《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](#_Toc92479273)

[二、连接时长的获取和修改 3](#_Toc92479274)

[statement\_timeout: 3](#_Toc92479275)

[tcp\_keepalives\_idle: 3](#_Toc92479276)

[三、数据库连接及访问 5](#_Toc92479277)

[1、数据库连接 5](#_Toc92479278)

[2、建表 7](#_Toc92479279)

[3、导入数据 7](#_Toc92479280)

[4、插入 9](#_Toc92479281)

[5、查询 10](#_Toc92479282)

[6、更新 11](#_Toc92479283)

[7、删除 12](#_Toc92479284)

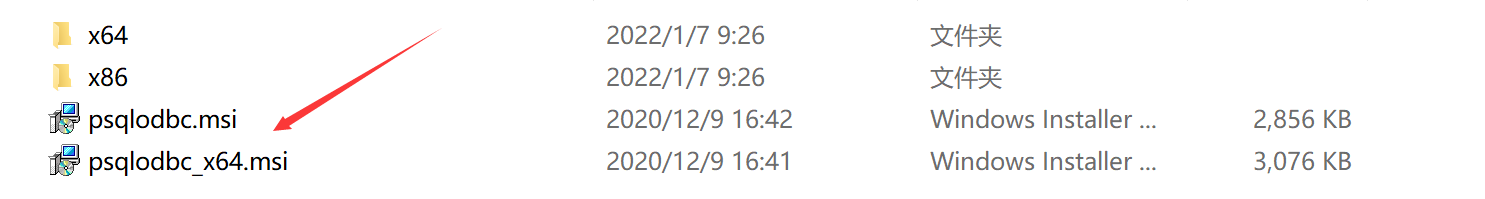
[四、问题及解决 13](#_Toc92479285)

[问题一： 13](#_Toc92479286)

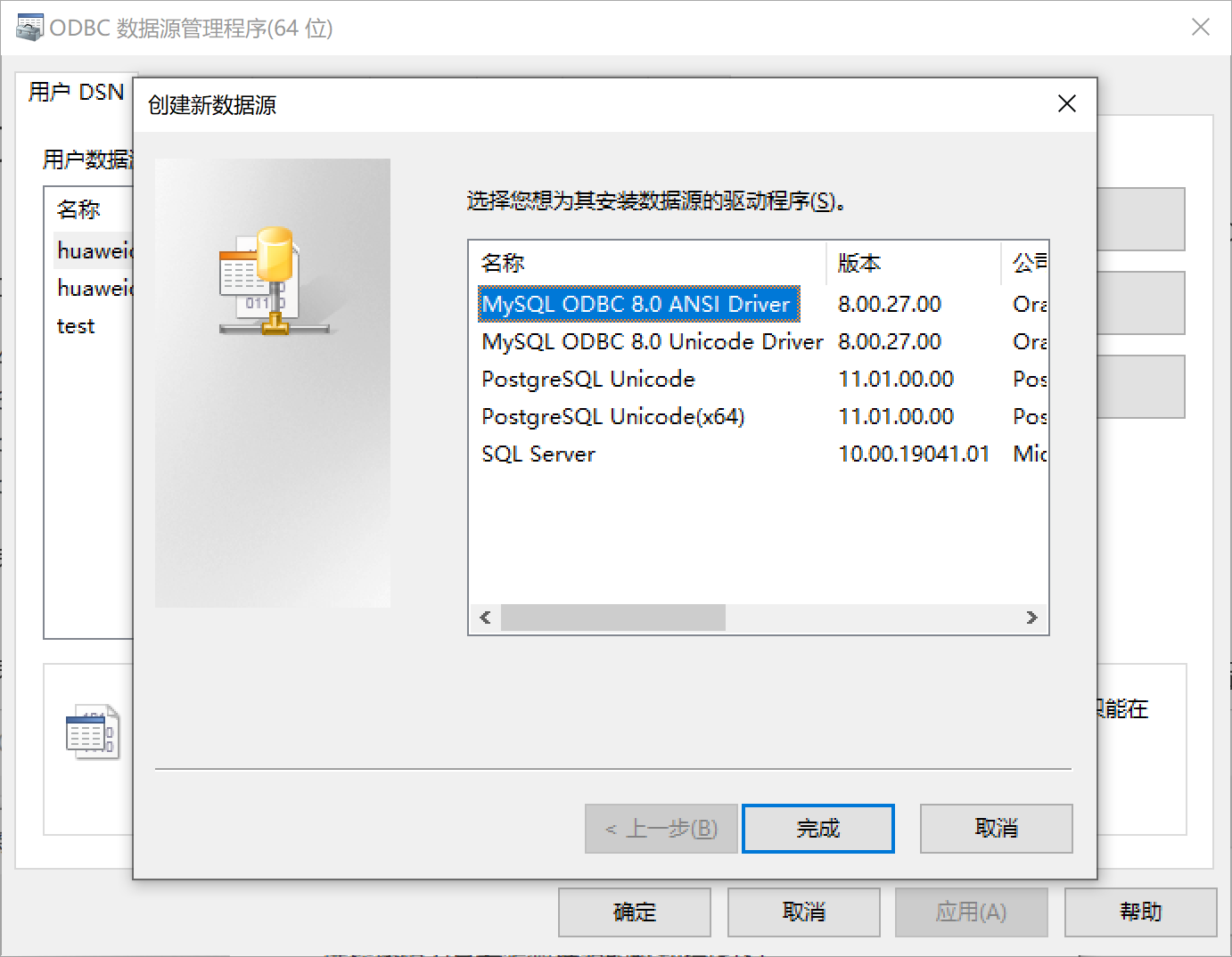
[问题二： 13](#_Toc92479287)

