

《数据库接口》 实验报告



学院： 计算机学院（国家示范性软件学院）

班级： 2019211308 2019211308 2019211308

姓名： 顾天阳 曾世茂 庞仕泽

学号： 2019211539 2019211532 2019211509

目录

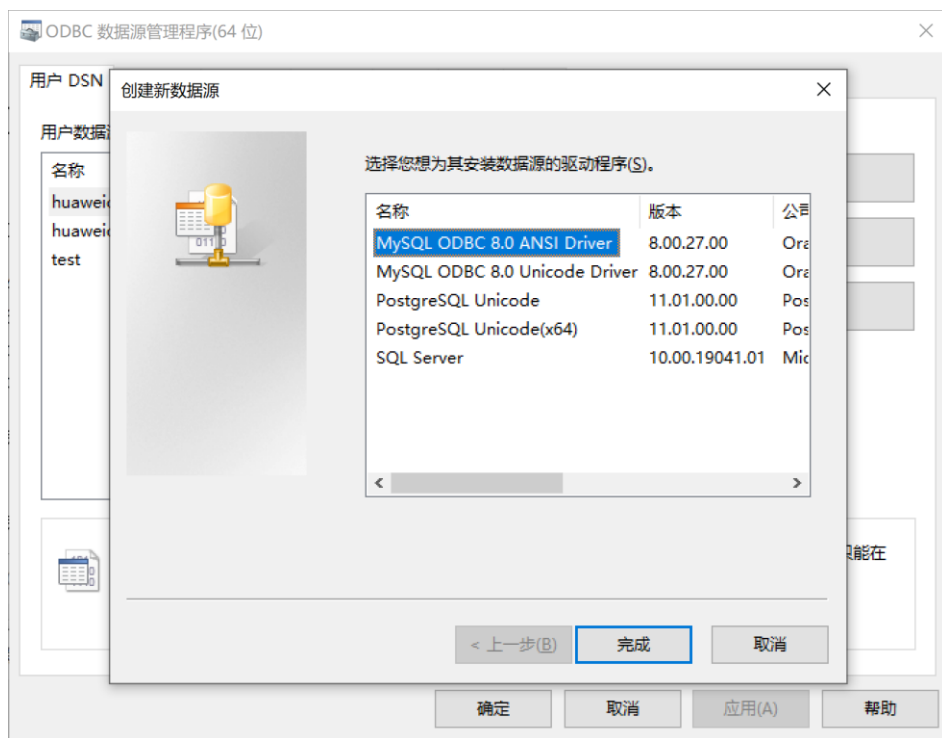
一、ODBC 访问接口环境配置	2
二、连接时长的获取和修改	3
statement_timeout:	3
tcp_keepalives_idle:	3
三、数据库连接及访问	5
1、数据库连接	5
2、建表	7
3、导入数据	7
4、插入	9
5、查询	10
6、更新	11
7、删除	12
四、问题及解决	13
问题一:	13
问题二:	13

一、ODBC 访问接口环境配置

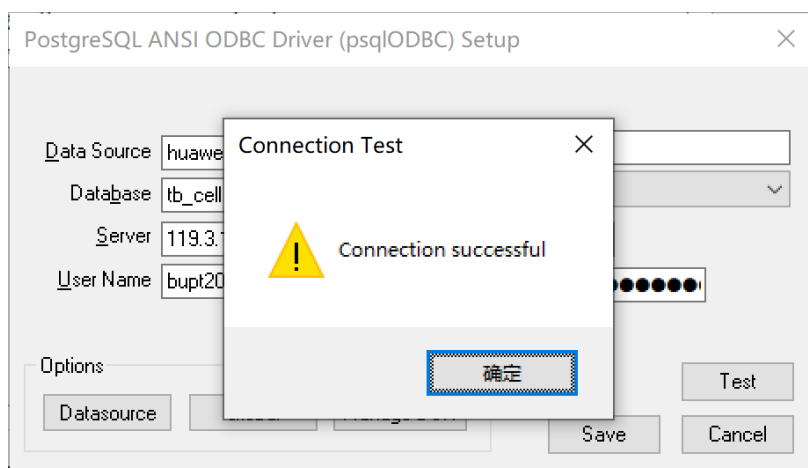
首先将 dws_8.1.x_odbc_driver_for_windows.zip 进行解压，点击 psqldbnc.msi 和 psqldbnc_x64.msi 安装驱动。

