Android程序设计

数据库和提供者

2019.5.12

isszym   sysu.edu.cn

内容

SQLite的基本操作

参考　参考

- 创建和打开数据库

第一种方式：继承SQLiteOpenHelper打开或创建数据库

```
public class DBOpenHandler extends SQLiteOpenHelper {
   private final String TAG = "MySQLiteOpenHelper";
   public DBOpenHandler(Context context, String dbName,
               SQLiteDatabase.CursorFactory factory, int dbVersion){
      super(context, dbName, factory, dbVersion);
      Log.d(TAG, "MySQLiteOpenHelper"); //这个方法会自动创建数据库
   }
   @Override
   public void onCreate(SQLiteDatabase db) {
      Log.d(TAG, "onCreate"); //可以在这里创建表
   }
   @Override   // oldVersion由系统提供， newVersion由构造器提供
   public void onUpgrade(SQLiteDatabase db,
                                    int oldVersion,int newVersion){
      Log.d(TAG, "onUpgrade"); //版本变化时做一些事，如，为某个表增加一列

   }
}
```

* CursorFactory用于查询时返回查询数据

使用方法：

```
private DBOpenHandler dbOpenHandler = new DBOpenHandler(this.getContext(),
                                                        "dbStu.db3", null, 1);
SQLiteDatabase db = dbOpenHandler.getReadableDatabase(); // 用于query
SQLiteDatabase db = dbOpenHandler.getWritableDatabase(); // 其它操作
……
db.close();                     // 记得关闭数据库操作
```

验证数据库是否创建成功：

```
List<Pair<String, String>> ll = db.getAttachedDbs();
for (int i = 0; i < ll.size(); i++) {
      Pair<String, String> p = ll.get(i);
      Log.d("测试", p.first + "=" + p.second);
}
```

打印结果

```
11-19 20:58:51.845 3422-3422/cn.bluemobi.dylan.sqlite D/SQLite: main
 =/data/data/cn.bluemobi.dylan.sqlite/databases/info
```

删除数据库：this.deleteDatabase(File file) ;

例如： this. deleteDatabase("dbStu.db3");

参考

第二种方式：Context.openOrCreateDatabase打开或创建数据库

```
SQLiteDatabase Context.openOrCreateDatabase(String dbName,int mode,
                                            CursorFactory factory)
   // 数据库名,模式,游标工厂   优点:可以指出操作模式（实际上会转化第三种方式）
```

mode: 操作模式，取值 0或MODE_PRIVATE为默认操作，取值CREATE_IF_NECESSARY表示如果数据库不存在则创建它。

例
```
SQLiteDatabase db = this.openOrCreateDatabase("dbStu.db3",
                                              MODE_PRIVATE, null);
```

第三种方式：SQLiteDatabase.openOrCreateDatabase打开或创建数据库

```
public static SQLiteDatabase openOrCreateDatabase (String path,
                             SQLiteDatabase.CursorFactory factory)
      // 数据库路径， 游标工厂    优点：可以指出数据库路径
      // 等价于openDatabase(path, factory, CREATE_IF_NECESSARY).
```

例

```
File dataBaseFile = new File(Environment.getExternalStorageDirectory()
                             + "/sqlite", "dbStu.db3");
if (!dataBaseFile.getParentFile().exists()) {  //其双亲文件为该文件的目录
   dataBaseFile.mkdirs();
}
SQLiteDatabase db=SQLiteDatabase.openOrCreateDatabase(dataBaseFile,null);
```

参考 参考

- 创建表和删除表 <u>参考</u>

```
String CREATE_TABLE = "CREATE TABLE stu(_id INTEGER PRIMARY KEY
                       autoincrement, num INTEGER, name varchar(24))";
db.execSQL(CREATE_TABLE);
db.execSQL("DROP TABLE stu");   //删除表，无返回值
```

- 插入操作

public long **insert** (String table, String nullColumnHack, ContentValues values)

参数	table	表名
	nullColumnHack	当values为null，即要插入一个空行时， 如果nullColumnHack为null，则会插入失败， 如果nullColumnHack指定了一个列名，则会插入成功，新行的这个列具有null值。
	values	列名和值的映射(map)。
返回值	新插入行的行ID(row ID), 或者 -1 （如果出错）	

例：
```
ContentValues cv = new ContentValues();
cv.put("num", 150010123); cv.put("name", "David");
db.insert("stu", null, cv);
```
或
```
String INSERT_DATA
       = "INSERT INTO stu(num, name) values (150010123, 'David')" ;
db.execSQL(INSERT_DATA);
```

- 修改操作

```
public int update(String table, ContentValues values,
                          String whereClause, String[] whereArgs)
```

参数：values为列名和值的映射，whereClause为带占位符?的条件，
　　　whereArgs为填入whereClause中?的字符串数组。
返回值：影响的行数。

例：

```
ContentValues cv = new ContentValues();
cv.put("num", 150010125);
cv.put("name", "Jane");
db.update("stu", cv, "_id=?", new String[]{"5"});
```

或

```
String UPDATE_DATA
       = "UPDATE stu set num=?, name=? WHERE _id=?" ;
db.execSQL(UPDATE_DATA,Object[]{150010125, "Jane", 5});
```

总结一下execSQL的两种格式:

```
public void execSQL (String sql)
public void execSQL (String sql, Object[ ] bindArgs)
```

- 删除操作

```
public int delete(String table, String whereClause,
                                             String[] whereArgs)
```

参数　　whereClause为带?的条件， whereArgs为填入whereClause中?的字符串数组。
返回值　返回影响的行数(>=0)。 whereClause为"1"，则删除所有行。

例：
```
db.delete("stu", "num=?",  new String[]{"1500100120"});
```

或

```
db.execSQL("DELETE from stu WHERE num= 1500100120");
```

删除表stu的全部数据:

```
db.execSQL("delete from stu");    // 没有返回
db.delete("stu",null,null);       // 删除所有行,返回删除的数量
```

• 查询操作

```
public Cursor query(String table,  String[] columns,
                    String selection, String[] selectionArgs,
                    String groupBy,String having,
                    String OrderBy,String limit)
```

参数　columns为列名数组，selection为带?的条件，sleletionArgs为填入selection中?的字符串数组。
返回值　指向结果集的游标。

```
public Cursor rawQuery(String sql, String[] selectionArgs)
```

参数：sql为带?的SQL语句，selectionArgs为填入SQL中?的字符串数组。
返回值：指向结果集的游标。

例1

```
String table="Orders";
String[] columns=new String[]{"CustomerName","SUM(OrderPrice)"};
String selection="Country=?";
String[] selectionArgs=new String[]{"China"};
String groupBy="CustomerName";
String having="SUM(OrderPrice)>500";
String orderBy="CustomerName";
Cursor c=db.query(table,columns,
       selection,selectionArgs,groupBy,having,orderBy,"10,100");
```

例2：

```
Cursor cursor = db.query("stu", new String[]{"_id","num","name" },
                "name like '?%'", new String[]{"张"}, null,null, null,
                "10,20");
while (cursor.moveToNext()) {
   System.out.println(cursor.getInt(cursor.getColumnIndex("_id")));
   System.out.println(cursor.getString(cursor.getColumnIndex("num")));
   System.out.println(cursor.getString(cursor.getColumnIndex("name")));
}


Cursor cursor = db.query("stu", new String[]{"_id","num","name" },
                null,null,null,null,null, null);  //query all
```

例3：

```
   Cursor cursor = db.rawQuery("select count(*) from stu", null);
   int count = 0;
   if(cursor.moveToFirst()){
       count = cursor.getLong(0);
   }
```

参考

• Cursor的方法

moveToFirst()	把游标移动到第一条，返回true(成功)或false
moveToLast()	移动到最后一条
move(int offset)	往后移动offset条记录
moveToNext()	到下一条
moveToPrevious()	移动到上一条
getCount()	得到总记录条数
isFirst()	判断当前游标是否指向为第一条记录
isLast()	判断当前游标是否为最后一条数据
getColumnIndex(String colName)	通过列名称获得列索引（从0开始）
getInt(int columnIndex)	通过列索引获得整数列的值(getShort/getLong)
getString(int columnIndex)	通过列索引获得字符串列的值
getFloat(int columnIndex)	通过列索引获得浮点数列的值(getDouble)
getExtras(int columnIndex)	通过列索引获得Bundle列的值

• SQLiteDatabase对象的方法

public void execSQL(String sql,Object[] args)	执行sql语句(增改删)
Cursor rawQuery(String sql,String[] args)	执行sql查询（select语句）
public void beginTransaction()	开始事务
public void endTransaction()	结束事务

, age int

,

SQLite的应用实例

- 本节给出了两种操作SQLite的方法，一种采用执行SQL语句(execSQL)的方法，另一种采用ContentValues带入参数的方法。后一种方法可以直接配合ContentProvider使用。实例还给出了单元测试方法的演示。

- 执行SQL语句(execSQL)的方法：

```
SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
db.execSQL("INSERT INTO stu(num,name) values(?,?)",new Object[]{num, name});
db.close();// 记得关闭数据库操作
```

- ContentValues带入参数的方法

```
SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
ContentValues cv = new ContentValues();
cv.put("num", 1700001);
cv.put("name", "学生1");
Uri uri = db.insert("stu", null, cv);
db.close();
```

项目：SqliteExec

```
class DBOpenHandler extends SQLiteOpenHelper {
    public DBOpenHandler(Context context, String name,
                   SQLiteDatabase.CursorFactory factory, int version) {
       super(context, name, factory, version); //第一次创建数据库时调用
    }
    @Override
    public void onCreate(SQLiteDatabase db) {   //第一次创建数据库时调用
        // 创建一个t_users表，id主键，自动增长，字符类型的username和pass;
       db.execSQL("CREATE TABLE stu(_id integer primary key autoincrement, num
               varchar(24), name varchar(24)), age int");
    }
    @Override     // 升级数据库时调用
    public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion){
        if (oldVersion < 2){
             try {
                 db.execSQL("ALTER TABLE stu ADD COLUMN age int");
             }catch(Exception ex){
             }
        }
    }
}
```

MainActivity.java、
AndroidManifest.xml
activity_main.xml
在新建项目后未修改

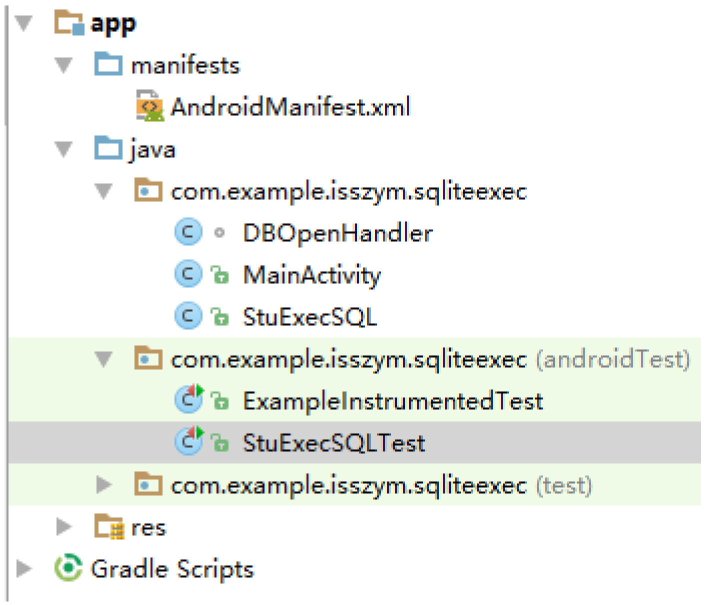
```
public class StuExecSQL {
    private DBOpenHandler dbOpenHandler;
    public StuExecSQL(Context context) {
        this.dbOpenHandler = new DBOpenHandler(context, "dbStu.db3", null, 2);
    }
    public void insert(String num,String name) {
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
        db.execSQL("INSERT INTO stu(num,name) values(?,?)",
                                              new Object[]{num, name});
        db.close();// 记得关闭数据库操作
    }
    public void delete(Integer id) {
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
        if(id<0)
            db.execSQL("DELETE FROM stu");
        else
            db.execSQL("DELETE FROM stu WHERE _id=?", new Object[]{id.toString()});
        db.close();
    }
    public void update(int id, String num,String name) {
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
        db.execSQL("UPDATE stu SET num=?,name=? WHERE" + " _id=?",
                new Object[]{num, name, id});
        db.close();
    }
    public Cursor query(String cond) {
        SQLiteDatabase db = dbOpenHandler.getReadableDatabase();
        Cursor cursor = (cond.isEmpty())?
                db.rawQuery("select * from stu",null):
                db.rawQuery("select * from stu where num like ? or name like ?",
                        new String[]{"%"+cond+"%","%"+cond+"%"},null);
        return cursor;
    }


    public long getCount() {//得到记录总数
        SQLiteDatabase db = dbOpenHandler.getReadableDatabase();
        Cursor cursor = db.rawQuery("select count(*) from stu", null);
        cursor.moveToFirst();
        long count = cursor.getLong(0);
        db.close();
        return count;
    }
}
```

StuExecSQLTest.java

```
@RunWith(AndroidJUnit4.class)
public class StuExecSQLTest {
    private static final String TAG = "测试";    // 准备好TAG标识用于LOG输出
    StuExecSQL db;
    long ind = 0;
    @Test
    public void useAppContext() throws Exception {
        Context appContext = InstrumentationRegistry.getTargetContext();
        assertEquals("com.example.isszym.sqliteexec", appContext.getPackageName());
    }
    @Before
    public void createDatabse() throws Throwable {
        Context appContext = InstrumentationRegistry.getTargetContext();
        assertEquals("com.example.isszym.sqliteexec", appContext.getPackageName());
        db = new StuExecSQL(appContext);
        ind = db.getCount();;
        Log.d(TAG, "建数据库成功");
    }
    @Test
    public void testInserts() throws Throwable {
        int count = 10;
        for(int i= 0;i<count;i++){
            testInsert();
        }
    }
```

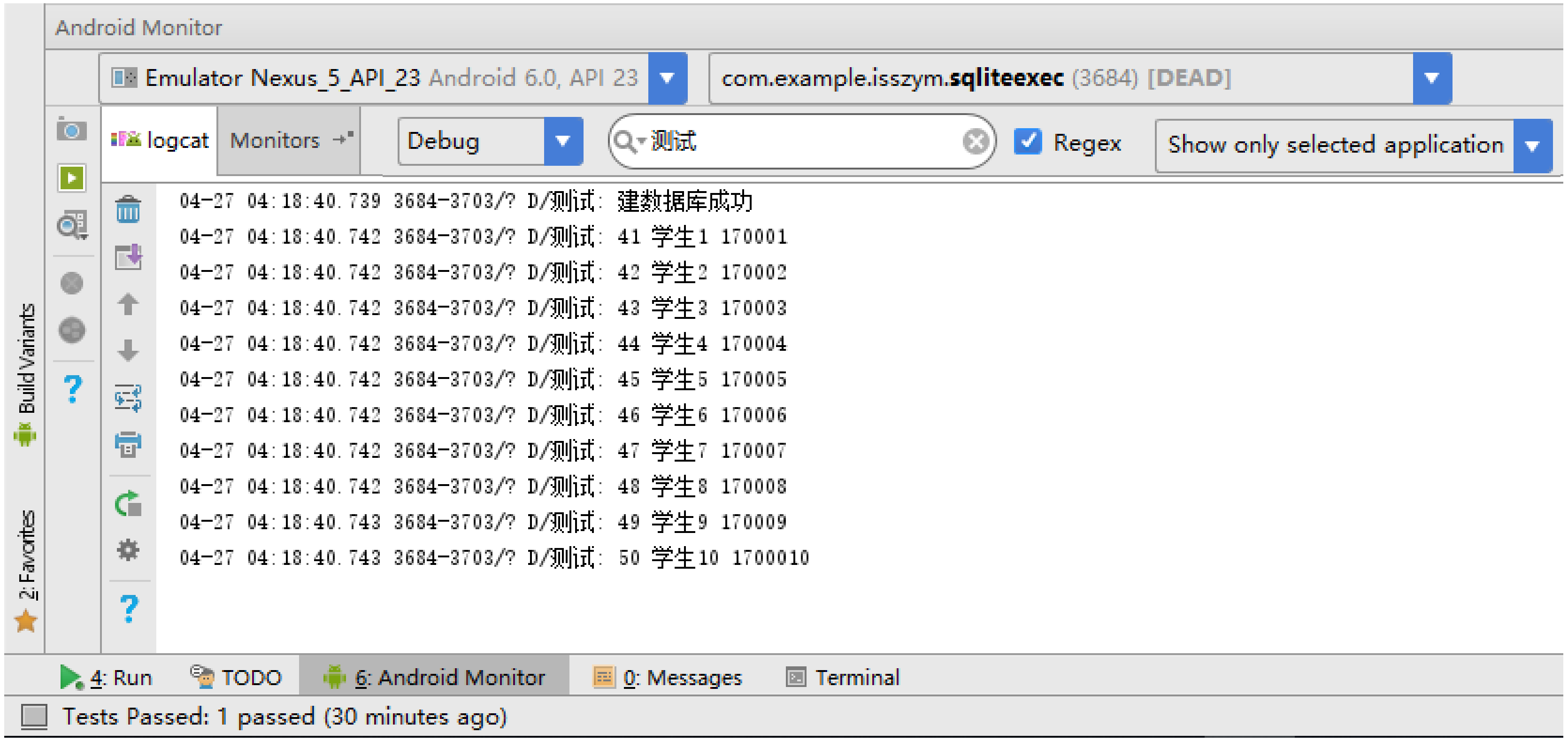
- 点击双箭头执行所有测试方法。
- 点击单箭头执行一个测试方法。运行这个方法前会先执行所有的@Before方法，再执行这个方法，最后执行所有的@After方法。

```
    @Test
    public void testInsert() throws Throwable {
        ind++;
        db.insert("17000"+ind,"学生"+ind);
        Log.d(TAG, "插入成功");
    }@Test
    public void testUpdate() throws Throwable {
        db.update(3,"170005", "张三");
        Log.d(TAG, "修改成功");
    }
    @Test
    public void testDelete() throws Throwable {
        db.delete(-1);
        Log.d(TAG, "删除成功");
    }
    @Test
    public void testGetCount() throws Throwable {
        Log.d(TAG, "总记录数: "+db.getCount());
        //assertEquals(db.getCount(), 10);
    }
    @Test
    public void testQueryAll() throws Throwable {
        Cursor cursor = db.query("");
        while (cursor.moveToNext()) {
            Log.d(TAG, cursor.getLong(0)+" "+cursor.getString(1)+" "+cursor.getString(2));
        }
        cursor.close();
    }

}
```



```
app
    manifests
        AndroidManifest.xml
    java
        com.example.isszym.sqliteexec
            DBOpenHandler
            MainActivity
            StuExecSQL
        com.example.isszym.sqliteexec (androidTest)
            ExampleInstrumentedTest
            StuExecSQLTest
        com.example.isszym.sqliteexec (test)
    res
Gradle Scripts
```

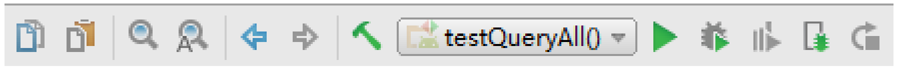
测试：先运行testDelete删除所有记录，再运行testInserts插入10条记录，最后testQueryAll查出所有记录，AndroidMonitor/logcat:


Android Monitor
Emulator Nexus_5_API_23 Android 6.0, API 23
com.example.isszym.sqliteexec (3684) [DEAD]
logcat
Monitors
Debug
测试
Regex
Show only selected application
4: Run
TODO
6: Android Monitor
0: Messages
Terminal
Build Variants
2: Favorites

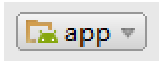
```
04-27 04:18:40.739 3684-3703/? D/测试: 建数据库成功
04-27 04:18:40.742 3684-3703/? D/测试: 41 学生1 170001
04-27 04:18:40.742 3684-3703/? D/测试: 42 学生2 170002
04-27 04:18:40.742 3684-3703/? D/测试: 43 学生3 170003
04-27 04:18:40.742 3684-3703/? D/测试: 44 学生4 170004
04-27 04:18:40.742 3684-3703/? D/测试: 45 学生5 170005
04-27 04:18:40.742 3684-3703/? D/测试: 46 学生6 170006
04-27 04:18:40.742 3684-3703/? D/测试: 47 学生7 170007
04-27 04:18:40.742 3684-3703/? D/测试: 48 学生8 170008
04-27 04:18:40.743 3684-3703/? D/测试: 49 学生9 170009
04-27 04:18:40.743 3684-3703/? D/测试: 50 学生10 1700010
```

Tests Passed: 1 passed (30 minutes ago)

再运行app：



改为



项目：SqliteCv

```java
public class StuCvSQL {
    private DBOpenHandler dbOpenHandler;
    public StuCvSQL(Context context) {
        this.dbOpenHandler = new DBOpenHandler(context, "dbStu.db3", null, 1);
    }

    public Uri insert(Uri uri,ContentValues cv) {
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
        db.insert("stu", null, cv);
        db.close();
        return uri;
    }

    public int delete(Uri uri, String where, String[] whereArgs) {
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
        int ret = db.delete("stu", where, whereArgs);
        db.close();
        return ret;//影响的记录数
    }

    public int update(Uri uri, ContentValues cv, String where, String[] whereArgs) {
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
        int ret = db.update("stu", cv, where, whereArgs);
        db.close();
        return ret; //影响的记录数
    }
```

MainActivity.java、
AndroidManifest.xml
activity_main.xml
在新建项目后未修改

```
    public Cursor query(Uri uri, String[] projection, String where,
                                    String[] whereArgs, String sortOrder){
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();
        Cursor cursor = db.query("stu", projection, where,
                                 whereArgs,null,null,sortOrder,null);
        // db.close();
        return cursor;
    }

    public long getCount() {//得到记录总数
        SQLiteDatabase db = dbOpenHandler.getReadableDatabase();
        Cursor cursor = db.rawQuery("select count(*) from stu", null);
        cursor.moveToFirst();
        db.close();
        return cursor.getLong(0);
    }
}
```

```
@RunWith(AndroidJUnit4.class)
public class StuCvSQLTest {
    static long ind = 0;
    private static final String TAG = "测试";    // 准备好TAG标识用于LOG输出
    StuCvSQL stuCvSQL;
    Uri uri;
    @Before
    public void createUri() throws Throwable {
        uri= Uri.parse("content://com.example.provider/");
    }
    @Before
    public void createDatabse() throws Throwable {
        Context appContext = InstrumentationRegistry.getTargetContext();
        assertEquals("com.example.isszym.sqlitecv", appContext.getPackageName());
        stuCvSQL = new StuCvSQL(appContext);
        ind = stuCvSQL.getCount();
        Log.d(TAG, "建数据库成功");
    }
    @Test
    public void testInserts() throws Throwable {
        int count = 10;
        for(int i= 0;i<count;i++){
            testInsert();
        }
    }
    @Test
    public void testInsert() throws Throwable {
        ContentValues cv = new ContentValues();
        ind++;
        cv.put("num", 170000+ind);
        cv.put("name", "学生"+ind);
        stuCvSQL.insert(uri,cv);
        Log.d(TAG, "插入成功: " + 170000+ind);
    }
    @Test
    public void testUpdate() throws Throwable {
        ContentValues cv = new ContentValues();
        cv.put("name", "张三");
        int cnt = stuCvSQL.update(uri,cv,"num=?",new  String[]{"170002"});
        Log.d(TAG, "成功修改"+cnt+"个记录");
    }
    @Test
    public void testDelete() throws Throwable {
        // stuCvSQL.delete(uri,"num=?", new String[]{"170004"});
        int cnt = stuCvSQL.delete(uri, null,null); //delete All
        Log.d(TAG, "成功删除"+cnt+"个记录");
    }
    @Test
    public void testGetCount() throws Throwable {
        long count =stuCvSQL.getCount();
        Log.d(TAG, "总记录数："+count);
    }
    @Test
    public void testQuery() throws Throwable {
        // String cond="1";
        // String where="num like ? or name like ?";
        // String[] whereArgs=new String[]{"%"+cond+"%","%"+cond+"%"};
        // String sortOrder = "num";
        // Cursor cursor = stuCvSQL.query(uri,projection,where,whereArgs,sortOrder);
        String[] projection=new String[]{"_id","num","name"};
        Cursor cursor = stuCvSQL.query(uri,projection,null,null,null); //query all
        while (cursor.moveToNext()) {
            Log.d(TAG,""+cursor.getInt(cursor.getColumnIndex("_id"))
                    +" "+cursor.getString(cursor.getColumnIndex("name"))
                    +" "+cursor.getString(cursor.getColumnIndex("num")));
        }
        cursor.close();
    }
}
```