一、ODBC访问接口环境配置

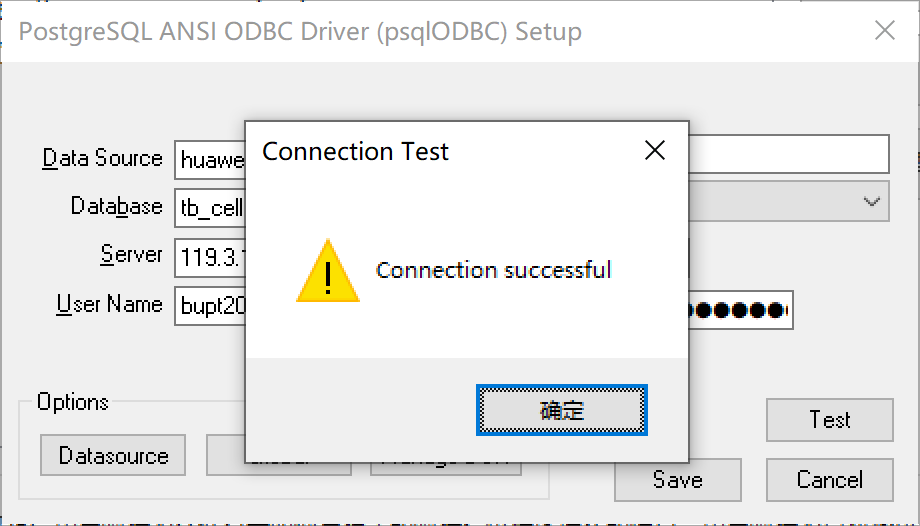
首先将dws\_8.1.x\_odbc\_driver\_for\_windows.zip进行解压，点击psqlodbc.msi和psqlodbc\_x64.msi安装驱动。



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

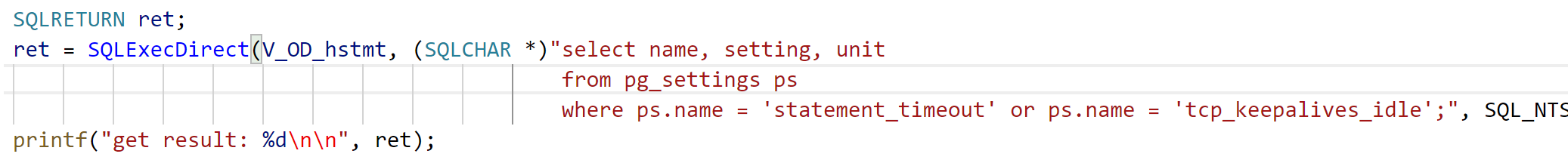
statement\_timeout:

控制语句执行时长，单位是ms。超过设定值，该语句将被中止。

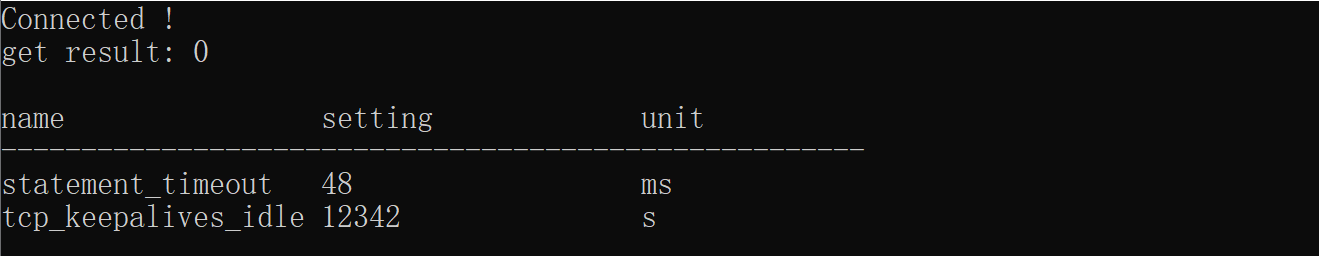
tcp\_keepalives\_idle:

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。

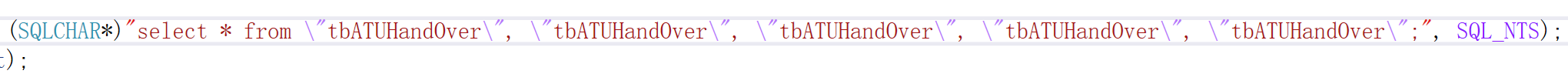
首先我们查询PostgreSQL默认的这两个参数。



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。

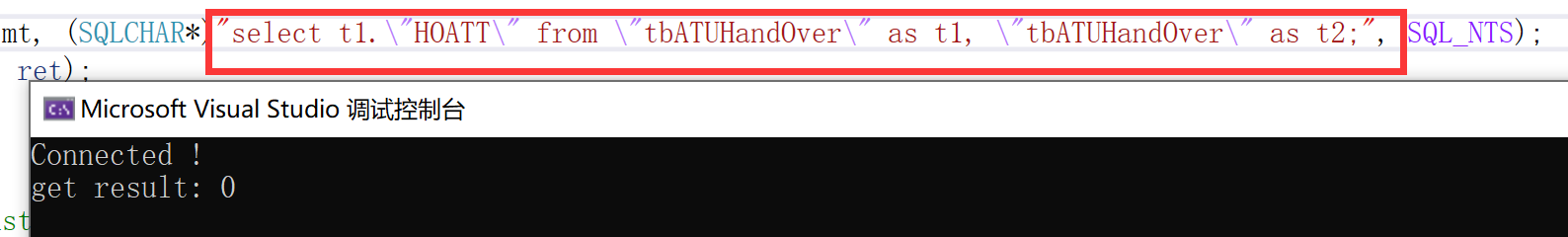


为了验证超时，当我试图执行一个有5个自然连接的查询时，返回结果为-1，表示查询失败。

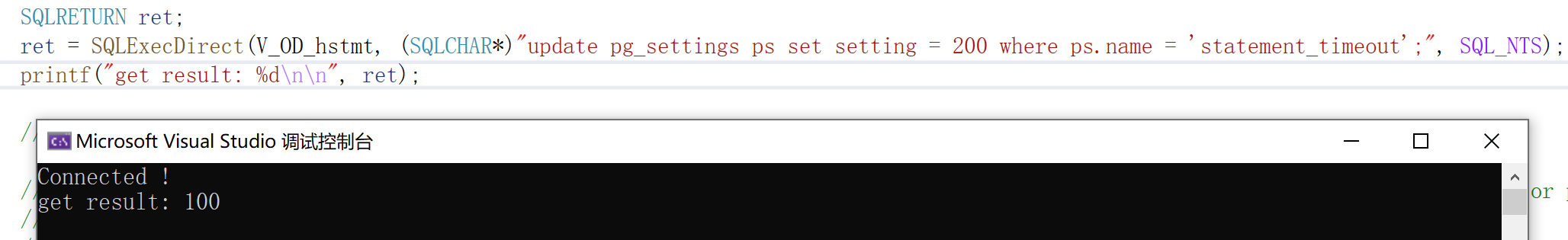




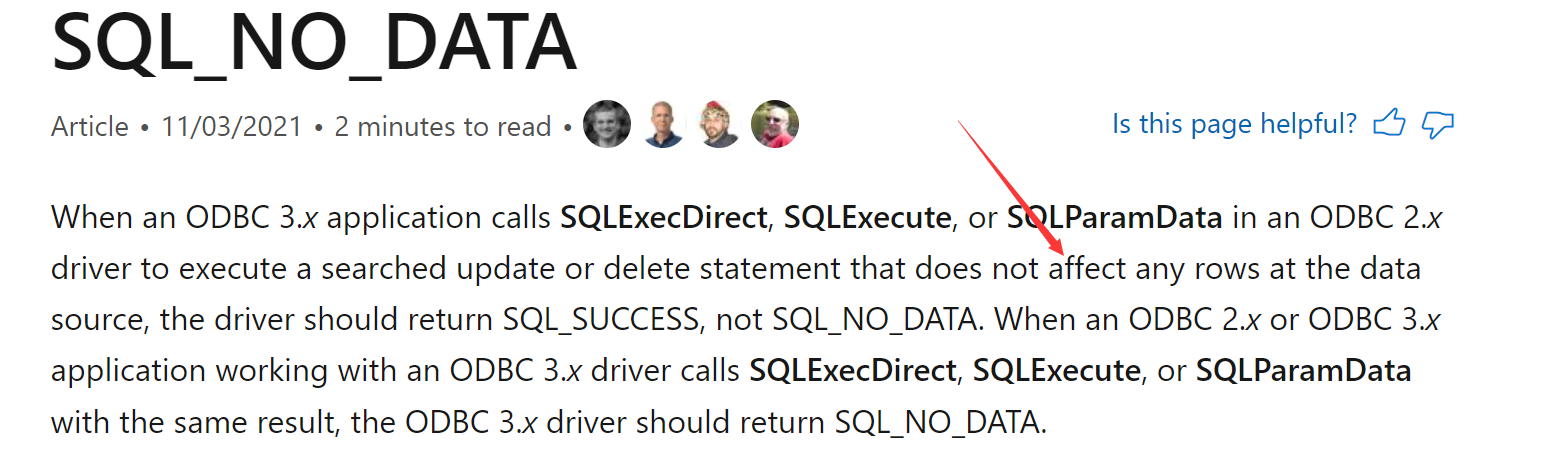
当我把连接的表减少至2个时，返回结果为1，表示查询成功，这表明当查询超过系统设置的statement\_timeout之后，会中止查询。



下面，当我试图修改这两个参数时，出现了返回结果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。



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。



修改对应的statement\_timeout和tcp\_keepalives\_idle：



再次查询statement\_timeout和tcp\_keepalives\_idle，发现数值已经成功加上100。



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. }
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);
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. }
20. // 设置连接属性
21. // SQLSetConnectAttr(V\_OD\_hdbc, SQL\_ATTR\_AUTOCOMMIT, SQL\_AUTOCOMMIT\_ON, 0);
23. // 连接数据源
24. V\_OD\_erg = SQLConnect(V\_OD\_hdbc, (SQLCHAR\*)"huaweicloud", SQL\_NTS,
25. (SQLCHAR\*)"bupt2019dbs37", SQL\_NTS, (SQLCHAR\*)"bupt2019dbs37@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");
34. // 设置语句属性
35. SQLSetStmtAttr(V\_OD\_hstmt, SQL\_ATTR\_QUERY\_TIMEOUT, (SQLPOINTER\*)3, 0);
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。



