

连接到 GBase 8s 数据库示例

南大通用数据技术股份有限公司

General Data Technologies Co., Ltd.



版权所有© GBASE 2023

天津总公司：天津市高新区华苑产业园区开华道 22 号普天创新园东塔 20-23 层

电 话：022-58815678

传 真：022-58815679

北京分公司：北京市朝阳区安定路五号院 13 号北投投资大厦 A 座 606-608

电 话：010-88866866

传 真：010-88864556

<http://www.gbase.cn>

E-mail:info@gbase.cn

连接到 GBase 8s 数据库示例，南大通用数据技术股份有限公司
版权所有© GBASE 2023，保留所有权利。

作者：廖晋清

终审者：

版权声明

本文档所涉及的软件著作权、版权和知识产权已依法进行了相关注册、登记，由南大通用数据技术股份有限公司合法拥有，受《中华人民共和国著作权法》、《计算机软件保护条例》、《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的南大通用公司的版权信息由南大通用公司合法拥有，受法律的保护，南大通用公司对本文档可能涉及到的非南大通用公司的信息不承担任何责任。在法律允许的范围内，您可以查阅，并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经南大通用公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将视为侵权，南大通用公司具有依法追究其责任的权利。

本文档中包含的信息如有更新，恕不另行通知。您对本文档的任何问题，可直接向南大通用数据技术股份有限公司告知或查询。

未经本公司明确授予的任何权利均予保留。

通讯方式

南大通用数据技术股份有限公司

中国天津市高新区华苑产业园区开华道 22 号普天创新园东塔 20-23 层

电话：400-013-9696

邮箱：info@gbase.cn

商标声明

GBASE[®] 是南大通用数据技术股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由南大通用公司合法拥有，受法律保护。未经南大通用公司书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯南大通用公司商标权的，南大通用公司将依法追究其法律责任。

目 录

1. 概述	1
2. 客户端软件安装	2
2.1. WINDOWS 下的 JDBC 驱动安装	2
2.2. WINDOWS 下的 CSDK 安装及连接配置	2
2.2.1. CSDK 的安装	2
2.2.2. CSDK 的配置	5
2.2.3. ODBC 的配置	8
2.3. LINUX 下的 JDBC 安装	11
2.4. LINUX 下的 CSDK 安装	12
2.4.1. CSDK 的安装	12
2.4.2. CSDK 的配置	12
2.4.3. ODBC 的配置	13
3. 数据库连接示例	17
3.1. JAVA 连接到数据库	17
3.1.1. JDBC 原生连接	17
3.2. C# 连接到数据库	20
3.2.1. ADO.NET 方式 (.net framework) 连接到数据库-Windows	20
3.2.2. ODBC 方式 (.net framework) 连接到数据库-Windows	25
3.2.3. ODBC 方式 (netcore3) 连接到数据库-Linux	28
3.3. C 连接到数据库	36
3.3.1. ESQCL 方式 (嵌入式 C) 连接到数据库-Linux	36
3.4. PHP 连接到数据库	38
3.4.1. ODBC 方式连接到数据库-Linux	38
3.4.2. PDO_GBASEDBT 方式连接到数据库-Linux	39
3.5. PYTHON3 连接到数据库	42
3.5.1. Pyodbc 方式连接到数据库-Linux	42
3.5.2. Pyodbc 方式连接到数据库-Windows	43
3.5.3. DbtPy 方式连接到数据库-Linux	45
3.5.4. DbtPy 方式连接到数据库-Windows	48
3.5.5. JayDeBeApi (JDBC) 方式连接到数据库	51
3.6. PERL 连接到数据库	52
3.6.1. DBI::GBASEDBT 方式连接到数据库-Linux	52
3.6.2. DBI::ODBC 方式连接到数据库-Linux	55
3.7. GO 连接到数据库	57
3.7.1. ODBC 方式连接到数据库-Linux	57
3.8. C++ 连接到数据库	60
3.8.1. SOCI 方式连接到数据库-Linux	60
3.9. DELPHI 连接到数据库	62
3.9.1. ODBC 方式连接到数据库-Windows	62

1. 概述

本文档以示例的形式介绍 GBase 8s 数据库的连接操作，仅保证使用相同环境下的测试能通过，不同的环境下仅供参考。

示例使用的数据库版本为 GBase 8s V8.8 3.0.0_1 版本，JDBC 版本为 JDBC_3.3.0_3，CSDK 的版本为 clientsdk_3.0.0_1（Linux）/clientsdk_3.0.0_1（Windows）。

示例使用的数据库服务器信息如下：

表 1 示例使用的数据库信息

序号	参数名称	示例参数值	说明信息
1	主机名称	bd.gbasedbt.com 或者 a02.gbasedbt.com	数据库服务器的主机名称或者 IP 地址
2	端口号	9088	数据库服务器使用的端口号
3	数据库服务名称	gbase01	数据库服务名称（DBSERVERNAME）
4	数据库名	utf8	数据库名称（DBNAME）
5	数据库字符集	zh_CN.utf8	数据库字符集（DB_LOCALE）
6	客户端字符集	zh_CN.utf8	客户端字符集（DB_LOCALE）
7	GLU 支持	1	全球语言支持（GL_USEGLU）

