

Android数据库高手秘籍(1)：SQLite命令

2015-09-05 安卓应用频道

(点击上方蓝字，可快速关注我们)

来源：郭霖

网址：http://blog.csdn.net/guolin_blog/article/details/38461239

要想熟练地操作任何一个数据库，最最基本的要求就是要懂SQL语言，这也是每个程序员都应该掌握的技能。虽说SQL博大精深，要想精通确实很难，但最基本的一些建表命令，增删改查，大家还是必须要学会的。

SQL(Structured Query Language)是一种标准的数据库查询语言，即所有的关系型数据库都会支持它，只不过每种数据库对SQL语言的支持与标准存在着细微的不同。我们无须关心其它数据库对SQL语言的支持情况，这里我们只要把重点放在SQLite上就可以了。下面我将使用模拟器来对SQLite支持的各种命令进行演示，如果你想用手机的话也可以，但要确保你的手机已经Root，并且包含sqlite3这个命令文件。

首先确保模拟器已经连接上了电脑，然后在命令行输入adb shell进入控制台，如下图所示：

```
C:\Users\Tony>adb shell
#
```

注意#符号表示我们当前已经是超级用户了，如果显示的是\$符号，表示当前只是普通用户而已，这时还需要输入su命令切换一下用户身份才行。

有了超级用户权限之后，我们能做的事情就很多了，这里我们先查看一下系统自带的联系人表吧。进入到/data/data目录下，如下图所示：

```
# cd data/data
cd data/data
#
```

所有应用程序的本地存储文件都是存放在这个目录下面的。为了要让不同应用程序之间的数据容易区别开来，Android是使用应用程序包名进行分开管理，也就是说每个应用程序的本地存储文件都会存放在自己应用程序包名的那个目录下，这里我们ls一下看看有多少子目录：

```
# ls
ls
com.android.backupconfirm
com.android.browser
com.android.calculator2
com.android.calendar
com.android.camera
com.android.certinstaller
com.android.contacts
com.android.customlocale2
com.android.defcontainer
com.android.deskclock
com.android.development
com.android.email
com.android.emulator.connectivity.test
com.android.emulator.gps.test
com.android.exchange
com.android.fallback
com.android.gallery
com.android.gesture.builder
com.android.htmlviewer
com.android.inputmethod.latin
com.android.inputmethod.pinyin
com.android.keychain
com.android.launcher
com.android.mms
com.android.music
com.android.netspeed
com.android.packageinstaller
com.android.phone
com.android.protips
com.android.providers.applications
com.android.providers.calendar
com.android.providers.contacts
```

OK，确实有很多，毕竟手机上所有的应用程序都在这里。其中，com.android.providers.contacts中存放的就是联系人的相关数据，我们进入到这个目录再ls一下：

```
# cd com.android.providers.contacts
cd com.android.providers.contacts
# ls
ls
databases
files
lib
shared_prefs
#
```

可以看到，目前有databases、files、lib和shared_prefs这几个子目录。其中databases肯定是用于存放数据库文件的，files是用于存放普通文本文件的，lib是用于存放so库的，shared_prefs则是用于存放shared文件的。这是Android数据持久化的几种可选方式，对这部分内容不太了解的朋友可以参考《第一行代码——Android》的第六章。

接着进入到databases目录中，再ls：

```
# cd databases
cd databases
# ls
ls
contacts2.db
contacts2.db-journal
profile.db
profile.db-journal
```

其中后缀名为journal的文件是日志文件，我们不用管，contacts2.db和profile.db才是真正的数据库文件，可以使用sqlite3命令来打开数据库，如下图所示：

```
# sqlite3 contacts2.db
sqlite3 contacts2.db
SQLite version 3.7.4
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

好的，数据库已经打开了，那么我们怎么才能知道当前数据库中有哪些表呢？很简单，.table命令就可以做到了：