执行testDelete：

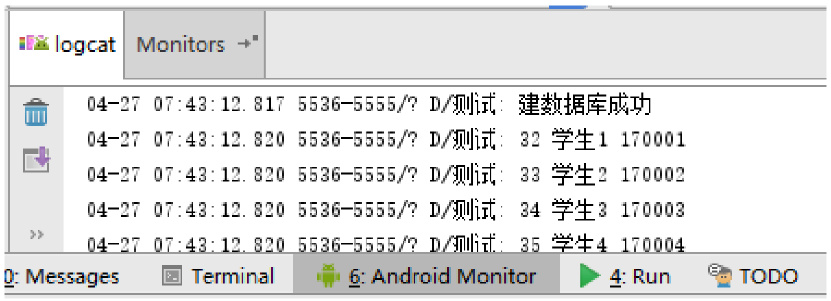
```
D/测试: 建数据库成功
D/测试: 成功删除10个记录
```

执行两次testInserts的第二次：

```
D/测试: 建数据库成功D/测试: 插入成功：17000011
D/测试: 插入成功：17000012
D/测试: 插入成功：17000013
D/测试: 插入成功：17000014
D/测试: 插入成功：17000015
D/测试: 插入成功：17000016
D/测试: 插入成功：17000017
D/测试: 插入成功：17000018
D/测试: 插入成功：17000019
D/测试: 插入成功：17000020
```

执行testQuery：

```
D/测试: 建数据库成功
D/测试: 32 学生1 170001
D/测试: 33 学生2 170002
D/测试: 34 学生3 170003
D/测试: 35 学生4 170004
……
D/测试: 49 学生18 170018
D/测试: 50 学生19 170019
D/测试: 51 学生20 170020
```



```
04-27 07:43:12.817 5536-5555/? D/测试: 建数据库成功
04-27 07:43:12.820 5536-5555/? D/测试: 32 学生1 170001
04-27 07:43:12.820 5536-5555/? D/测试: 33 学生2 170002
04-27 07:43:12.820 5536-5555/? D/测试: 34 学生3 170003
04-27 07:43:12.820 5536-5555/? D/测试: 35 学生4 170004
```

SimpleCursorAdapter

- Cursor是指向从数据库中取出的记录集的指针。SimpleCursorAdapter的使用与SimpleAdapter相似， 只是把ArrayList(<HashMap>)替换成了Cursor。

```
Cursor cursor = db.rawQuery("select _id,photo,num,name from stu", null);
SimpleCursorAdapter adapter = new SimpleCursorAdapter(MainActivity.this,
                       R.layout.list_item, cursor,
                       new String[]{"num", "name"},
                       new int[]{R.id.num, R.id.name}, 0);
listView.setAdapter(adapter);
```

- 由于数据库中保存的是图片名称，本例通过自定义SimpleCursorAdapter显示图片。

```
MySimpleCursorAdapter adapter = new MySimpleCursorAdapter(MainActivity.this,
              R.layout.list_item, cursor,
              new String[]{"photo", "num", "name"},
              new int[]{R.id.photo,R.id.num, R.id.name}, 0);
listView.setAdapter(adapter);
```

SimpleCursorAdapter的回调函数newView()用于生成每行的视图。
SimpleCursorAdapter的回调函数bindView()用于显示每行视图。因此，先在newView()中填充(inflate)布局*list_item*作为列表每一行的视图，然后再利用bindView()修改每一行的显示内容。其中的要显示的图片需要通过资源名称来获得资源Id。

工程名：ListViewCursor

```
public class MainActivity extends AppCompatActivity {
    ListView listView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        DBOpenHandler dbOpenHandler = new DBOpenHandler(this, "dbStu.db3", null, 1);
        SQLiteDatabase db = dbOpenHandler.getWritableDatabase();//insert/upatde/delete
        //db.execSQL("DROP TABLE stu");
        String CREATE_TABLE = "CREATE TABLE IF NOT EXISTS stu(_id INTEGER PRIMARY KEY
                    autoincrement,photo varchar(24), num INTEGER, name varchar(24))";
        db.execSQL(CREATE_TABLE);
        Cursor cursor = db.rawQuery("select _id,photo,num,name from stu", null);
        if (cursor != null) {
            MySimpleCursorAdapter adapter=new MySimpleCursorAdapter(MainActivity.this,
                         R.layout.list_item, cursor,new String[]{"photo","num", "name"},
                         new int[]{R.id.photo,R.id.num, R.id.name}, 0);
            listView = (ListView) findViewById(R.id.listView);
            listView.setAdapter(adapter);
        }
        db.close();
    Button btn=(Button)findViewById(R.id.button);
    btn.setOnClickListener(new Button.OnClickListener(){
        @Override
        public void onClick(View v) {
            AddRows();
        }
    });
    listView.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
       @Override
       public void onItemSelected(AdapterView<?> parent,View v,int pos,long id){
            // TODO Auto-generated method stub
            Cursor cursor = (Cursor) listView.getSelectedItem();
            String name = cursor.getString(1);
            String num = cursor.getString(2);
            System.out.println(num+" "+name);
       }

       @Override
       public void onNothingSelected(AdapterView<?> arg0) {
            // TODO Auto-generated method stub
       }
    });
}
```

```
  void AddRows() {
   DBOpenHandler dbOpenHandler = new DBOpenHandler(this, "dbStu.db3", null, 1);
   SQLiteDatabase db = dbOpenHandler.getWritableDatabase();//ins/upatde/del
   //db.execSQL("DROP TABLE stu");
   String CREATE_TABLE = "CREATE TABLE IF NOT EXISTS stu(_id INTEGER PRIMARY KEY
               autoincrement,photo varchar(24), num INTEGER, name varchar(24))";
   db.execSQL(CREATE_TABLE);
   String INSERT_DATA = "";
   INSERT_DATA="INSERT INTO stu(photo,num, name)values('cwt',150010121,'陈伟霆')";
   db.execSQL(INSERT_DATA);
   INSERT_DATA="INSERT INTO stu(photo,num, name)values('dlrb',150010122,'迪丽热巴')";
   db.execSQL(INSERT_DATA);
   INSERT_DATA="INSERT INTO stu(photo,num, name) values('zyx',150010123, '张艺兴')";
   db.execSQL(INSERT_DATA);
   INSERT_DATA="INSERT INTO stu(photo,num, name)values('lh',150010124, '鹿晗')";
   db.execSQL(INSERT_DATA);
   //db.execSQL("DELETE FROM stu;");
   db.close();
  }
}
```

MySimpleCursorAdapter.java

```java
class MySimpleCursorAdapter extends SimpleCursorAdapter {
    private LayoutInflater mInflater;
    private Context mContext;
    Cursor cur;   int layout;
    public MySimpleCursorAdapter(Context context, int layout, Cursor c,
                                              String[] from, int[] to, int flags) {
        super(context, layout, c, from, to, flags);
        this.layout = layout; mContext = context;
        mInflater = LayoutInflater.from(context);  cur = c;
    }
    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ImageView notePhoto = (ImageView) view.findViewById(R.id.photo);
        TextView noteNum = (TextView) view.findViewById(R.id.num);
        TextView noteName = (TextView) view.findViewById(R.id.name);
        Log.d("xxxx", "cur="+cursor.getCount()+",c_count="+cursor.getColumnCount());
        notePhoto.setImageResource(getResourceByReflect(cur.getString(1)));
        noteNum.setText("" + cur.getLong(2));
        noteName.setText("" + cur.getString(3));
    }
    @Override
    public View newView(Context arg0, Cursor arg1, ViewGroup arg2) {
        return mInflater.inflate(layout, arg2, false);
    }
    int getResource(String imageName) {
        //Context ctx = getBaseContext();
        int resId = mContext.getResources().getIdentifier(imageName, "drawable",
                                          mContext.getPackageName());
        return resId;
    }
    public int getResourceByReflect(String imageName) {
        Class drawable = R.drawable.class;
        Field field = null;
        int r_id;
        try {
            field = drawable.getField(imageName);
            r_id = field.getInt(field.getName());
        } catch (Exception e) {
            r_id = R.drawable.cwt;
            Log.e("ERROR", "PICTURE NOT  FOUND! ");
        }
        return r_id;
    }
}
```

DBOpenHandler.java

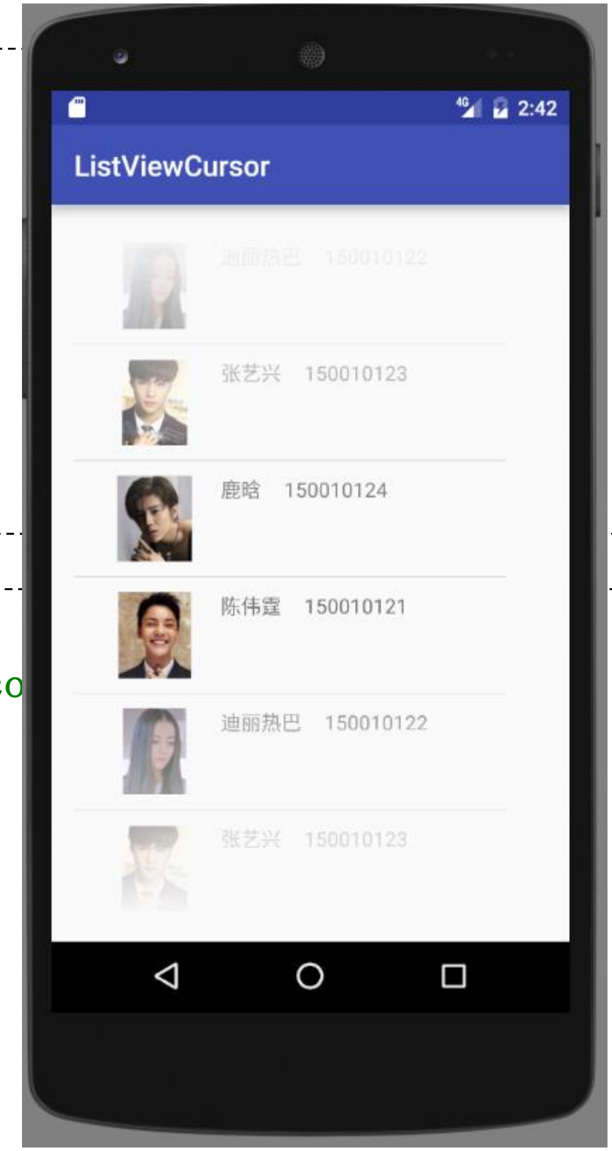
```
class DBOpenHandler extends SQLiteOpenHelper {
    private final String TAG = "MySQLiteOpenHelper";

    public DBOpenHandler(Context context, String dbName,
                          SQLiteDatabase.CursorFactory factory, int dbVersion) {
        //上下文,数据库名称,游标工厂，数据库版本号（>=1）    优点：可以处理版本变化
        super(context, dbName, factory, dbVersion);
        Log.d(TAG, "MySQLiteOpenHelper");
    }
    //可以在这里创建表
    @Override
    public void onCreate(SQLiteDatabase db) {
        Log.d(TAG, "onCreate");
    }
    //如果版本号发生变化，可以在这里做一些事，例如，为某个表增加一列
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.d(TAG, "onUpgrade");
    }

}
```

activity_main.xml

```
<ListView
    android:id="@+id/listView"
    android:layout_width="300dp"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:fadeScrollbars="true"
    android:requiresFadingEdge="vertical"
    android:fadingEdgeLength="200dp"/>
```

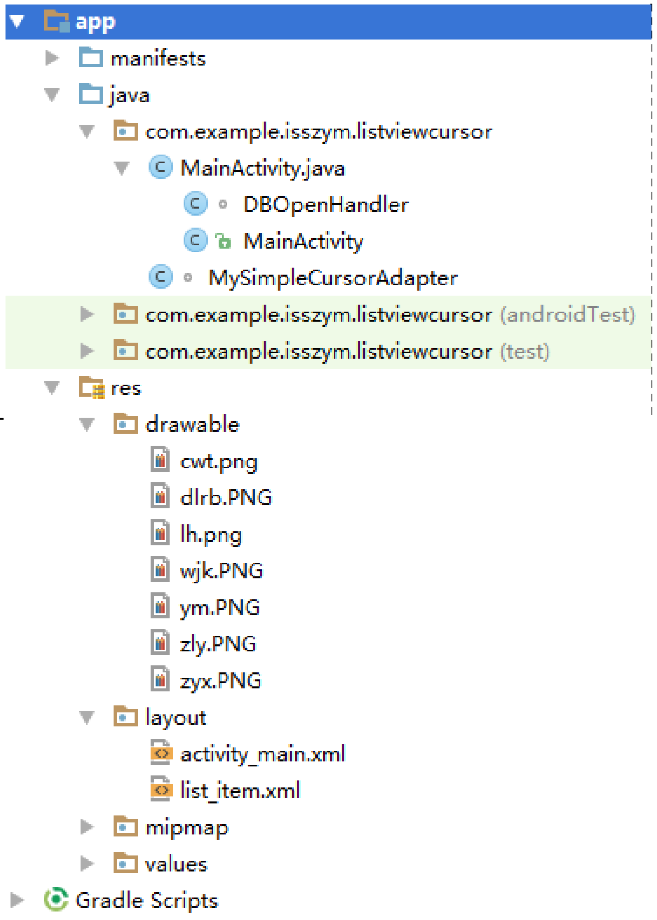
list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.co
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp" >
    <ImageView
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:id="@+id/photo"
        android:text="photo"
        android:src="@drawable/cwt"
        android:layout_marginLeft="@dimen/activity_horizontal_margin"/>
```



迪丽热巴 150010122
张艺兴 150010123
鹿晗 150010124
陈伟霆 150010121
迪丽热巴 150010122
张艺兴 150010123

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/name"
        android:text="姓名"
        android:layout_marginLeft="@dimen/activity_horizontal_margin" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/num"
        android:text="学号"
        android:layout_marginLeft
           ="@dimen/activity_horizontal_margin" />
</LinearLayout>
```



```
app
  manifests
  java
    com.example.isszym.listviewcursor
      MainActivity.java
        DBOpenHandler
        MainActivity
      MySimpleCursorAdapter
    com.example.isszym.listviewcursor (androidTest)
    com.example.isszym.listviewcursor (test)
  res
    drawable
      cwt.png
      dlrb.PNG
      lh.png
      wjk.PNG
      ym.PNG
      zly.PNG
      zyx.PNG
    layout
      activity_main.xml
      list_item.xml
    mipmap
    values
Gradle Scripts
```

ListView的其他常用属性

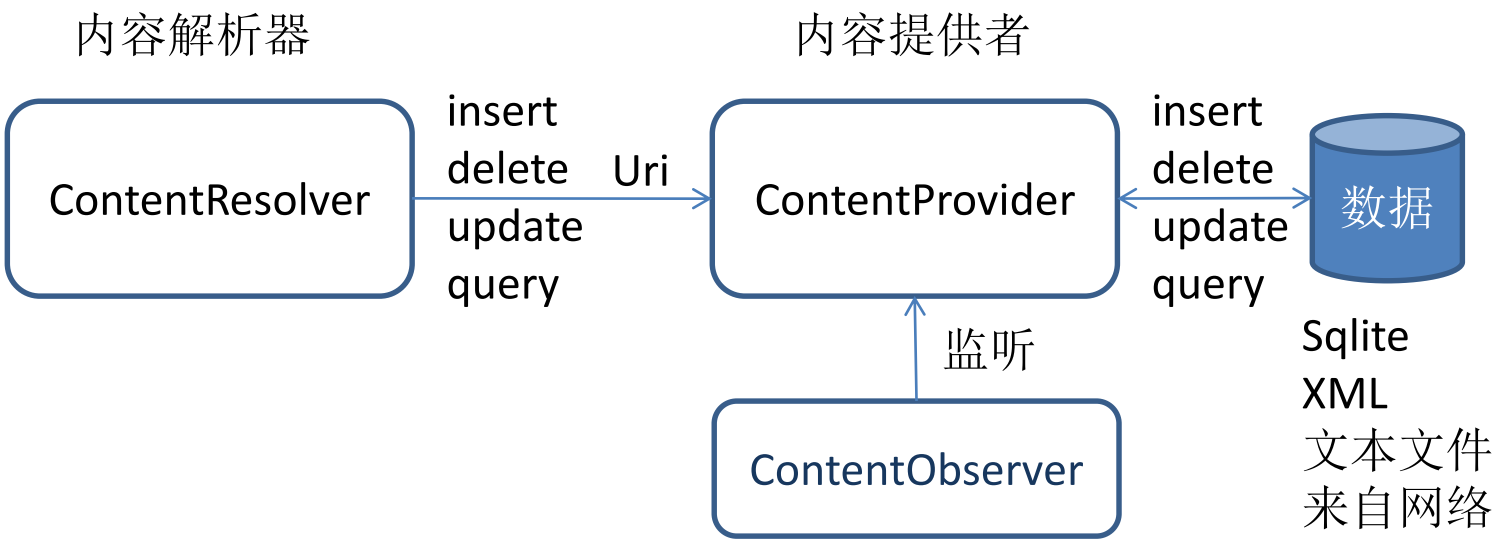
android:stackFromBottom="true"	显示列表的末尾内容
android:transcriptMode="alwaysScroll"	最新的条目自动滚动到可视范围内
android:cacheColorHint="#F00"	设置背景颜色，如果用图片做背景
android:cacheColorHint="#00000000"	如果要用background设置背景图
android:divider="@drawable/driver"	每一项之间用一个图片做为间隔
android:divider="@null"	去掉item之间的分割线
android:requriesFadingEdge="true"	上边和下边有黑色的阴影
android:fadingEdgeLength="16dp"	上下边阴影的长度。
android:scrollbars="none"	隐藏listView的滚动条
android:fadeScrollbars="true"	实现滚动条的自动隐藏和显示。
android:entries="@array/city"	设置列表内容
setVerticalScrollBarEnabled(true)	用于设置垂直滚动条

ContentProvider和ContentResolver

- 概述

参考

  - ContentProvider提供接口让其他app可以操作自己的数据并发出改变的消息。其他app则通过ContentResolver来操作这些数据。ContentObserver可以用来监听ContentProvider数据的改变消息。它们之间都是通过Uri （Uniform Resource Identifier）进行联系的。



- ContentProvider的主要方法：

方法	说明
public boolean **onCreate**()	在创建ContentProvider时调用
public Uri **insert**(Uri uri, ContentValues values)	根据uri进行插入数据。ContentValues 为HashMap，列名与列值的映射。
public int **update**(Uri uri, ContentValues values, String whereClause, String[] whereArgs)	修改uri指定的数据。whereClause为带占位符?的where条件，whereArgs为?对应的值。返回修改的行数。
public int **delete**(Uri uri, String whereClause, String[] whereArgs)	根据uri删除数据。返回删除的行数。
public String **getType**(Uri uri)	返回指定的uri中的数据的MIME类型
public Cursor **query**(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)	根据uri查询数据。projection[]为列名，即columns[]，selection为带占位符?的where条件， selectionArgs为?对应的值。

- ContentResolver的主要方法(参数含义见ContentProvider)：

方法	说明
ContentResolver **getContentResolver**();	从Context获得ContentResolver。
public Uri **insert**(Uri uri, ContentValues values)	添加数据到uri指定的ContentProvider中
public int **update**(Uri uri, ContentValues values, String selection, String[] selectionArgs)	修改uri指定的ContentProvider中的数据
public int **delete**(Uri uri, String selection, String[] selectionArgs)	从uri指定的ContentProvider中删除数据
public Cursor **query**(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)	查询uri指定的ContentProvider，除了uri，其他参数都可以为null，projection为null时取所有字段，selection为null时取所有记录。sortOrder 可以加入limit等，例如， id desc limit 10,100

• Uri

Uri （Uniform Resource Identifier）类似URL，但是它不是用于定位Web对象而是用于定位要操作的数据，其实用的格式如下：

$$\underbrace{\texttt{content}}_{\text{scheme}}\texttt{://}\underbrace{\texttt{com.abc.providers}}_{\text{authorities}}\texttt{/}\underbrace{\texttt{customers/100/6}}_{\text{path}}$$

scheme(方案)：	这里必须是content。
authorities(机构)：	用于区分不同的机构，即域名。
path(路径)：	用于区分机构中的不同数据。例如，路径words/30/6可以表示词汇库的30行6列。

在ContentResolver中要把给出的Uri字符串表示转换成Uri类型的变量值，然后再调用insert等方法去调用ContentProvider的对应方法：

```
Uri uri = Uri.parse(URI);
```

例如：

```
Uri uri = Uri.parse("content://com.abc.providers/customers/100/6");
```

- **UriMatcher**

  ContentProvider要使用UriMatcher匹配Uri的authority和path，然后根据匹配情况确定要完成的操作。

```
Uri uri= Uri.parse("content://com.abc.providers/customers/100/6");
//常量UriMatcher.NO_MATCH表示不匹配任何路径的返回码
UriMatcher sMatcher = new UriMatcher(UriMatcher.NO_MATCH);
//如果匹配content://com.abc.providers/customers，返回匹配码为1
sMatcher.addURI("com.abc.providers","customers", 1);

//如果match()方法匹配content://com.abc.providers/customers/*,
//      返回匹配码为2. #号为通配符
sMatcher.addURI("com.abc.providers", "customers/#", 2);
switch (sMatcher.match(uri)) {
   case 1:
      …
      break;
   case 2:
       …
      break;
   default://不匹配
      break;
}
```

- **ContentUris**

ContentUris类可以用于获取或并入Uri路径后面的ID部分。

➢ ContentUris.withAppendedId(Uri,int)往Uri并入一个id，例如：

```
Uri uri=Uri.parse("content://com.abc.providers/");
Uri.withAppendPath(uri,"customers");
Uri resultUri = ContentUris.withAppendedId(uri, 30);
//生成后的uri为：content://com.abc.providers/customers/30
```

➢ ContentUris.parseId(Uri uri)从Uri取回id，例如：

```
long personid = ContentUris.parseId(resultUri);//获取的结果为:30
```

- **ContentProvider例　　　　　　项目名:FirstProvider**

```
public class FirstProvider extends ContentProvider{
     @Override
    public boolean onCreate()  {
        System.out.println("===onCreate is called==="); return true;
    }
    @Override
    public String getType(Uri uri){    // 返回本实例提供的MIME类型
        System.out.println("~~getType is called~~"+sh(uri)); return null;
    }
    @Override
    public Cursor query(Uri uri, String[] projection,
                   String where,String[] whereArgs, String sortOrder) {
        System.out.println("===query is called==="+projection[0]+sh(uri));
        return null;
    }
    @Override
    public Uri insert(Uri uri, ContentValues cv){
        System.out.println("===insert iscalled==="+cv.get("name")+sh(uri));
        return uri;
    }
    @Override
    public int delete(Uri uri, String where, String[] whereArgs){
        System.out.println("===delete is called==="+sh(uri)); return 1;
    }
```

```java
    @Override
    public int update(Uri uri, ContentValues cv, String where,String[] whereArgs){
        System.out.println("===update is called==="+sh(uri));
        getContext().getContentResolver().notifyChange(uri, null);
        return 1;
    }
    private String sh(Uri uri){
        return "\r\n-----"+uri+"-----\r\n";
    }
}
```

AndroidManifest.xml

```xml
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name">
     ...
    <provider
        android:exported="true"
        android:name=".FirstProvider"
        android:authorities="com.example.providers.firstprovider">
    </provider>
</application>
```

- **项目名:FirstResolver**

```
public class MainActivity extends AppCompatActivity {
    ContentResolver resolver;
    ContentValues cv=new ContentValues();
    Uri uri = Uri.parse("content://com.example.providers.firstprovider/");
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
        Uri uri = Uri.parse("content://com.example.providers.firstprovider/");
        resolver = getContentResolver();
    }
    public void query(View vw) {
        Cursor c=resolver.query(uri,new String[]{"num","name"},"_id=?",new String[]{"6"},null);
        if(c!=null && c.moveToNext()){
            Log.d("测试","num:"+c.getString(0)+" name:"+c.getString(1));
        }
    }
    public void insert(View vw){
        cv.clear();  cv.put("num","170001");    cv.put("name","张三");
        Uri newUri = resolver.insert(uri,cv);  // null--错误
    }
    public void update(View vw) {
        cv.clear();   cv.put("num","170002");    cv.put("name","李四");
        int count = resolver.update(uri, cv, "_id=?", new String[]{"2"});// 影响的记录数
    }
    public void delete(View vw) {
        int count = resolver.delete(uri, "_id=?", new String[]{"5"}); // 影响的记录数
    }
}
```

```java
@RunWith(AndroidJUnit4.class)
public class ResolverTest {
    ContentResolver resolver;
    ContentValues cv = new ContentValues();
    Uri uri = Uri.parse("content://com.example.providers.firstprovider/");
    @Before
    public void useAppContext() throws Exception {
        Context appContext = InstrumentationRegistry.getTargetContext();
        assertEquals("com.example.isszym.newfirstresolver", appContext.getPackageName());
        resolver = appContext.getContentResolver();
    }
    @Test
    public void query() {
        Cursor c = resolver.query(uri,new String[]{"num","name"}, "_id=?", new String[]{"6"}, null);
        if(c!=null && c.moveToNext()){
            Log.d("测试","num:"+c.getString(0)+" name:"+c.getString(1));
        }
    }
    @Test
    public void insert(){
        cv.clear();   cv.put("num","170001");  cv.put("name","张三");
        Uri newUri = resolver.insert(uri,cv);  // null--错误
    }
    @Test
    public void update() {
        cv.clear();    cv.put("num","170002");    cv.put("name","李四");
        int count = resolver.update(uri, cv, "_id=?", new String[]{"2"});// 影响的记录数
    }
    @Test
    public void delete() {
        int count = resolver.delete(uri, "_id=?", new String[]{"5"}); // 影响的记录数
    }
}
```