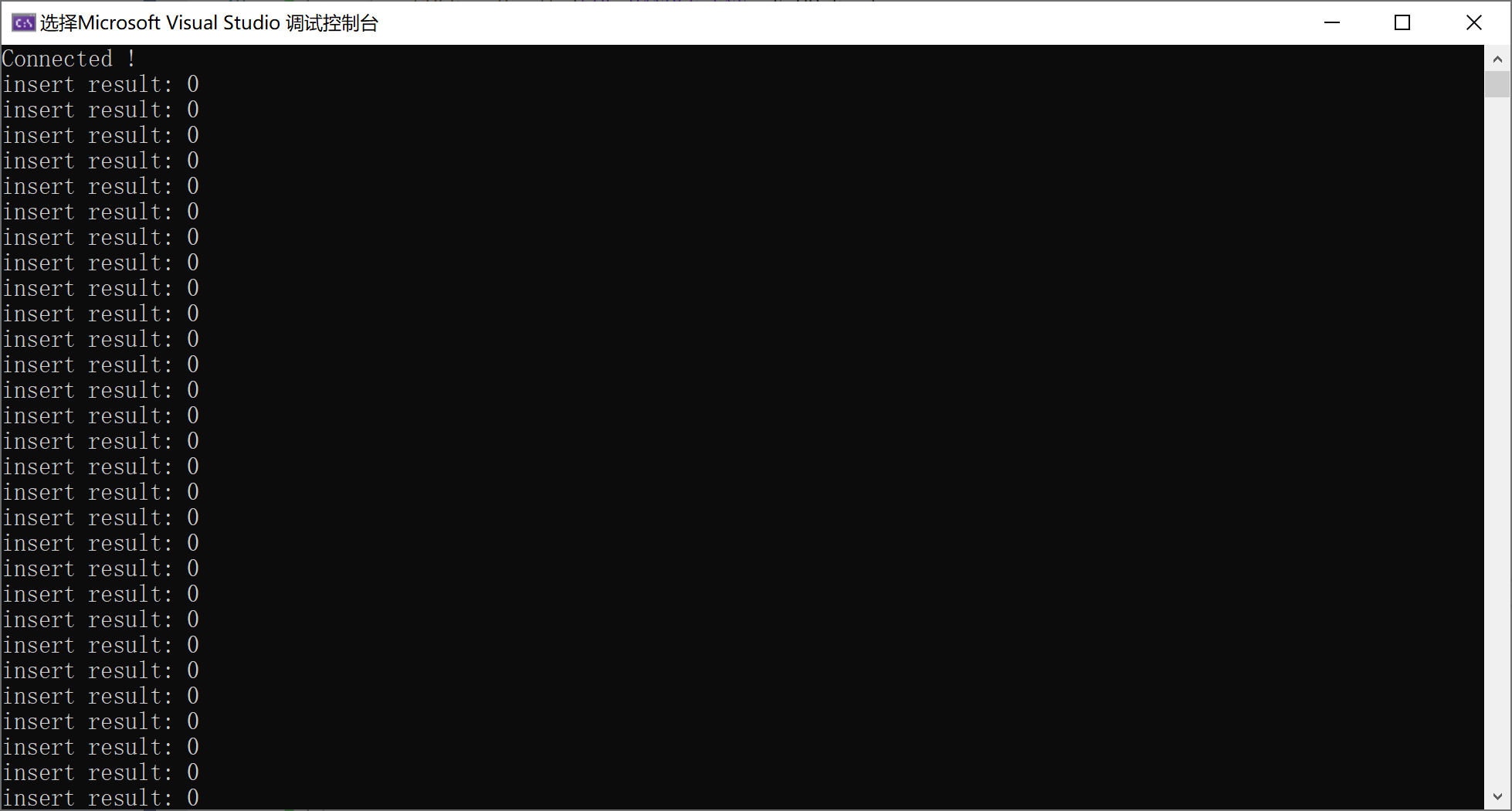
3、导入数据

由于初始tbATUHandOver表为空，所以我们需要导入数据。

导入代码片段如下：

1. SQLRETURN ret;
2. SQLPrepare(V\_OD\_hstmt, "insert into \"tbATUHandOver\" values(?, ?, ?)", SQL\_NTS);
4. **FILE**\* fp = fopen("data.txt", "r+");
5. **while** (fscanf(fp, "%s %s %d\n", SSECTOR\_ID, NSECTOR\_ID, &HOATT) == 3)
6. {
7. SQLBindParameter(V\_OD\_hstmt, 1, SQL\_PARAM\_INPUT, SQL\_C\_CHAR, SQL\_C\_CHAR, 0, 0, &SSECTOR\_ID, 0, NULL);
8. SQLBindParameter(V\_OD\_hstmt, 2, SQL\_PARAM\_INPUT, SQL\_C\_CHAR, SQL\_C\_CHAR, 0, 0, &NSECTOR\_ID, 0, NULL);
9. SQLBindParameter(V\_OD\_hstmt, 3, SQL\_PARAM\_INPUT, SQL\_C\_SLONG, SQL\_INTEGER, 0, 0, &HOATT, 0, NULL);
10. ret = SQLExecute(V\_OD\_hstmt);
11. printf("insert result: %d\n", ret);
12. }

