

深圳天软科技

# 天软 ODBC 驱动程序

*TinySoft ODBC Driver*

## 1 更新日志

### 1.1 更新日志

| 更新日期       | 更新说明  |
|------------|---|
| 2016-02-11 | 文档创建和发布   |
| 2016-02-17 | 新增 Excel 和 VB 应用开发实例  |
| 2016-07-11 | 实现 SQLConnect 的支持<br>新增 MATLAB、SAS 开发实例   |
| 2016-07-27 | 增加对 SQLFetchScroll 的支持<br>支持行集(rowset)  |
| 2016-07-29 | 修订某些情况下内存泄露的 BUG<br>完善对 rowset 的处理, 支持 MATLAB Native ODBC interface<br>增加对 MATLAB Native ODBC interface 的开发说明   |
| 2016-08-03 | 修订不调用 SQLFreeStmt 的情况下, 内存可能泄露的问题<br>MATLAB 示例代码中增加 close(curs), 否则会存在不释放的句柄  |
| 2016-08-11 | 脱离对 PCRE 库的依赖。调整正则表达式处理, 支持 ECMA Script 语法, 且大小写不敏感。<br>修改执行函数, 支持参数调用, 新增对 SQLBindParameter, SQLExecute, SQLPrepare, SQLNumParams 函数的支持。<br>新增 Python 开发实例<br>新增 VBA 参数化数据提取实例 |
| 2016-08-12 | 新增对 TaskAdmin 指令的支持, 新增 TaskAdmin 调用的 Python 实例。<br>修改 DSN 配置对话框, 支持 TaskAdmin 指令返回结果的设置<br>完善 SQLGetData, 新增对 long data 的支持。   |
| 2016-12-12 | 新增 JDBC-ODBC 桥的实例<br>修订 DSN 异常提示<br>修订 SQLGetTypeInfo 中对 UNICODE 的处理  |
| 2017-05-12 | 修订 R 实例, 增加 odbcClose   |

### 1.2 摘要

天软科技推出的天软 ODBC 驱动程序(以下简称 TinyODBC)结合了天软公司 TSL 语言的特点, 为用户查询天软数据平台提供了一种方便、统一的方式。因为大多数的数据处理软件都支持 ODBC 接口, 所以 TinyODBC 也提高了用户的整合能力。本文主要包括以下内容:

1. 天软 ODBC 驱动程序标准
2. 天软 ODBC 驱动程序安装
3. 天软 ODBC 应用开发实例

## 2 天软 ODBC 驱动程序标准

本节主要说明 TinyODBC 遵守的标准，支持的数据类型和可配置的参数等内容。

### 2.1 Conformance Level

TinyODBC 支持微软 ODBC 3.5 标准，函数基本支持微软 ODBC Core Interface Conformance Level，描述符字段支持 level 2。微软 ODBC Conformance Level 分为 core, level 1 和 level 2 三个等级，复杂度逐级提高。因天软数据平台不是关系数据库，且不提供数据的增、删、改和事务服务等原因，部分 core level 的函数也没有支持。目前函数支持情况见表 1：

表 1: TinyODBC 支持的函数

| 函数名                 | Conformance Level | 是否支持 | 说明   |
|---------------------|-------------------|------|--|
| SQLAllocHandle      | Core              | √    | 分配 environment, connection, statement 或 descriptor 句柄      |
| SQLBindCol          | Core              | √    | 绑定列，将输出缓冲区的地址，类型和长度等信息告知驱动程序，SQLFetch 等函数根据这些信息，将提取的列写入缓冲区 |
| SQLBindParameter    | Core              | √    | 绑定参数，TinyODBC 目前不支持参数                                      |
| SQLBrowseConnect    | Level 1           |      |  |
| SQLBulkOperations   | Level 1           |      |  |
| SQLCancel           | Core              |      | 取消正在处理中的 statement，TinyODBC 目前不支持异步处理                      |
| SQLCloseCursor      | Core              |      |  |
| SQLColAttribute     | Core              | √    | 获得列属性  |
| SQLColumnPrivileges | Level 2           |      |  |
| SQLColumns          | Core              | √    | 驱动程序模拟一张伪表   |
| SQLConnect          | Core              | √    | 连接数据库  |
| SQLCopyDesc         | Core              |      |  |
| SQLDataSources      | Core              |      |  |
| SQLDescribeCol      | Core              | √    | 返回列的描述信息   |
| SQLDescribeParam    | Level 2           |      |  |
| SQLDisconnect       | Core              |      |  |
| SQLDriverConnect    | Core              |      | 连接数据库  |
| SQLDrivers          | Core              |      |  |
| SQLEndTran          | Core              |      | 不支持事务服务  |
| SQLExecDirect       | Core              | √    | 执行指定的代码（TSL）   |
| SQLExecute          | Core              | √    |  |
| SQLFetch            | Core              | √    | 获取当前行集   |
| SQLFetchScroll      | Core              | √    | 获取行集   |

| 函数名                 | Conformance Level | 是否支持 | 说明  |
|---------------------|-------------------|------|---|
| SQLForeignKeys      | Level 2           |      |   |
| SQLFreeHandle       | Core              | √    | 释放 environment, connection, statement 或 descriptor 句柄 |
| SQLFreeStmt         | Core              | √    | 释放 statement  |
| SQLGetConnectAttr   | Core              | √    | 获取 connection 属性                                      |
| SQLGetCursorName    | Core              |      |   |
| SQLGetData          | Core              | √    | 获取当前行指定列的数据, 通常用于变长列, 如 binary 类型列                    |
| SQLGetDescField     | Core              | √    | 获取描述符字段   |
| SQLGetDescRec       | Core              |      |   |
| SQLGetDiagField     | Core              | √    | 获取诊断信息  |
| SQLGetDiagRec       | Core              | √    | 获取诊断信息  |
| SQLGetEnvAttr       | Core              |      |   |
| SQLGetFunctions     | Core              | √    | 由驱动管理器(DM)自动支持  |
| SQLGetInfo          | Core              | √    | 获取驱动程序或数据库信息  |
| SQLGetStmtAttr      | Core              | √    | 获取 statement 属性                                       |
| SQLGetTypeInfo      | Core              | √    | 驱动程序提供所支持的数据类型  |
| SQLMoreResults      | Level 1           | √    | 判断是否还有未处理的结果集, TinyODBC 暂不支持多结果集                      |
| SQLNativeSql        | Core              |      |   |
| SQLNumParams        | Core              | √    | 取语句中参数数量  |
| SQLNumResultCols    | Core              | √    | 返回结果集的列数  |
| SQLParamData        | Core              |      |   |
| SQLPrepare          | Core              | √    |   |
| SQLPrimaryKeys      | Level 1           |      |   |
| SQLProcedureColumns | Level 1           |      |   |
| SQLProcedures       | Level 1           |      |   |
| SQLPutData          | Core              |      |   |
| SQLRowCount         | Core              | √    | 返回结果集的行数  |
| SQLSetConnectAttr   | Core              | √    | 设置 connection 属性                                      |
| SQLSetCursorName    | Core              |      |   |
| SQLSetDescField     | Core              | √    | 设置描述符字段   |
| SQLSetDescRec       | Core              |      |   |
| SQLSetEnvAttr       | Core              |      |   |
| SQLSetPos           | Level 1           | √    | 设置游标位置  |
| SQLSetStmtAttr      | Core              | √    | 设置 statement 属性                                       |
| SQLSpecialColumns   | Core              |      |   |
| SQLStatistics       | Core              | √    | 驱动程序模拟一张伪表  |
| SQLTablePrivileges  | Level 2           |      |   |