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="query"
        android:text="查询" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="insert"
        android:text="插入" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="update"
        android:text="更新" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="delete"
        android:text="删除" />
</LinearLayout>
```

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.isszym.newfirstresolver">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

ContentObserver

- **注册 ContentObserver 方法**

```
public final void registerContentObserver(Uri uri,
                    boolean notifyForDescendents, ContentObserver observer)
```

用指定的 Uri 注册一个 ContentObserver 派生类实例，当给定的 Uri 发生改变时，回调该实例对象去处理，其中， uri 表示需要观察的 Uri，notifyForDescendents为false时表示精确匹配，即只匹配该Uri，为true表示可以同时匹配其派生的 Uri。

在ContentProvider中要调用ContentResolver的notifyChange方法发送改变的通知。

- **取消注册 ContentObserver 方法**

```
public final void unregisterContentObserver(ContentObserver observer)
```

取消对给定 Uri 的观察，其中，observer为正在观察的派生类实例。

- **观察特定 Uri 的步骤**

1、创建 ContentObserver 派生类，必须重载父类构造方法和onChange()方法。
2、用 ContentResolover对象的registerContentObserver()方法把上面的派生类注册为Uri的内容观察者。
3、调用 unregisterContentObserver()去取消注册，结束观察。

项目名：FirstObserver

activity_main.xml
AndroidManifest.xml建立后未改变

```
public class MainActivity extends AppCompatActivity {
    private Handler handler = new Handler() {
        public void handleMessage(android.os.Message msg) {
            switch (msg.what) {
                case 100: String body = (String) msg.obj;
                    TextView tv = (TextView) findViewById(R.id.tv);
                    tv.setText(body); break;
            }
        }
    };
    MyContentObserver mContentObserver;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Uri uri = Uri.parse("content://com.example.providers.firstprovider/#");
        mContentObserver = new MyContentObserver(handler);
        this.getContentResolver().registerContentObserver(uri, true,mContentObserver);
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mContentObserver!=null)
            this.getContentResolver().unregisterContentObserver(mContentObserver);
    }
}
```

```java
class MyContentObserver extends ContentObserver {
    Handler handler;
    MyContentObserver(Handler handler) {
        super(handler);
        this.handler = handler;
    }

    @Override
    public void onChange(boolean selfChange) {
        super.onChange(selfChange);
        Log.d("测试","ContentObserver onChange() selfChange=" + selfChange);
        handler.obtainMessage(100, "ContentObserver Changed!").sendToTarget();
    }

    @Override
    public boolean deliverSelfNotifications() {
        Log.d("测试","ContentObserver deliverSelfNotifications!");
        return true;
    }
}
```

先运行FirstProvider，然后运行FirstObserver，最后运行FirstResolver，然后依次点击FirstResolver的四个按钮（查询、插入、更新、删除）后，**FirstProvider的logcat显示如下：**

```
I/System.out: ===query is called===name
I/System.out: -----content://com.example.providers.firstprovider/-----
I/System.out: ===insert iscalled===张三
I/System.out: -----content://com.example.providers.firstprovider/-----
I/System.out: ===update is called===
I/System.out: -----content://com.example.providers.firstprovider/-----
I/System.out: ===delete is called===
I/System.out: -----content://com.example.providers.firstprovider/-----
```

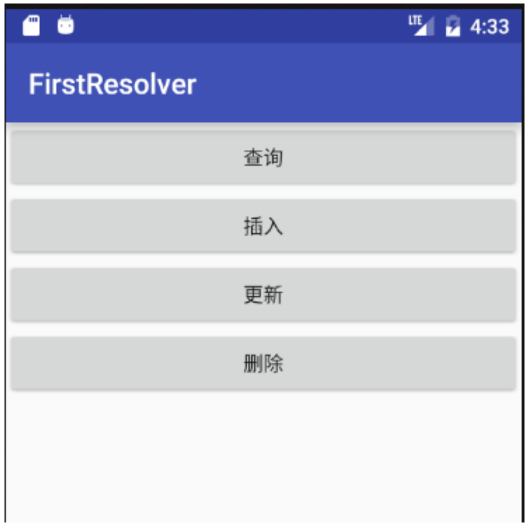
FirstObserver的logcat显示如下：

```
com.example.isszym.firstobserver D/测试: ContentObserver onChange() selfChange=false
```

为什么只捕捉到一个观察事件？

因为在ContentProvider中只有更新事件里发送了改变通知：

```
getContext().getContentResolver().notifyChange(uri, null);
```



通信录操作

- 概述
  - Android通信录(contact2.db)一共包括三张表：data, raw_contacts, contact。安卓系统提供了ContentProvider接口，app可以使用ContentResolver访问前两个表，它们的URI为：
    content://com.android.contacts/raw_contacts
    content://com.android.contacts/data
  - raw_contacts表的每行存储了联系人的一般信息。
  - data表存储了联系人的详细信息，表的每一行存储一个特定数据类型的信息，比如Email、Address或Phone等。
  - Contact表的每一行表示一个联系人，它综合了raw_contacts的若干行，是**由系统自动维护**的。当增加一个raw_contacts新记录时，系统会首先查看contact表是否有记录跟插入的记录表示同一个人（电话号码或邮件地址相同），如果找到了，则把找到的这个contact记录的_id填入raw_contacts记录的contact_id字段，如果没有找到，则系统自动插入一个新的contact记录并把它的_id填入新插入raw_contacts记录的contact_id字段。

- 通信录的表

contact表

_id	主键，自动增长
name_raw_contact_id	对应raw_contacts表中的id,即界面显示的contact
photo_id	模糊大头贴id
photo_file_id	清晰大头贴id
custom_ringtone	个人铃声
send_to_voicemail	来电转接语音信箱（0 false，1 true）
times_contacted	联系人联系次数
last_time_contacted	最近一次联系的时间
starred	是否收藏
has_phone_number	是否有号码（至少一个）
lookup	查找此联系人的key值

data表

_id	主键，自动增长
package_id	包id
mimetype_id	数据类型id，对应mimetypes表
raw_contact_id	对应contact表的id
is_primary,is_super_primay	是否为默认号码
data_version	此数据记录的版本
data1-data15	同类数据的不同形式的表达方式

raw_contacts表

_id	主键，自动增长
account_id	记录所属账户id
sourceid	数据源id
backupid	数据备份时产生的id
raw_contact_is_read_only	是否只读（1只读，0可删除）M中没用
version	数据发送变化时改变
dirty	version变化，值为1，需同步数据
deleted	标记是否为删除的记录
contact_id	对应contacts表中的id
aggregation_mode	标记是否需要合并（default:0?immediate:1 suspended:2 disabled:3 ）
aggregation_needed	是否需要合并（1需要，0不需要）
custom_rington	与该记录相关的手机铃声
send_to_voicemail	来电转接语音信箱
times_contacted	与该联系人联系的次数
last_time_connected	最近一次联系时间
starred	是否收藏（1收藏，0没收藏）
pinned	是否被固定
display_name	联系人显示名称
display_name_alt	联系人显示名称的替代表示（如西方名字“名在前”）
display_name_source	名字显示数据类型：email、phone、name
phonetic_name	发音名字
phonetic_name_style	名字不同发音风格（undefined=0,pinyin=3,japanese=4,korean=5）
sort_key	排序字段
phonebook_label	首字母
phonebook_bucket	对应首字母的排序
sync1、2、3、4	同步相关，保存uri，同步状态，服务器版本，错误代码等等

- ## 项目 Contact

```
public class Contact {
    Context context;
    ContentResolver resolver;
    String TAG = "测试";
    Contact(Context context){
        this.context = context;
        resolver = context.getContentResolver();
    }
    // 直接添加联系人：先往raw_contacts中插入一个联系人，然后再往data中加入姓名和电话号码
    public void addContact(String name,String phone,String email){
        //插入raw_contacts表，并获取_id属性
        Uri uri1 = Uri.parse("content://com.android.contacts/raw_contacts");
        ContentValues values = new ContentValues();
        Uri rawContactUri = resolver.insert(ContactsContract.RawContacts.CONTENT_URI, values);
        //Uri rawContactUri = ContactsContract.RawContacts.CONTENT_URI;
        long rawContactId = ContentUris.parseId(rawContactUri);
        //插入data表
        Uri uri2 = Uri.parse("content://com.android.contacts/data");
        //add Name
        values.put("raw_contact_id", rawContactId);
        values.put(ContactsContract.Contacts.Data.MIMETYPE,"vnd.android.cursor.item/name");
        values.put("data2", name);
        values.put("data1", name);
        resolver.insert(uri2, values);
        values.clear();
        //add Phone
        values.put("raw_contact_id", rawContactId);
        values.put(ContactsContract.Contacts.Data.MIMETYPE,"vnd.android.cursor.item/phone_v2");
        values.put("data2", "2");
        values.put("data1", phone);
        resolver.insert(uri2, values);
        values.clear();
        //add email
        values.put("raw_contact_id", rawContactId);
        values.put(ContactsContract.Contacts.Data.MIMETYPE,"vnd.android.cursor.item/email_v2");
        values.put("data2", "2");
        values.put("data1", email);
        resolver.insert(uri2, values);
    }

    public void updatePhoneNumber(int id,String phoneNumber)throws Exception{
        Uri uri = Uri.parse("content://com.android.contacts/data");   // 表data
        ContentValues values = new ContentValues();
        values.put("data1", phoneNumber);
        resolver.update(uri, values, "mimetype=? and raw_contact_id=?",
                                  new String[]{"vnd.android.cursor.item/phone_v2",""+id});
    }

    public void deleteAllContacts()throws Exception {
        Uri uri1 = Uri.parse("content://com.android.contacts/raw_contacts");  // 表raw_contacts
        Uri uri2 = Uri.parse("content://com.android.contacts/data");
        resolver.delete(uri2, null, null);
        resolver.delete(uri1, null, null);
        //Cursor cursor = resolver.query(uri1,
        //                        new String[]{ContactsContract.Contacts.Data._ID},null,
        //     null, null);
        //while(cursor.moveToNext()){
        //  int id = cursor.getInt(0);
        //    resolver.delete(uri1, "_id=", new String[]{""+id});
        //    resolver.delete(uri2, "raw_contact_id=?", new String[]{""+id});
        //}
    }
```

```
public void deleteContact(String display_name)throws Exception{
    Uri uri = Uri.parse("content://com.android.contacts/raw_contacts");  // 表raw_contacts

    Cursor cursor = resolver.query(uri, new String[]{ContactsContract.Contacts.Data._ID},
                                            "display_name=?",new String[]{display_name},null);
    if(cursor.moveToNext()){
        int id = cursor.getInt(0);
        //根据id删除data中的相应数据
        resolver.delete(uri, "display_name=?", new String[]{display_name});
        uri = Uri.parse("content://com.android.contacts/data");
        resolver.delete(uri, "raw_contact_id=?", new String[]{""+id});
    }
}

//查询指定电话的联系人姓名，邮箱
public void queryNameByNumber(String phoneNumber) throws Exception {
    Uri uri = Uri.parse("content://com.android.contacts/data/phones/filter/" + phoneNumber);
    Cursor cursor = resolver.query(uri, new String[]{"display_name"}, null, null, null);
    if (cursor.moveToFirst()) {
        String name = cursor.getString(0);
        Log.i(TAG, name);
    }
    cursor.close();
}


// 获得联系人个数
public long getCount() throws Exception {
    Uri uri = Uri.parse("content://com.android.contacts/contacts");
    Cursor cursor = resolver.query(uri, new String[]{"_id"}, null, null, null);
    long count = cursor.getCount();
    cursor.close();
    return count;
}
```

```
//查询所有联系人的姓名，电话，邮箱
    public void queryAllContacts() throws Exception {
        Uri uri = Uri.parse("content://com.android.contacts/contacts");
        Cursor cursor = resolver.query(uri, new String[]{"_id"}, null, null, null);
        while (cursor.moveToNext()) {
            int rawContactId = cursor.getInt(0);
            StringBuilder sb = new StringBuilder("contractID=");
            sb.append(rawContactId);
            uri = Uri.parse("content://com.android.contacts/contacts/" + rawContactId + "/data");
            Cursor cursor1 = resolver.query(uri, new String[]{"mimetype", "data1", "data2"}, null, null, null);
            while (cursor1.moveToNext()) {
                String data1 = cursor1.getString(cursor1.getColumnIndex("data1"));
                String mimeType = cursor1.getString(cursor1.getColumnIndex("mimetype"));
                if ("vnd.android.cursor.item/name".equals(mimeType)) { //姓名
                    sb.append(",name=" + data1);
                } else if ("vnd.android.cursor.item/email_v2".equals(mimeType)) { //邮件
                    sb.append(",email=" + data1);
                } else if ("vnd.android.cursor.item/phone_v2".equals(mimeType)) { //手机号
                    sb.append(",phone=" + data1);
                }
            }
            cursor1.close();
            Log.i(TAG, sb.toString());
        }
        cursor.close();
    }
```

```
// 使用事务添加联系人
public void addContactByTransaction(String name,String phoneNumber,String email) throws Exception {
    Uri uri = Uri.parse("content://com.android.contacts/raw_contacts");
    // 用ContentProviderOperation批量增加联系人
    ArrayList<ContentProviderOperation> operations = new ArrayList<ContentProviderOperation>();
    ContentProviderOperation op1 = ContentProviderOperation.newInsert(uri)
            .withValue("account_name", null)
            .build();
    operations.add(op1);

    uri = Uri.parse("content://com.android.contacts/data");
    ContentProviderOperation op2 = ContentProviderOperation.newInsert(uri)
            .withValueBackReference("raw_contact_id", 0)
            .withValue("mimetype", "vnd.android.cursor.item/name")
            .withValue("data2", name)
            .build();
    operations.add(op2);

    ContentProviderOperation op3 = ContentProviderOperation.newInsert(uri)
            .withValueBackReference("raw_contact_id", 0)
            .withValue("mimetype", "vnd.android.cursor.item/phone_v2")
            .withValue("data1", phoneNumber)
            .withValue("data2", "2")
            .build();
    operations.add(op3);

    ContentProviderOperation op4 = ContentProviderOperation.newInsert(uri)
            .withValueBackReference("raw_contact_id", 0)
            .withValue("mimetype", "vnd.android.cursor.item/email_v2")
            .withValue("data1", email)
            .withValue("data2", "2")
            .build();
    operations.add(op4);

    resolver.applyBatch("com.android.contacts", operations);
}
```

```
// 采用系统常量添加联系人
public void insertContact(String familyName,String givenName,String phoneNumber) {
    //  ContactsContract.RawContacts.CONTENT_URI="content://com.android.contacts/raw_contacts"
    ContentValues values = new ContentValues();
    Uri rawContactUri = resolver.insert(ContactsContract.RawContacts.CONTENT_URI, values);
    long rawContactId = ContentUris.parseId(rawContactUri);

    //插入姓名：ContactsContract.Data.CONTENT_URI="content://com.android.contacts/data"
    //                  .StructuredName.CONTENT_ITEM_TYPE = "vnd.android.cursor.item/name"
    values.clear();
    values.put(ContactsContract.Data.RAW_CONTACT_ID, rawContactId);
    values.put(ContactsContract.Data.MIMETYPE,
                           ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE);
    values.put(ContactsContract.CommonDataKinds.StructuredName.GIVEN_NAME, givenName);// .GIVEN_NAME="data2"
    values.put(ContactsContract.CommonDataKinds.StructuredName.FAMILY_NAME, familyName);// .FAMILY_NAME="data3"
    resolver.insert(ContactsContract.Data.CONTENT_URI, values);

    //插入电话号码    .Phone.CONTENT_ITEM_TYPE="vnd.android.cursor.item/phone_v2"
    //                .Phone.TYPE_MOBILE = 2
    values.clear();
    values.put(ContactsContract.Contacts.Data.RAW_CONTACT_ID, rawContactId);
    values.put(ContactsContract.Contacts.Data.MIMETYPE,
                                ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);
    values.put(ContactsContract.CommonDataKinds.Phone.NUMBER, phoneNumber);          //  data1,-
    values.put(ContactsContract.CommonDataKinds.Phone.TYPE,
                                ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE);//data2,2
    resolver.insert(ContactsContract.Data.CONTENT_URI, values);
}
```

```
    //查询联系人
    public void showAllContacts() {
        Uri uri = ContactsContract.Contacts.CONTENT_URI;
        // String[] projection = {ContactsContract.Contacts.DISPLAY_NAME,ContactsContract.Contacts._COUNT};
        // String selection = ContactsContract.Contacts.DISPLAY_NAME +"=?";
        //  String[] selectionArgs = {"aaa"};
        String sortOrder = ContactsContract.Contacts._ID;
        Cursor c = resolver.query(uri, null, null, null, sortOrder);
        while(c.moveToNext()) {
            String rawContactId = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));
            Log.i(TAG, "========== _ID="+rawContactId+" ==========");
            Log.i(TAG, "DISPLAY_NAME="+c.getString(c.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)));
            int phoneCount = c.getInt(c.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));
            if(phoneCount>0) {
                Cursor phones = resolver.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                        ContactsContract.CommonDataKinds.Phone.CONTACT_ID+ " = " + rawContactId, null, null);
                while(phones.moveToNext()){
                    Log.i(TAG, "NUMBER=" +
                      phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER)));
                }

            }
        }
    }
}
```

```java
@RunWith(AndroidJUnit4.class)
public class ContactTest {
    static long ind = 0;
    private static final String TAG = "测试";   // 准备好TAG标识用于LOG输出
    Uri uri;
    Contact contact;
    @Before
    public void createUri() throws Throwable {
        Context appContext = InstrumentationRegistry.getTargetContext();
        uri = Uri.parse("content://com.android.contacts/contacts");
        contact = new Contact(appContext);
        ind = contact.getCount();
    }

    @Test
    public void testBatchInsert() throws Throwable {
        int count = 10;
        for(int i= 0;i<count;i++){
            testInsert();
        }
    }
        @Test
    public void testInsert() throws Throwable {
        ind++;
        contact.addContact("136427830"+ind,"王"+ind,"w"+ind+"@123.com");
        Log.d(TAG, "插入成功---136427830"+ind);
    }

    @Test
    public void testUpdate() throws Throwable {
        contact.updatePhoneNumber(10,"13600010005");
        Log.d(TAG, "修改成功");
    }
    @Test
    public void testDelete() throws Throwable {
        contact.deleteContact("王29");
        Log.d(TAG, "删除成功");
    }

    @Test
    public void testDeleteAll() throws Throwable {
        contact.deleteAllContacts();
        Log.d(TAG, "删除成功");
    }

    @Test
    public void testGetCount() throws Throwable {
       long count = contact.getCount();
       Log.d(TAG, "总记录数："+count);
    }

    @Test
    public void testQuery() throws Throwable {
        contact.queryAllContacts();
    }
}
```

```java
public class MainActivity extends AppCompatActivity {
    String TAG="test";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setRight();
    }
    void  setRight() {
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS)
                != PackageManager.PERMISSION_GRANTED
                || ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE_CONTACTS)
                != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ_CONTACTS}, 1);
        }
    }

}
```

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.isszym.contact">
    <application>
         ...
    </application>
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
</manifest>
```

先执行app获得权限，然后执行testDeleteAll，再执行testBatchInsert，得到logcat：

```
D/测试: 插入成功---1364278301
D/测试: 插入成功---1364278302
com.example.isszym.contact D/测试: 插入成功---1364278303
com.example.isszym.contact D/测试: 插入成功---1364278304
com.example.isszym.contact D/测试: 插入成功---1364278305
com.example.isszym.contact D/测试: 插入成功---1364278306
com.example.isszym.contact D/测试: 插入成功---1364278307
com.example.isszym.contact D/测试: 插入成功---1364278308
com.example.isszym.contact D/测试: 插入成功---1364278309
com.example.isszym.contact D/测试: 插入成功---13642783010
```

最后执行testQuery，得到logcat：

```
com.example.isszym.contact I/测试: contractID=51,name=1364278301,phone=王1,email=w1@123.com
com.example.isszym.contact I/测试: contractID=52,name=1364278302,phone=王2,email=w2@123.com
com.example.isszym.contact I/测试: contractID=53,name=1364278303,phone=王3,email=w3@123.com
com.example.isszym.contact I/测试: contractID=54,name=1364278304,phone=王4,email=w4@123.com
com.example.isszym.contact I/测试: contractID=55,name=1364278305,phone=王5,email=w5@123.com
com.example.isszym.contact I/测试: contractID=56,name=1364278306,phone=王6,email=w6@123.com
com.example.isszym.contact I/测试: contractID=57,name=1364278307,phone=王7,email=w7@123.com
com.example.isszym.contact I/测试: contractID=58,name=1364278308,phone=王8,email=w8@123.com
测试: contractID=59,name=1364278309,phone=王9,email=w9@123.com
测试: contractID=60,name=13642783010,phone=王10,email=w10@123.com
```

短信和拨号记录

- 概述

app可以通过使用ContentResolver获取安卓系统的短信记录，可用URI有以下这些:

content://sms	所有短信	content://sms/inbox	收件箱	content://sms/sent	已发送
content://sms/draft	草稿	content://sms/outbox	发件箱	content://sms/failed	发送失败
content://sms/queued	待发送列表				

app可以通过使用ContentResolver获取拨号记录，可用的Uri为:

`content://call_log/calls`	**所有拨入拨出电话记录**

app还可以通过ContentObserver监测短信记录和拨号记录是否发生变化。

• 数据库

➢ 短信使用的数据库是mmssms.db

sms表：

_id	INTEGER	一个自增字段，从1开始
thread_id	INTEGER	序号，同一发信人的id相同
address	TEXT	发件人手机号码
person	INTEGER	联系人列表里的序号，陌生人为null
date	INTEGER	发件日期
protocol	INTEGER	协议： 0 SMS_RPOTO, 1 MMS_PROTO
read	INTEGER	是否阅读: 0未读，1已读
status	INTEGER	状态:1接收 0 complete 64 pending, 128 failed
type	INTEGER	ALL = 0; INBOX =1; SENT = 2; DRAFT = 3; OUTBOX = 4; FAILED = 5; QUEUED = 6;
subject	TEXT	短信的主题
body	TEXT	短信内容
service_center	TEXT	短信服务中心号码编号
reply_path_present	INTEGER	TP-Reply-Path
locked	INTEGER	是否锁定

➢ 拨号记录使用的数据库为contact2.db/calls

表calls：

type	INTEGER	通话类型 （1-接入，2-打出，3-未接）
number	TEXT	电话号码
date	INTEGER	通话日期
duration	INTEGER	通话时长
new	INTEGER	是否被告知
is_read	INTEGER	是否被读
countryiso	TEXT	城市代码
geocoded_location	TEXT	地理位置
voicemail_uri	TEXT	语音邮件地址
name	TEXT	缓存名字*
numbertype	INTEGER	缓存电话类型*
lookup_uri	TEXT	缓存查找联系人*
matched_number	TEXT	缓存匹配电话号码*
normalized_number	TEXT	缓存标准版本电话号码*
photo_id	INTEGER	缓存照片id*
formatted_number	TEXT	缓存格式化电话*

带有*的表示不一定有值

- 项目名: SMS

```
public class MainActivity extends AppCompatActivity {
    private ContentResolver resolver = null;
    private static final String URI_SMS_LOG = "content://sms";
    private static final String URI_CALL_LOG = "content://call_log/calls";

    private Handler handler = new Handler() {
        public void handleMessage(android.os.Message msg) {
            switch (msg.what) {
                case 100:
                    String body = (String) msg.obj; Log.d("测试",body);
                    break;
            }
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setRight();
        resolver = getContentResolver();
        ContentObserver smsObserver = new  SmsContentObserver(this, handler);
        // 第二个参数,true表示观察所有派生Uri
        resolver.registerContentObserver(Uri.parse(URI_SMS_LOG), true, smsObserver);
       readSmsLog();
       readCallLog();
    }
```

＊其它权限码见附录

```
void  setRight() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_SMS)
            != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ_SMS}, 1);
    }
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.RECEIVE_SMS)
            != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.RECEIVE_SMS}, 1);
    }
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CALL_LOG)
            != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ_CALL_LOG}, 1);
    }
 }
 private void readSmsLog() {
    Cursor cursor_sms = resolver.query(Uri.parse(URI_SMS_LOG),
          new String[]{"_id", "address", "body", "date", "type"}, null, null, "_id desc limit 0,10");
    if (cursor_sms != null){
        while(cursor_sms.moveToNext()){
            String id = cursor_sms.getString(cursor_sms.getColumnIndex("_id"));
            String address = cursor_sms.getString(cursor_sms.getColumnIndex("address"));
            String body = cursor_sms.getString(cursor_sms.getColumnIndex("body"));
            String date = cursor_sms.getString(cursor_sms.getColumnIndex("date"));
            String type = cursor_sms.getString(cursor_sms.getColumnIndex("type"));
            String sms_type = "0".equals(type)?"待发":("1".equals(type)?"收到":"已发送");
            Log.d("测试","sms: "+ id + ":" + address + ":" + body + ":" + date + ":" + sms_type);

        }
        cursor_sms.close();
    }
 }
```

```
    private void readCallLog() {
        Cursor cursor_call = resolver.query(Uri.parse(URI_CALL_LOG),
                        new String[]{"_id", "number", "date", "type"}, null, null, "_id desc limit 0,10");
        if (cursor_call != null){
            while(cursor_call.moveToNext()){
                String id = cursor_call.getString(cursor_call.getColumnIndex("_id"));
                String number = cursor_call.getString(cursor_call.getColumnIndex("number"));
                String date = cursor_call.getString(cursor_call.getColumnIndex("date"));
                String type = cursor_call.getString(cursor_call.getColumnIndex("type"));
                String call_type = "1".equals(type)?"接入":("2".equals(type)?"打出":"未接");
                Log.d("测试","call: " + id + ":" + number + ":" + date + ":" + call_type );
            }
            cursor_call.close();
        }
    }
}
```

