姓名重复
        if (!name_set.contains(name.getText().toString().trim()) || s.equals(name.getText().toString().trim())) {
```

若修改后没有出现姓名重复，则更新列表数据

```
if (!name_set.contains(name.getText().toString().trim()) || s.equals(name.getText().toString().trim())) {
    map.put("name", name.getText().toString().trim());
    EditText birthday = (EditText) view_in.findViewById(R.id.dialoglayout_birthday_edit);
    map.put("birthday", birthday.getText().toString().trim());
    EditText gift = (EditText) view_in.findViewById(R.id.dialoglayout_gift_edit);
    map.put("gift", gift.getText().toString().trim());
    String phoneNumber = findPhone(name.getText().toString().trim());
    map.put("phone", phoneNumber);
    adapter.notifyDataSetChanged();
```

更新数据库数据，其中第一个参数 s 是原姓名字段的字符串

```
mydb.update(s, map.get("name").toString().trim(),
        map.get("birthday").toString().trim(), map.get("gift").toString().trim());
```

更新 nameSet，删去旧的姓名，加入新的姓名

```
nameSet.remove(s);
nameSet.add(name.getText().toString().trim());
```

若有出现姓名重复，则弹出 Toast 提示

```
} else
    Toast.makeText(MainActivity.this, "名字重复啦，请检查", Toast.LENGTH_SHORT).show();
```

4. SQLite 数据库搭建

   构造函数

```
public class myDB extends SQLiteOpenHelper{
    private static final String DB_NAME = "Contacts.db";
    private static final String TABLE_NAME = "Contacts";
    private static final int DB_VERSION = 1;

    public myDB(Context context, String name, SQLiteDatabase.CursorFactory cursorFactory, int version){
        super(context, DB_NAME, cursorFactory, DB_VERSION);
    }
```

onCreate 函数。OnCreate 函数会在数据库对象第一次产生时被调用，之后对于同一个数据库对象均不会再次调用。在这里，myDB 对象第一次调用 getWritableDatabase 方法时调用一次 onCreate 函数而之后每一次调用 getWritableDatabase 方法都不会再调用 onCreate 函数。所以对数据库的创建数据表操作可以写到 onCreate 函数中。

```
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_TABLE = "create table " + TABLE_NAME
            + "(name text, "
            + "birthday text, "
            + "gift text);";
    db.execSQL(CREATE_TABLE);
}
```

insert 函数负责将记录插入数据库。先使用 getWritableDatabase 方法获取数据库对象。然后 ContentValues 对象存储记录的属性及其属性值。最后调用数据库对象的 insert 方法完成数据插入操作。

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

update 函数负责更新数据库中的数据。先使用 getWritableDatabase 方法获取数据库对象。然后使用 ContentValues 对象存储更新后的记录，whereClause 存储条件判断语句，whereArg 存储针对 whereClause 中的？的替换参数(按顺序)。最后调用数据库对象的 update 方法完成数据更新操作。

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

delete 函数流程同 update。

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

query 函数负责返回数据库中的全部数据，返回类型是 Listu 对象，存储一个个 Map 对象，每个 Map 对象对应一条记录。

```
public List<Map<String, String>> query() {
    SQLiteDatabase db = getWritableDatabase();
    Cursor cursor = db.query(TABLE_NAME, new String[]{"name", "birthday", "gift"},
            null, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();

    List<Map<String, String>> list = new ArrayList<>();
    while (cursor != null && cursor.moveToNext()) {
            final String name = cursor.getString(cursor.getColumnIndex("name"));
            final String birthday = cursor.getString(cursor.getColumnIndex("birthday"));
            final String gift = cursor.getString(cursor.getColumnIndex("gift"));
            list.add(new HashMap<String, String>(){{
                put("name", name);
                put("birthday", birthday);
                put("gift", gift);
            }});
    }
    db.close();
    return list;
}
```

5. 通讯录访问

在 AndroidManifest.xml 文件中加入 permission

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

先获取一个 cursor 可以遍历通讯录