x64	2022/1/7 9:26	文件夹	
x86	2022/1/7 9:26	文件夹	
psqldbnc.msi	2020/12/9 16:42	Windows Installer ...	2,856 KB
psqldbnc_x64.msi	2020/12/9 16:41	Windows Installer ...	3,076 KB

接着打开 ODBC 数据源管理器，点击添加数据源。



选择驱动程序 PostgreSQL Unicode(x64)，建立数据源 huaweicloud，并测试连接。



至此 ODBC 访问接口环境的配置已经完成。

二、连接时长的获取和修改

statement_timeout:

控制语句执行时长，单位是 ms。超过设定值，该语句将被中止。

tcp_keepalives_idle:

空闲事务超时后，终止任何已经闲置超过这个参数所指定的时间（以秒计）的打开事务的会话。这使得该会话所持有的任何锁被释放，并且其所持有的连接槽可以被重用，它也允许只对这个事务可见的元组被清理。

首先我们查询 PostgreSQL 默认的这两个参数。

```
SQLRETURN ret;
ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR *) "select name, setting, unit
from pg_settings ps
where ps.name = 'statement_timeout' or ps.name = 'tcp_keepalives_idle';", SQL_NTS);
printf("get result: %d\n\n", ret);
```

执行结果如下，表明当执行语句超过 48ms 会中止，空闲事务超过 12342s 会中断。

```
Connected !
get result: 0
```

name	setting	unit
statement_timeout	48	ms
tcp_keepalives_idle	12342	s

为了验证超时，当我试图执行一个有 5 个自然连接的查询时，返回结果为 -1，表示查询失败。

```
(SQLCHAR*) "select * from \tbATUHandOver\", \tbATUHandOver\", \tbATUHandOver\", \tbATUHandOver\", \tbATUHandOver\";", SQL_NTS);
);
```

```
Connected !
get result: -1
```

当我把连接的表减少至 2 个时，返回结果为 1，表示查询成功，这表明当查询超过系统设置的 statement_timeout 之后，会中止查询。

```
mt, (SQLCHAR*) "select t1.\HOATT\" from \tbATUHandOver\" as t1, \tbATUHandOver\" as t2;", SQL_NTS);
ret);
```

Microsoft Visual Studio 调试控制台

```
Connected !
get result: 0
```

下面，当我试图修改这两个参数时，出现了返回结果 100。


```
SQLRETURN ret;
ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*) "update pg_settings ps set setting = 200 where ps.name = 'statement_timeout';", SQL_NTS);
printf("get result: %d\n\n", ret);
```


Microsoft Visual Studio 调试控制台

```
Connected !
get result: 100
```

查阅资料后发现，100 表示返回的是 SQL_NO_DATA，这个值意味着数据库中没有数据行受到影响。

SQL_NO_DATA

Article • 11/03/2021 • 2 minutes to read • 

Is this page helpful? 

When an ODBC 3.x application calls **SQLExecDirect**, **SQLExecute**, or **SQLParamData** in an ODBC 2.x driver to execute a searched update or delete statement that does not affect any rows at the data source, the driver should return **SQL_SUCCESS**, not **SQL_NO_DATA**. When an ODBC 2.x or ODBC 3.x application working with an ODBC 3.x driver calls **SQLExecDirect**, **SQLExecute**, or **SQLParamData** with the same result, the ODBC 3.x driver should return **SQL_NO_DATA**.