```
sqlite> .table
.table
_sync_state          phone_lookup         view_data_usage_stat
_sync_state_metadata photo_files          view_entities
accounts             properties           view_groups
activities            raw_contacts         view_raw_contacts
agg_exceptions        search_index         view_raw_entities
android_metadata     search_index_content view_stream_items
calls                 search_index_docsize view_v1_contact_methods
contacts              search_index_segdir  view_v1_extensions
data                  search_index_segments view_v1_group_membership
data_usage_stat       search_index_stat    view_v1_groups
default_directory     settings             view_v1_organizations
directories            status_updates       view_v1_people
groups                stream_item_photos   view_v1_phones
mimetypes             stream_items         view_v1_photos
name_lookup           v1_settings          visible_contacts
nickname_lookup       view_contacts        voicemail_status
packages              view_data
sqlite>
```

哇，竟然有这么多张表！是的，联系人的数据结构非常复杂，很多的数据都是分表存储的。这里我们随便挑一张表，比如说accounts表，如果我想知道这张表中有哪些列应该怎么办呢？在MySQL中可以使用desc accounts这个命令，但SQLite却不认识这个命令，毕竟它们是有差异化的。SQLite中可以使用pragma table_info(TABLE_NAME)这个命令来查看表的数据结构，如下图所示：

```
sqlite> pragma table_info(accounts);
pragma table_info(accounts);
0|account_name|TEXT|0||0
1|account_type|TEXT|0||0
2|data_set|TEXT|0||0
sqlite> _
```

可以看到，一共显示了三条结果，表示accounts表中共有三列。但是，所有的字段都缩在了了一行里面，并用“|”符号分隔，这样我们很难看出每个字段的含义。很简单，只需要换一种显示模式就行了，比如说line模式就挺不错的。输入.mode line命令切换显示模式，然后重新运行pragma命令，结果如下图所示：

```
sqlite> .mode line
.mode line
sqlite> pragma table_info(accounts);
pragma table_info(accounts);
      cid = 0
      name = account_name
      type = TEXT
  notnull = 0
dflt_value =
      pk = 0

      cid = 1
      name = account_type
      type = TEXT
  notnull = 0
dflt_value =
      pk = 0

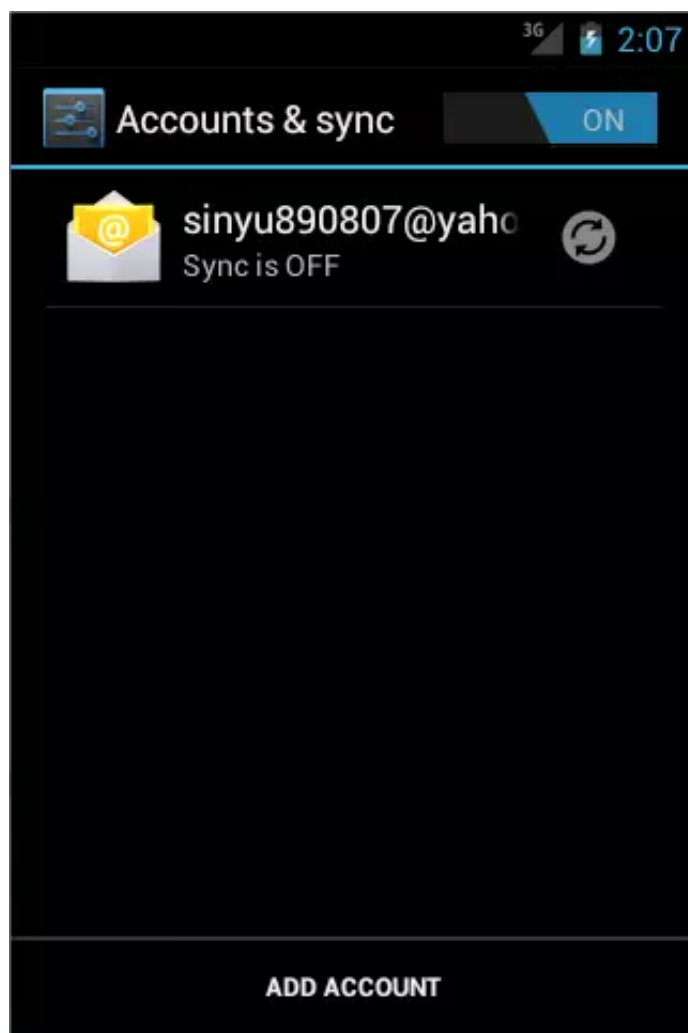
      cid = 2
      name = data_set
      type = TEXT
  notnull = 0
dflt_value =
      pk = 0
sqlite>
```

怎么样，这样就清晰多了吧？这三列的列名分别是account_name、account_type和data_set，数据类型都是TEXT(字符串)，允许为空，并且都不是主键。好，那我现在想查一查accounts表中的数据呢？这就太简单了，使用select语句就可以了，如下所示：