- 通过类Calls可以取到的常量：Calls.MISSED_TYPE, Calls.INCOMING_TYPE, Calls.OUTGOING_TYPE
- resolver.query(Calls.CONTENT_URI, new String[] {Calls.NUMBER, Calls.TYPE, Calls.NEW}, null, null, Calls.DEFAULT_SORT_ORDER);

```
class SmsContentObserver extends ContentObserver {
    private Handler handler;
    private Context context;

    public SmsContentObserver(Context context, Handler
            handler) {
        super(handler);
        this.handler = handler;
        this.context = context;
    }

    @Override
    public void onChange(boolean selfChange) {
        ContentResolver cr = context.getContentResolver();
        Cursor c = cr.query(Uri.parse("content://sms"), null, null, null, "date desc ");
        String sb = "";
        if (c!=null && c.moveToNext()) {
            String sendNumber = c.getString(c.getColumnIndex("address")); //发件人手机号码
            String body = c.getString(c.getColumnIndex("body"));  //信息内容
            //int hasRead = c.getInt(c.getColumnIndex("read")); // 表示是否已经读
            sb = "短信   "+sendNumber + ":" + body  + ":";
        }
        handler.obtainMessage(100, sb.toString()).sendToTarget();
    }
}
```

```
class CallContentObserver extends ContentObserver {
    private Handler handler;
    private Context context;

    public CallContentObserver(Context context, Handler
            handler) {
        super(handler);
        this.handler = handler;
        this.context = context;
    }

    @Override
    public void onChange(boolean selfChange) {
        ContentResolver resolver = context.getContentResolver();
        Cursor cursor_call = resolver.query(Uri.parse("content://call_log/calls"),
                      new String[]{"_id", "number", "date", "type"}, null, null, "_id desc limit 0,10");
        String sb="";
        if(cursor_call != null && cursor_call.moveToNext()){
            String id = cursor_call.getString(cursor_call.getColumnIndex("_id"));
            String number = cursor_call.getString(cursor_call.getColumnIndex("number"));
            String date = cursor_call.getString(cursor_call.getColumnIndex("date"));
            String type = cursor_call.getString(cursor_call.getColumnIndex("type"));
            String call_type = "1".equals(type)?"接入":("2".equals(type)?"打出":"未接");
            sb= "拨号   " + id + ":" + number + ":" + date + ":" + call_type;
        }
        cursor_call.close();
        handler.obtainMessage(100, sb.toString()).sendToTarget();
    }
}
```

运行之后显示所有短信和拨出电话

```
D/测试: sms: 7:王2:Hello:1557392435574:已发送
D/测试: sms: 6:王1:Are you Okey?:1556504464802:已发送
D/测试: sms: 5:王1:I'm John:1556504436799:已发送
D/测试: sms: 4:王1:Hi:1556504424007:已发送
D/测试: sms: 3:王10:I'm David:1556504389250:已发送
D/测试: sms: 2:王10:How are you?:1556504371198:已发送
D/测试: sms: 1:王10:Hello:1556504343800:已发送
D/测试: call: 14:1:1557382635182:打出
D/测试: call: 13:2:1557382621037:打出
D/测试: call: 12:13600015411:1556816506094:打出
D/测试: call: 11:13600015411:1556815647226:打出
D/测试: call: 10:1:1556504279149:打出
D/测试: call: 9:2:1556504268353:打出
D/测试: call: 8:10:1556504260304:打出
D/测试: call: 7:1:1556504251032:打出
D/测试: call: 6:2:1556504241055:打出
D/测试: call: 5:10:1556504231322:打出
```

通过观察器，收发短信和来电拨电话时会显示有关信息：

```
D/测试: 短信   王2:Hi:
D/测试: 短信   王1:Hello:
D/测试: 拨号   15:10:1557392995007:打出
D/测试: 拨号   15:10:1557392995007:打出
D/测试: 拨号   16:1:1557393017789:打出
D/测试: 拨号   16:1:1557393017789:打出
D/测试: 拨号   16:1:1557393017789:打出
```

附录1、单元测试

参考

@Before	在执行测试方法之前要执行的方法
@After	在执行测试方法之后要执行的方法
@Test	要测试的方法，可以输出测试值、捕捉异常
@Ignore	忽略的测试方法
@BeforeClass	针对所有测试只执行一次的方法，必须为static void
@AfterClass	针对所有测试只执行一次的方法，必须为static void

@Test(expected=ArithmeticException.class) 检查被测方法是否抛出ArithmeticException异常

一个JUnit4的单元测试用例执行顺序为：
@BeforeClass -> @Before -> @Test -> @After -> @AfterClass;

附录2、assertEquals

参考

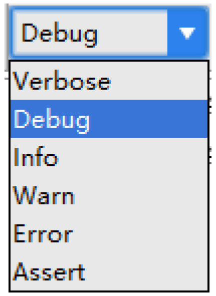
```
public static void assertFalse(boolean condition)
public static void assertTrue(boolean condition)
public static void assertTrue(String message, boolean condition)
   // Asserts that a condition is true.
   //   If it isn't it throws an AssertionError with the given message.
public static void fail()
public static void fail(String message)
// Fails a test with the given message.
 assertEquals(Double.NaN, Double.NaN, *)
public static void assertEquals(double expected, double actual)
public static void assertEquals(String message,double expected,double actual)
public static void assertEquals(String message,double expected,double actual,
                                double delta)
//Asserts that two doubles or floats are equal to within a positive delta.
//     If they are not, an AssertionError is thrown with the given message.
//       If the expected value is infinity then the delta value is ignored.
//  message - the identifying message for the AssertionError (null okay)
//  expected - expected value
//  actual - the value to check against expected
//  delta - the maximum delta between expected and actual for which both numbers
//          are still considered equal.
```

```
 public static void assertArrayEquals(String message,double [] expecteds,
         double [] actuals) throws ArrayComparisonFailure
//  Asserts that two double arrays are equal. If they are not,
//     an AssertionError is thrown with the given message.
//     If expecteds and actuals are null, they are considered equal.
public static void assertNotNull(String message,.Object object)
public static void assertSame(String message, Object expected,Object actual)
public static void assertNotSame(String message, Object unexpected,Object actual)
// Asserts that two objects do not refer to the same object. If they do refer to
//    the same object, an AssertionError is thrown with the given message.

public static <T> void assertThat(T actual, Matcher<T> matcher)
// Asserts that actual satisfies the condition specified by matcher. If not,
//   an AssertionError is thrown with information about the matcher and
//   failing value. Example:
//        assertThat(0, is(1));
// fails: failure message:
//            expected: is <1>
//            got value: <0>
//        assertThat(0, is(not(1))) // passes
//    T - the static type accepted by the matcher (this can flag obvious
//                          compile-time problems such as assertThat(1, is("a"))
//    actual - the computed value being compared
//    matcher - an expression, built of Matchers, specifying allowed values
```

附录3、Log

- 采用Log可以输出信息到监控日志（Android Monitor/logcat）中，共有五类Log语句：

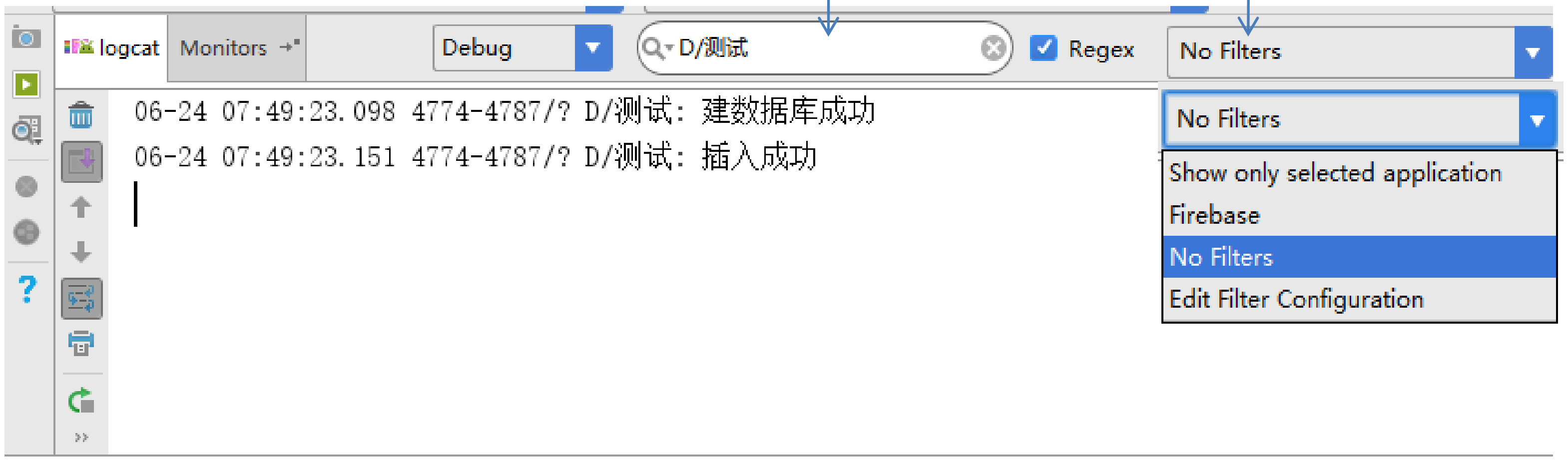
Log.v	verbose(啰嗦)
Log.d	debug(调试)
Log.i	information(信息)
Log.w	warning(警告)
Log.e	error(错误)



- 格式：　　　Log.i(Tag,Messaage)
- 查看方法：

设置过滤子串

设置为不过滤



```
06-24 07:49:23.098 4774-4787/? D/测试: 建数据库成功
06-24 07:49:23.151 4774-4787/? D/测试: 插入成功
```

附录4、SQLite的数据类型 参考

亲和类型	描述
TEXT	数值型数据在被插入之前，需要先被转换为文本格式，之后再插入到目标字段中。
NUMERIC	当文本数据被插入到亲缘性为NUMERIC的字段中时，如果转换操作不会导致数据信息丢失以及完全可逆，那么SQLite就会将该文本数据转换为INTEGER或REAL类型的数据，如果转换失败，SQLite仍会以TEXT方式存储该数据。对于NULL或BLOB类型的新数据，SQLite将不做任何转换，直接以NULL或BLOB的方式存储该数据。需要额外说明的是，对于浮点格式的常量文本，如"30000.0"，如果该值可以转换为INTEGER同时又不会丢失数值信息，那么SQLite就会将其转换为INTEGER的存储方式。
INTEGER	对于亲缘类型为INTEGER的字段，其规则等同于NUMERIC，唯一差别是在执行CAST表达式时。
REAL	其规则基本等同于NUMERIC，唯一的差别是不会将"30000.0"这样的文本数据转换为INTEGER存储方式。
NONE	不做任何的转换，直接以该数据所属的数据类型进行存储。

数据类型	亲和类型
INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8	INTEGER

CHARACTER(20)
VARCHAR(255)
VARYING CHARACTER(255)
NCHAR(55)
NATIVE CHARACTER(70)
NVARCHAR(100)
TEXT
CLOB
	TEXT

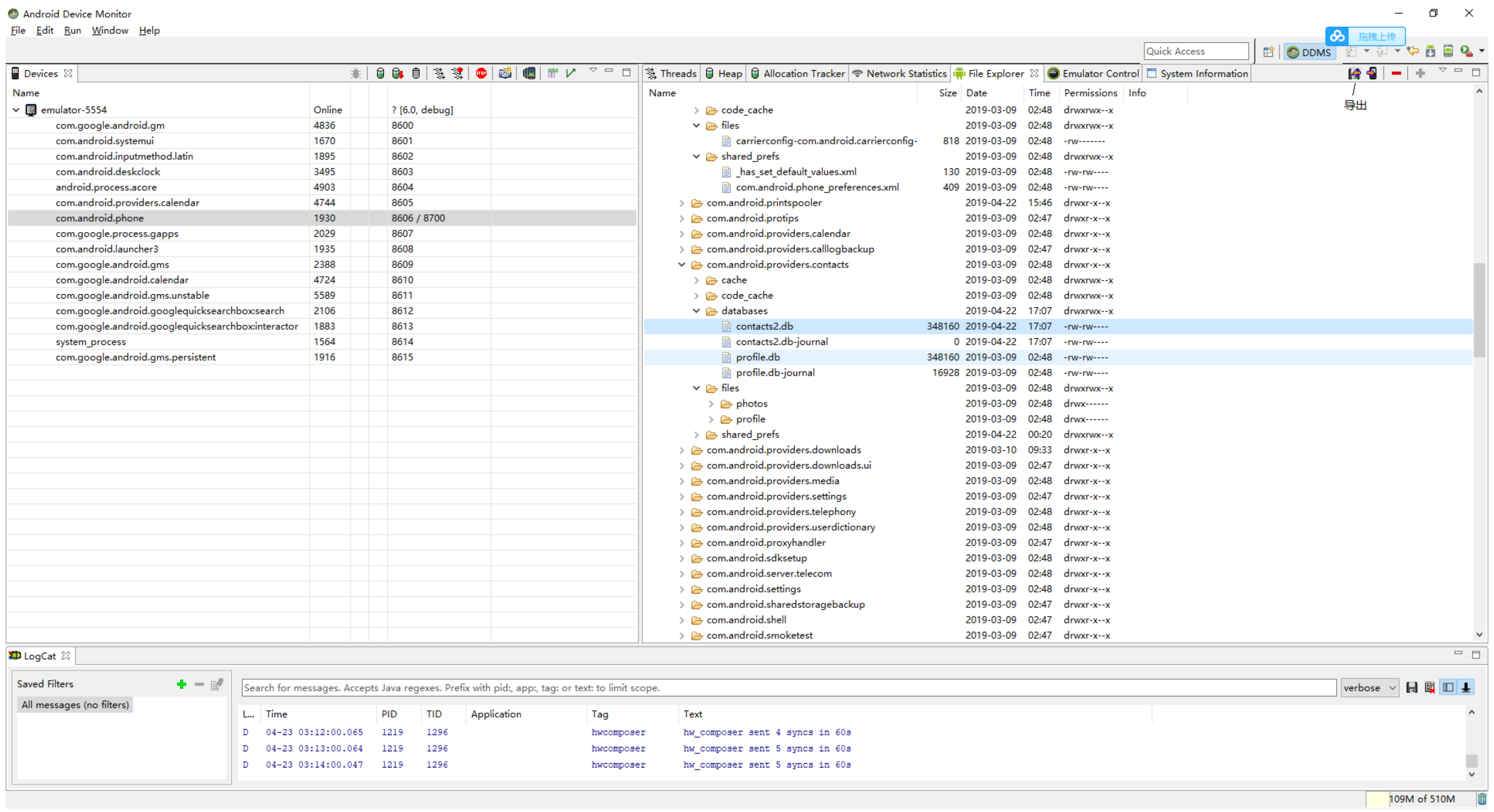
BLOB no datatype specified	NONE
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME TIME TIMESTAMP	NUMERIC

SQLite 没有单独的 Boolean 存储类。相反，布尔值被存储为整数 0（false）和 1（true）。 SQLite 没有一个单独的用于存储日期和/或时间的存储类，但 SQLite 能够把日期和时间存储为 TEXT、REAL 或 INTEGER 值。

附录5、通讯录数据库

导出数据库并且用Navicat打开：

contacts2.db



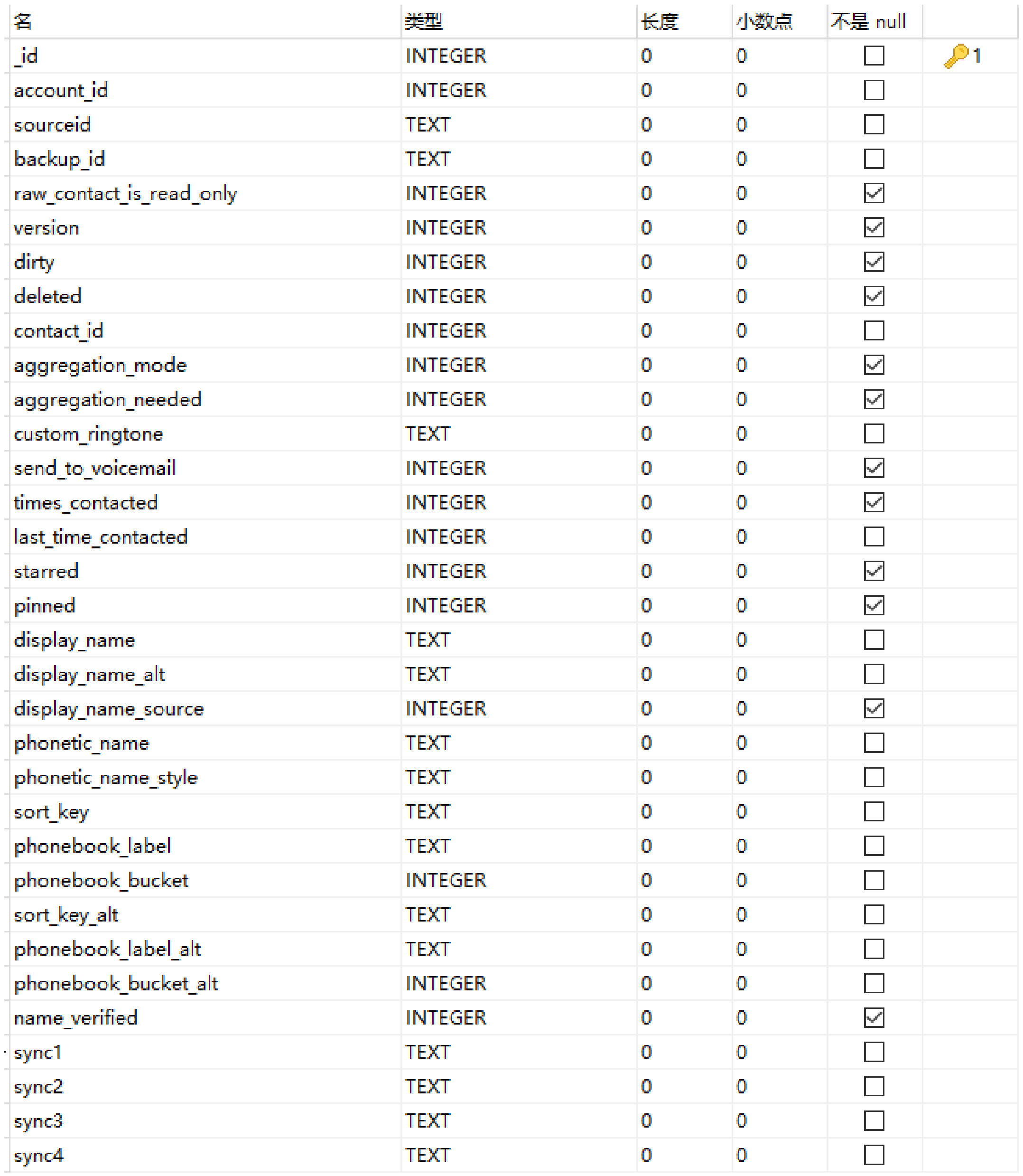
Name		
emulator-5554	Online	? [6.0, debug]
com.google.android.gm	4836	8600
com.android.systemui	1670	8601
com.android.inputmethod.latin	1895	8602
com.android.deskclock	3495	8603
android.process.acore	4903	8604
com.android.providers.calendar	4744	8605
com.android.phone	1930	8606 / 8700
com.google.process.gapps	2029	8607
com.android.launcher3	1935	8608
com.google.android.gms	2388	8609
com.google.android.calendar	4724	8610
com.google.android.gms.unstable	5589	8611
com.google.android.googlequicksearchbox:search	2106	8612
com.google.android.googlequicksearchbox:interactor	1883	8613
system_process	1564	8614
com.google.android.gms.persistent	1916	8615

Name	Size	Date	Time	Permissions	Info
code_cache		2019-03-09	02:48	drwxrwx--x	
files		2019-03-09	02:48	drwxrwx--x	
carrierconfig-com.android.carrierconfig-	818	2019-03-09	02:48	-rw-------	
shared_prefs		2019-03-09	02:48	drwxrwx--x	
_has_set_default_values.xml	130	2019-03-09	02:48	-rw-rw----	
com.android.phone_preferences.xml	409	2019-03-09	02:48	-rw-rw----	
com.android.printspooler		2019-04-22	15:46	drwxr-x--x	
com.android.protips		2019-03-09	02:47	drwxr-x--x	
com.android.providers.calendar		2019-03-09	02:48	drwxr-x--x	
com.android.providers.calllogbackup		2019-03-09	02:47	drwxr-x--x	
com.android.providers.contacts		2019-03-09	02:48	drwxr-x--x	
cache		2019-03-09	02:48	drwxrwx--x	
code_cache		2019-03-09	02:48	drwxrwx--x	
databases		2019-04-22	17:07	drwxrwx--x	
contacts2.db	348160	2019-04-22	17:07	-rw-rw----	
contacts2.db-journal	0	2019-04-22	17:07	-rw-rw----	
profile.db	348160	2019-03-09	02:48	-rw-rw----	
profile.db-journal	16928	2019-03-09	02:48	-rw-rw----	
files		2019-03-09	02:48	drwxrwx--x	
photos		2019-03-09	02:48	drwx------	
profile		2019-03-09	02:48	drwx------	
shared_prefs		2019-04-22	00:20	drwxrwx--x	
com.android.providers.downloads		2019-03-10	09:33	drwxr-x--x	
com.android.providers.downloads.ui		2019-03-09	02:47	drwxr-x--x	
com.android.providers.media		2019-03-09	02:48	drwxr-x--x	
com.android.providers.settings		2019-03-09	02:47	drwxr-x--x	
com.android.providers.telephony		2019-03-09	02:48	drwxr-x--x	
com.android.providers.userdictionary		2019-03-09	02:48	drwxr-x--x	
com.android.proxyhandler		2019-03-09	02:47	drwxr-x--x	
com.android.sdksetup		2019-03-09	02:48	drwxr-x--x	
com.android.server.telecom		2019-03-09	02:48	drwxr-x--x	
com.android.settings		2019-03-09	02:48	drwxr-x--x	
com.android.sharedstoragebackup		2019-03-09	02:47	drwxr-x--x	
com.android.shell		2019-03-09	02:47	drwxr-x--x	
com.android.smoketest		2019-03-09	02:47	drwxr-x--x	

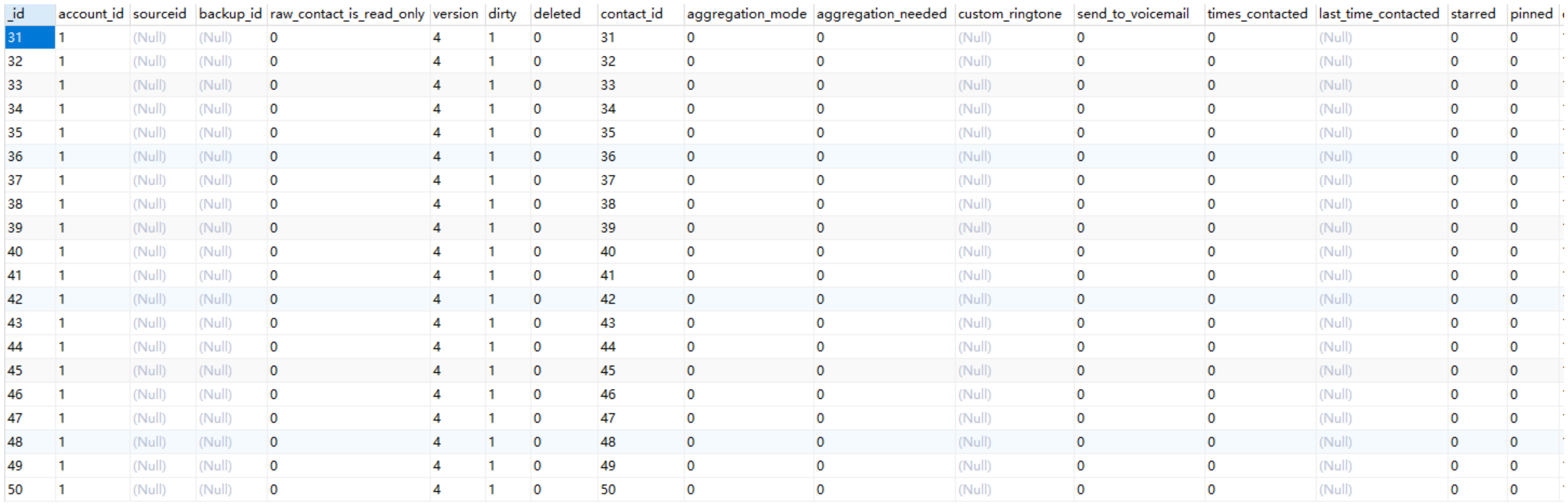
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

```
L...  Time                 PID    TID    Application    Tag           Text
D     04-23 03:12:00.065   1219   1296                  hwcomposer    hw_composer sent 4 syncs in 60s
D     04-23 03:13:00.064   1219   1296                  hwcomposer    hw_composer sent 5 syncs in 60s
D     04-23 03:14:00.047   1219   1296                  hwcomposer    hw_composer sent 5 syncs in 60s
```