经过思考，这大概率是由于客户端权限不够，无法自行修改默认配置，所以接下来我将在华为云上演示连接时长的获取和修改。

执行如下 SQL 语句获取 statement_timeout 和 tcp_keepalives_idle：

1. SELECT name, setting, unit

```
2. FROM pg_settings ps
3. WHERE ps.name = 'statement_timeout'
4. OR ps.name = 'tcp_keepalives_idle';
```

结果如下。

```
1 SELECT name, setting, unit
2 FROM pg_settings ps
3 WHERE ps.name = 'statement_timeout'
4 OR ps.name = 'tcp_keepalives_idle';
```

SQL执行记录 消息 结果集1 X

以下是SELECT name, setting, unit FROM pg_settings ps WHERE ps.name = 'statement_timeout' OR ps.name = 'tcp_keepalives_idle'; ① 该表不可编辑。

	name	setting	unit
1	statement_timeout	1800	ms
2	tcp_keepalives_idle	60	s

修改对应的 statement_timeout 和 tcp_keepalives_idle:

```
1 update pg_settings ps
2 set setting = setting + 100
3 where ps.name = 'statement_timeout' or ps.name = 'tcp_keepalives_idle';
```

SQL执行记录 消息

-----开始执行-----

【拆分SQL完成】: 将执行SQL语句数量: (1条)

【执行SQL: (1)】

update pg_settings ps

set setting = setting + 100

where ps.name = 'statement_timeout' or ps.name = 'tcp_keepalives_idle';

执行成功, 耗时: [12ms.]

再次查询 statement_timeout 和 tcp_keepalives_idle, 发现数值已经成功加上 100。

```
1 SELECT name, setting, unit
2 FROM pg_settings ps
3 WHERE ps.name = 'statement_timeout'
4 OR ps.name = 'tcp_keepalives_idle';
```

SQL执行记录 消息 结果集1 X

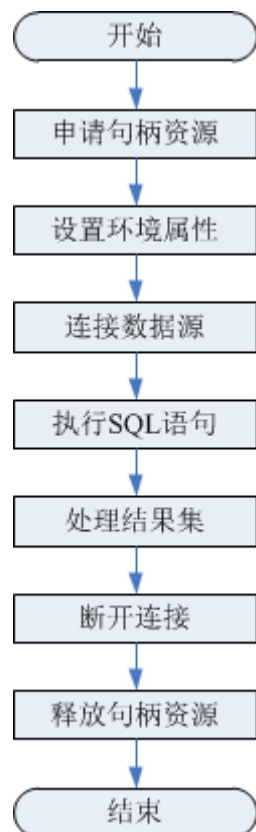
以下是SELECT name, setting, unit FROM pg_settings ps WHERE ps.name = 'statement_timeout' OR ps.name = 'tcp_keepalives_idle'; ① 该表不可编辑。

	name	setting	unit
1	statement_timeout	1900	ms
2	tcp_keepalives_idle	160	s

三、数据库连接及访问

1、数据库连接

ODBC 开发应用程序的流程如下:



引入 SQL 语句之前，首先需要申请环境句柄、连接句柄，之后再连接到数据源。另外，执行 SQL 语句之前，我们还需要申请语句句柄。

代码片段如下：

```
1. // 申请环境句柄
2. V_OD_erg = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &V_OD_Env)
   ;
3. if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
4. {
5.     printf("Error AllocHandle\n");
6.     exit(0);
7. }
8.
9. // 设置环境属性（版本信息）
10. SQLSetEnvAttr(V_OD_Env, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0
   );
11.
12. // 申请连接句柄
13. V_OD_erg = SQLAllocHandle(SQL_HANDLE_DBC, V_OD_Env, &V_OD_hdbc);
14. if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
15. {
16.     SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
17.     exit(0);
```

```

18. }
19.
20. // 设置连接属性
21. // SQLSetConnectAttr(V_OD_hdbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_0
    N, 0);
22.
23. // 连接数据源
24. V_OD_erg = SQLConnect(V_OD_hdbc, (SQLCHAR*)"huaweicloud", SQL_NTS,
25.     (SQLCHAR*)"bupt2019db37", SQL_NTS, (SQLCHAR*)"bupt2019db37@123"
        , SQL_NTS);
26. if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
27. {
28.     printf("Error SQLConnect %d\n", V_OD_erg);
29.     SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
30.     exit(0);
31. }
32. printf("Connected !\n");
33.
34. // 设置语句属性
35. SQLSetStmtAttr(V_OD_hstmt, SQL_ATTR_QUERY_TIMEOUT, (SQLPOINTER*)3, 0)
    ;
36.
37. // 申请语句句柄
38. SQLAllocHandle(SQL_HANDLE_STMT, V_OD_hdbc, &V_OD_hstmt);