| 函数名       | Conformance Level | 是否支持 | 说明         |
|-----------|-------------------|------|------------|
| SQLTables | Core              | √    | 驱动程序模拟一张伪表 |

## 2.2 数据类型

TinyODBC 支持的数据类型见表 2:

表 2: TinyODBC 支持的数据类型说明

| ODBC 类型名      | 对应的 TSL 类型    | 客户应用程序可以绑定的 C 语言类型   | 说明                        |
|---------------|---------------|--|---------------------------|
| SQL_BIGINT    | TSL_TINT64    | SQL_BIGINT,SQL_C_SBIGINT,SQL_C_UBIGINT   | 64 位整数                    |
| SQL_BINARY    | TSL_TBINARY   | SQL_BINARY,SQL_VARBINARY,SQL_LONGVARBINARY   | 二进制类型                     |
| SQL_DOUBLE    | TSL_TNUMBER   | SQL_C_DOUBLE   | 64 位浮点数                   |
| SQL_INTEGER   | TSL_TINT      | SQL_C_LONG,SQL_C_SLONG,SQL_C_ULONG   | 32 位整数                    |
| SQL_VARCHAR   | TSL_TSZSTRING | ANSI 字符集:<br>SQL_VARCHAR,SQL_LONGVARCHAR,SQL_C_CHAR<br>UNICODE 字符集:<br>SQL_WVARCHAR,SQL_WLONGVARCHAR,<br>SQL_C_WCHAR | 字符串                       |
| SQL_TIMESTAMP | TSL_TNUMBER   | SQL_TIMESTAMP,SQL_TYPE_TIMESTAMP, 对应<br>TIMESTAMP_STRUCT 结构  | 日期时间, 见<br>2.2.1 节和 2.3 节 |

### 2.2.1 结果集列数据类型的判断规则

不同于传统关系数据库中每列类型唯一, TSL 中矩阵单元格可是任意类型, 因此 TinyODBC 按照下列规则判断结果集的列类型, 并将结果通过 SQLColAttribute 或 SQLDescribeCol 或 SQLGetDescField 等函数告知客户端应用程序。

1. 结果集中首行的类型决定列类型, 对照关系见表 2, 除非发生以下情况;
2. 如果该列后面行的类型与首行类型不同 (TSL\_TNIL 类型除外), 则该列一律设定为 SQL\_VARCHAR 类型。
3. 如果结果集中首行是 TSL\_TNIL 类型, 则由后面行的第一个非 NIL 值决定类型, 如果该列所有行的值都为 NIL, 则该列为 SQL\_VARCHAR 类型。
4. 因 TSL 无内置 datetime 或 timestamp 类型, 如果列符合 DSN 配置中 DATETIMECOLUMNAUTO, DATETIMECOLUMNLIST 和 DATETIMECOLUMNREGEXP 所设定的规则, 则强行将该列设置为 SQL\_TIMESTAMP 类型。参见 2.3 节。

### 2.2.2 列类型强制转换

用户通过 SQLBindCol 等函数绑定列时，可能发生客户端设定的列类型和 TinyODBC 判断的列类型不一致的情况。比如客户通过 SQLExecDirect 执行 “return rand(3,2);”，TinyODBC 会判断三列为 SQL\_DOUBLE 类型，但是用户应用程序可为这三列绑定 SQL\_VARCHAR 类型。这种情况下，TinyODBC 必须进行数据类型转换。但是可能存在 TinyODBC 无法进行类型转换的情况，例如，TinyODBC 判断列为 SQL\_VARCHAR，而用户应用程序将该列绑定到 SQL\_DOUBLE 类型，对于这种情况，TinyODBC 通常返回 NULL 给应用程序。

## 2.3 DSN 配置或 DSN 字符串

表 3: DSN 配置项目

| 项目                   | 名称               | 说明  |
|----------------------|------------------|---|
| DSN                  | 数据源名             | 在 odbcad 中设置，用户根据需要自行设置   |
| HOST                 | 主机名              | IP 地址或域名，例：tsl.tinysoft.com.cn  |
| PORT                 | 端口号              | 例：443   |
| USERNAME             | 用户名              |   |
| PASSWORD             | 密码               |   |
| DATETIMECOLUMNAUTO   | 自动判断是否日期时间型      | 如果为‘Y’则 TinyODBC 对所有行类型为 TSL_TNUMBER 的列进行扫描，如果所有行的值均在 (33208,73051) 范围内，则判断该列为日期时间型           |
| DATETIMECOLUMNLIST   | 时间日期型列名列表        | 以半角逗号 “,” 分隔的列名或列号列表，如 “col1, 2” 将列名为 “col1” 和 “2”（或第 2 列）设定为日期时间型，且大小写不敏感，列号从第 0 列起。         |
| DATETIMECOLUMNREGEXP | 时间日期型列名正则表达式     | 正则表达式，当列名能够匹配该正则表达式时，则判断该列为日期时间型，例如正则式 “\w*_Datetime” 将所有列名后缀为 “_Datetime” 的列视为日期时间型，且大小写不敏感。 |
| TASKADMINROW         | TaskAdmin 指令返回结果 | TaskAdmin 指令返回结果是字符串，如果选择 “1 行”，则返回字符串都在一行中，如果选择 “多行”，则驱动程序将每行返回一行字符串。                        |

## 3 TinyODBC 驱动程序安装

### 3.1 驱动程序的安装

下面以 Windows 10 专业版为例说明 TinyODBC 驱动程序的安装方法。TinyODBC 的发布方式分为两种，一种是独立软件安装包发布，另一种是随天软金融分析.NET 发布。下面分别加以介绍：