涉及到使用的其它组件，参考具体的页面。

第三章使用到的示例代码，存放于 <https://gbasedbt.com/dl/Demo4GBase8s> 目录下。

2. 客户端软件安装

客户端软件主要包括 CSDK, JDBC 两种。在 Linux 或者 Unix 环境下, 和 Windows 环境下分别配置。

2.1. Windows 下的 JDBC 驱动安装

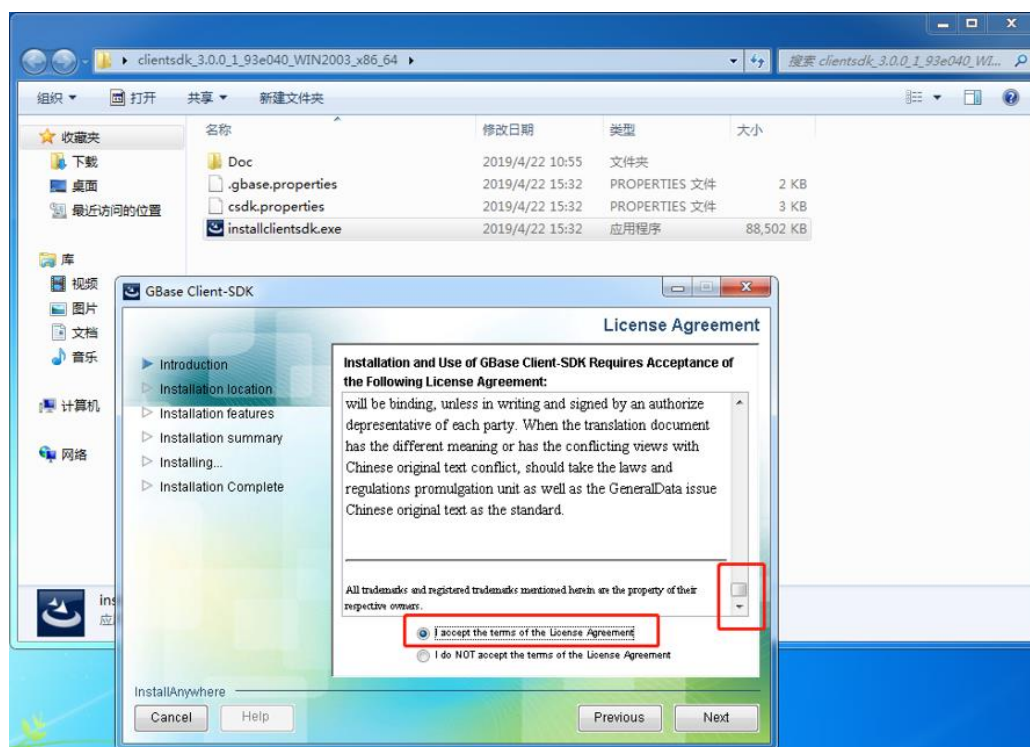
说明: JDBC_3.0.0_1 以上版本直接提供对应的 jar 包, 对应的名称一般形如: gbasedbtjdbc_3.3.0_3.jar, 不再需要安装操作。仅需要随 Server 软件获取即可。