```

2、建表

连接到数据源之后，我们基于 ODBC 创建 tbATUHandOver 表，包括 SSECTOR_ID、NSECTOR_ID 和 HOATT 字段。

建表代码片段如下：

```

1. // 直接执行 SQL 语句
2. SQLRETURN ret;
3. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"create table \"tbATUHandOver\"(\"SSECTOR_ID\" varchar(50), \"NSECTOR_ID\" varchar(50), \"HOATT\"
    \" int);", SQL_NTS);
4. printf("create result: %d\n", ret);

```

程序执行如下，返回结果为 1 表示建表成功。

```

Connected !
create result: 1

```

3、导入数据

导入代码片段如下:

结果显示数据均导入成功。

下面我们选择 tbATUHandOver 表中的全部数据进行验证。

选择全部数据代码片段如下:

```
1. SQLRETURN ret;
2. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"select * from \"tbATUHandO
   ver\"", SQL_NTS);
3. int i = 0;
4. V_OD_erg = SQLFetch(V_OD_hstmt);
5. printf("%-20s%-20s-10s\n", "SECTOR_ID", "NSECTOR_ID", "HOATT");
6. printf("-----\n");
```



```

7. while (V_OD_erg != SQL_NO_DATA)
8. {
9.     SQLGetData(V_OD_hstmt, 1, SQL_C_DEFAULT, (SQLPOINTER)&SSECTOR_ID,
        100, NULL);
10.    SQLGetData(V_OD_hstmt, 2, SQL_C_DEFAULT, (SQLPOINTER)&NSECTOR_ID,
        100, NULL);
11.    SQLGetData(V_OD_hstmt, 3, SQL_C_DEFAULT, (SQLPOINTER)&HOATT, 100,
        NULL);
12.    printf("%-20s%-20s-10d\n", SSECTOR_ID, NSECTOR_ID, HOATT);
13.    V_OD_erg = SQLFetch(V_OD_hstmt);
14.    i++;
15. };

```

结果显示数据可以全部查出。

Microsoft Visual Studio 调试控制台

Connected !

SSECTOR_ID	NSECTOR_ID	HOATT
15113-129	253890-1	1
15113-129	253914-1	1
238397-1	253931-0	2
238397-1	253927-2	2
238397-2	238397-1	2
238397-2	7400-130	1
253890-0	253934-1	2
253890-1	253914-1	5
253890-1	253935-0	5
253890-1	253899-0	4
253890-1	253890-2	1
253890-2	253890-1	3
253890-2	253912-2	2
253890-2	253890-0	2
253891-0	253891-2	2
253891-1	253923-1	4
253891-1	253891-2	2
253891-2	253891-1	4
253891-2	253899-0	2
253891-2	253899-1	1
253892-1	253912-2	2
253893-0	238397-2	2
253893-0	259778-1	2
253893-0	253893-2	1
253893-0	5641-129	1
253893-2	253893-0	2
253893-2	253905-0	2

4、插入

添加一条 SSECTOR_ID: '123', NSECTOR_ID: '456', HOATT: 8 到数据库。

插入代码片段如下：

```

1. SQLRETURN ret;
2. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"insert into \"tbATUHandOver\" values ('123', '456', 9)", SQL_NTS);
3. printf("insert result: %d\n\n", ret);

```

结果显示插入成功。

Connected !
insert result: 0

插入之后查询该表，发现数据已经添加成功！

259778-2	259778-0	2
259778-2	259778-1	2
5641-129	253893-0	3
5641-129	253939-0	1
5641-130	253903-1	4
5641-130	5641-129	2
5641-130	253939-2	2
7201-128	253934-2	1
7400-128	253930-130	1
7400-130	238397-2	1
123	456	9