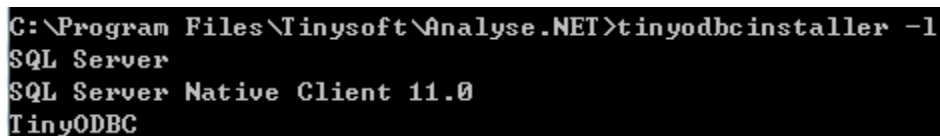
#### 3.1.1 独立软件安装包

选择所需要的 32 位或 64 位 TinyODBC 软件安装包进行安装。安装程序主要完成的工作是：复制驱动程序，注册驱动程序，并将安装目录添加到系统环境变量 Path 中。该安装包包含 TinyODBC 驱动程序所需所有文件，不依赖于天软金融分析.NET 客户端。

### 3.1.2 随天软金融分析.NET 平台安装

这种方式下，TinyODBC 驱动程序有关文件均存放在天软金融分析.NET 平台安装的安装目录中，安装和配置步骤是：

1. 以**管理员**身份运行 cmd.exe。
2. 将当前目录设置为天软金融分析.NET 平台的安装目录，64 位版通常是 C:\Program Files\Tinysoft\Analyse.NET，32 位版通常是 C:\Program Files (x86)\Tinysoft\Analyse.NET。
3. 检查当前目录下是否包含 TinyODBC.dll 和 TinyODBCInstaller.exe 文件。
4. 注册驱动程序，以 64 位版为例：TinyODBCInstaller -i "C:\Program Files\Tinysoft\Analyse.NET" TinyODBC.dll。
5. 执行命令：TinyODBCInstaller -l



```
C:\Program Files\Tinysoft\Analyse.NET>tinyodbcinstaller -l
SQL Server
SQL Server Native Client 11.0
TinyODBC
```

如显示类似上图中所示，有 TinyODBC 这一行，说明驱动已注册成功。

6. 建议将天软金融分析.NET 平台的安装目录设置到系统环境变量的 Path 中。

**注意：**无论采用那种方式安装，为保证应用程序能正确地获得系统环境变量，请在安装完成后关闭所有应用程序，并再次打开，或者重新启动计算机。

## 3.2 DSN 配置

### 3.2.1 DSN 配置程序

配置 DSN 需要使用操作系统中的 DSN 配置程序 odbcad32。但是 32 位或 64 位数据源的配置程序 odbcad32 的启动方式是不同的，如果选择了不恰当的 odbcad32，可能出现无法找到驱动程序的问题，因此这里分类加以说明：

1. 64 位 Windows 下配置 64 位数据源：控制面板->管理工具->数据源（ODBC）
2. 64 位 Windows 下配置 32 位数据源。运行 Windows 安装目录下 SysWOW64\odbcad32.exe
3. 32 位 Windows 下配置 32 位数据源。控制面板->管理工具->数据源（ODBC）

### 3.2.2 配置 DSN

运行 odbcad32，点击“添加”按钮，选择 TinyODBC，点击“完成”按钮后，弹出 TinyODBC 驱动程序的 DSN 配置窗口。参照表 3 的内容填写完成“连接设置”和“Datetime 列设置”后，点击“确定”按钮，驱动程序会将配置好的 DSN 信息写入系统注册表。



## 4 TinyODBC 应用开发实例

### 4.1 C 语言开发实例

使用 C 语言通过 ODBC API 查询数据一般程序是：

1. 分配 ODBC 环境句柄

```
SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &hEnv);
```

2. 分配 ODBC 连接句柄

```
SQLAllocHandle(SQL_HANDLE_DBC, hEnv, &hDbc)
```

3. 连接数据库

```
SQLDriverConnect(hDbc,  
GetDesktopWindow(),  
_T("DSN=t1"),          //这里的 t1 与 3.2.2 节中图上 DSN 对应  
SQL_NTS,  
NULL,
```



```
0,  
NULL,  
SQL_DRIVER_COMPLETE));
```

#### 4. 分配语句句柄

```
SQLAllocHandle(SQL_HANDLE_STMT, hDbc, &hStmt);
```

#### 5. 执行指定的语句

```
SQLExecDirect(hStmt, _T("return rand(5,3);"), SQL_NTS);
```

#### 6. 取得结果集的列数

```
SQLNumResultCols(hStmt, &sNumResults);
```

#### 7. 取得结果集的行数

```
SQLRowCount(hStmt, &cRowCount);
```

#### 8. 逐列获取返回列的类型和列宽等信息。

```
SQLColAttribute(hStmt,  
                iCol,  
                SQL_DESC_CONCISE_TYPE,  
                NULL,  
                0,  
                NULL,  
                &ssType));
```

#### 9. 根据上述信息逐列分配内存缓冲区, 并将缓冲区提交驱动程序进行绑定。

```
SQLBindCol(hStmt,  
iCol,  
SQL_C_TCHAR,  
(SQLPOINTER) wszBuffer,  
(cchDisplay + 1) * sizeof(WCHAR),  
indPtr)
```

#### 10. 调用 SQLFetch 提取数据, 直至返回 SQL\_NO\_DATA\_FOUND。驱动程序将提取的数据写入上一步绑定的缓冲区中, 供应用程序访问。

```
SQLFetch(hStmt);
```

#### 11. 处理完成后, 释放所有资源。

```
SQLFreeHandle(SQL_HANDLE_STMT, hStmt);  
SQLDisconnect(hDbc);  
SQLFreeHandle(SQL_HANDLE_DBC, hDbc);  
SQLFreeHandle(SQL_HANDLE_ENV, hEnv);
```

## 4.2 Excel 开发实例

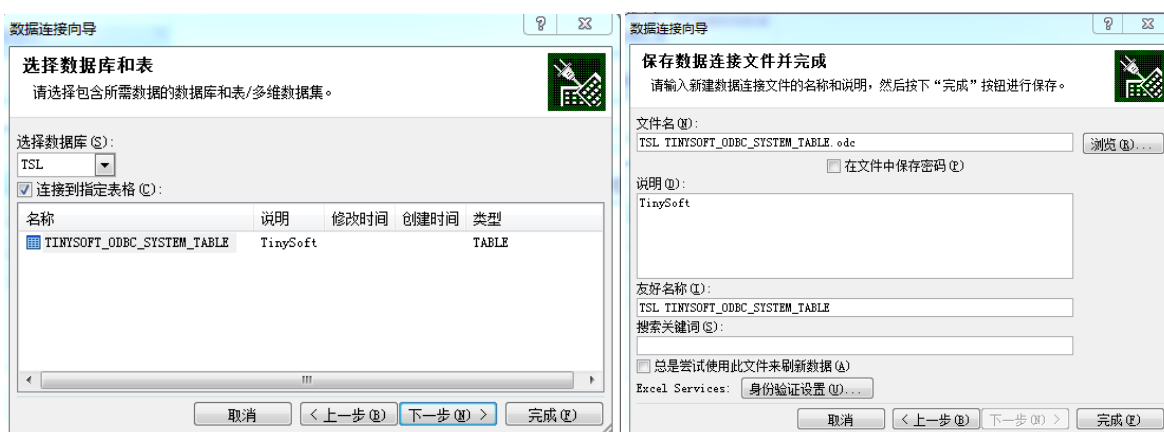
微软 Excel 内置支持支持 ODBC 接口, 且内置 VBA 通过 ADO 可实现 ODBC 接口的访问, Excel 通过 TinyODBC 访问天软数据平台非常方便。要说明的是, 32 位的 Excel 必须使用 32 位的 TinyODBC 驱动