下载地址: https://gbasedbt.com/dl/GBase8s-JDBC/gbasedbtjdbc_3.3.0_3.jar

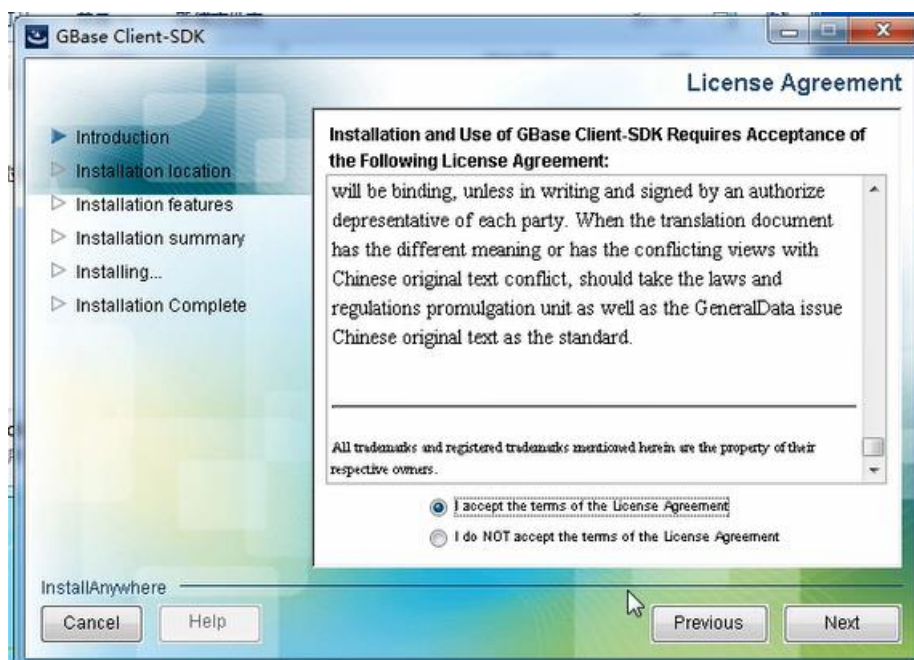
2.2. Windows 下的 CSDK 安装及连接配置

2.2.1. CSDK 的安装

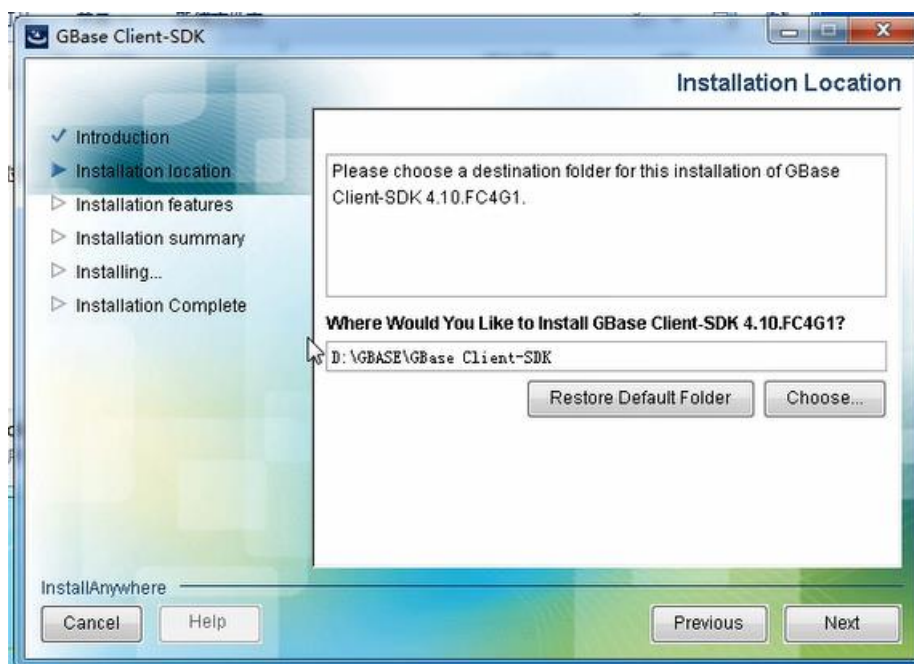
CSDK 需要使用管理员权限进行安装。CSDK 安装包解压, 以管理员身份运行 installclientsdk.exe 开始安装。



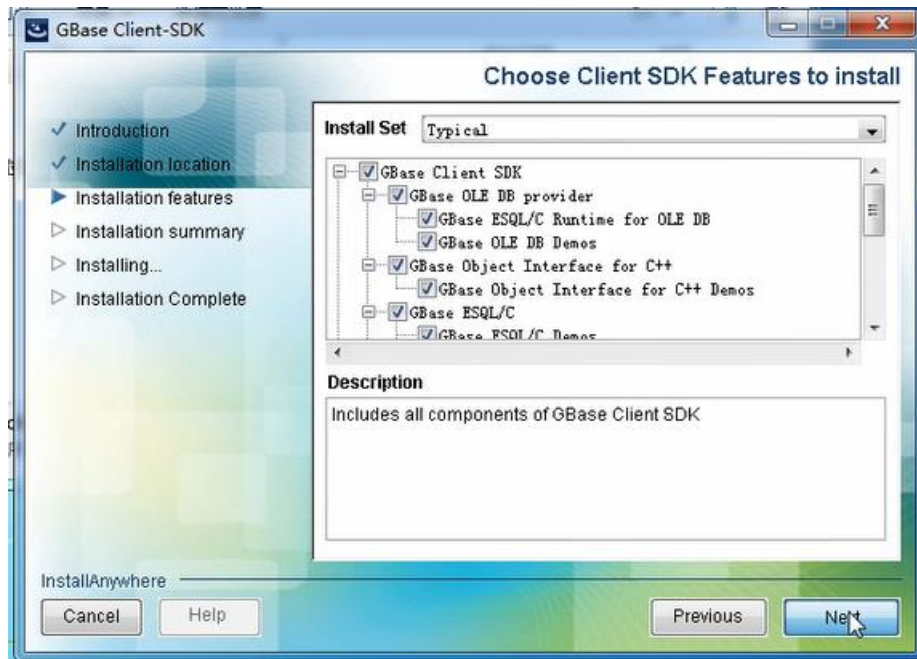
安装介绍协议页面, 需要将滚动条拉到最下面, 才可以点击接受协议的单选框。



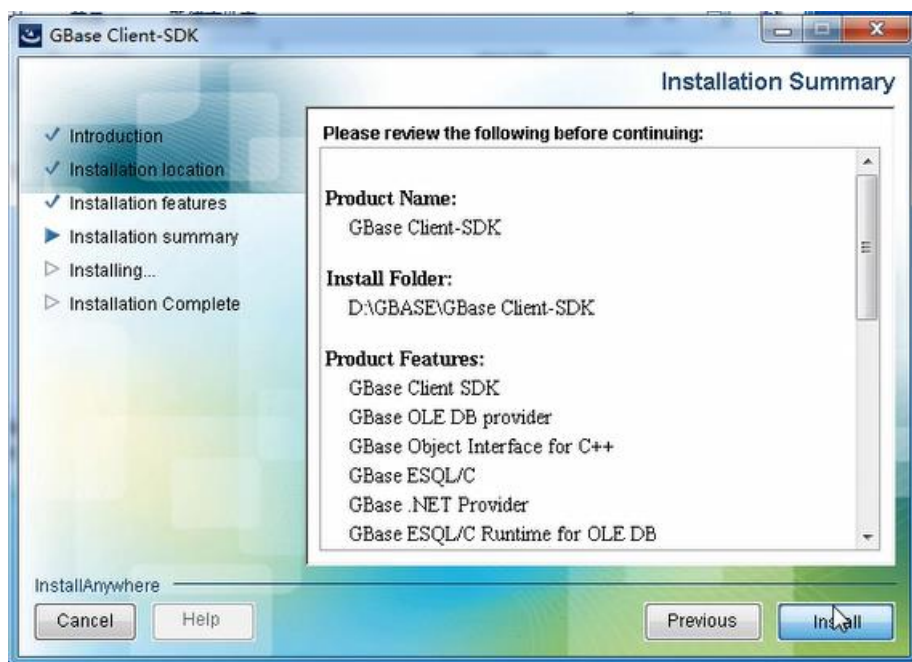
设置安装路径，这里我们使用 D:\GBASE\GBase Client-SDK 目录