5、查询

查询 tbATUHandOver 中，所有 HOATT 字段值大于 4 的数据行，并且按照 HOATT 字段升序排列。

查询代码片段如下：

```

1. SQLRETURN ret;
2. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"select * from \"tbATUHandO
   ver\" where \"HOATT\" > 4 order by \"HOATT\" asc", SQL_NTS);
3. int i = 0;
4. V_OD_erg = SQLFetch(V_OD_hstmt);
5. printf("%-20s%-20s%-10s\n", "SSECTOR_ID", "NSECTOR_ID", "HOATT");
6. printf("-----\n");
7. while (V_OD_erg != SQL_NO_DATA)
8. {
9.     SQLGetData(V_OD_hstmt, 1, SQL_C_DEFAULT, (SQLPOINTER)&SSECTOR_ID,
        100, NULL);
10.    SQLGetData(V_OD_hstmt, 2, SQL_C_DEFAULT, (SQLPOINTER)&NSECTOR_ID,
        100, NULL);
11.    SQLGetData(V_OD_hstmt, 3, SQL_C_DEFAULT, (SQLPOINTER)&HOATT, 100,
        NULL);
12.    printf("%-20s%-20s%-10d\n", SSECTOR_ID, NSECTOR_ID, HOATT);
13.    V_OD_erg = SQLFetch(V_OD_hstmt);
14.    i++;
15. };

```

结果如下，按照 HOATT 字段升序排列。

Microsoft Visual Studio 调试控制台

Connected !

SSECTOR_ID	NSECTOR_ID	HOATT
253890-1	253914-1	5
253890-1	253935-0	5
253895-1	253916-1	5
253899-2	259772-1	5
253914-1	253890-1	5
253916-2	253941-1	5
253927-2	253921-2	5
253932-1	253934-2	5
253935-0	259772-0	5
259772-0	253935-0	5
259772-1	253899-2	5
259772-1	253923-2	5
253912-0	253935-2	6
253939-0	253939-2	6
253934-2	253932-1	6
253894-2	253939-0	6
253923-2	259772-1	6
253939-2	5641-130	6
253935-2	259772-0	8
253910-2	253911-1	8
259772-0	253935-2	9
123	456	9
253911-1	253910-2	10

6、更新

将 4 中插入的数据的 HOATT 字段值改为 66，并打印该行信息。

更新打印代码片段如下：

```
1. SQLRETURN ret;
2. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"update \"tbATUHandOver\" s
   et \"HOATT\" = 66 where \"SSECTOR_ID\" = '123'", SQL_NTS);
3. printf("update result: %d\n\n", ret);
4.
5. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"select * from \"tbATUHandO
   ver\" where \"SSECTOR_ID\" = '123'", SQL_NTS);
6. int i = 0;
7. V_OD_erg = SQLFetch(V_OD_hstmt);
8. printf("%-20s%-20s%-10s\n", "SSECTOR_ID", "NSECTOR_ID", "HOATT");
9. printf("-----\n");
10. while (V_OD_erg != SQL_NO_DATA)
11. {
12.     SQLGetData(V_OD_hstmt, 1, SQL_C_DEFAULT, (SQLPOINTER)&SSECTOR_ID,
        100, NULL);
13.     SQLGetData(V_OD_hstmt, 2, SQL_C_DEFAULT, (SQLPOINTER)&NSECTOR_ID,
        100, NULL);
14.     SQLGetData(V_OD_hstmt, 3, SQL_C_DEFAULT, (SQLPOINTER)&HOATT, 100,
        NULL);
15.     printf("%-20s%-20s%-10d\n", SSECTOR_ID, NSECTOR_ID, HOATT);
16.     V_OD_erg = SQLFetch(V_OD_hstmt);
17.     i++;
18. };
```