结果显示数据均导入成功。

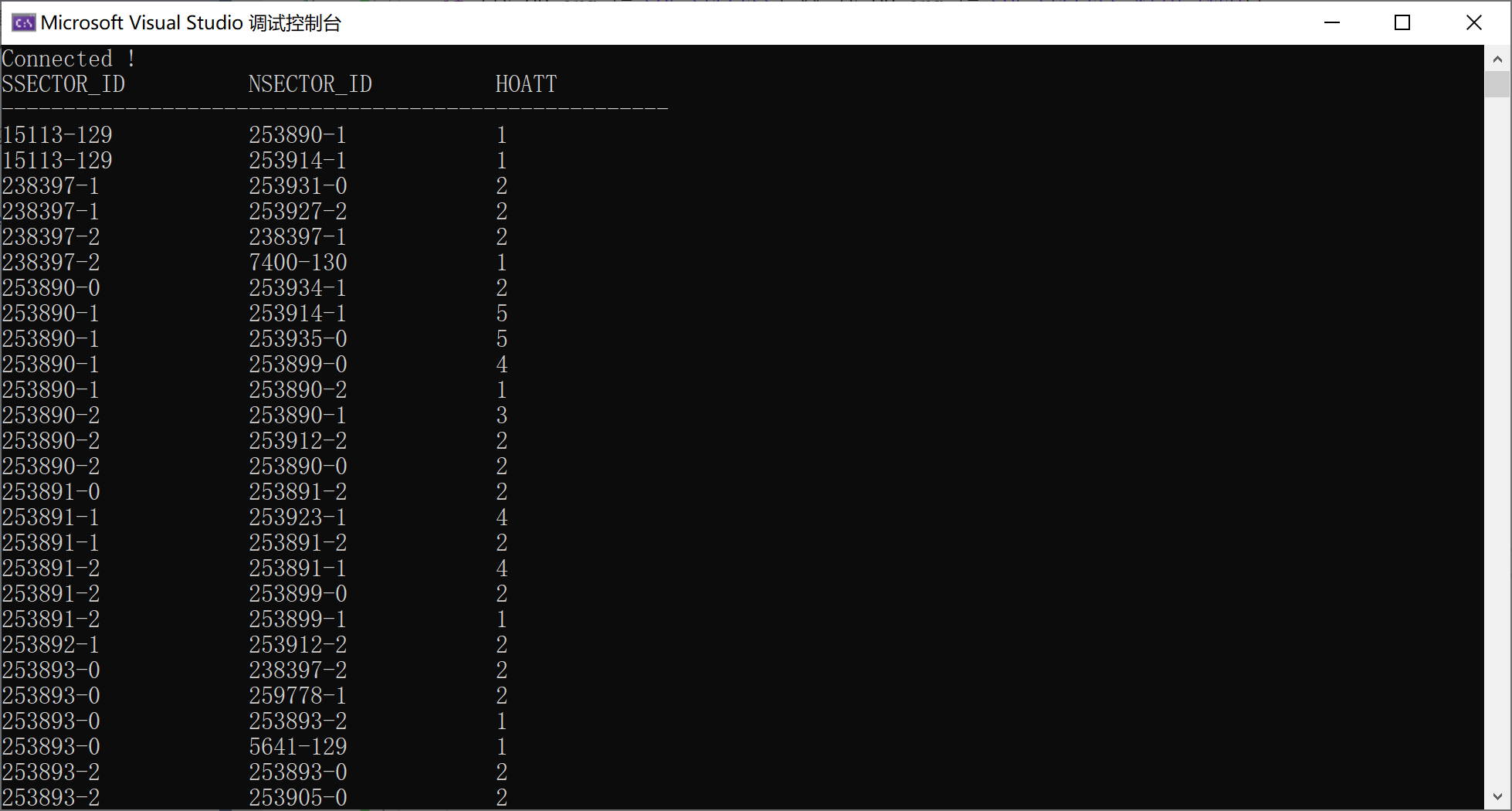


下面我们选择tbATUHandOver表中的全部数据进行验证。

选择全部数据代码片段如下：

1. SQLRETURN ret;
2. ret = SQLExecDirect(V\_OD\_hstmt, (SQLCHAR\*)"select \* from \"tbATUHandOver\"", 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. };