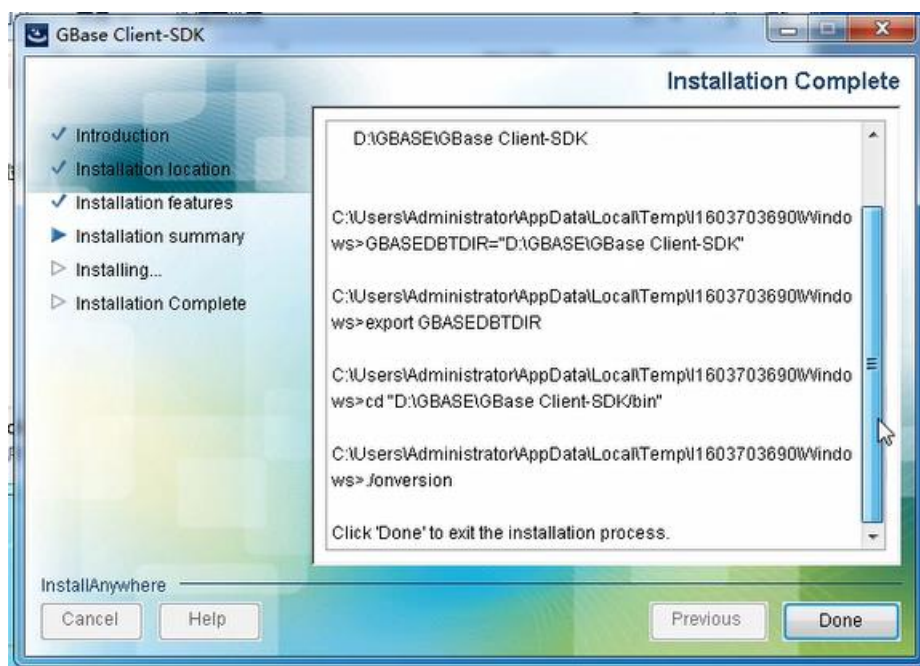
安装的功能浏览，CSDK 默认安装的组件包括：OLEDB 运行环境、C++接口、ESQL/C、.NET 驱动、客户端程序 LIBDMI、ODBC 驱动、通用数据库工具和 GLS。



安装组件预览，执行安装

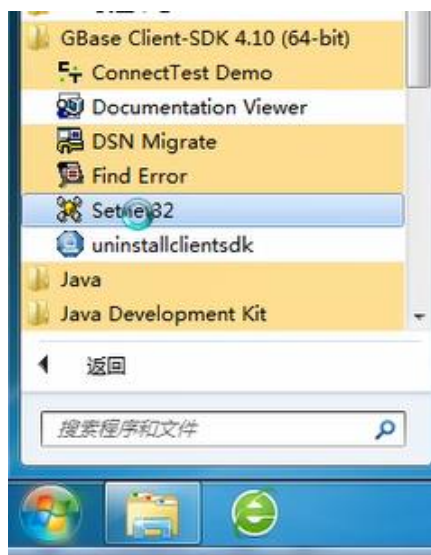


完成安装



2.2.2. CSDK 的配置

安装完成后,需要对客户端连接进行设置。在开始菜单里找到 GBase Client-SDK 4.10(64-bit) 目录,使用 管理员权限运行 打开 Setnet32 程序



在环境 (Environment) 选项卡,根据数据库的实现情况设置:

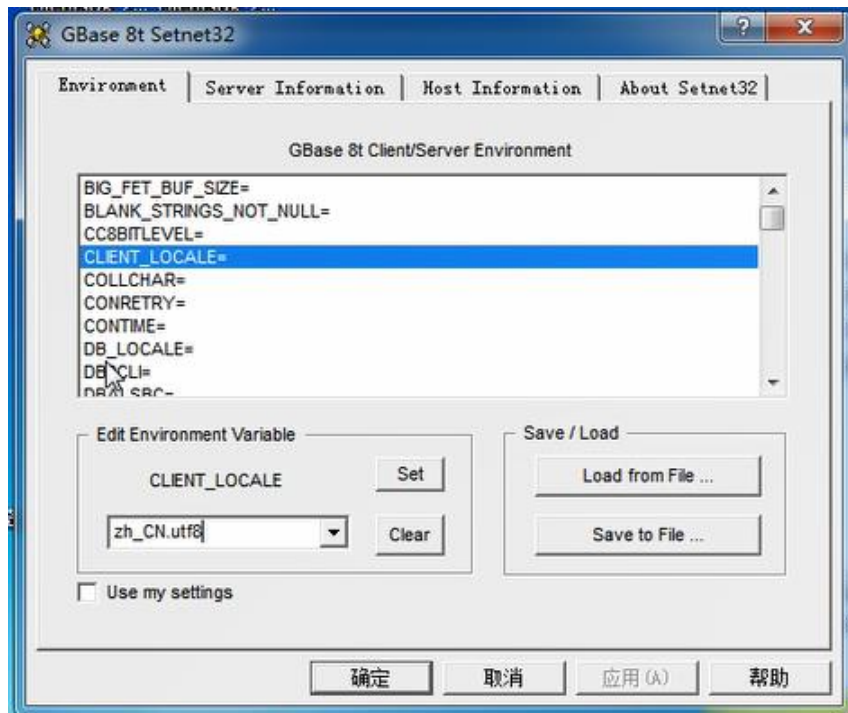
CLIENT_LOCALE: zh_CN.utf8

DB_LOCALE: zh_CN.utf8

GBASEBTDIR: gbase01

GL_USEGLU: 1

等参数。



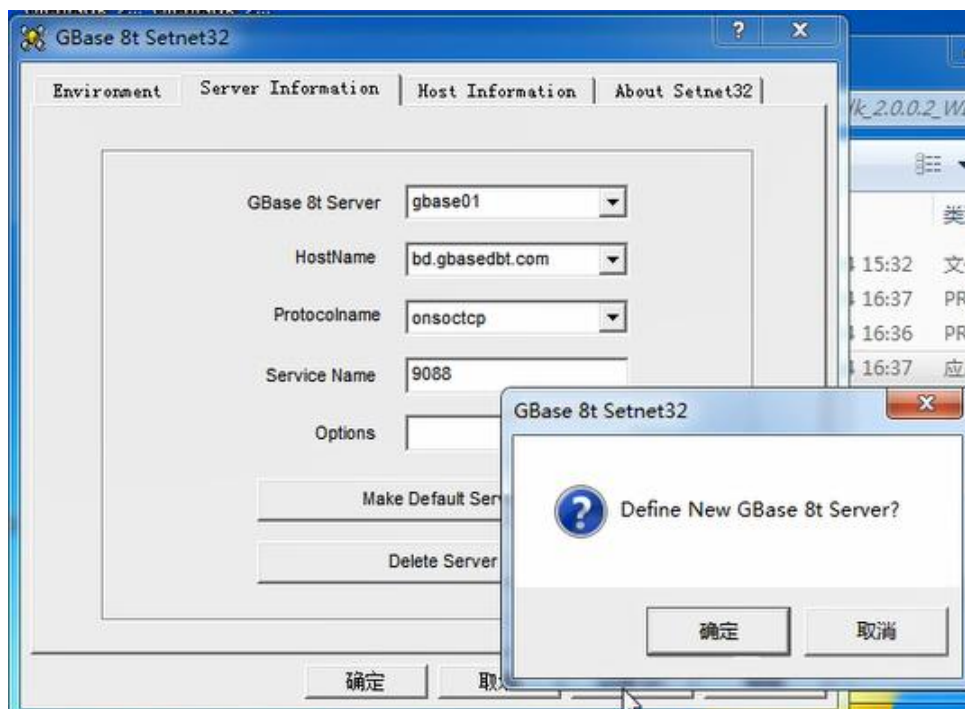
在服务器信息（Server Information）选项卡中设置数据库服务器信息：

HostName: bd.gbasedbt.com 主机名或者域名

Protocolname: onsoctcp 协议名称

Service Name: 9088 数据库使用的端口号

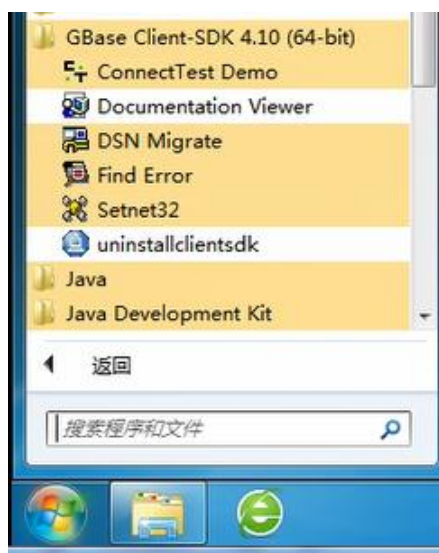
可以设置为默认的数据库服务器



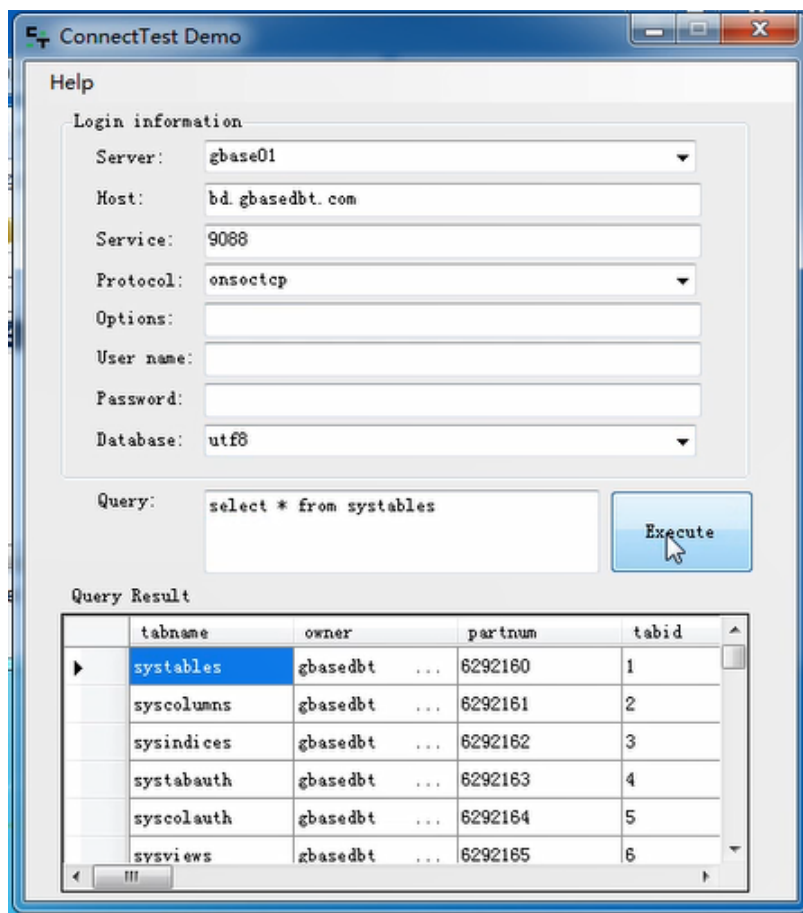
在主机信息（Host Information）选项卡中设置主机信息
设置用户、密码选项及密码