结果如下，数据已经修改成功。

```
Connected !
update result: 0
```

SSECTOR_ID	NSECTOR_ID	HOATT
123	456	66

7、删除

删除 4 中插入的数据，并打印整张表。

删除打印代码片段如下：

```
1. SQLRETURN ret;
2. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"delete from \"tbATUHandOve
   r\" where \"SSECTOR_ID\" = '123'", SQL_NTS);
3. printf("delete result: %d\n\n", ret);
4.
5. ret = SQLExecDirect(V_OD_hstmt, (SQLCHAR*)"select * from \"tbATUHandO
   ver\"", SQL_NTS);
6. int i = 0;
7. V_OD_erg = SQLFetch(V_OD_hstmt);
8. printf("%-20s%-20s%-10s\n", "SSECTOR_ID", "NSECTOR_ID", "HOATT");
9. printf("-----\n");
10. while (V_OD_erg != SQL_NO_DATA)
11. {
12.     SQLGetData(V_OD_hstmt, 1, SQL_C_DEFAULT, (SQLPOINTER)&SSECTOR_ID,
        100, NULL);
13.     SQLGetData(V_OD_hstmt, 2, SQL_C_DEFAULT, (SQLPOINTER)&NSECTOR_ID,
        100, NULL);
14.     SQLGetData(V_OD_hstmt, 3, SQL_C_DEFAULT, (SQLPOINTER)&HOATT, 100,
        NULL);
15.     printf("%-20s%-20s%-10d\n", SSECTOR_ID, NSECTOR_ID, HOATT);
16.     V_OD_erg = SQLFetch(V_OD_hstmt);
17.     i++;
18. };
```

结果显示删除成功，并打印整张表。

```
Microsoft Visual Studio 调试控制台
Connected !
delete result: 0

SSECTOR_ID      NSECTOR_ID      HOATT
-----
15113-129        253890-1         1
15113-129        253914-1         1
238397-1         253931-0         2
238397-1         253927-2         2
238397-2         238397-1         2
238397-2         7400-130         1
253890-0         253934-1         2
253890-1         253914-1         5
253890-1         253935-0         5
253890-1         253899-0         4
253890-1         253890-2         1
253890-2         253890-1         3
253890-2         253912-2         2
253890-2         253890-0         2
253891-0         253891-2         2
253891-1         253923-1         4
253891-1         253891-2         2
253891-2         253891-1         4
253891-2         253899-0         2
253891-2         253899-1         1
253892-1         253912-2         2
253893-0         238397-2         2
253893-0         259778-1         2
253893-0         253893-2         1
253893-0         5641-129         1
```

四、问题及解决

问题一：

在编写 ODBC 程序之后，执行时出现 Error SQLConnect -1，无法连接到远程数据库。这个问题困扰了我半天，查阅了很多资料都没有解决。于是我试了一下连接本地 MySQL 数据库，发现可以连上，于是我判定代码是没有问题的，很大可能是环境的问题。

```
Microsoft Visual Studio 调试控制台
Error SQLConnect -1

D:\ODBC\Test\Test\Debug\Test.exe (进程 18156) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . .
```

于是我调用了 SQLGetDiaField 函数，发现未找到数据源，同时又检查了华为方面的方案，也都没有违背。这就让我更加坚信是 ODBC 配置环境的问题。

```
SQLSTATE : IM002
[Microsoft][ODBC 驱动程序管理器] 未发现数据源名称并且未指定默认驱动程序
```

我又考虑到之前本地 MySQL 数据库是 64 位的，而我安装的驱动程序是 32 位的，我猜想可能是不兼容 32 位驱动。于是我便下载安装了 64 位的驱动程序 PostgreSQL Unicode(x64)，结果令人惊喜！成功连接！

```
Connected !
get result: 0
```

问题二：

在连接到数据源之后，我进行了第一个选择操作，发现返回结果为 -1，表示查询失败：

```
Connected !
get result: -1
```

之后试了很多次，依旧返回-1，后来询问了班里同学得知，公网连接数据库原始为空，需要自己手动建表并导入数据，问题得以解决。

在公网IP数据库里导入数据 原始是空的