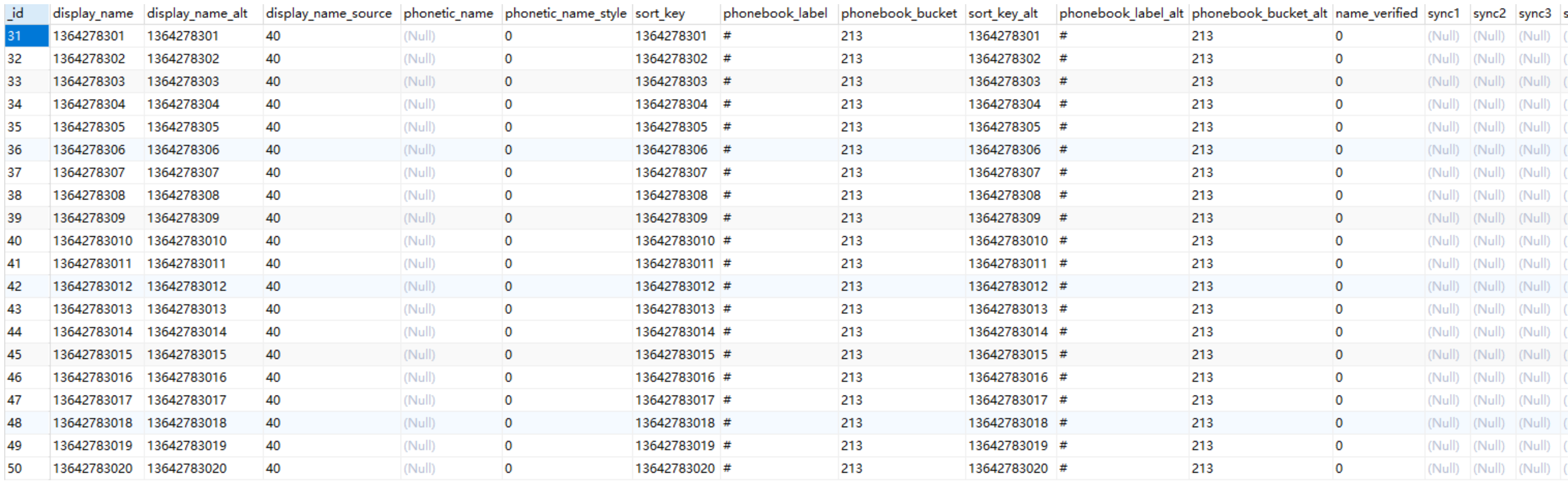
- raw_contacts



名	类型	长度	小数点	不是 null	
_id	INTEGER	0	0	☐	🔑1
account_id	INTEGER	0	0	☐	
sourceid	TEXT	0	0	☐	
backup_id	TEXT	0	0	☐	
raw_contact_is_read_only	INTEGER	0	0	☑	
version	INTEGER	0	0	☑	
dirty	INTEGER	0	0	☑	
deleted	INTEGER	0	0	☑	
contact_id	INTEGER	0	0	☐	
aggregation_mode	INTEGER	0	0	☑	
aggregation_needed	INTEGER	0	0	☑	
custom_ringtone	TEXT	0	0	☐	
send_to_voicemail	INTEGER	0	0	☑	
times_contacted	INTEGER	0	0	☑	
last_time_contacted	INTEGER	0	0	☐	
starred	INTEGER	0	0	☑	
pinned	INTEGER	0	0	☑	
display_name	TEXT	0	0	☐	
display_name_alt	TEXT	0	0	☐	
display_name_source	INTEGER	0	0	☑	
phonetic_name	TEXT	0	0	☐	
phonetic_name_style	TEXT	0	0	☐	
sort_key	TEXT	0	0	☐	
phonebook_label	TEXT	0	0	☐	
phonebook_bucket	INTEGER	0	0	☐	
sort_key_alt	TEXT	0	0	☐	
phonebook_label_alt	TEXT	0	0	☐	
phonebook_bucket_alt	INTEGER	0	0	☐	
name_verified	INTEGER	0	0	☑	
sync1	TEXT	0	0	☐	
sync2	TEXT	0	0	☐	
sync3	TEXT	0	0	☐	
sync4	TEXT	0	0	☐	

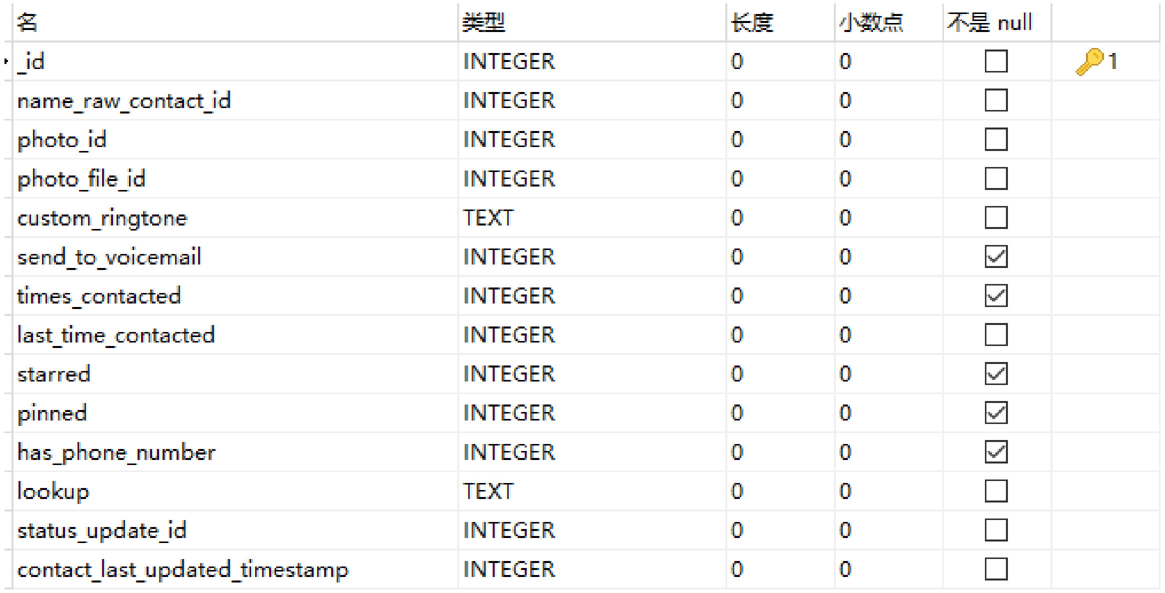


_id	account_id	sourceid	backup_id	raw_contact_is_read_only	version	dirty	deleted	contact_id	aggregation_mode	aggregation_needed	custom_ringtone	send_to_voicemail	times_contacted	last_time_contacted	starred	pinned
31	1	(Null)	(Null)	0	4	1	0	31	0	0	(Null)	0	0	(Null)	0	0
32	1	(Null)	(Null)	0	4	1	0	32	0	0	(Null)	0	0	(Null)	0	0
33	1	(Null)	(Null)	0	4	1	0	33	0	0	(Null)	0	0	(Null)	0	0
34	1	(Null)	(Null)	0	4	1	0	34	0	0	(Null)	0	0	(Null)	0	0
35	1	(Null)	(Null)	0	4	1	0	35	0	0	(Null)	0	0	(Null)	0	0
36	1	(Null)	(Null)	0	4	1	0	36	0	0	(Null)	0	0	(Null)	0	0
37	1	(Null)	(Null)	0	4	1	0	37	0	0	(Null)	0	0	(Null)	0	0
38	1	(Null)	(Null)	0	4	1	0	38	0	0	(Null)	0	0	(Null)	0	0
39	1	(Null)	(Null)	0	4	1	0	39	0	0	(Null)	0	0	(Null)	0	0
40	1	(Null)	(Null)	0	4	1	0	40	0	0	(Null)	0	0	(Null)	0	0
41	1	(Null)	(Null)	0	4	1	0	41	0	0	(Null)	0	0	(Null)	0	0
42	1	(Null)	(Null)	0	4	1	0	42	0	0	(Null)	0	0	(Null)	0	0
43	1	(Null)	(Null)	0	4	1	0	43	0	0	(Null)	0	0	(Null)	0	0
44	1	(Null)	(Null)	0	4	1	0	44	0	0	(Null)	0	0	(Null)	0	0
45	1	(Null)	(Null)	0	4	1	0	45	0	0	(Null)	0	0	(Null)	0	0
46	1	(Null)	(Null)	0	4	1	0	46	0	0	(Null)	0	0	(Null)	0	0
47	1	(Null)	(Null)	0	4	1	0	47	0	0	(Null)	0	0	(Null)	0	0
48	1	(Null)	(Null)	0	4	1	0	48	0	0	(Null)	0	0	(Null)	0	0
49	1	(Null)	(Null)	0	4	1	0	49	0	0	(Null)	0	0	(Null)	0	0
50	1	(Null)	(Null)	0	4	1	0	50	0	0	(Null)	0	0	(Null)	0	0

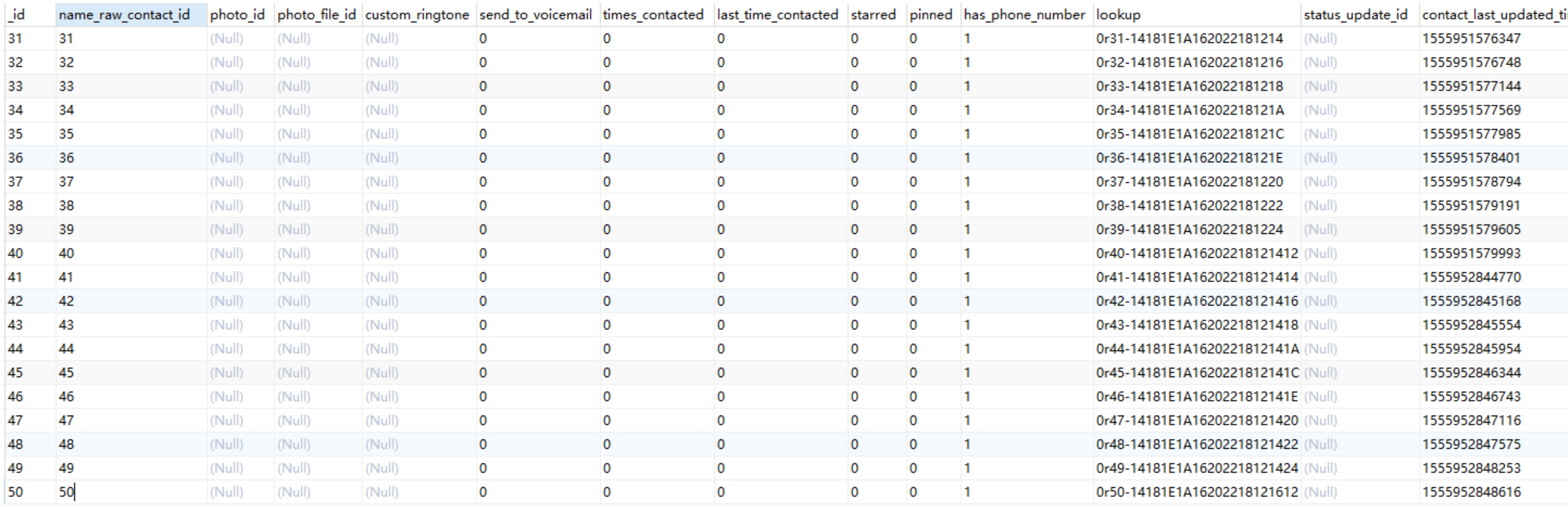


_id	display_name	display_name_alt	display_name_source	phonetic_name	phonetic_name_style	sort_key	phonebook_label	phonebook_bucket	sort_key_alt	phonebook_label_alt	phonebook_bucket_alt	name_verified	sync1	sync2	sync3
31	1364278301	1364278301	40	(Null)	0	1364278301	#	213	1364278301	#	213	0	(Null)	(Null)	(Null)
32	1364278302	1364278302	40	(Null)	0	1364278302	#	213	1364278302	#	213	0	(Null)	(Null)	(Null)
33	1364278303	1364278303	40	(Null)	0	1364278303	#	213	1364278303	#	213	0	(Null)	(Null)	(Null)
34	1364278304	1364278304	40	(Null)	0	1364278304	#	213	1364278304	#	213	0	(Null)	(Null)	(Null)
35	1364278305	1364278305	40	(Null)	0	1364278305	#	213	1364278305	#	213	0	(Null)	(Null)	(Null)
36	1364278306	1364278306	40	(Null)	0	1364278306	#	213	1364278306	#	213	0	(Null)	(Null)	(Null)
37	1364278307	1364278307	40	(Null)	0	1364278307	#	213	1364278307	#	213	0	(Null)	(Null)	(Null)
38	1364278308	1364278308	40	(Null)	0	1364278308	#	213	1364278308	#	213	0	(Null)	(Null)	(Null)
39	1364278309	1364278309	40	(Null)	0	1364278309	#	213	1364278309	#	213	0	(Null)	(Null)	(Null)
40	13642783010	13642783010	40	(Null)	0	13642783010	#	213	13642783010	#	213	0	(Null)	(Null)	(Null)
41	13642783011	13642783011	40	(Null)	0	13642783011	#	213	13642783011	#	213	0	(Null)	(Null)	(Null)
42	13642783012	13642783012	40	(Null)	0	13642783012	#	213	13642783012	#	213	0	(Null)	(Null)	(Null)
43	13642783013	13642783013	40	(Null)	0	13642783013	#	213	13642783013	#	213	0	(Null)	(Null)	(Null)
44	13642783014	13642783014	40	(Null)	0	13642783014	#	213	13642783014	#	213	0	(Null)	(Null)	(Null)
45	13642783015	13642783015	40	(Null)	0	13642783015	#	213	13642783015	#	213	0	(Null)	(Null)	(Null)
46	13642783016	13642783016	40	(Null)	0	13642783016	#	213	13642783016	#	213	0	(Null)	(Null)	(Null)
47	13642783017	13642783017	40	(Null)	0	13642783017	#	213	13642783017	#	213	0	(Null)	(Null)	(Null)
48	13642783018	13642783018	40	(Null)	0	13642783018	#	213	13642783018	#	213	0	(Null)	(Null)	(Null)
49	13642783019	13642783019	40	(Null)	0	13642783019	#	213	13642783019	#	213	0	(Null)	(Null)	(Null)
50	13642783020	13642783020	40	(Null)	0	13642783020	#	213	13642783020	#	213	0	(Null)	(Null)	(Null)

- contacts

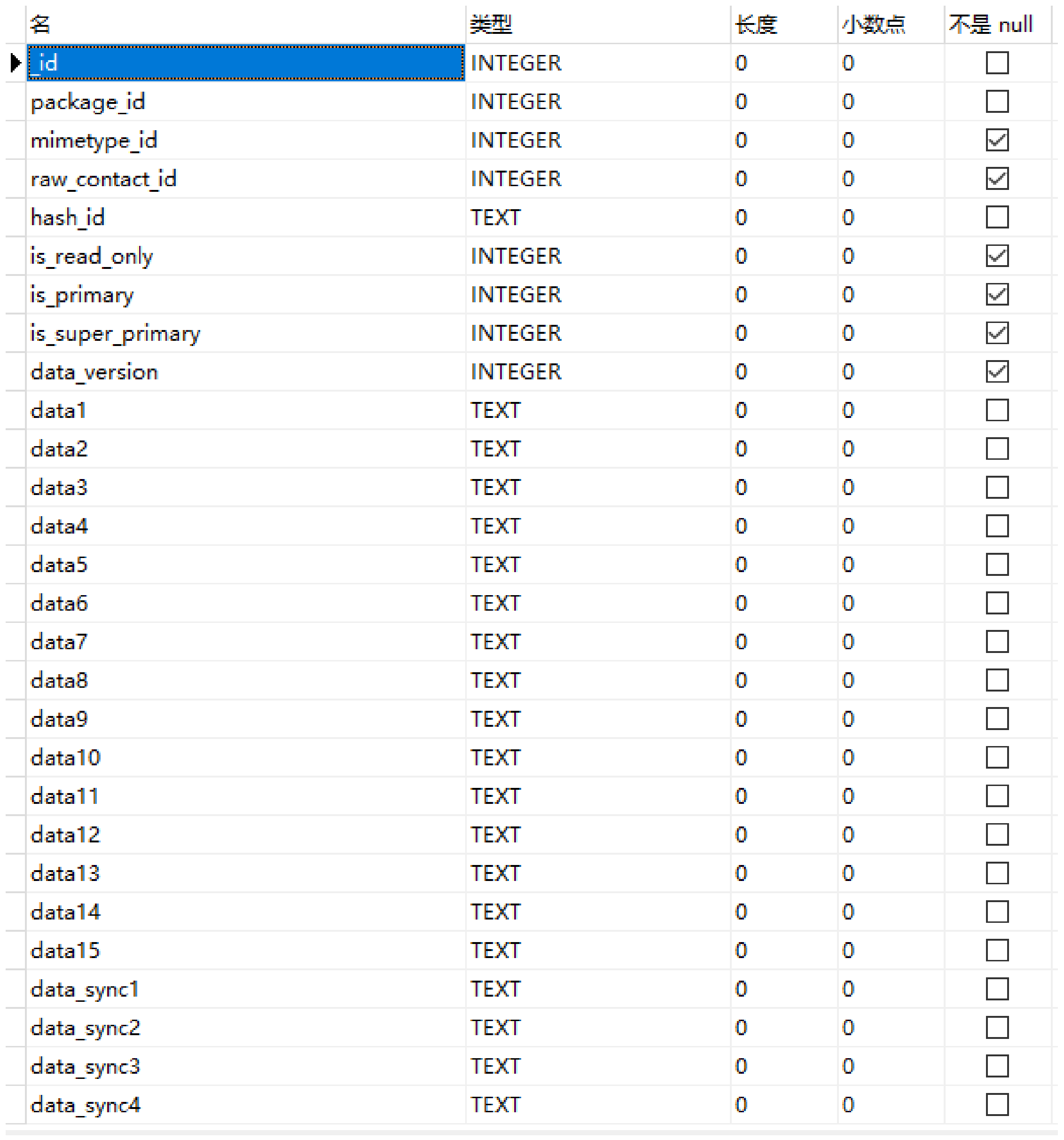


名	类型	长度	小数点	不是 null	
_id	INTEGER	0	0	☐	🔑 1
name_raw_contact_id	INTEGER	0	0	☐	
photo_id	INTEGER	0	0	☐	
photo_file_id	INTEGER	0	0	☐	
custom_ringtone	TEXT	0	0	☐	
send_to_voicemail	INTEGER	0	0	☑	
times_contacted	INTEGER	0	0	☑	
last_time_contacted	INTEGER	0	0	☐	
starred	INTEGER	0	0	☑	
pinned	INTEGER	0	0	☑	
has_phone_number	INTEGER	0	0	☑	
lookup	TEXT	0	0	☐	
status_update_id	INTEGER	0	0	☐	
contact_last_updated_timestamp	INTEGER	0	0	☐	

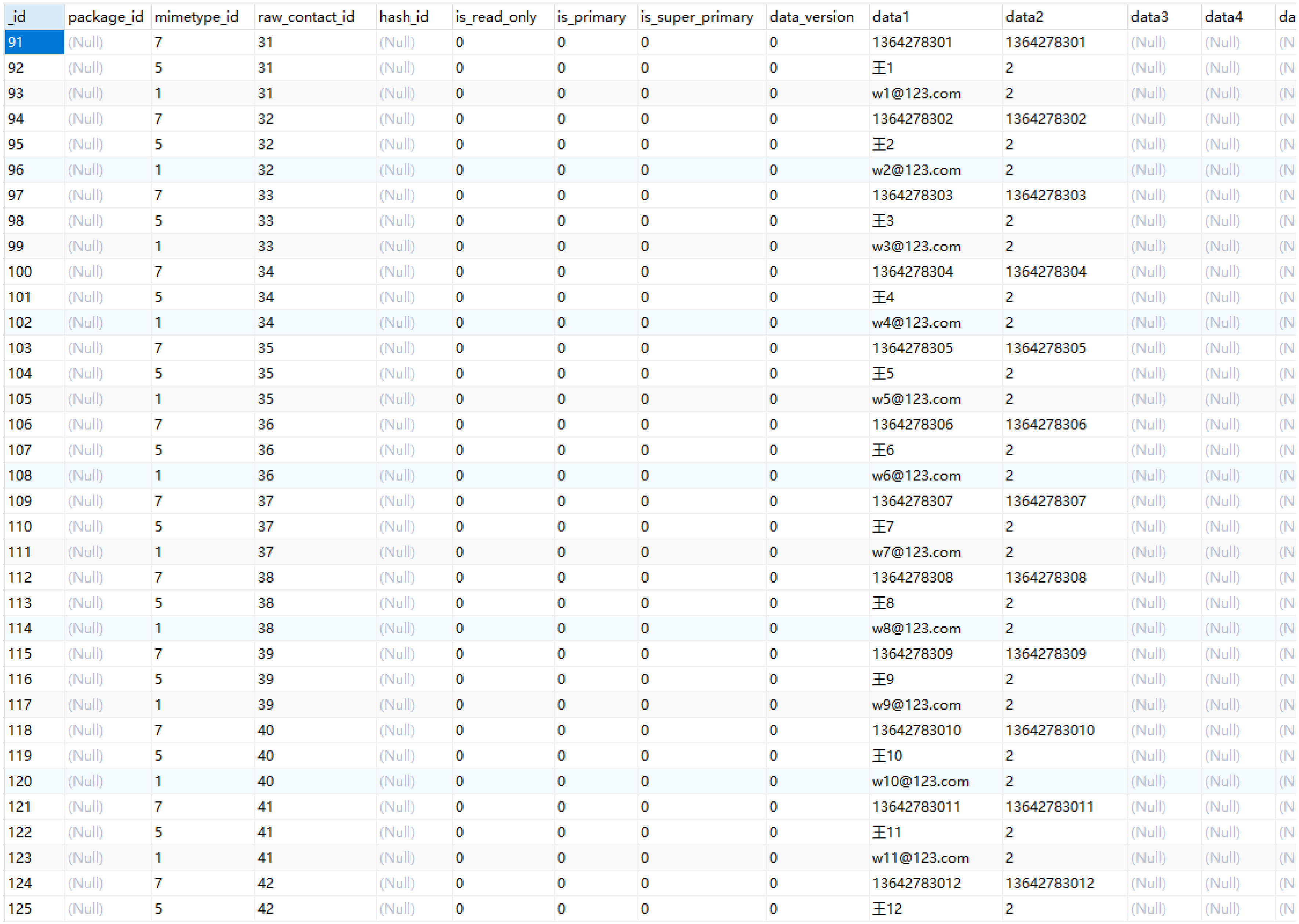


_id	name_raw_contact_id	photo_id	photo_file_id	custom_ringtone	send_to_voicemail	times_contacted	last_time_contacted	starred	pinned	has_phone_number	lookup	status_update_id	contact_last_updated_ti
31	31	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r31-14181E1A162022181214	(Null)	1555951576347
32	32	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r32-14181E1A162022181216	(Null)	1555951576748
33	33	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r33-14181E1A162022181218	(Null)	1555951577144
34	34	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r34-14181E1A16202218121A	(Null)	1555951577569
35	35	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r35-14181E1A16202218121C	(Null)	1555951577985
36	36	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r36-14181E1A16202218121E	(Null)	1555951578401
37	37	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r37-14181E1A162022181220	(Null)	1555951578794
38	38	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r38-14181E1A162022181222	(Null)	1555951579191
39	39	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r39-14181E1A162022181224	(Null)	1555951579605
40	40	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r40-14181E1A16202218121412	(Null)	1555951579993
41	41	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r41-14181E1A16202218121414	(Null)	1555952844770
42	42	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r42-14181E1A16202218121416	(Null)	1555952845168
43	43	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r43-14181E1A16202218121418	(Null)	1555952845554
44	44	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r44-14181E1A1620221812141A	(Null)	1555952845954
45	45	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r45-14181E1A1620221812141C	(Null)	1555952846344
46	46	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r46-14181E1A1620221812141E	(Null)	1555952846743
47	47	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r47-14181E1A16202218121420	(Null)	1555952847116
48	48	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r48-14181E1A16202218121422	(Null)	1555952847575
49	49	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r49-14181E1A16202218121424	(Null)	1555952848253
50	50	(Null)	(Null)	(Null)	0	0	0	0	0	1	0r50-14181E1A16202218121612	(Null)	1555952848616