在开始菜单里找到 GBase Client-SDK 4.10(64-bit)目录，使用 管理员权限运行 ConnectTest Demo 进行连接测试

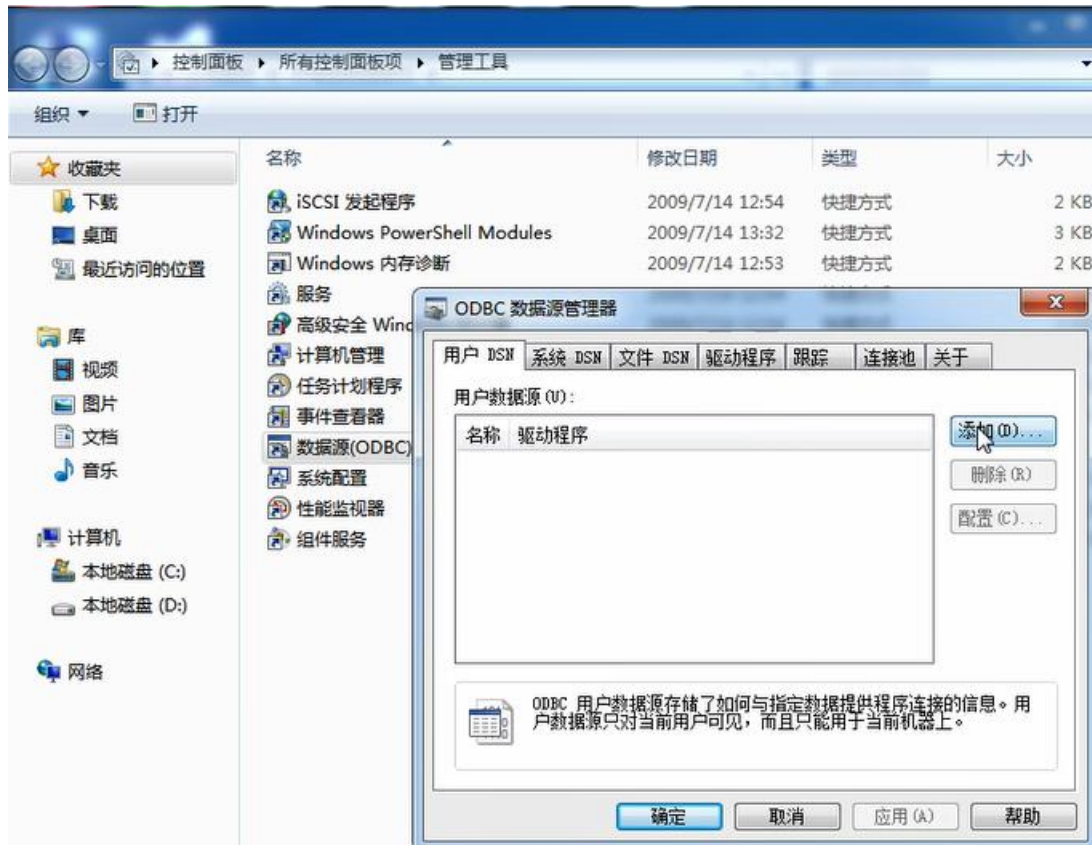


出现的 ConnectTest Demo 界面下，会自动加载配置好的数据库服务器信息，选择 Database（这里我们使用 utf8），填写测试语句 `select * from systables` 进行测试，能获取到数据即为成功。