程序。本节以 Excel 2007 为例，说明 Excel 如何通过 TinyODBC 访问天软数据平台。下面的实例均以“提取万科 A(SZ000002)截止到今天前 30 个交易日的开盘价和收盘价”为例编写 TSL 语句。

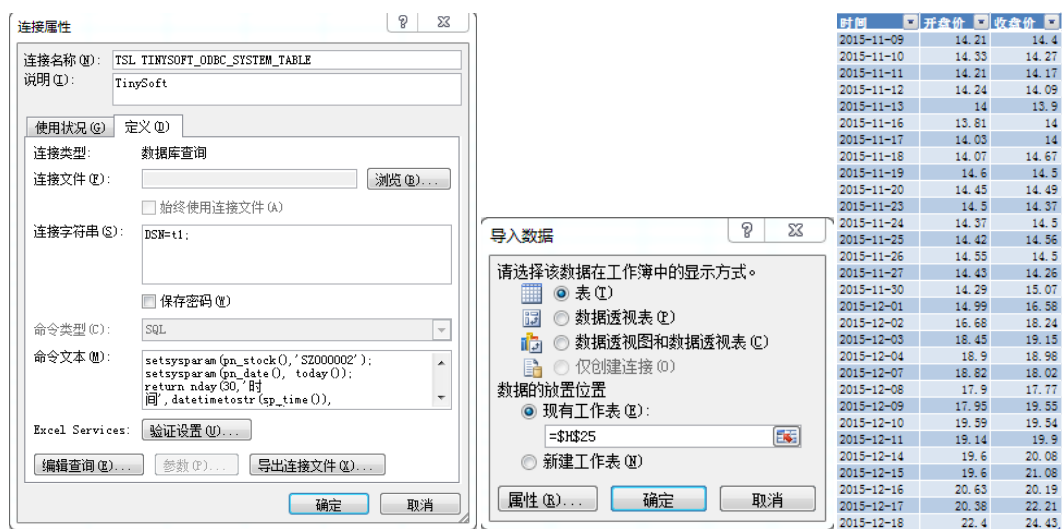
```
setsysparam(pn_stock(), 'SZ000002');
setsysparam(pn_date(), today());
return nday(30, '时间', datetimetoe(str(sp_time()),
    '开盘价', open(),
    '收盘价', close()));
```

#### 4.2.1 通过 ODBC 接口实现 Excel 中数据提取

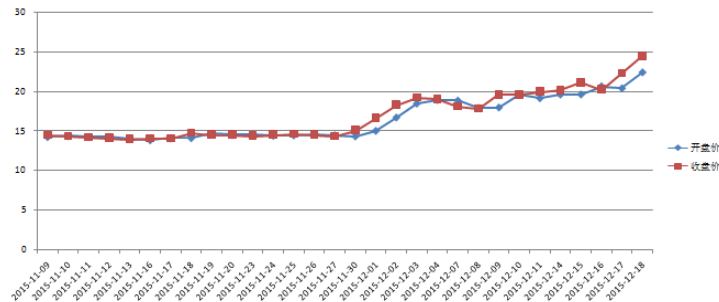
1. 创建数据连接：打开 Excel 文件，菜单中选择“数据”->“连接”->“添加”->“浏览更多”->“+连接到新数据源”->选择“ODBC DSN”->选择为 TinyODBC 所创建的 DSN，点击下一步->出现如下左图所示窗口，点击下一步->出现如下右图所示窗口，为创建的数据连接命名后，点击完成。



2. 编辑查询语句：菜单中选择“数据”->“连接”->选择上一步创建的连接，点击“属性”->点击定义->在“命令文本”输入框中输入 TSL 语句，如下左图所示->点击确定。



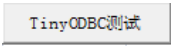
3. 获取数据：菜单中选择“数据”->“现有连接”->选择在上一步中创建的连接，点击“打开”->选择数据在工作表的显示方式“表”，设置数据工作表中显示的位置，如上中图所示，点击“确定”。结果如上右图所示。
4. 也可用按上述方式提取的数据，在 Excel 中制作图表。



5. 如数据发生了变化，可在工作表中选中单元格，单击鼠标右键，选择“刷新”，或通过菜单“数据”->“全部刷新”，对工作表中的数据进行刷新。

## 4.2.2 使用 VBA , ADO 实现 Excel 中数据提取

### 4.2.2.1 无参数数据提取

1. 打开 Excel 文件，菜单选择“开发工具”->“插入”->“命令按钮（ActiveX 控件）”->在工作表中画出按钮的形状->在按钮的属性窗口中修改按钮的标题。 
2. 菜单选择“开发工具”->“设计模式”->对准按钮单击鼠标右键->选择“查看代码”，进入 Visual Basic 开发窗口->选择所设计按钮的 Click 事件代码编辑窗口后，输入下列代码：

```
'变量
Dim conn As New ADODB.Connection, strConn As String, strSQL As String, rs As New ADODB.Recordset, i As Integer

'连接服务器
strConn = "Provider=MSDASQL.1;Persist Security Info=False;Data Source=t1"
conn.Open strConn '这里的 t1 与创建的 DSN 相对应

'执行 TSL 语句，生成游标
strSQL = "setsysparam(pn_stock(),'SZ000002');
setsysparam(pn_date(), today()); return nday(30,'时间',datetimetostr(sp_time()), '开盘价',open(), '收盘价',close());"
rs.CursorLocation = adUseClient '使用本地游标
```

```

rs.Open strSQL, conn, adOpenStatic, adLockReadOnly '打开结果集

'显示游标数据
i = 0
With rs
    If Not (.BOF And .EOF) Then '遍历记录集显示在 Excel 工作表中
        Cells(1, 1) = .Fields(0).Name
    Cells(1, 2) = .Fields(1).Name
        Cells(1, 3) = .Fields(2).Name
    For i = 1 To CLng(.RecordCount)
        Cells(i+1, 1) = .Fields(0)
            Cells(i+1, 2) = .Fields(1)
            Cells(i+1, 3) = .Fields(2)
            .MoveNext
        Next
    End If
End With