- data表

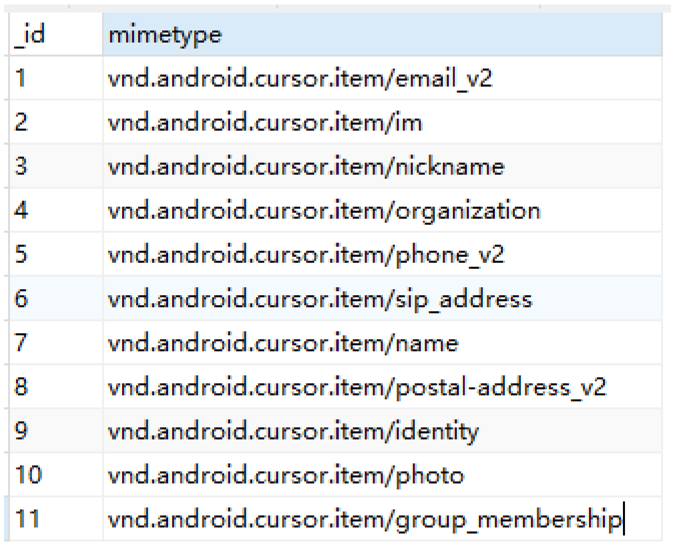


名	类型	长度	小数点	不是 null
id	INTEGER	0	0	☐
package_id	INTEGER	0	0	☐
mimetype_id	INTEGER	0	0	☑
raw_contact_id	INTEGER	0	0	☑
hash_id	TEXT	0	0	☐
is_read_only	INTEGER	0	0	☑
is_primary	INTEGER	0	0	☑
is_super_primary	INTEGER	0	0	☑
data_version	INTEGER	0	0	☑
data1	TEXT	0	0	☐
data2	TEXT	0	0	☐
data3	TEXT	0	0	☐
data4	TEXT	0	0	☐
data5	TEXT	0	0	☐
data6	TEXT	0	0	☐
data7	TEXT	0	0	☐
data8	TEXT	0	0	☐
data9	TEXT	0	0	☐
data10	TEXT	0	0	☐
data11	TEXT	0	0	☐
data12	TEXT	0	0	☐
data13	TEXT	0	0	☐
data14	TEXT	0	0	☐
data15	TEXT	0	0	☐
data_sync1	TEXT	0	0	☐
data_sync2	TEXT	0	0	☐
data_sync3	TEXT	0	0	☐
data_sync4	TEXT	0	0	☐



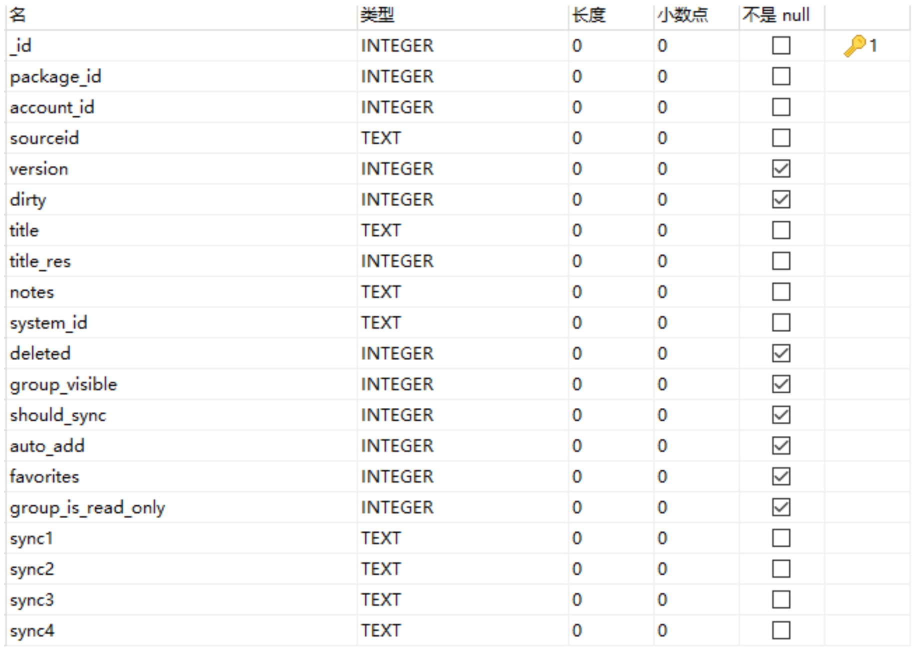
_id	package_id	mimetype_id	raw_contact_id	hash_id	is_read_only	is_primary	is_super_primary	data_version	data1	data2	data3	data4	da
91	(Null)	7	31	(Null)	0	0	0	0	1364278301	1364278301	(Null)	(Null)	(N
92	(Null)	5	31	(Null)	0	0	0	0	王1	2	(Null)	(Null)	(N
93	(Null)	1	31	(Null)	0	0	0	0	w1@123.com	2	(Null)	(Null)	(N
94	(Null)	7	32	(Null)	0	0	0	0	1364278302	1364278302	(Null)	(Null)	(N
95	(Null)	5	32	(Null)	0	0	0	0	王2	2	(Null)	(Null)	(N
96	(Null)	1	32	(Null)	0	0	0	0	w2@123.com	2	(Null)	(Null)	(N
97	(Null)	7	33	(Null)	0	0	0	0	1364278303	1364278303	(Null)	(Null)	(N
98	(Null)	5	33	(Null)	0	0	0	0	王3	2	(Null)	(Null)	(N
99	(Null)	1	33	(Null)	0	0	0	0	w3@123.com	2	(Null)	(Null)	(N
100	(Null)	7	34	(Null)	0	0	0	0	1364278304	1364278304	(Null)	(Null)	(N
101	(Null)	5	34	(Null)	0	0	0	0	王4	2	(Null)	(Null)	(N
102	(Null)	1	34	(Null)	0	0	0	0	w4@123.com	2	(Null)	(Null)	(N
103	(Null)	7	35	(Null)	0	0	0	0	1364278305	1364278305	(Null)	(Null)	(N
104	(Null)	5	35	(Null)	0	0	0	0	王5	2	(Null)	(Null)	(N
105	(Null)	1	35	(Null)	0	0	0	0	w5@123.com	2	(Null)	(Null)	(N
106	(Null)	7	36	(Null)	0	0	0	0	1364278306	1364278306	(Null)	(Null)	(N
107	(Null)	5	36	(Null)	0	0	0	0	王6	2	(Null)	(Null)	(N
108	(Null)	1	36	(Null)	0	0	0	0	w6@123.com	2	(Null)	(Null)	(N
109	(Null)	7	37	(Null)	0	0	0	0	1364278307	1364278307	(Null)	(Null)	(N
110	(Null)	5	37	(Null)	0	0	0	0	王7	2	(Null)	(Null)	(N
111	(Null)	1	37	(Null)	0	0	0	0	w7@123.com	2	(Null)	(Null)	(N
112	(Null)	7	38	(Null)	0	0	0	0	1364278308	1364278308	(Null)	(Null)	(N
113	(Null)	5	38	(Null)	0	0	0	0	王8	2	(Null)	(Null)	(N
114	(Null)	1	38	(Null)	0	0	0	0	w8@123.com	2	(Null)	(Null)	(N
115	(Null)	7	39	(Null)	0	0	0	0	1364278309	1364278309	(Null)	(Null)	(N
116	(Null)	5	39	(Null)	0	0	0	0	王9	2	(Null)	(Null)	(N
117	(Null)	1	39	(Null)	0	0	0	0	w9@123.com	2	(Null)	(Null)	(N
118	(Null)	7	40	(Null)	0	0	0	0	13642783010	13642783010	(Null)	(Null)	(N
119	(Null)	5	40	(Null)	0	0	0	0	王10	2	(Null)	(Null)	(N
120	(Null)	1	40	(Null)	0	0	0	0	w10@123.com	2	(Null)	(Null)	(N
121	(Null)	7	41	(Null)	0	0	0	0	13642783011	13642783011	(Null)	(Null)	(N
122	(Null)	5	41	(Null)	0	0	0	0	王11	2	(Null)	(Null)	(N
123	(Null)	1	41	(Null)	0	0	0	0	w11@123.com	2	(Null)	(Null)	(N
124	(Null)	7	42	(Null)	0	0	0	0	13642783012	13642783012	(Null)	(Null)	(N
125	(Null)	5	42	(Null)	0	0	0	0	王12	2	(Null)	(Null)	(N

- mimetypes



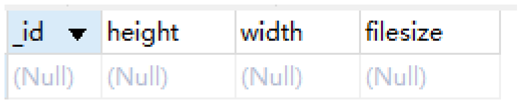
_id	mimetype
1	vnd.android.cursor.item/email_v2
2	vnd.android.cursor.item/im
3	vnd.android.cursor.item/nickname
4	vnd.android.cursor.item/organization
5	vnd.android.cursor.item/phone_v2
6	vnd.android.cursor.item/sip_address
7	vnd.android.cursor.item/name
8	vnd.android.cursor.item/postal-address_v2
9	vnd.android.cursor.item/identity
10	vnd.android.cursor.item/photo
11	vnd.android.cursor.item/group_membership

- groups



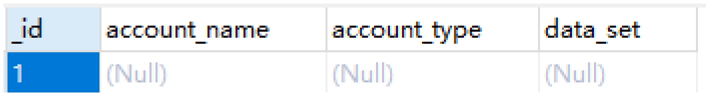
名	类型	长度	小数点	不是 null	
_id	INTEGER	0	0	☐	🔑1
package_id	INTEGER	0	0	☐	
account_id	INTEGER	0	0	☐	
sourceid	TEXT	0	0	☐	
version	INTEGER	0	0	☑	
dirty	INTEGER	0	0	☑	
title	TEXT	0	0	☐	
title_res	INTEGER	0	0	☐	
notes	TEXT	0	0	☐	
system_id	TEXT	0	0	☐	
deleted	INTEGER	0	0	☑	
group_visible	INTEGER	0	0	☑	
should_sync	INTEGER	0	0	☑	
auto_add	INTEGER	0	0	☑	
favorites	INTEGER	0	0	☑	
group_is_read_only	INTEGER	0	0	☑	
sync1	TEXT	0	0	☐	
sync2	TEXT	0	0	☐	
sync3	TEXT	0	0	☐	
sync4	TEXT	0	0	☐	

- photo_files



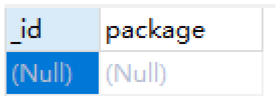
_id	height	width	filesize
(Null)	(Null)	(Null)	(Null)

- accounts



_id	account_name	account_type	data_set
1	(Null)	(Null)	(Null)

- packages

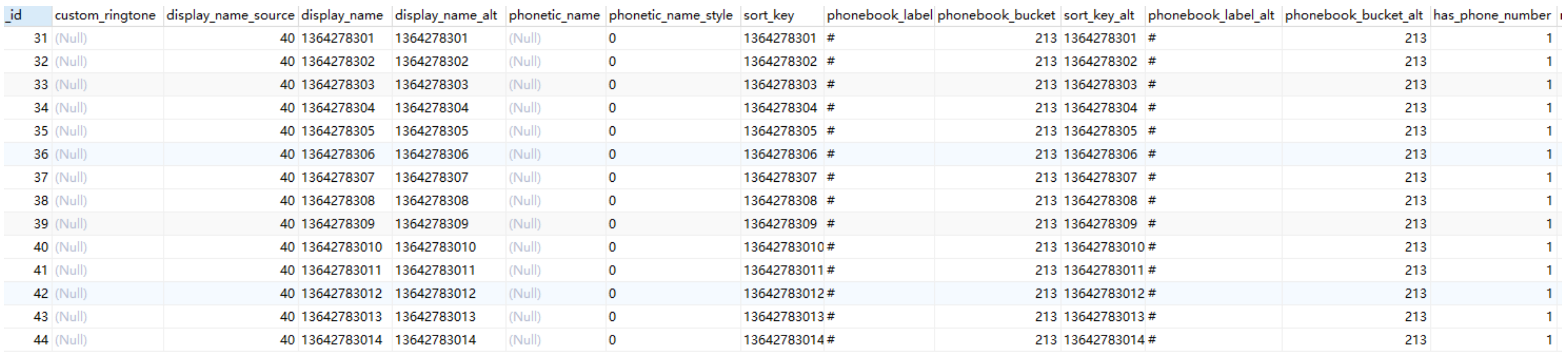


_id	package
(Null)	(Null)

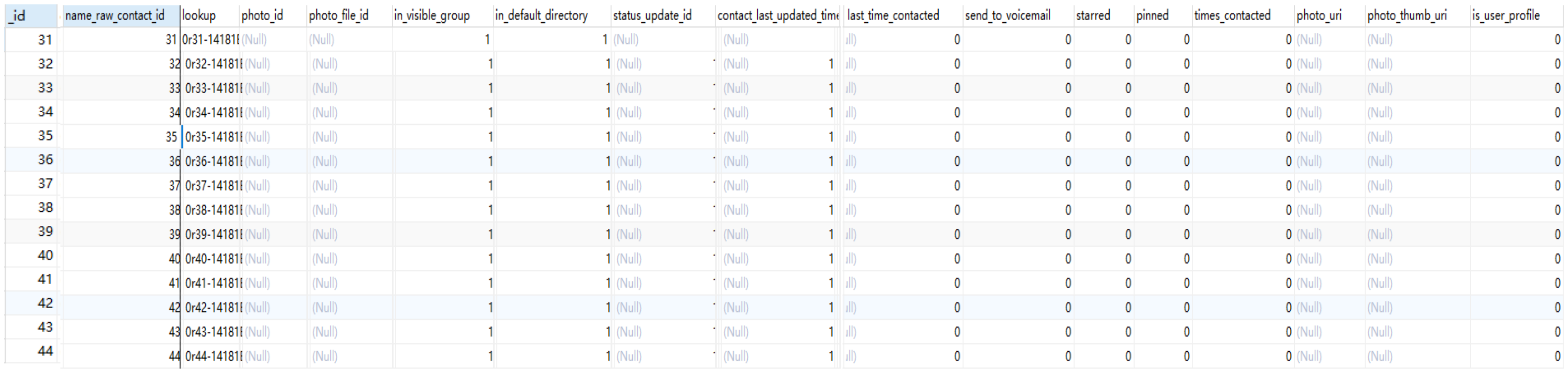
附录6、通讯录的View

- view_contacts

```
SELECT contacts._id AS _id,contacts.custom_ringtone AS custom_ringtone,
name_raw_contact.display_name_source AS display_name_source,
name_raw_contact.display_name AS display_name,
name_raw_contact.display_name_alt AS display_name_alt,
name_raw_contact.phonetic_name AS phonetic_name,
name_raw_contact.phonetic_name_style AS phonetic_name_style,
name_raw_contact.sort_key AS sort_key,
name_raw_contact.phonebook_label AS phonebook_label,
name_raw_contact.phonebook_bucket AS phonebook_bucket,
name_raw_contact.sort_key_alt AS sort_key_alt,
name_raw_contact.phonebook_label_alt AS phonebook_label_alt,
name_raw_contact.phonebook_bucket_alt AS phonebook_bucket_alt,
has_phone_number, name_raw_contact_id, lookup, photo_id, photo_file_id,
CAST(EXISTS (SELECT _id FROM visible_contacts WHERE contacts._id=visible_contacts._id) AS INTEGER) AS in_visible_group,
CAST(EXISTS (SELECT _id FROM default_directory WHERE contacts._id=default_directory._id) AS INTEGER) AS in_default_directory,
status_update_id, contacts.contact_last_updated_timestamp,
contacts.last_time_contacted AS last_time_contacted,
contacts.send_to_voicemail AS send_to_voicemail,
contacts.starred AS starred, contacts.pinned AS pinned,
contacts.times_contacted AS times_contacted,
(CASE WHEN photo_file_id IS NULL THEN (CASE WHEN photo_id IS NULL OR photo_id=0 THEN NULL ELSE
'content://com.android.contacts/contacts/'||contacts._id|| '/photo' END) ELSE
'content://com.android.contacts/display_photo/'||photo_file_id END) AS photo_uri,
 (CASE WHEN photo_id IS NULL OR photo_id=0 THEN NULL ELSE 'content://com.android.contacts/contacts/'||contacts._id||
'/photo' END) AS photo_thumb_uri,
0 AS is_user_profile
FROM contacts JOIN raw_contacts AS name_raw_contact
ON(name_raw_contact_id=name_raw_contact._id)
```



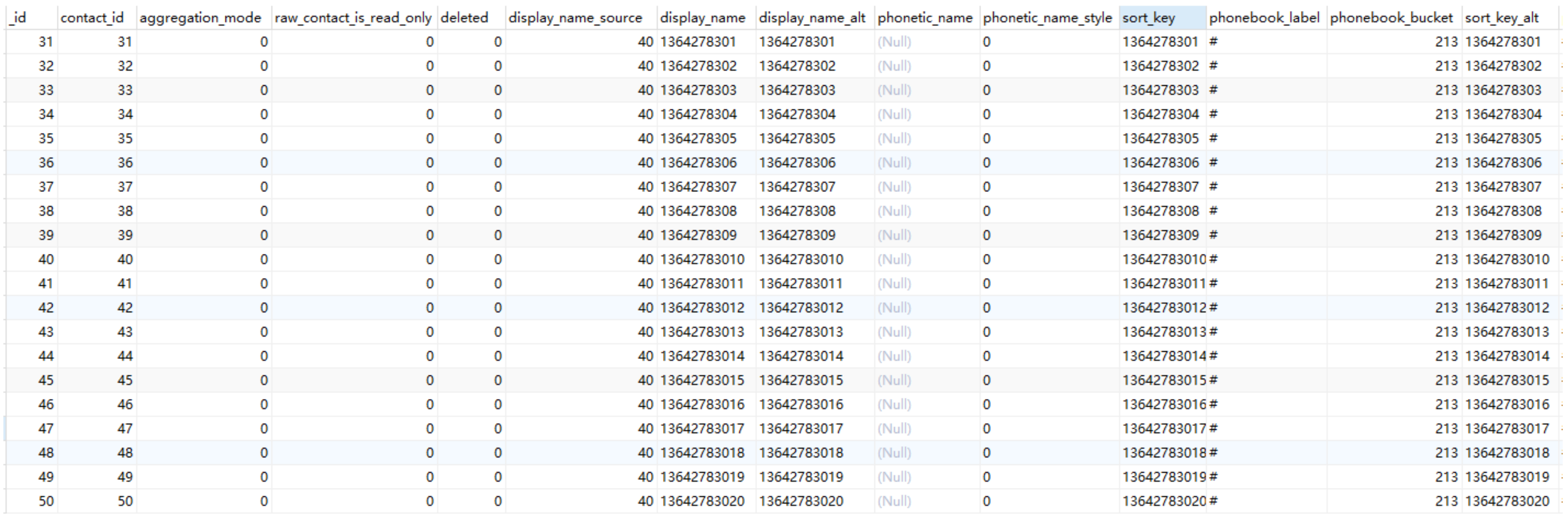
_id	custom_ringtone	display_name_source	display_name	display_name_alt	phonetic_name	phonetic_name_style	sort_key	phonebook_label	phonebook_bucket	sort_key_alt	phonebook_label_alt	phonebook_bucket_alt	has_phone_number
31	(Null)	40	1364278301	1364278301	(Null)	0	1364278301	#	213	1364278301	#	213	1
32	(Null)	40	1364278302	1364278302	(Null)	0	1364278302	#	213	1364278302	#	213	1
33	(Null)	40	1364278303	1364278303	(Null)	0	1364278303	#	213	1364278303	#	213	1
34	(Null)	40	1364278304	1364278304	(Null)	0	1364278304	#	213	1364278304	#	213	1
35	(Null)	40	1364278305	1364278305	(Null)	0	1364278305	#	213	1364278305	#	213	1
36	(Null)	40	1364278306	1364278306	(Null)	0	1364278306	#	213	1364278306	#	213	1
37	(Null)	40	1364278307	1364278307	(Null)	0	1364278307	#	213	1364278307	#	213	1
38	(Null)	40	1364278308	1364278308	(Null)	0	1364278308	#	213	1364278308	#	213	1
39	(Null)	40	1364278309	1364278309	(Null)	0	1364278309	#	213	1364278309	#	213	1
40	(Null)	40	13642783010	13642783010	(Null)	0	13642783010	#	213	13642783010	#	213	1
41	(Null)	40	13642783011	13642783011	(Null)	0	13642783011	#	213	13642783011	#	213	1
42	(Null)	40	13642783012	13642783012	(Null)	0	13642783012	#	213	13642783012	#	213	1
43	(Null)	40	13642783013	13642783013	(Null)	0	13642783013	#	213	13642783013	#	213	1
44	(Null)	40	13642783014	13642783014	(Null)	0	13642783014	#	213	13642783014	#	213	1



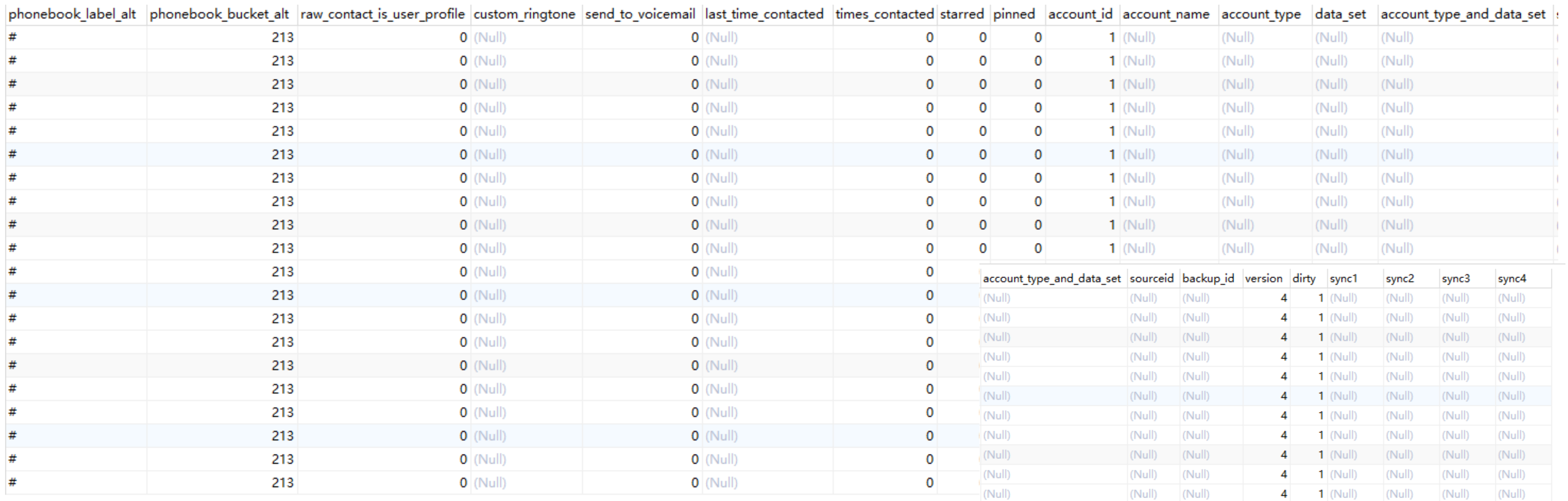
_id	name_raw_contact_id	lookup	photo_id	photo_file_id	in_visible_group	in_default_directory	status_update_id	contact_last_updated_time	last_time_contacted	send_to_voicemail	starred	pinned	times_contacted	photo_uri	photo_thumb_uri	is_user_profile
31	31	0r31-14181	(Null)	(Null)	1	1	(Null)	(Null)	ll) 0	0	0	0	0	(Null)	(Null)	0
32	32	0r32-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
33	33	0r33-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
34	34	0r34-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
35	35	0r35-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
36	36	0r36-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
37	37	0r37-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
38	38	0r38-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
39	39	0r39-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
40	40	0r40-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
41	41	0r41-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
42	42	0r42-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
43	43	0r43-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0
44	44	0r44-14181	(Null)	(Null)	1	1	(Null)	(Null) 1	ll) 0	0	0	0	0	(Null)	(Null)	0

• view_raw_contacts

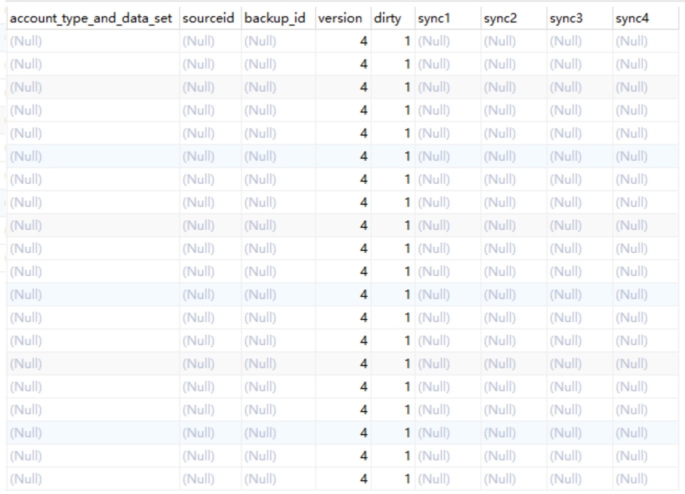
SELECT raw_contacts._id AS _id,contact_id, aggregation_mode, raw_contact_is_read_only, deleted, display_name_source, display_name, display_name_alt, phonetic_name, phonetic_name_style, sort_key, phonebook_label, phonebook_bucket, sort_key_alt, phonebook_label_alt, phonebook_bucket_alt, 0 AS raw_contact_is_user_profile,
custom_ringtone,send_to_voicemail,last_time_contacted,times_contacted,starred,pinned, raw_contacts.account_id,accounts.account_name AS account_name,accounts.account_type AS account_type,accounts.data_set AS data_set,(CASE WHEN accounts.data_set IS NULL THEN accounts.account_type ELSE accounts.account_type||'/'||accounts.data_set END) AS account_type_and_data_set,raw_contacts.sourceid AS sourceid,raw_contacts.backup_id AS backup_id,raw_contacts.version AS version,raw_contacts.dirty AS dirty,raw_contacts.sync1 AS sync1,raw_contacts.sync2 AS sync2,raw_contacts.sync3 AS sync3,raw_contacts.sync4 AS sync4
FROM raw_contacts **JOIN** accounts **ON** (raw_contacts.account_id=accounts._id)