```
sqlite> select * from accounts;
select * from accounts;
account_name =
account_type =
      data_set =
```

恩？怎么只有一条空数据啊。貌似模拟器上默认就是这样的，如果你用的是手机的话，这里

应该就可以查到真正的数据了。不过没关系，我们可以在设置里面手动添加一个邮箱账户，如下图所示：



现在再来重新查询一遍accounts表，如下所示：

```
sqlite> select * from accounts;
select * from accounts;
account_name =
account_type =
data_set =

account_name = sinyu890807@yahoo.com
account_type = com.android.email
data_set =
sqlite>
```

OK，添加的新账户已经成功查出来了。

除了查询命令之外，还有其它的增删改命令都和标准的SQL语法是相同的，即insert、delete和update，由于比较简单，我就不再赘述了。比较值得一提的是，每个SQLite数据库中都还有一个隐藏的sqlite_master表，这里记载了当前数据库中所有表的建表语句，可以使用

select * from sqlite_master命令进行查看：

```

    type = table
    name = search_index_segdir
tbl_name = search_index_segdir
rootpage = 60
    sql = CREATE TABLE 'search_index_segdir' (level INTEGER, idx INTEGER, start_block INTEGER, leaves_end_block INTEGER, end_block INTEGER, root BLOB, PRIMARY KEY (level, idx))

    type = index
    name = sqlite_autoindex_search_index_segdir_1
tbl_name = search_index_segdir
rootpage = 61
    sql =

    type = table
    name = search_index_docsize
tbl_name = search_index_docsize
rootpage = 62
    sql = CREATE TABLE 'search_index_docsize' (docid INTEGER PRIMARY KEY, size BLOB)

    type = table
    name = search_index_stat
tbl_name = search_index_stat
rootpage = 63
    sql = CREATE TABLE 'search_index_stat' (id INTEGER PRIMARY KEY, value BLOB)
sqlite>

```

结果太多了是不是？一屏根本就显示不下嘛。不要着急，别忘了我们使用的是select命令，可以使用where语句来过滤出我们想要查询的那部分内容，如下图所示：

```

sqlite> select * from sqlite_master where name='accounts';
select * from sqlite_master where name='accounts';
    type = table
    name = accounts
tbl_name = accounts
rootpage = 49
    sql = CREATE TABLE accounts (account_name TEXT, account_type TEXT, data_set TEXT)
sqlite>