```

3. 为支持 ADO，应在代码编辑器的“工具”->“引用”中打开 Microsoft ActiveX Date Objects 2.x Library。

4. 退出“设计模式”，点击按钮执行代码，结果见下图。

| 时间         | 收盘价   | 收盘价   |
|------------|-------|-------|
| 2015-11-09 | 14.21 | 14.4  |
| 2015-11-10 | 14.33 | 14.27 |
| 2015-11-11 | 14.21 | 14.17 |
| 2015-11-12 | 14.24 | 14.09 |
| 2015-11-13 | 14    | 13.9  |
| 2015-11-16 | 13.81 | 14    |
| 2015-11-17 | 14.03 | 14    |
| 2015-11-18 | 14.07 | 14.67 |
| 2015-11-19 | 14.6  | 14.5  |
| 2015-11-20 | 14.45 | 14.49 |
| 2015-11-23 | 14.5  | 14.37 |
| 2015-11-24 | 14.37 | 14.5  |
| 2015-11-25 | 14.42 | 14.56 |
| 2015-11-26 | 14.55 | 14.5  |
| 2015-11-27 | 14.43 | 14.26 |
| 2015-11-30 | 14.29 | 15.07 |
| 2015-12-01 | 14.99 | 16.58 |
| 2015-12-02 | 16.68 | 18.24 |
| 2015-12-03 | 18.45 | 19.15 |
| 2015-12-04 | 18.9  | 18.98 |
| 2015-12-07 | 18.82 | 18.02 |
| 2015-12-08 | 17.9  | 17.77 |
| 2015-12-09 | 17.95 | 19.55 |
| 2015-12-10 | 19.59 | 19.54 |
| 2015-12-11 | 19.14 | 19.9  |
| 2015-12-14 | 19.6  | 20.08 |
| 2015-12-15 | 19.6  | 21.08 |
| 2015-12-16 | 20.63 | 20.19 |
| 2015-12-17 | 20.38 | 22.21 |
| 2015-12-18 | 22.4  | 24.43 |

要说明的是上述代码只是示例性质的，完整的代码应补充错误处理功能。

#### 4.2.2.2 参数化数据提取

```
Dim conn As New ADODB.Connection, cmd As New ADODB.Command,
```

```
strConn As String
    Dim rs As ADODB.Recordset
    Dim par As ADODB.Parameter

    '连接数据库
    strConn = "Provider=MSDASQL.1;Persist Security Info=False;Data
Source=t1"
    conn.CursorLocation = adUseClient    '使用本地游标
    conn.Open strConn
    '设置 Command 对象
    cmd.ActiveConnection = conn
    cmd.CommandText = "setsysparam(pn_stock(),{?});
setsysparam(pn_date(), today()); return nday({?}, '时间
', datetimetostr(sp_time()), '开盘价', open(), '收盘价', close());"
    '设置执行参数
    Set par = cmd.CreateParameter(, adChar, adParamInput, 8,
Cells(46, 2))    '取 46 行第 2 列的内容作为证券代码参数
    cmd.Parameters.Append par
    Set par = cmd.CreateParameter(, adInteger, adParamInput, 0,
Cells(47, 2))    '取 47 行第 2 列的内容作为交易日天数参数
    cmd.Parameters.Append par
    '执行
    Set rs = cmd.Execute
    '设置表头
    Cells(50, 1) = rs.Fields(0).Name
    Cells(50, 2) = rs.Fields(1).Name
    Cells(50, 3) = rs.Fields(2).Name
    With rs
        If Not (.BOF And .EOF) Then    '遍历记录集显示在 Excel 工作表中
            For i = 1 To CLng(.RecordCount)
                Cells(i + 50, 1) = .Fields(0)
                Cells(i + 50, 2) = .Fields(1)
                Cells(i + 50, 3) = .Fields(2)
                .MoveNext
            Next
        End If
    End With
```

### 4.3 Visual Basic 开发实例

本节以 Visual Studio 2015 开发环境为例介绍在 Visual Basic 中通过 TinyODBC 驱动程序访问天软数据平台。

1. 在 Visual Studio 中创建一个 Visual Basic 窗口项目。
2. 选择“工具箱”在 Form 中放置一个按钮，并为按钮命名，修改按钮标题。
3. Form 代码编辑器中，在文件头部加入：Imports System.Data.Odbc
4. 选择“工具箱”->“容器”->“数据”->“DataGridView”放置在 Form 中，进行简单布局。
5. 双击放置的按钮进入代码编辑模式，输入下列代码：

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button4.Click
Dim strConnection As String =
"DSN=tl;HOST=tsl.tinysoft.com.cn;PORT=443;USERNAME=tsodbctest;PASSWORD=bl
ablabla;DATETIMECOLUMNAUTO=N;DATETIMECOLUMNLIST=0;" '指定 0 列为日期时间型
Dim sqlConnection1 As New OdbcConnection(strConnection)
Dim dataAdapter As New OdbcDataAdapter
Dim dst As New DataSet
Dim dt As New DataTable

sqlConnection1.Open() '打开连接
Dim sql As String = "setsysparam(pn_stock(),'SZ000002');
setsysparam(pn_date(), today()); return nday(30,'时间',sp_time(), '开盘价
',open(), '收盘价',close());"
Dim cmd As OdbcCommand = New OdbcCommand(sql, sqlConnection1)