_id	contact_id	aggregation_mode	raw_contact_is_read_only	deleted	display_name_source	display_name	display_name_alt	phonetic_name	phonetic_name_style	sort_key	phonebook_label	phonebook_bucket	sort_key_alt
31	31	0	0	0	40	1364278301	1364278301	(Null)	0	1364278301	#	213	1364278301
32	32	0	0	0	40	1364278302	1364278302	(Null)	0	1364278302	#	213	1364278302
33	33	0	0	0	40	1364278303	1364278303	(Null)	0	1364278303	#	213	1364278303
34	34	0	0	0	40	1364278304	1364278304	(Null)	0	1364278304	#	213	1364278304
35	35	0	0	0	40	1364278305	1364278305	(Null)	0	1364278305	#	213	1364278305
36	36	0	0	0	40	1364278306	1364278306	(Null)	0	1364278306	#	213	1364278306
37	37	0	0	0	40	1364278307	1364278307	(Null)	0	1364278307	#	213	1364278307
38	38	0	0	0	40	1364278308	1364278308	(Null)	0	1364278308	#	213	1364278308
39	39	0	0	0	40	1364278309	1364278309	(Null)	0	1364278309	#	213	1364278309
40	40	0	0	0	40	13642783010	13642783010	(Null)	0	13642783010	#	213	13642783010
41	41	0	0	0	40	13642783011	13642783011	(Null)	0	13642783011	#	213	13642783011
42	42	0	0	0	40	13642783012	13642783012	(Null)	0	13642783012	#	213	13642783012
43	43	0	0	0	40	13642783013	13642783013	(Null)	0	13642783013	#	213	13642783013
44	44	0	0	0	40	13642783014	13642783014	(Null)	0	13642783014	#	213	13642783014
45	45	0	0	0	40	13642783015	13642783015	(Null)	0	13642783015	#	213	13642783015
46	46	0	0	0	40	13642783016	13642783016	(Null)	0	13642783016	#	213	13642783016
47	47	0	0	0	40	13642783017	13642783017	(Null)	0	13642783017	#	213	13642783017
48	48	0	0	0	40	13642783018	13642783018	(Null)	0	13642783018	#	213	13642783018
49	49	0	0	0	40	13642783019	13642783019	(Null)	0	13642783019	#	213	13642783019
50	50	0	0	0	40	13642783020	13642783020	(Null)	0	13642783020	#	213	13642783020



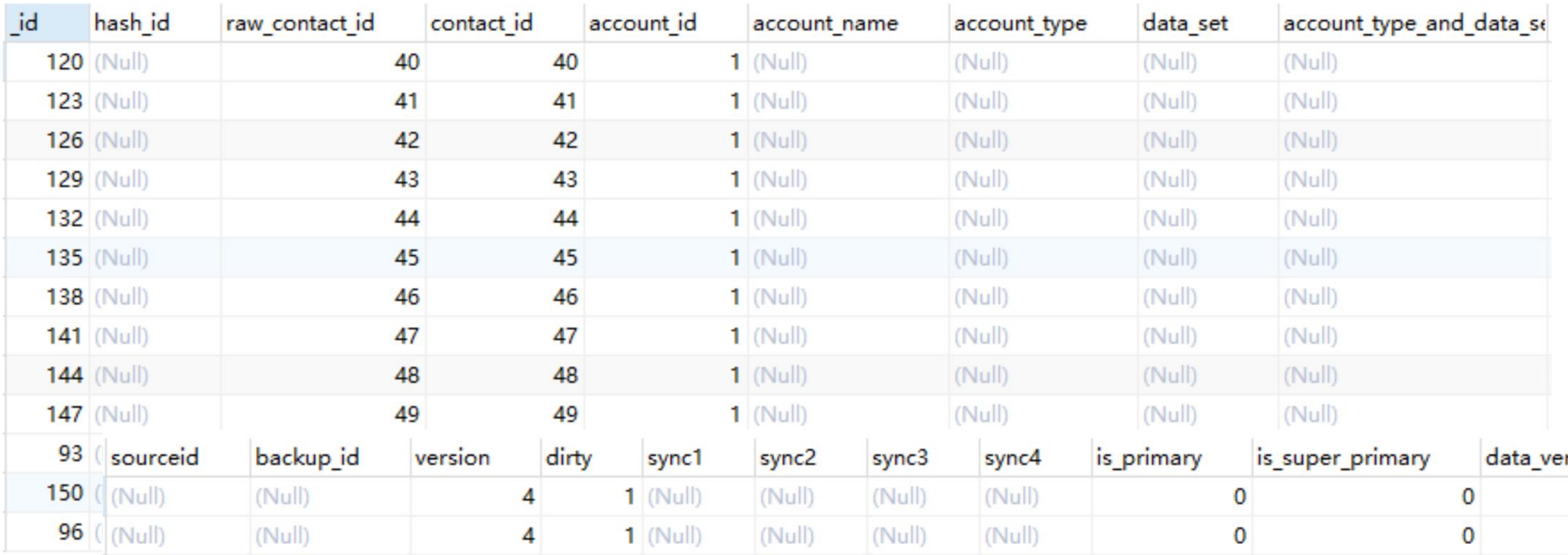
phonebook_label_alt	phonebook_bucket_alt	raw_contact_is_user_profile	custom_ringtone	send_to_voicemail	last_time_contacted	times_contacted	starred	pinned	account_id	account_name	account_type	data_set	account_type_and_data_set
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0	0	0	1	(Null)	(Null)	(Null)	(Null)
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							
#	213	0	(Null)	0	(Null)	0							



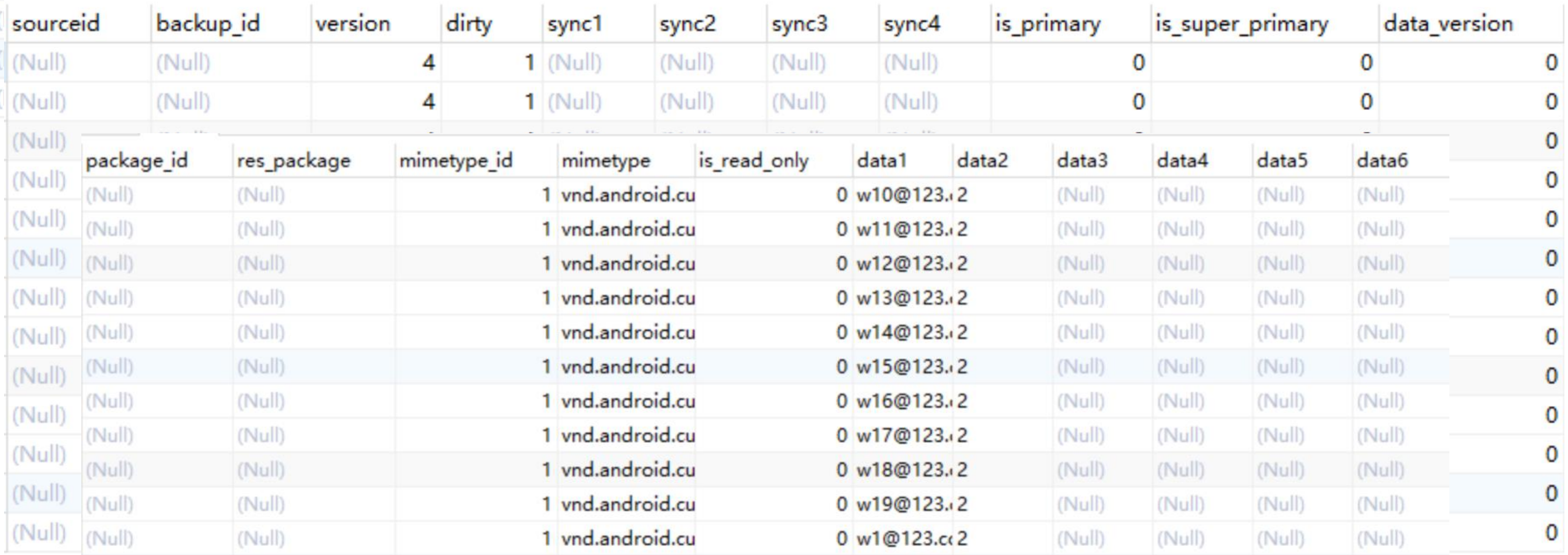
account_type_and_data_set	sourceid	backup_id	version	dirty	sync1	sync2	sync3	sync4
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)

- view_data

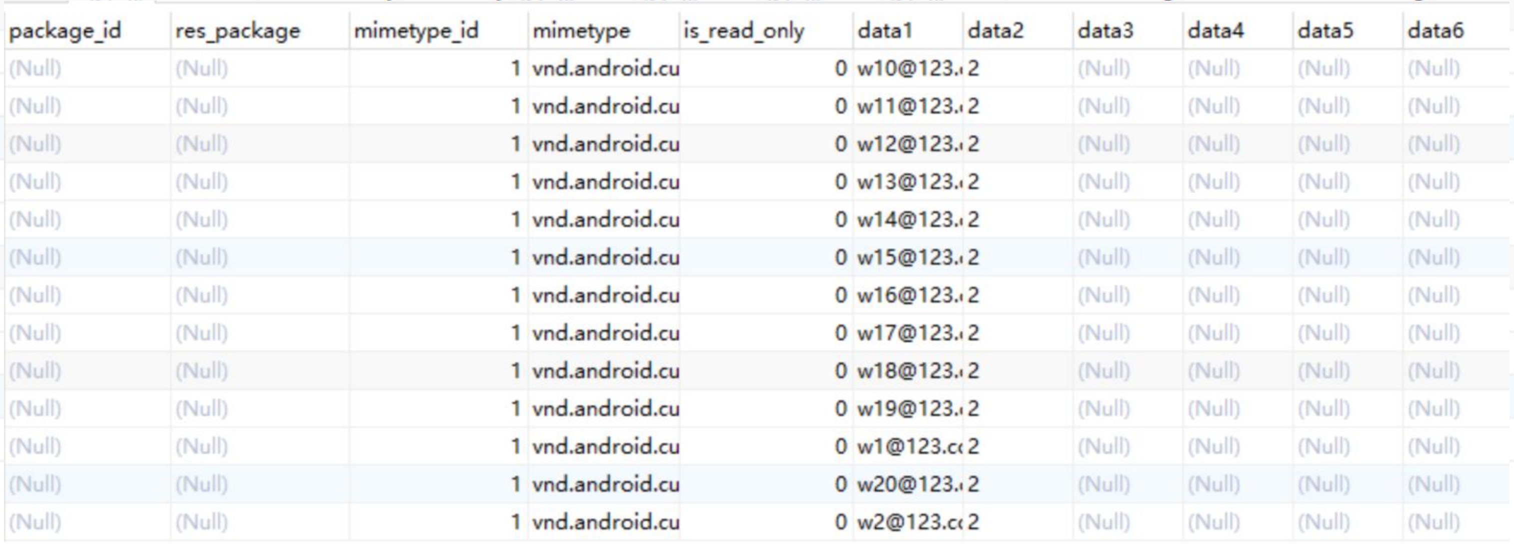
```
SELECT data._id AS _id,hash_id, raw_contact_id, raw_contacts.contact_id AS contact_id,
raw_contacts.account_id,accounts.account_name AS account_name,accounts.account_type AS account_type,accounts.data_set
AS data_set,(CASE WHEN accounts.data_set IS NULL THEN accounts.account_type ELSE
accounts.account_type||'/'||accounts.data_set END) AS account_type_and_data_set,raw_contacts.sourceid AS
sourceid,raw_contacts.backup_id AS backup_id,raw_contacts.version AS version,raw_contacts.dirty AS dirty,raw_contacts.sync1 AS
sync1,raw_contacts.sync2 AS sync2,raw_contacts.sync3 AS sync3,raw_contacts.sync4 AS sync4, is_primary, is_super_primary,
data_version, data.package_id,package AS res_package,data.mimetype_id,mimetype AS mimetype, is_read_only, data1, data2,
data3, data4, data5, data6, data7, data8, data9, data10, data11, data12, data13, data14, data15, carrier_presence, data_sync1,
data_sync2, data_sync3, data_sync4, contacts.custom_ringtone AS custom_ringtone,contacts.send_to_voicemail AS
send_to_voicemail,contacts.last_time_contacted AS last_time_contacted,contacts.times_contacted AS
times_contacted,contacts.starred AS starred,contacts.pinned AS pinned, name_raw_contact.display_name_source AS
display_name_source, name_raw_contact.display_name AS display_name, name_raw_contact.display_name_alt AS
display_name_alt, name_raw_contact.phonetic_name AS phonetic_name, name_raw_contact.phonetic_name_style AS
phonetic_name_style, name_raw_contact.sort_key AS sort_key, name_raw_contact.phonebook_label AS phonebook_label,
name_raw_contact.phonebook_bucket AS phonebook_bucket, name_raw_contact.sort_key_alt AS sort_key_alt,
name_raw_contact.phonebook_label_alt AS phonebook_label_alt, name_raw_contact.phonebook_bucket_alt AS
phonebook_bucket_alt, has_phone_number, name_raw_contact_id, lookup, photo_id, photo_file_id, CAST(EXISTS (SELECT _id
FROM visible_contacts WHERE contacts._id=visible_contacts._id) AS INTEGER) AS in_visible_group, CAST(EXISTS (SELECT _id FROM
default_directory WHERE contacts._id=default_directory._id) AS INTEGER) AS in_default_directory, status_update_id,
contacts.contact_last_updated_timestamp, (CASE WHEN photo_file_id IS NULL THEN (CASE WHEN photo_id IS NULL OR
photo_id=0 THEN NULL ELSE 'content://com.android.contacts/contacts/'||raw_contacts.contact_id|| '/photo' END) ELSE
'content://com.android.contacts/display_photo/'||photo_file_id END) AS photo_uri, (CASE WHEN photo_id IS NULL OR
photo_id=0 THEN NULL ELSE 'content://com.android.contacts/contacts/'||raw_contacts.contact_id|| '/photo' END) AS
photo_thumb_uri, 0 AS raw_contact_is_user_profile, groups.sourceid AS group_sourceid
 FROM data JOIN mimetypes ON (data.mimetype_id=mimetypes._id) JOIN raw_contacts ON
(data.raw_contact_id=raw_contacts._id) JOIN accounts ON (raw_contacts.account_id=accounts._id) JOIN contacts ON
(raw_contacts.contact_id=contacts._id) JOIN raw_contacts AS name_raw_contact
ON(name_raw_contact_id=name_raw_contact._id) LEFT OUTER JOIN packages ON (data.package_id=packages._id) LEFT OUTER
JOIN groups ON (mimetypes.mimetype='vnd.android.cursor.item/group_membership' AND groups._id=data.data1)
```



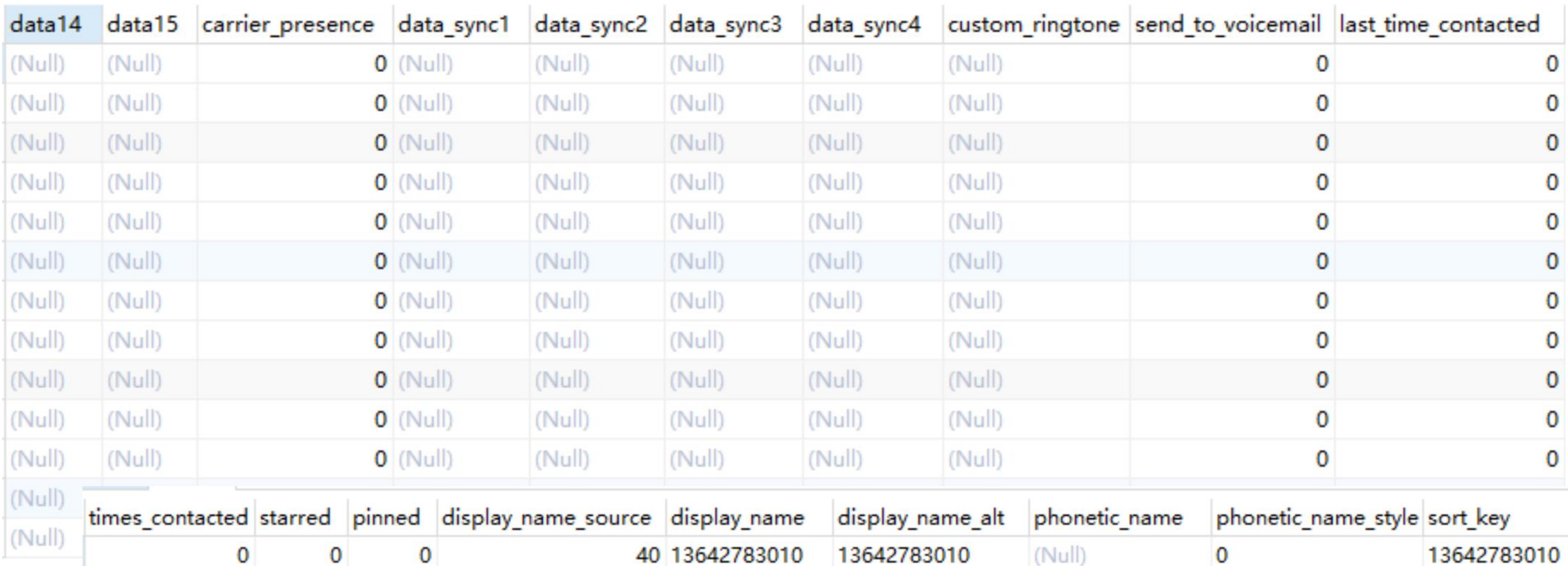
_id	hash_id	raw_contact_id	contact_id	account_id	account_name	account_type	data_set	account_type_and_data_se
120	(Null)	40	40	1	(Null)	(Null)	(Null)	(Null)
123	(Null)	41	41	1	(Null)	(Null)	(Null)	(Null)
126	(Null)	42	42	1	(Null)	(Null)	(Null)	(Null)
129	(Null)	43	43	1	(Null)	(Null)	(Null)	(Null)
132	(Null)	44	44	1	(Null)	(Null)	(Null)	(Null)
135	(Null)	45	45	1	(Null)	(Null)	(Null)	(Null)
138	(Null)	46	46	1	(Null)	(Null)	(Null)	(Null)
141	(Null)	47	47	1	(Null)	(Null)	(Null)	(Null)
144	(Null)	48	48	1	(Null)	(Null)	(Null)	(Null)
147	(Null)	49	49	1	(Null)	(Null)	(Null)	(Null)
93	(							
150	(							
96	(							



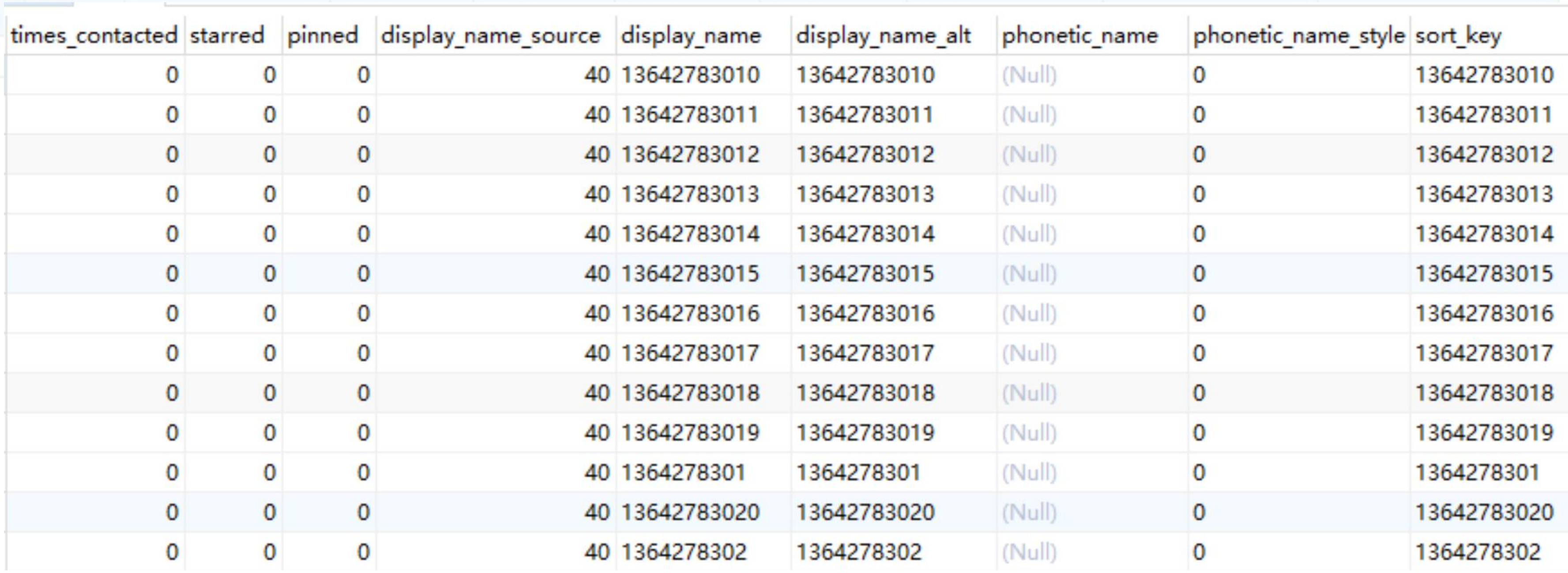
sourceid	backup_id	version	dirty	sync1	sync2	sync3	sync4	is_primary	is_super_primary	data_version
(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)	0	0	0
(Null)	(Null)	4	1	(Null)	(Null)	(Null)	(Null)	0	0	0
(Null)										0
(Null)										0
(Null)										0
(Null)										0
(Null)										0
(Null)										0
(Null)										0
(Null)										0
(Null)										0
(Null)										0
(Null)										0



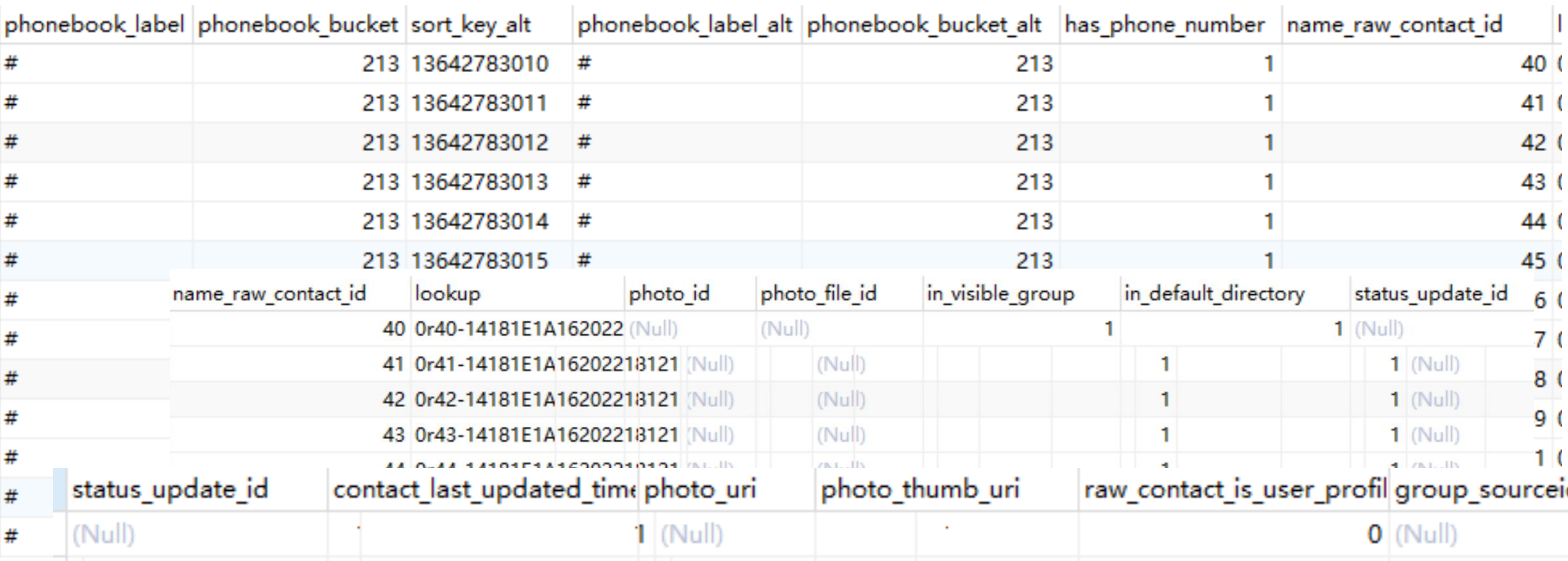
package_id	res_package	mimetype_id	mimetype	is_read_only	data1	data2	data3	data4	data5	data6
(Null)	(Null)	1	vnd.android.cu	0	w10@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w11@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w12@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w13@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w14@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w15@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w16@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w17@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w18@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w19@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w1@123.c	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w20@123.	2	(Null)	(Null)	(Null)	(Null)
(Null)	(Null)	1	vnd.android.cu	0	w2@123.c	2	(Null)	(Null)	(Null)	(Null)