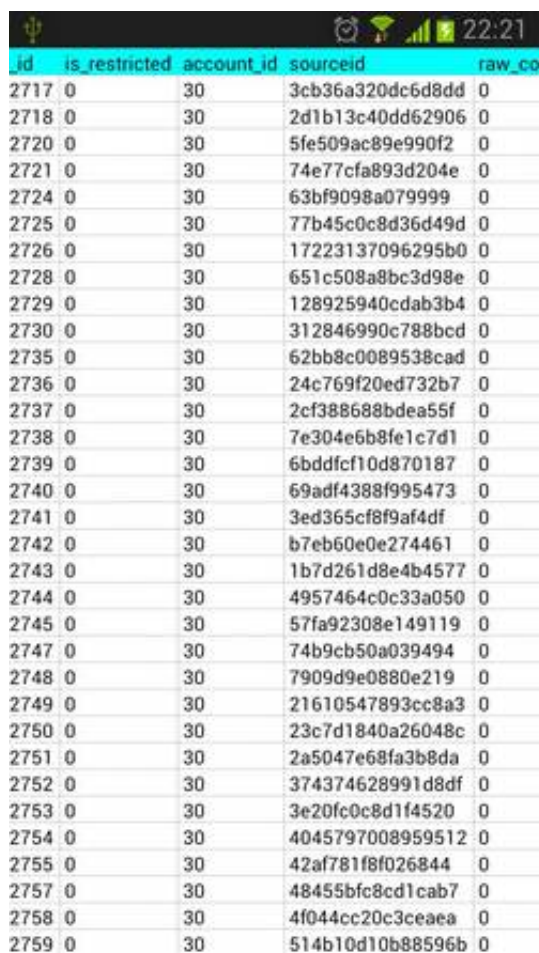
```

OK，CREATE TABLE accounts (account_name TEXT, account_type TEXT, data_set TEXT) 这就是accounts表的建表语句了，通过这种方式我们可以查询到任意一张表的建表语句，从而对我们学习和分析数据库表结构有所帮助。

有些朋友可能会觉得，每次都要输入select命令来查询表中的数据太麻烦了。没错，而且还要保证手机是连接在电脑上的时候才能查询，确实太不方便。幸运的是，有些手机软件已经

提供了数据库表查询的功能，使得我们随时随地都可以方便地查看数据库中的数据，比如Root Explorer这款软件就不错。

仍然是确保你的手机已经Root，然后安装Root Explorer，打开软件之后按照我们前面介绍的路径，进入/data/data/com.android.providers.contacts/databases，点击contacts2.db数据库，选择内置数据库查看器，然后随便点击一张表就可以查看到里面的数据了，如下图所示：



id	is_restricted	account_id	sourceid	raw_contact_id
2717	0	30	3cb36a320dc6d8dd	0
2718	0	30	2d1b13c40dd62906	0
2720	0	30	5fe509ac89e990f2	0
2721	0	30	74e77cfa893d204e	0
2724	0	30	63bf9098a079999	0
2725	0	30	77b45c0c8d36d49d	0
2726	0	30	17223137096295b0	0
2728	0	30	651c508a8bc3d98e	0
2729	0	30	128925940cdab3b4	0
2730	0	30	312846990c788bcd	0
2735	0	30	62bb8c0089538cad	0
2736	0	30	24c769f20ed732b7	0
2737	0	30	2cf388688bdea55f	0
2738	0	30	7e304e6b8fe1c7d1	0
2739	0	30	6bddfcf10d870187	0
2740	0	30	69adf4388f995473	0
2741	0	30	3ed365cf8f9af4df	0
2742	0	30	b7eb60e0e274461	0
2743	0	30	1b7d261d8e4b4577	0
2744	0	30	4957464c0c33a050	0
2745	0	30	57fa92308e149119	0
2747	0	30	74b9cb50a039494	0
2748	0	30	7909d9e0880e219	0
2749	0	30	21610547893cc8a3	0
2750	0	30	23c7d1840a26048c	0
2751	0	30	2a5047e68fa3b8da	0
2752	0	30	374374628991d8df	0
2753	0	30	3e20fc0c8d1f4520	0
2754	0	30	4045797008959512	0
2755	0	30	42af781f8f026844	0
2757	0	30	48455bfc8cd1cab7	0
2758	0	30	4f044cc20c3ceaea	0
2759	0	30	514b10d10b88596b	0

使用这种方法，我们可以随时查看数据库表中的最新数据，直观又方便，在程序开发的时候可以起到非常大的帮助。

好了，今天的讲解就到这里，下篇文章当中我将带领大家探究Android数据库中更多的奥秘。感兴趣的朋友请继续阅读 Android数据库高手秘籍(2)：创建表和LitePal的基本用法。

安卓应用频道

微信号：AndroidPD



打造东半球最好的 安卓技术 微信号

商务合作QQ：2302462408

投稿网址：top.jobbole.com

拉勾网
程序员
高薪工作范儿

BAT、小米等顶尖互联网公司900000+职位open



拉勾招聘

长按二维码关注拉勾网



拉勾 专注互联网职业机会
www.lagou.com

速戳阅读原文进入拉勾主战场

阅读原文



微信扫一扫

关注该公众号