dataAdapter.SelectCommand = cmd
dataAdapter.Fill(dst, "info")
dt = dst.Tables("info")
sqlConnection1.Close() '关闭连接
DataGridView1.AutoGenerateColumns = True '自动创建列
DataGridView1.DataSource = dt
End Sub
```

6. 启动应用程序，点击按钮执行，显示如下：



| 时间         | 开盘价   | 收盘价   |
|------------|-------|-------|
| 2015/11/9  | 14.21 | 14.4  |
| 2015/11/10 | 14.33 | 14.27 |
| 2015/11/11 | 14.21 | 14.17 |
| 2015/11/12 | 14.24 | 14.09 |
| 2015/11/13 | 14    | 13.9  |
| 2015/11/16 | 13.81 | 14    |
| 2015/11/17 | 14.03 | 14    |
| 2015/11/18 | 14.07 | 14.67 |
| 2015/11/19 | 14.6  | 14.5  |

## 4.4 MATLAB 开发实例

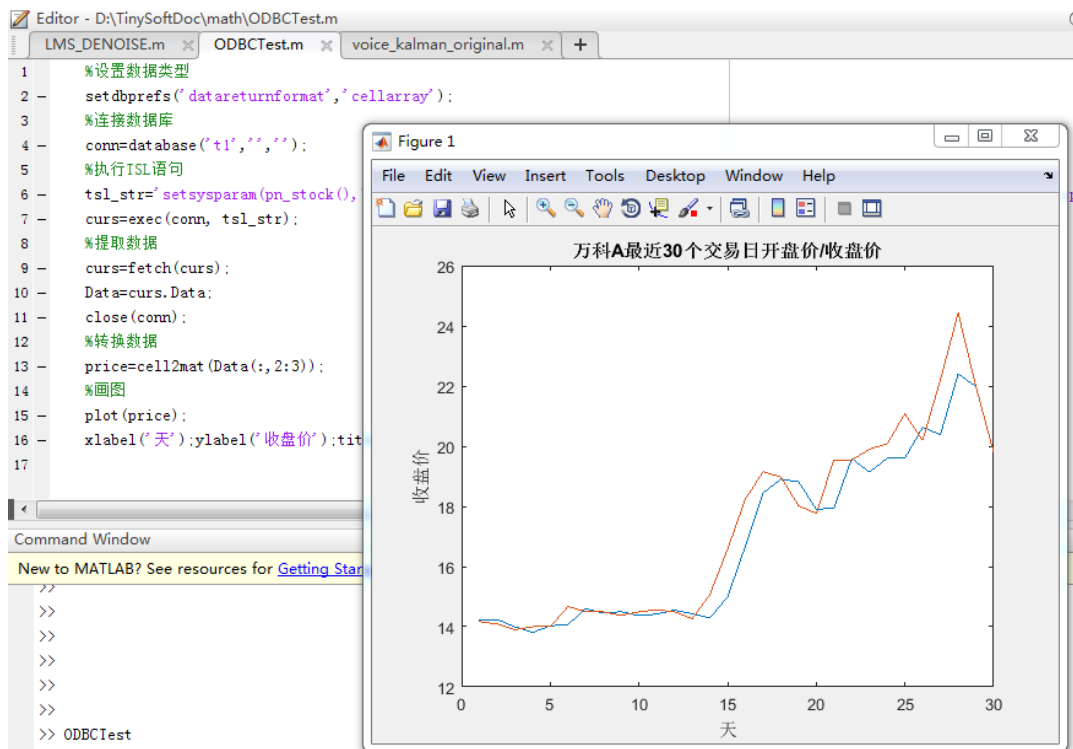
因为 Native ODBC 方式效率高，强烈建议 MATLAB 中使用 Native ODBC 方式调用 TinyODBC。

### 4.4.1 JDBC-ODBC 桥方式

下面是 MATLAB 中以 JDBC-ODBC 桥方式通过 TinyODBC 访问天软平台示例：

```
%设置数据类型
setdbprefs('datareturnformat','cellarray');
%连接数据库
conn=database('t1','','');
%执行 TSL 语句
tsl_str='setsysparam(pn_stock(),"SZ000002");
setsysparam(pn_date(), today()); return nday(30,"时间
",datetimetostr(sp_time()), "开盘价",open(), "收盘价",close());';
curs=exec(conn, tsl_str);
%提取数据
curs=fetch(curs);
Data=curs.Data;
close(curs);
close(conn);
%转换数据
price=cell2mat(Data(:,2:3));
%画图
plot(price);
xlabel('天');ylabel('收盘价');title('万科 A 最近 30 个交易日开盘价/收盘价
');
```

运行结果如下图：



#### 4.4.2 Native ODBC 方式

下面是 MATLAB（2015 及更高版）中以 Native ODBC 方式通过 TinyODBC 访问天软平台示例：

```

%设置数据类型
setdbprefs('datareturnformat','cellarray');
%matlab 默认每次 fetch 取 100000 行，下面两条语句可以调整每次取多少行
%setdbprefs('FetchInBatches','yes')
%setdbprefs('FetchBatchSize','2')
%连接数据库
conn=database.ODBCConnection('t1','','');
%执行 TSL 语句
tsl_str='setsysparam(pn_stock(),"SZ000002"); setsysparam(pn_date(),
today()); return nday(1000,"时间",datetimetoe(str(sp_time()), "开盘价
",open(), "收盘价",close()));';
curs=exec(conn, tsl_str);
%提取数据
curs=fetch(curs);
Data=curs.Data;
close(curs);
close(conn);
%转换数据
price=cell2mat(Data(:,2:3));

```



```
%画图
plot(price);
xlabel('天');ylabel('收盘价');title('万科 A 最近 30 个交易日开盘价/收盘价');
```

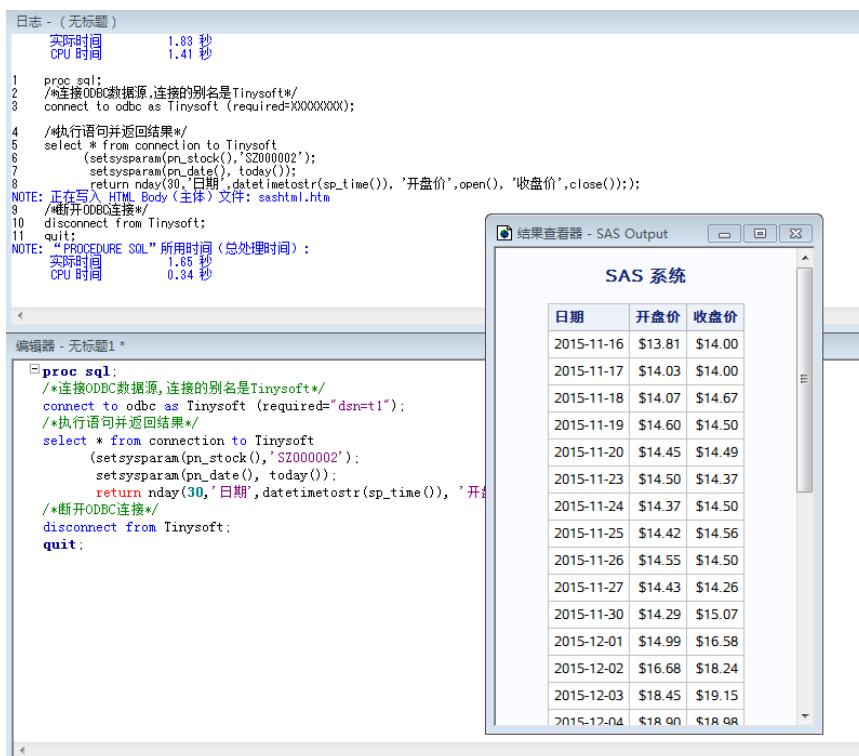
说明：目前 MATLAB 不支持 ODBC 参数调用。

## 4.5 SAS 开发实例

下面是 SAS 下用 Pass-Through 模式通过 TinyODBC 访问天软平台：