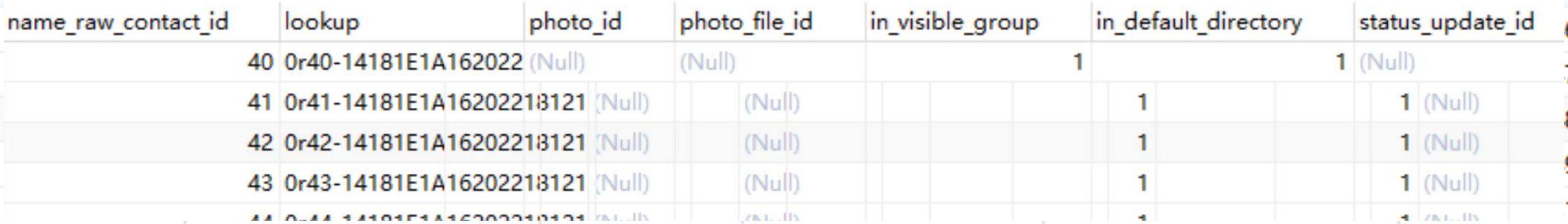
data14	data15	carrier_presence	data_sync1	data_sync2	data_sync3	data_sync4	custom_ringtone	send_to_voicemail	last_time_contacted
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)	(Null)	0	(Null)	(Null)	(Null)	(Null)	(Null)	0	0
(Null)									
(Null)									



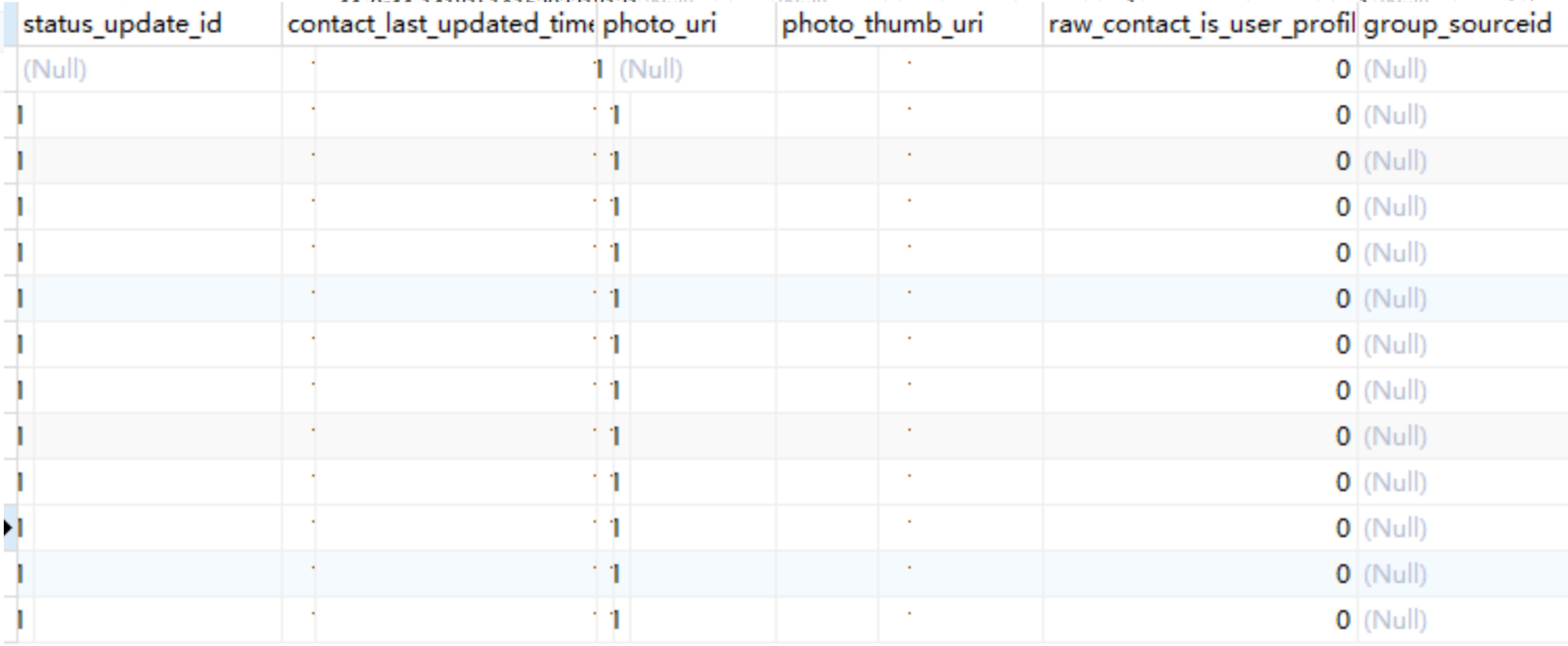
times_contacted	starred	pinned	display_name_source	display_name	display_name_alt	phonetic_name	phonetic_name_style	sort_key
0	0	0	40	13642783010	13642783010	(Null)	0	13642783010
0	0	0	40	13642783011	13642783011	(Null)	0	13642783011
0	0	0	40	13642783012	13642783012	(Null)	0	13642783012
0	0	0	40	13642783013	13642783013	(Null)	0	13642783013
0	0	0	40	13642783014	13642783014	(Null)	0	13642783014
0	0	0	40	13642783015	13642783015	(Null)	0	13642783015
0	0	0	40	13642783016	13642783016	(Null)	0	13642783016
0	0	0	40	13642783017	13642783017	(Null)	0	13642783017
0	0	0	40	13642783018	13642783018	(Null)	0	13642783018
0	0	0	40	13642783019	13642783019	(Null)	0	13642783019
0	0	0	40	1364278301	1364278301	(Null)	0	1364278301
0	0	0	40	13642783020	13642783020	(Null)	0	13642783020
0	0	0	40	1364278302	1364278302	(Null)	0	1364278302



phonebook_label	phonebook_bucket	sort_key_alt	phonebook_label_alt	phonebook_bucket_alt	has_phone_number	name_raw_contact_id
#	213	13642783010	#	213	1	40
#	213	13642783011	#	213	1	41
#	213	13642783012	#	213	1	42
#	213	13642783013	#	213	1	43
#	213	13642783014	#	213	1	44
#	213	13642783015	#	213	1	45
#						6
#						7
#						8
#						9
#						1
#						
#						



name_raw_contact_id	lookup	photo_id	photo_file_id	in_visible_group	in_default_directory	status_update_id
40	0r40-14181E1A162022	(Null)	(Null)	1	1	(Null)
41	0r41-14181E1A16202218121	(Null)	(Null)	1	1	(Null)
42	0r42-14181E1A16202218121	(Null)	(Null)	1	1	(Null)
43	0r43-14181E1A16202218121	(Null)	(Null)	1	1	(Null)



status_update_id	contact_last_updated_time	photo_uri	photo_thumb_uri	raw_contact_is_user_profil	group_sourceid
(Null)	1	(Null)		0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)
1	1			0	(Null)

```
//查询所有联系人的姓名，电话，邮箱
public void queryAllContacts() throws Exception {
   Uri uri = Uri.parse("content://com.android.contacts/contacts");
   ContentResolver resolver = getContentResolver();
   Cursor cursor = resolver.query(uri, new String[]{"_id"}, null, null, null);
   while (cursor.moveToNext()) {
     int contractID = cursor.getInt(0);
     StringBuilder sb = new StringBuilder("contractID=");
     sb.append(contractID);
     uri=Uri.parse("content://com.android.contacts/contacts/"+contractID+"/data");
     Cursor cursor1 = resolver.query(uri, new String[]{"mimetype", "data1",
                             "data2"}, null, null, null);
     while (cursor1.moveToNext()) {
        String data1 = cursor1.getString(cursor1.getColumnIndex("data1"));
        String mimeType = cursor1.getString(cursor1.getColumnIndex("mimetype"));
        if ("vnd.android.cursor.item/name".equals(mimeType)){ //姓名
          sb.append(",name=" + data1);
        } else if ("vnd.android.cursor.item/email_v2".equals(mimeType)){//邮件
          sb.append(",email=" + data1);
        } else if ("vnd.android.cursor.item/phone_v2".equals(mimeType)){//手机号
          sb.append(",phone=" + data1);
        }
     }
     cursor1.close();
     Log.i(TAG, sb.toString());
   }
   cursor.close();
}
```

视图：view_data_restricted

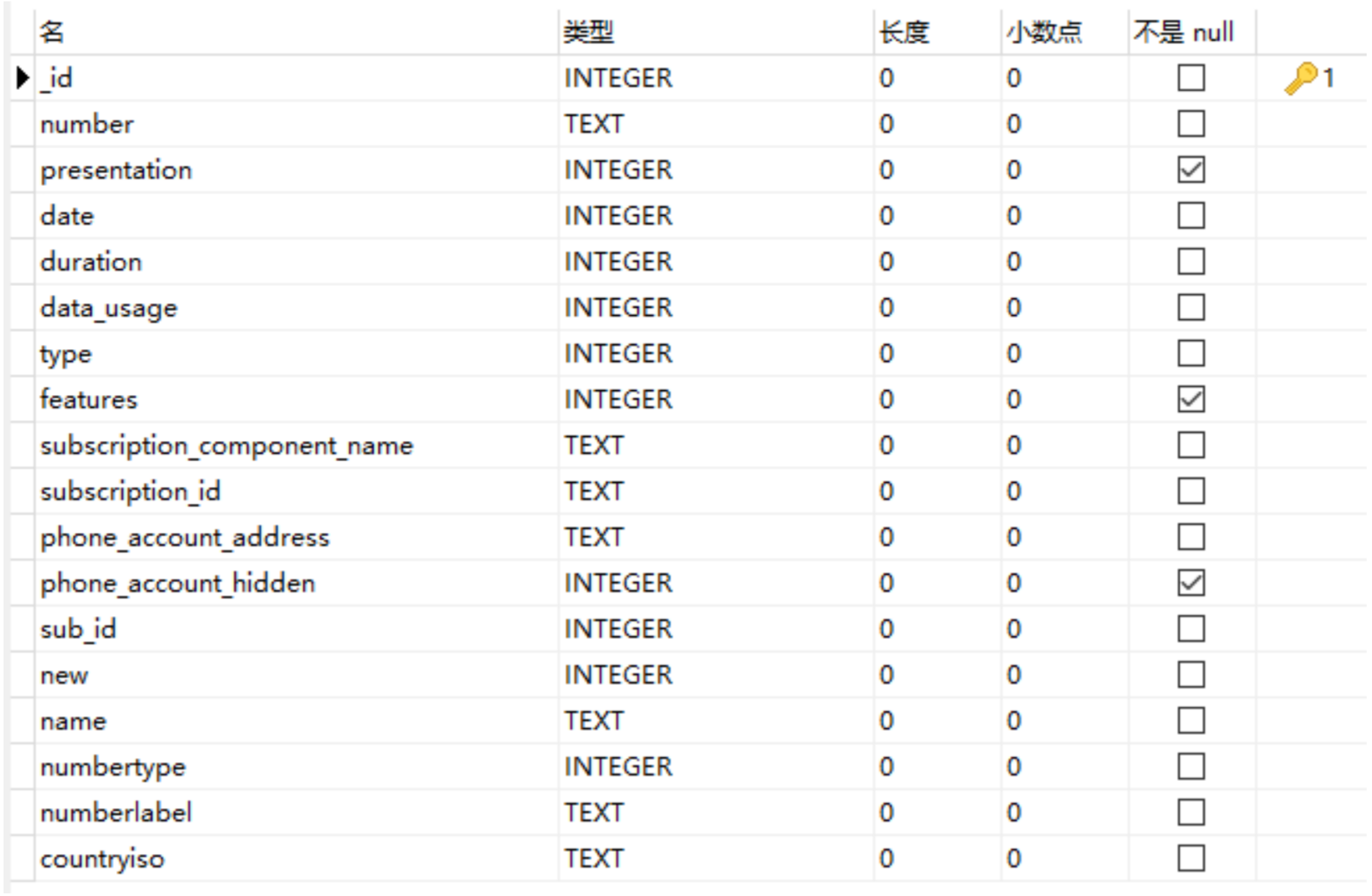
```java
//查询指定电话的联系人姓名，邮箱
public void queryContactNameByNumber(String number) throws Exception {
    Uri uri = Uri.parse("content://com.android.contacts/data/phones/filter/" + number);
    ContentResolver resolver = getContentResolver();
    Cursor cursor = resolver.query(uri, new String[]{"display_name"}, null, null, null);
    if (cursor.moveToFirst()) {
        String name = cursor.getString(0);
        Log.i(TAG, name);
    }
    cursor.close();
}
```

```java
contentResolver.query(android.provider.ContactsContract.CommonDataKinds.
Phone.CONTENT_URI,   null, null, null, null);
```

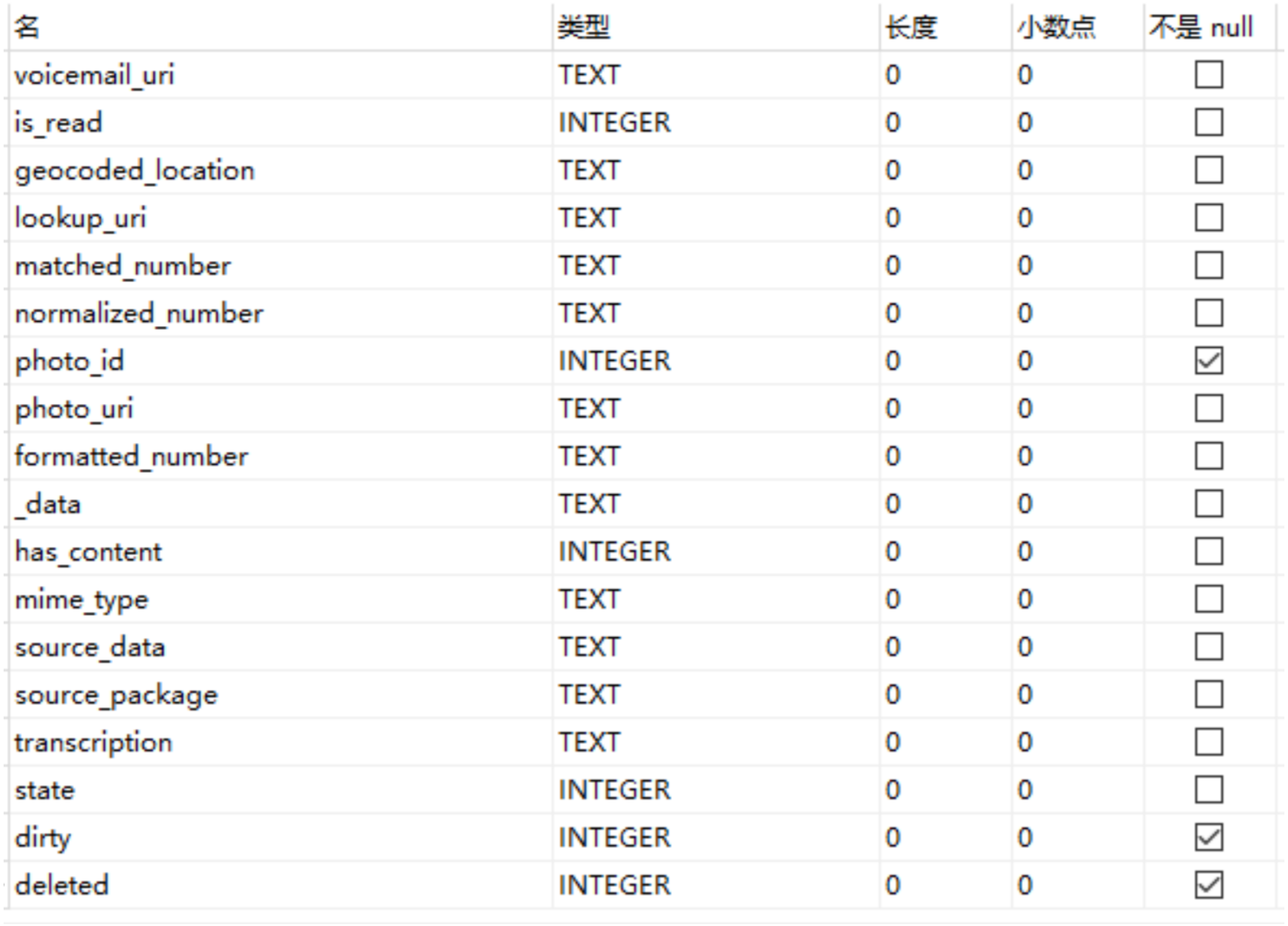
```sql
SELECT data_version, contact_id, lookup, data12, data11, data10, mimetype, data15, data14, data13, data_sync
data_sync3, data_sync2, data_sync4, account_type, custom_ringtone, status_updates.status AS status, data1, d
ata4, data5, data2, data3, data8, data9, group_sourceid, data6, account_name, data7, display_name, in_visible_
group, contacts_status_updates.status_res_package AS contact_status_res_package, is_primary, contacts_status
_updates.status_ts AS contact_status_ts, raw_contact_id, times_contacted, contacts_status_updates.status AS c
ontact_status, status_updates.status_res_package AS status_res_package, status_updates.status_icon AS status
_icon, contacts_status_updates.status_icon AS contact_status_icon, presence.mode AS mode, version, last_time
_contacted, res_package, _id, status_updates.status_ts AS status_ts, dirty, is_super_primary, photo_id, send_to_
voicemail, contacts_status_updates.status_label AS contact_status_label, status_updates.status_label AS status_
label, starred, agg_presence.mode AS contact_presence, sourceid FROM view_data_restricted data LEFT OUTE
R JOIN agg_presence ON (agg_presence.presence_contact_id=contact_id) LEFT OUTER JOIN status_updates co
ntacts_status_updates ON (status_update_id=contacts_status_updates.status_update_data_id) LEFT OUTER JOI
N presence ON (presence_data_id=data._id) LEFT OUTER JOIN status_updates ON (status_updates.status_upd
ate_data_id=data._id) WHERE (1 AND mimetype = 'vnd.android.cursor.item/phone_v2')
```

附录7、呼叫记录

Sqlite数据库contact2.db的表calls

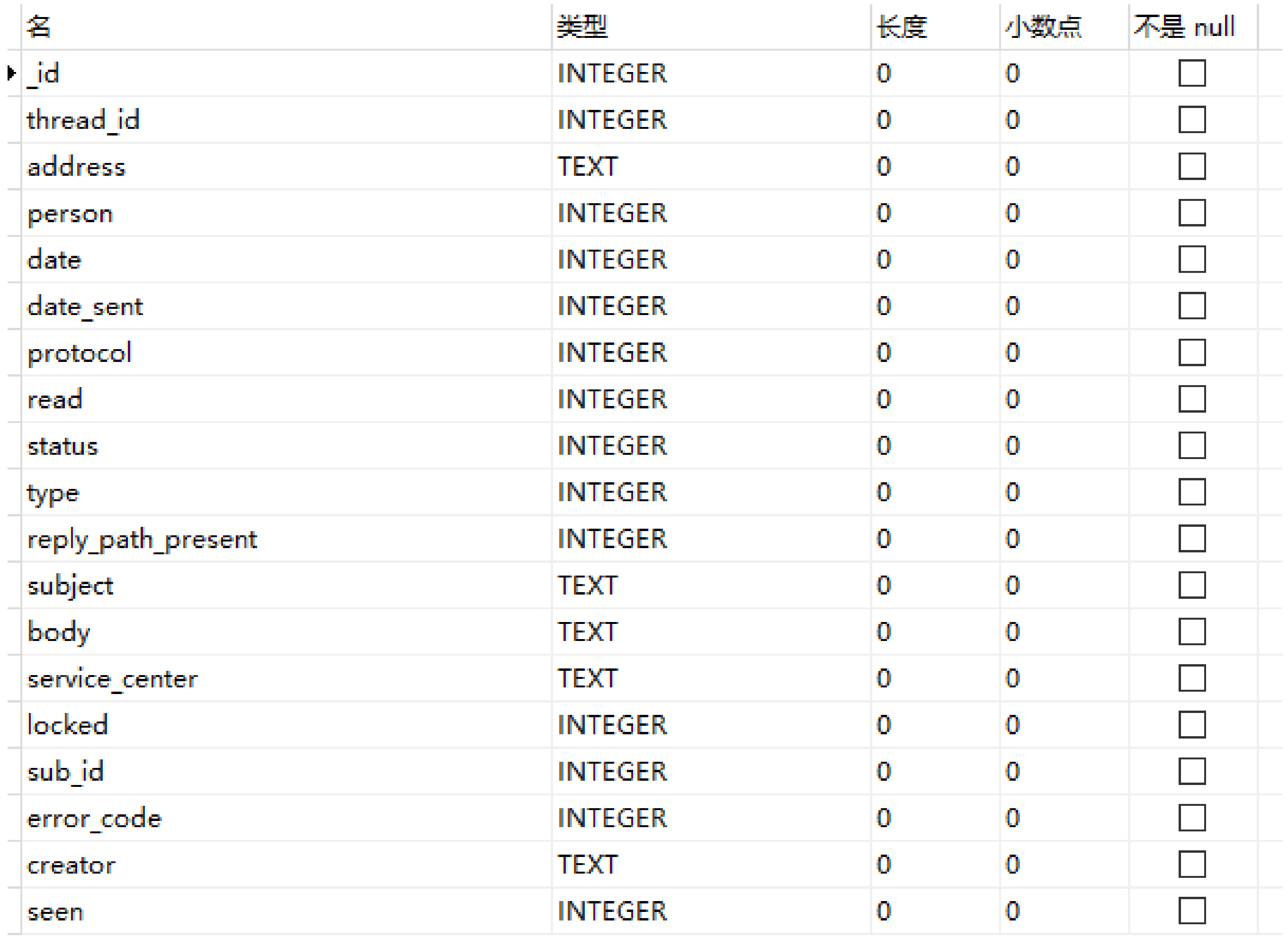


名	类型	长度	小数点	不是 null	
_id	INTEGER	0	0	☐	🔑1
number	TEXT	0	0	☐	
presentation	INTEGER	0	0	☑	
date	INTEGER	0	0	☐	
duration	INTEGER	0	0	☐	
data_usage	INTEGER	0	0	☐	
type	INTEGER	0	0	☐	
features	INTEGER	0	0	☑	
subscription_component_name	TEXT	0	0	☐	
subscription_id	TEXT	0	0	☐	
phone_account_address	TEXT	0	0	☐	
phone_account_hidden	INTEGER	0	0	☑	
sub_id	INTEGER	0	0	☐	
new	INTEGER	0	0	☐	
name	TEXT	0	0	☐	
numbertype	INTEGER	0	0	☐	
numberlabel	TEXT	0	0	☐	
countryiso	TEXT	0	0	☐	



名	类型	长度	小数点	不是 null
voicemail_uri	TEXT	0	0	☐
is_read	INTEGER	0	0	☐
geocoded_location	TEXT	0	0	☐
lookup_uri	TEXT	0	0	☐
matched_number	TEXT	0	0	☐
normalized_number	TEXT	0	0	☐
photo_id	INTEGER	0	0	☑
photo_uri	TEXT	0	0	☐
formatted_number	TEXT	0	0	☐
_data	TEXT	0	0	☐
has_content	INTEGER	0	0	☐
mime_type	TEXT	0	0	☐
source_data	TEXT	0	0	☐
source_package	TEXT	0	0	☐
transcription	TEXT	0	0	☐
state	INTEGER	0	0	☐
dirty	INTEGER	0	0	☑
deleted	INTEGER	0	0	☑

Sqlite数据库mmssms.db的表sms



名	类型	长度	小数点	不是 null
_id	INTEGER	0	0	☐
thread_id	INTEGER	0	0	☐
address	TEXT	0	0	☐
person	INTEGER	0	0	☐
date	INTEGER	0	0	☐
date_sent	INTEGER	0	0	☐
protocol	INTEGER	0	0	☐
read	INTEGER	0	0	☐
status	INTEGER	0	0	☐
type	INTEGER	0	0	☐
reply_path_present	INTEGER	0	0	☐
subject	TEXT	0	0	☐
body	TEXT	0	0	☐
service_center	TEXT	0	0	☐
locked	INTEGER	0	0	☐
sub_id	INTEGER	0	0	☐
error_code	INTEGER	0	0	☐
creator	TEXT	0	0	☐
seen	INTEGER	0	0	☐