```
public String findPhone(String name) {
    String number = "";
    Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
            null, null, null, null);
```

遍历通讯录每条记录，判断记录是否存在电话号码

```
while (cursor != null && cursor.moveToNext()) {
    int isHas = Integer.parseInt(cursor.getString(cursor.getColumnIndex(ContactsContract
            .Contacts.HAS_PHONE_NUMBER)));
```

如果存在电话号码，先获取联系人姓名，判断这个姓名是否是我们要查找的联系人姓名，若是则将这条记录的电话号码添加到 number 字符串内。

```
if (isHas > 0) {
    String nname = cursor.getString(cursor.getColumnIndex(ContactsContract
            .Contacts.DISPLAY_NAME));
    if (!nname.equals(name)) continue;
    String contactID = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts._ID));
    Cursor phone = getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
            null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID + "=" + contactID, null, null);
    while (phone != null && phone.moveToNext()) {
        number += phone.getString(phone.getColumnIndex(ContactsContract.CommonDataKinds
            .Phone.NUMBER)) + " ";
    }

}
```

因为一条记录可能不止一个电话号码，所以需要使用一个 cursor 对象来遍历电话号码，将所有遍历到的电话号码存到 number 字符串内。

（3） 实验遇到困难以及解决思路

1. Cursor 的 moveToFirst 方法问题。一开始参考网上的代码，发现很多人创建 cursor 对象后都会先调用其 moveToFirst 方法。于是模仿了一波发现应用总是第一条记录没有。后来将 moveToFirst 方法去掉后就正常了。原因是 moveToFirst 方法使 cursor 对象返回到第一条记录。但我执行 while 循环时，里面的条件使用的是 cursor.moveToNext()，这样导致了 cursor 对象遍历时总是跳过第一条记录从第二条记录开始取数据，所以总是取不到第一条记录。

2. 长按连带点击事件触发问题。一开始没有考虑这个问题，写完代码后发现每次长按总会同时触发长按事件与点击事件，导致出现两个对话框，交互体验很差。解决方法是设置一个 tag，然后给对话框添加一个 dismiss 事件监听器。当有对话框弹出时设置 tag 为 false，当对话框

弹出时 tag 设置为 true。对于每个对话框事件，开始时都判断一下 tag，若 tag 为 false 则取消本次对话框弹出行为。这样当长按事件触发弹出对话框后，点击事件即使触发但是因为 tag 已经设置为 false 所以也不会弹出对话框。这样就解决了两个对话框弹出的问题了。

3. 应用登入后出现闪退。原因是应用访问通讯录但没有通讯录访问权限。解决方法是手动在系统设置中给这个应用分配访问通讯录权限。

四、 课后实验结果

五、 实验思考及感想

1. SQLiteOpenHelper 对象中的 onCreate 方法只有在数据库对象产生时才会被系统调用（第一次调用 getWritableDatabase 或 getReadableDatabase），同一个对象之后若要重新调用这个方法只能人为手动调用。

2. 本次实验中，寻找姓名在通讯录中对应的电话号码，我使用的方法是每次插入或修改记录时都重新遍历一次通讯录。这样当通讯录记录比较多时效率会很差。一种可行方法是将电话号码也存入数据库，但这样需要在通讯录信息修改后通知数据库也修改数据，否则会出现数据不准确的现象。

3. 在应用中每次添加或更新删除记录时，都会调用一次数据库操作。这种方法当数据库不在本地而在远端时，效率会比较低。一种可行的方法是每次进入应用时，先把数据缓存到一个列表，然后每次对数据操作都是对这个列表的数据进行操作而不进行数据库操作。在应用退出时或是在等待一段时间后再将列表中变化的数据批量更新到数据库。但这种方法也有一个弊端，那就是当应用出现不正常行为时，数据会丢失。另一种可行方法就是使用多线程，数据库操作都交给另一个线程来处理，这样即使网络环境处于拥塞状态，应用也不会因为需要连接远程数据库而导致应用等待时间过长，导致与用户的交互变慢变卡。