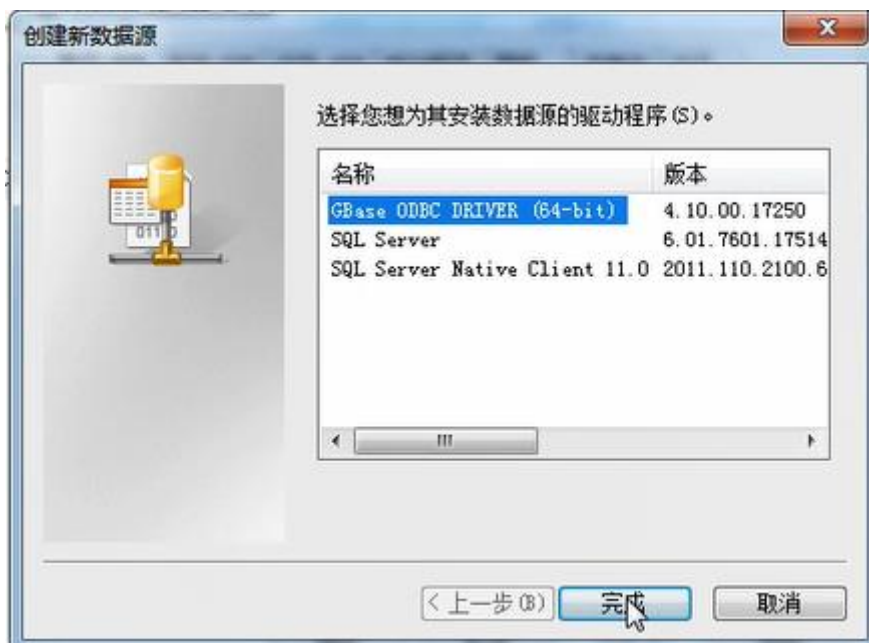


2.2.3. ODBC 的配置

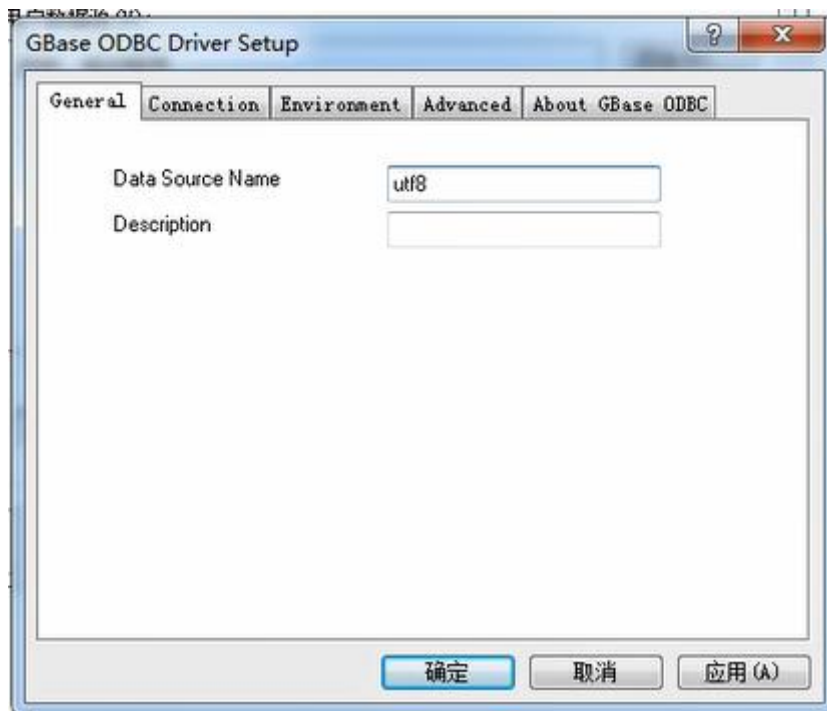
安装了 64 位的 CSDK,则需要配置 64 位的数据源。在 控制面板 – 所有控制面板项 – 管理工具 中配置 数据源 (ODBC) 开启 ODBC 配置,添加 用户或者系统 DSN



使用 GBase ODBC DRIVER (64-bit) 驱动



在通用 (General) 选项卡中, 填写数据源名称: utf8



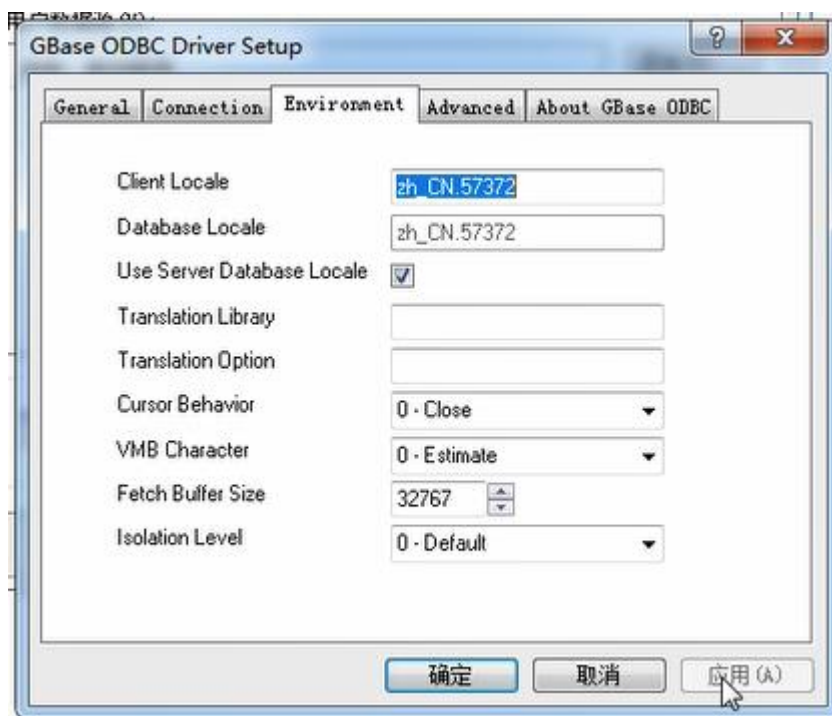
在连接 (Connection) 选项卡中, 会读到现有的 CSDK 连接信息, 只需选择相应的 DataBase Name, 我们这里使用 utf8



然后, 应用和测试连接



测试连接成功后，在环境（Environment）选项卡中，确认环境是否正确。



2.3. Linux 下的 JDBC 安装

说明：JDBC_3.0.0_1 以上版本直接提供对应的 jar 包，对应的名称一般形如：
gbasedbtjdbc_3.3.0_3.jar，不再需要安装操作。仅需要随 Server 软件获取即可。

下载地址：https://gbasedbt.com/dl/GBase8s-JDBC/gbasedbtjdbc_3.3.0_3.jar

2.4. Linux 下的 CSDK 安装

2.4.1. CSDK 的安装

CSDK 需要使用 root 用户权限进行安装。需要预先创建 gbasdbt 用户组及 gbasedbt 用户。

1)、创建 gbasedbt 用户组及 gbasedbt 用户

```
[root@localhost ~]# groupadd -g 1000 gbasedbt
[root@localhost ~]# useradd -g 1000 -d /home/gbasedbt -m -s /bin/bash gbasedbt
```

2)、解压缩 CSDK 软件包

```
[root@localhost ~]# mkdir csdk
[root@localhost ~]# cd csdk/
[root@localhost csdk]# tar -xvf ../clientsdk_3.0.0_1_93e040_RHLE6_x86_64.tar
csdk.properties
doc/
doc/Glsapi_machine_notes_4.10.txt
doc/Libcpp_machine_notes_4.10.txt
doc/ESQLC_machine_notes_4.10.txt
doc/Odbc_machine_notes_4.10.txt
installclientsdk
.gbase.properties
```

3)、执行静默安装，自动完成安装

```
[root@localhost csdk]# ./installclientsdk -i silent \
-DUSER_INSTALL_DIR=/opt/gbase -DLICENSE_ACCEPTED=TRUE
```

表 3 Linux 下 CSDK 安装涉及的参数说明

序号	参数名称	示例参数值	说明信息
1	-i	silent	指定使用静默安装
2	-DUSER_INSTALL_DIR=	/opt/gbase	指定安装目录
3	-DLICENSE_ACCEPTED=	TRUE	指定接受协议