```
procsql;
/*连接 ODBC 数据源,连接的别名是 Tinysoft*/
connectto odbc as Tinysoft (required="dsn=t1");
/*执行语句并返回结果*/
select * from connection to Tinysoft
      (setsysparam(pn_stock(),'SZ000002');
       setsysparam(pn_date(), today()));
return nday(30,'日期',datetimetoepr(sp_time()), '开盘价',open(), '
收盘价',close()););
/*断开 ODBC 连接*/
disconnectfrom Tinysoft;
quit;
```

运行结果如下图：



## 4.6 R 开发实例

下面是 R 用 TinyODBC 访问天软平台:

```

library(RODBC)
channel<-odbcConnect("t1")
sqlstr="setsysparam(pn_stock(),'SZ000002');
setsysparam(pn_date(), today()); return nday(30,'日期
',datetimetoepr(sp_time()), '开盘价',open(), '收盘价',close());"
t=sqlQuery(channel, sqlstr)
odbcClose(channel)

```

说明:

1. 目前 RODBC 不支持 ODBC 参数调用。
2. 数据访问完成后请立即使用 odbcClose 函数释放连接, 请勿使用 odbcCloseAll 函数, 因这个函数使用受限。

运行结果如下:

```
> library(RODBC);
> channel<-odbcConnect("t1");
> sqlstr="setsysparam(pn_stock(),'SZ000002'); setsysparam(pn_date(), today())$
> t=sqlQuery(channel, sqlstr);
> t
```

|    | 日期         | 开盘价   | 收盘价   |
|----|------------|-------|-------|
| 1  | 2015-12-03 | 18.45 | 19.15 |
| 2  | 2015-12-04 | 18.90 | 18.98 |
| 3  | 2015-12-07 | 18.82 | 18.02 |
| 4  | 2015-12-08 | 17.90 | 17.77 |
| 5  | 2015-12-09 | 17.95 | 19.55 |
| 6  | 2015-12-10 | 19.59 | 19.54 |
| 7  | 2015-12-11 | 19.14 | 19.90 |
| 8  | 2015-12-14 | 19.60 | 20.08 |
| 9  | 2015-12-15 | 19.60 | 21.08 |
| 10 | 2015-12-16 | 20.63 | 20.19 |
| 11 | 2015-12-17 | 20.38 | 22.21 |
| 12 | 2015-12-18 | 22.40 | 24.43 |
| 13 | 2016-07-04 | 21.99 | 21.99 |
| 14 | 2016-07-05 | 19.79 | 19.79 |
| 15 | 2016-07-06 | 19.10 | 19.80 |
| 16 | 2016-07-07 | 19.10 | 18.82 |
| 17 | 2016-07-08 | 18.60 | 18.75 |
| 18 | 2016-07-11 | 18.50 | 18.27 |
| 19 | 2016-07-12 | 18.27 | 18.12 |
| 20 | 2016-07-13 | 18.00 | 18.32 |
| 21 | 2016-07-14 | 18.14 | 17.96 |
| 22 | 2016-07-15 | 17.90 | 17.89 |
| 23 | 2016-07-18 | 17.84 | 17.43 |
| 24 | 2016-07-19 | 17.39 | 17.11 |
| 25 | 2016-07-20 | 16.89 | 17.08 |
| 26 | 2016-07-21 | 16.91 | 17.02 |
| 27 | 2016-07-22 | 17.17 | 17.39 |
| 28 | 2016-07-25 | 17.26 | 17.70 |
| 29 | 2016-07-26 | 17.60 | 17.78 |
| 30 | 2016-07-27 | 17.78 | 17.42 |

```
> |
```

## 4.7 Python 开发实例

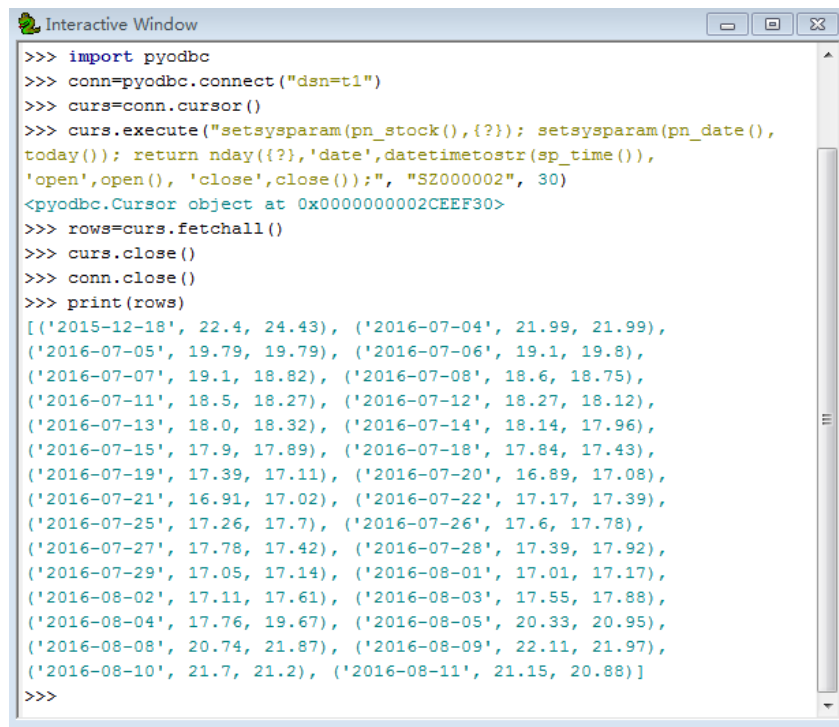
### 4.7.1.1 参数调用

下面是 Python pyodbc 使用 TinyODBC 参数调用访问天软平台：

```
import pyodbc
conn=pyodbc.connect("dsn=t1")
curs=conn.cursor()
curs.execute("setsysparam(pn_stock(),{?}); setsysparam(pn_date(),
today()); return nday({?},'date',datetimetostr(sp_time()),
'open',open(), 'close',close());", "SZ000002", 30)
rows=curs.fetchall()
curs.close()
conn.close()
print(rows)
```

说明：pyodbc 不支持汉字列名。

运行结果如下：