结果显示数据可以全部查出。



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。



插入之后查询该表，发现数据已经添加成功！



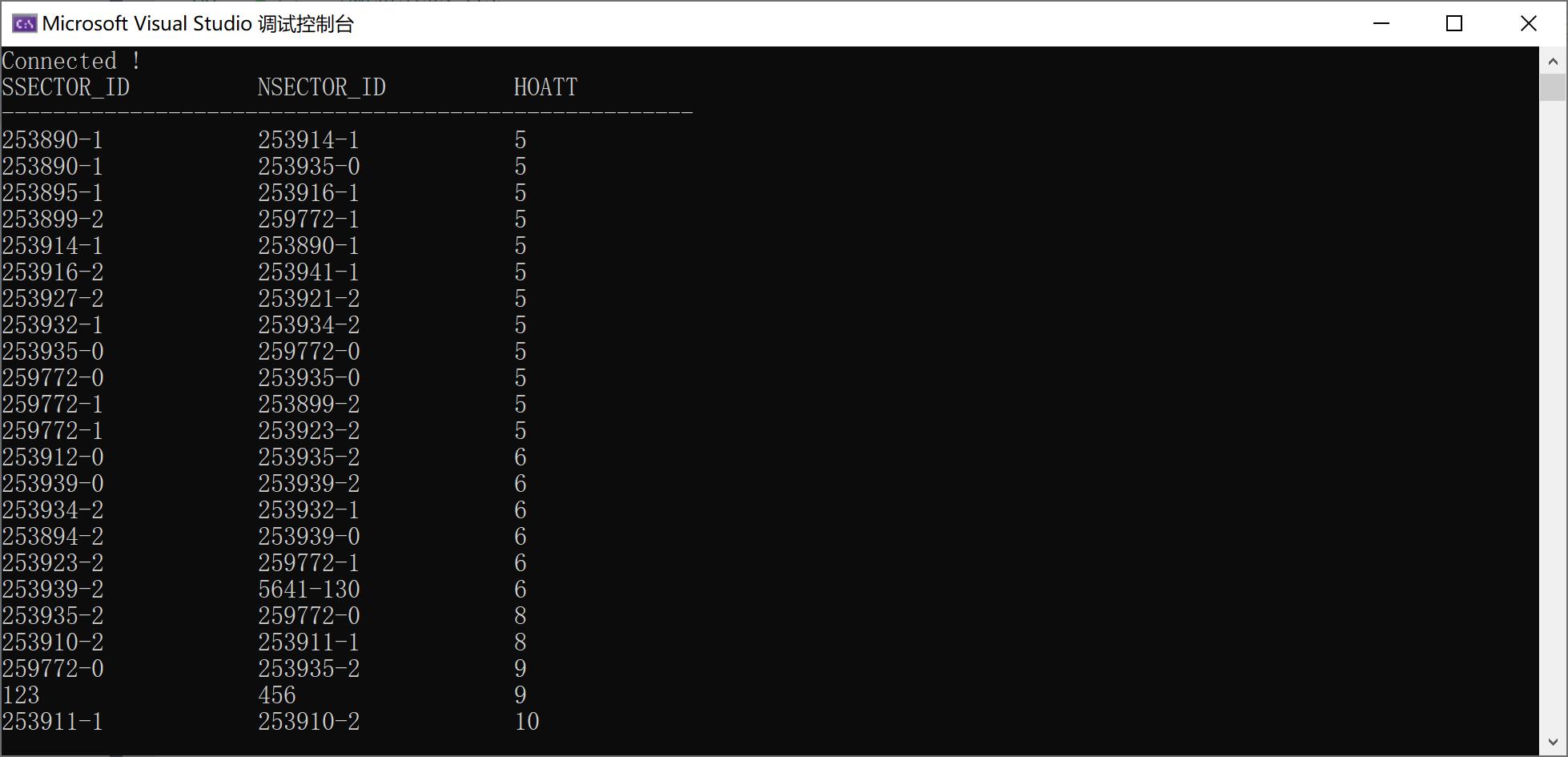
5、查询

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 \"tbATUHandOver\" 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。



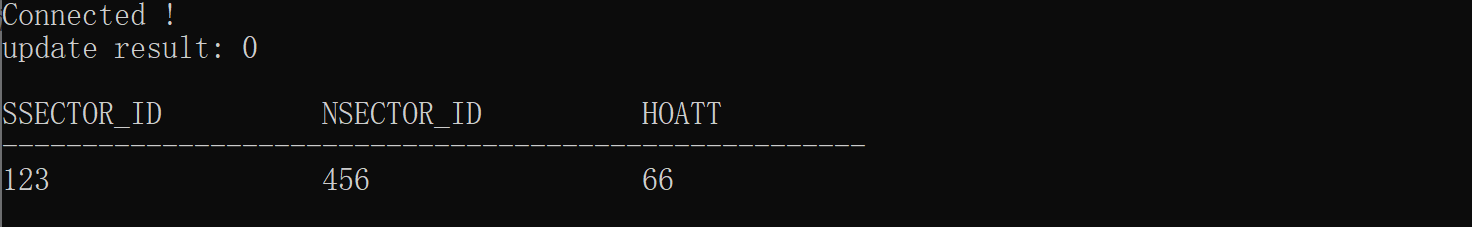
6、更新

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\" set \"HOATT\" = 66 where \"SSECTOR\_ID\" = '123'", SQL\_NTS);
3. printf("update result: %d\n\n", ret);
5. ret = SQLExecDirect(V\_OD\_hstmt, (SQLCHAR\*)"select \* from \"tbATUHandOver\" 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。