```

Interactive Window
>>> import pyodbc
>>> conn=pyodbc.connect("dsn=t1")
>>> curs=conn.cursor()
>>> curs.execute("setsysparam(pn_stock(),{?}); setsysparam(pn_date(),
today()); return nday({?}, 'date', datetimetostr(sp_time()),
'open', open(), 'close', close());", "S2000002", 30)
<pyodbc.Cursor object at 0x0000000002CEE30>
>>> rows=curs.fetchall()
>>> curs.close()
>>> conn.close()
>>> print(rows)
[('2015-12-18', 22.4, 24.43), ('2016-07-04', 21.99, 21.99),
('2016-07-05', 19.79, 19.79), ('2016-07-06', 19.1, 19.8),
('2016-07-07', 19.1, 18.82), ('2016-07-08', 18.6, 18.75),
('2016-07-11', 18.5, 18.27), ('2016-07-12', 18.27, 18.12),
('2016-07-13', 18.0, 18.32), ('2016-07-14', 18.14, 17.96),
('2016-07-15', 17.9, 17.89), ('2016-07-18', 17.84, 17.43),
('2016-07-19', 17.39, 17.11), ('2016-07-20', 16.89, 17.08),
('2016-07-21', 16.91, 17.02), ('2016-07-22', 17.17, 17.39),
('2016-07-25', 17.26, 17.7), ('2016-07-26', 17.6, 17.78),
('2016-07-27', 17.78, 17.42), ('2016-07-28', 17.39, 17.92),
('2016-07-29', 17.05, 17.14), ('2016-08-01', 17.01, 17.17),
('2016-08-02', 17.11, 17.61), ('2016-08-03', 17.55, 17.88),
('2016-08-04', 17.76, 19.67), ('2016-08-05', 20.33, 20.95),
('2016-08-08', 20.74, 21.87), ('2016-08-09', 22.11, 21.97),
('2016-08-10', 21.7, 21.2), ('2016-08-11', 21.15, 20.88)]
>>>

```

#### 4.7.1.2 TaskAdmin 调用

查询在线用户信息

```

import pyodbc
conn=pyodbc.connect("dsn=t1")
curs=conn.cursor()
curs.execute("taskadmin ou")
rows=curs.fetchall()
curs.close()
conn.close()
print(rows)

```

## 4.8 使用 JDBC-ODBC 桥访问天软平台

用户可使用 JDBC-ODBC 桥的方式在 Java 平台下通过 TinyODBC 访问天软平台。要说明的是 Oracle 公司从 Java 8 起不再支持 JDBC-ODBC 桥，另外目前这种方式只支持 Windows 平台。

下面是实例代码，如果用户使用 Java 中间件访问天软平台，可参考代码中 dbURL 部分进行设置。

```
//测试使用 JDBC 访问天软 ODBC
```

```
import java.sql.*;
```

```
public class JDBCtest {
    public static void main(String args[]) {
        ResultSet rs = null;
        PreparedStatement pstmt = null;
        Statement stmt = null;
        Connection conn=null;
        try {
            String driverName = "sun.jdbc.odbc.JdbcOdbcDriver";
            Class.forName(driverName);
            // 设置 URL，连接天软平台
            String dbURL =
"jdbc:odbc:Driver={TinyODBC};HOST=tsl.tinysoft.com.cn;PORT=443;USERNAME=usertest;PASSWOR
D=blabla";

            conn = DriverManager.getConnection(dbURL);
            // 创建一个语句
            stmt = conn.createStatement();
            // SQL 语句
            String sqlStr;
            sqlStr ="setsysparam(pn_stock(),'SZ000002'); ";
            sqlStr += "setsysparam(pn_date(), today()); ";
            sqlStr += "return nday(30,'时间',datetimetostr(sp_time()),'开盘价',open(),'收盘价
';close());";

            // 执行语句
            rs = stmt.executeQuery(sqlStr);
            // 取结果集描述
            ResultSetMetaData rsmd = rs.getMetaData();
            // 取列数
            int cols = rsmd.getColumnCount();
            // 显示列名
            for (int i = 1; i <= cols; i++) {
                System.out.print(rsmd.getColumnName(i));
                System.out.print("\t");
            }
        }
    }
}
```

```
System.out.println("");
// 显示所有结果集
while (rs.next())
{
    // 显示每行中各列
    for (int i = 1; i <= cols; i++) {
        int type = rsmd.getColumnType(i);
        // 根据列类型取各列显示
        switch (type) {
            case Types.DATE:
                System.out.print(rs.getDate(i));
                break;
            case Types.TIME:
                System.out.print(rs.getTime(i));
                break;
            case Types.TIMESTAMP:
                System.out.print(rs.getTimestamp(i));
                break;
            case Types.REAL:
            case Types.FLOAT:
                System.out.print(String.valueOf(rs.getFloat(i)));
                break;
            case Types.DOUBLE:
                System.out.print(String.valueOf(rs.getDouble(i)));
                break;
            case Types.TINYINT:
                System.out.print(String.valueOf(rs.getByte(i)));
                break;
            case Types.SMALLINT:
                System.out.print(String.valueOf(rs.getShort(i)));
                break;
            case Types.INTEGER:
                System.out.print(String.valueOf(rs.getInt(i)));
                break;
```

```
        case Types.BIGINT:
            System.out.print(String.valueOf(rs.getLong(i)));
            break;
        case Types.DECIMAL:
        case Types.NUMERIC:
            System.out.print(rs.getBigDecimal(i).toString());
            break;
        case Types.CHAR:
        case Types.LONGVARCHAR:
        case Types.VARCHAR:
            System.out.print(rs.getString(i));
            break;
        default:
            System.out.println("Unsupported data type!");
            return;
    }
    System.out.print("\t");
}
System.out.println("");
};
}
catch (SQLException ex) {
    System.out.println(ex.getMessage());
    return;
}
catch (Exception ex) {
    ex.printStackTrace();
    return;
}
finally {
    try {
        if (rs != null) {
            rs.close();
        }
        if (stmt != null) {
```

```
        stmt.close();
    }
    if (pstmt != null) {
        pstmt.close();
    }
    if (conn != null) {
        conn.close();
    }
}
catch (SQLException ex) {
    ex.printStackTrace();
}
}
}
```

运行结果如下:

```
D:\TinySoftDoc\Java>javac JDBCTest.java
D:\TinySoftDoc\Java>java JDBCTest
时间 开盘价 收盘价
2016-11-01 25.2 25.79
2016-11-02 25.45 24.9
2016-11-03 24.77 24.87
2016-11-04 24.84 24.59
2016-11-07 24.47 24.16
2016-11-08 24.16 24.22
2016-11-09 24.43 26.3
2016-11-10 26.0 26.56
2016-11-11 26.15 25.99
2016-11-14 26.23 25.7
2016-11-15 25.6 26.94
2016-11-16 26.64 26.68
2016-11-17 26.62 27.0
2016-11-18 27.8 27.72
2016-11-21 27.27 27.4
2016-11-22 27.04 27.3
2016-11-23 27.9 27.22
2016-11-24 27.08 26.93
2016-11-25 26.79 27.2
2016-11-28 27.36 26.38
2016-11-29 26.17 26.16
2016-11-30 27.35 26.98
2016-12-01 26.56 26.97
2016-12-02 26.71 26.45
2016-12-05 25.08 25.5
2016-12-06 25.15 25.38
2016-12-07 25.35 25.14
2016-12-08 25.1 24.74
2016-12-09 24.61 24.81
2016-12-12 24.78 23.79
D:\TinySoftDoc\Java>
```