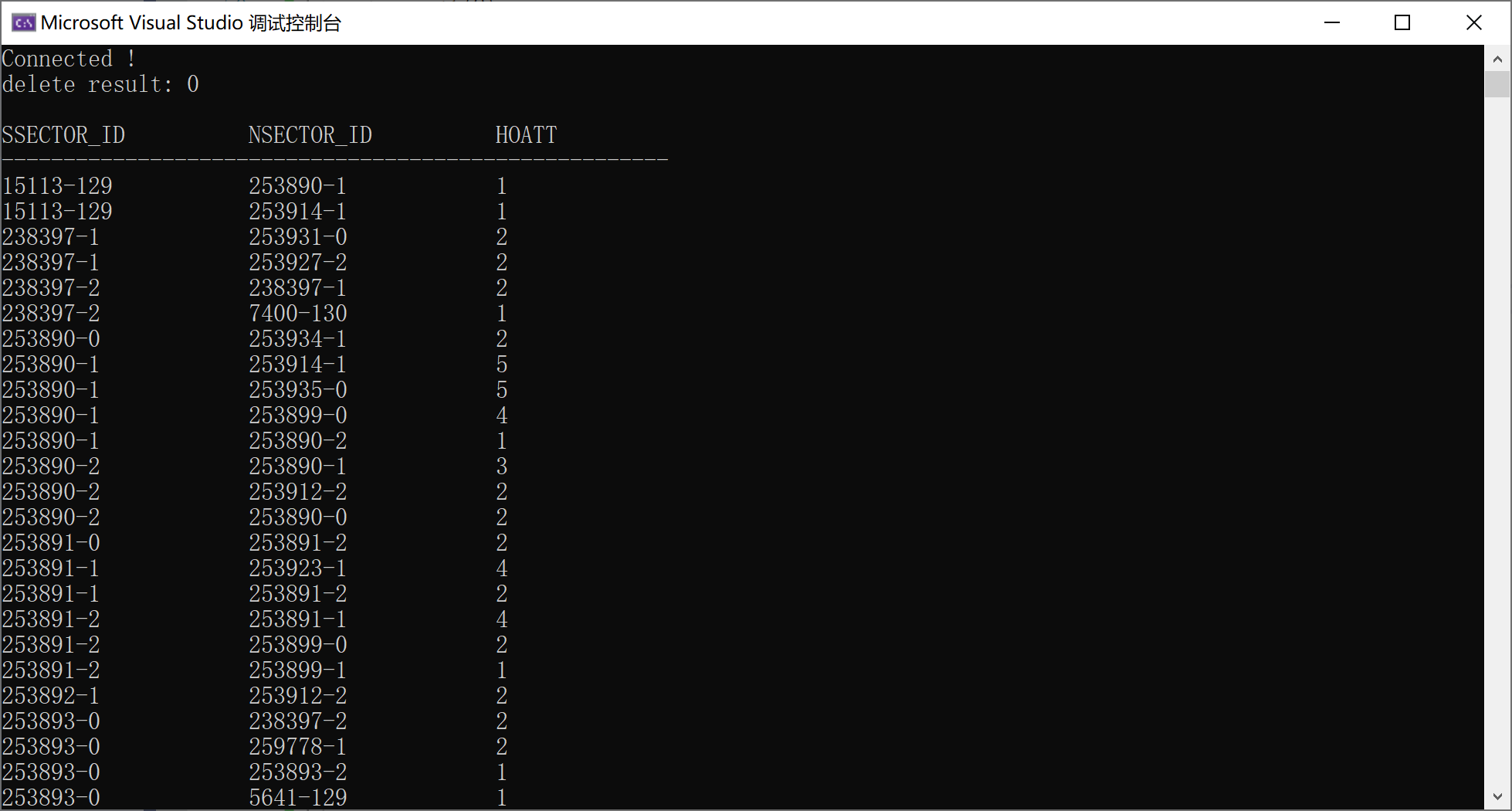
7、删除

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 \"tbATUHandOver\" where \"SSECTOR\_ID\" = '123'", SQL\_NTS);
3. printf("delete result: %d\n\n", ret);
5. ret = SQLExecDirect(V\_OD\_hstmt, (SQLCHAR\*)"select \* from \"tbATUHandOver\"", 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. };

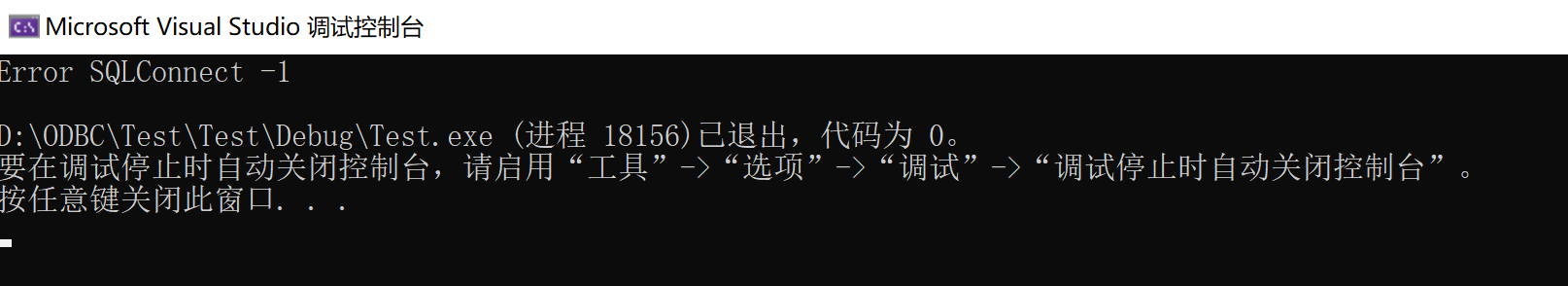
结果显示删除成功，并打印整张表。



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。



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。



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！



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：



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。