2.4.2. CSDK 的配置

CSDK 安装完成后，需要对客户端连接进行设置。以下使用 gbasedbt 用户来说明。

1)、在用户的目录下的用户环境配置文件.bash_profile 中增加数据库的环境。

根据数据库的实现情况设置：

```
# .bash_profile
export GBASEBTDIR=/opt/gbase
export GBASEBTSERVER=gbase01
export PATH=${GBASEBTDIR}/bin:${PATH}
export LD_LIBRARY_PATH=${GBASEBTDIR}/lib:${GBASEBTDIR}/lib/cli:${GBASEBTDIR}/lib/esql:${LD_LIBRARY_PATH}

export DB_LOCALE=zh_CN.utf8
export CLIENT_LOCALE=zh_CN.utf8
export GL_USEGLU=1
export DBDATE="Y4MD-"
```

2)、修改 GBASEBTSQLHOSTS 配置文件

在\$GBASEBTDIR/etc/目录下创建 sqlhosts (默认的 GBASEBTSQLHOSTS) 配置文件，内容为连接到数据库服务器的信息。

```
# GBASEBTSQLHOSTS
gbase01 onsoctcp bd.gbasedbt.com 9088
```

3)、测试数据库连接

```
[gbasedbt@localhost ~]$ dbaccess - -
> connect to "utf8@gbase01" user "gbasedbt";
  输入密码: <输入用户密码>

已连接。

Elapsed time: 4.978 sec

> select dbservername from dual;

(expression)  gbase01

查询到 1 行。

Elapsed time: 0.312 sec
```

2.4.3. ODBC 的配置

安装了 64 位的 CSDK，则需要配置 64 位的数据源。Linux 下的 ODBC 需要 unixODBC。如果需要对所有用户生效，需要在系统级配置。

1)、确认 unixODBC 已经安装

```
[root@localhost ~]# rpm -qa | grep -i unixODBC
unixODBC-devel-2.3.1-14.el7.x86_64
unixODBC-2.3.1-14.el7.x86_64
```

2)、在/etc/profile 配置文件里增加 CSDK 的配置环境

```
# /etc/profile
# Add for GBase 8s ODBC
export GBASEBTDIR=/opt/gbase
export PATH=${GBASEBTDIR}/bin:${PATH}
export LD_LIBRARY_PATH=${GBASEBTDIR}/lib:${GBASEBTDIR}/lib/cli:${GBASEBTDIR}/lib/esql:${LD_LIBRARY_PATH}

export DB_LOCALE=zh_CN.utf8
export CLIENT_LOCALE=zh_CN.utf8
export GL_USEGLU=1

export ODBCINI=/etc/odbc.ini
```

3)、配置/etc/odbcinst.ini 配置文件，根据 CSDK 环境，配置如下：

```
; /etc/odbcinst.ini
; ODBC Driver for GBase 8s
[GBase ODBC DRIVER]
Driver=/opt/gbase/lib/cli/iclit09b.so
Setup=/opt/gbase/lib/cli/iclit09b.so
APILevel=1
ConnectFunctions=YYY
DriverODBCVer=03.51
FileUsage=0
SQLLevel=1
smProcessPerConnect=Y
```

4)、配置 ODBCINI 配置文件，根据 CSDK 环境，配置如下：

```
[ODBC Data Sources]
utf8=GBase ODBC DRIVER
;
; Define ODBC Database Driver's Below - Driver Configuration Section
;
[utf8]
;Driver=/opt/gbase/lib/cli/iclit09b.so
Driver=GBase ODBC DRIVER
Description=GBase ODBC DRIVER
Database=utf8
LogonID=gbasedbt
pwd=GBase123
Servername=gbase01
CursorBehavior=0
CLIENT_LOCALE=zh_CN.utf8
DB_LOCALE=zh_CN.utf8
GL_USEGLU=1
```

```

TRANSLATIONDLL=/opt/gbase/lib/esql/igo4a304.so
; ISOLATIONLEVEL=1 # 使用该参数 (简写: ISOLVL) 设置默认的隔离级别, 0-5
;
; UNICODE connection Section
;
[ODBC]
; uncomment the below line for UNICODE connection
UNICODE=UCS-2      # 如果需要使用 unicode 连接数据库, 这里需要去除注释, 值改为 UCS-2
;
; Trace file Section
;
Trace=0
TraceFile=/tmp/odbctrace.out
InstallDir=/opt/gbase
TRACEDLL=idmrs09a.so

```

5)、ODBC 连接测试

确认当前用户环境变量中包括 2) 中配置的环境变量, 3) 已经完成。通过 isql 测试 ODBC 配置正确。

特别说明: 有的开发语言使用 **unicode** 方式访问数据库, 这时需要使用 **iusql** 确认 ODBC 可以连接。CentOS 7 默认带的 **unixODBC** 版本为 2.3.1, 需要升级到更高的版本。

```

[root@localhost ~]# env | egrep '(GBASEDBT|ODBCINI)'
GBASEDBTSERVER=gbase01
ODBCINI=/etc/odbc.ini
GBASEDBTDIR=/opt/gbase
[root@localhost ~]# isql -v utf8
+-----+
| Connected! |
|          |
| sql-statement |
| help [tablename] |
| quit      |
|          |
+-----+
SQL> select dbservername from dual;
+-----+
|          |
+-----+
| gbase01  |
+-----+
SQLRowCount returns -1
1 rows fetched
>

```

```
[root@localhost ~]# iusql -v utf8
```

```
+-----+
| Connected! |
|          |
| sql-statement |
| help [tablename] |
| quit |
|          |
+-----+
```

```
SQL> select dbservername from dual;
```

```
+-----+
|          |
+-----+
| gbase01 |
+-----+
```

```
SQLRowCount returns -1
```

```
1 rows fetched
```

3. 数据库连接示例

3.1. Java 连接到数据库

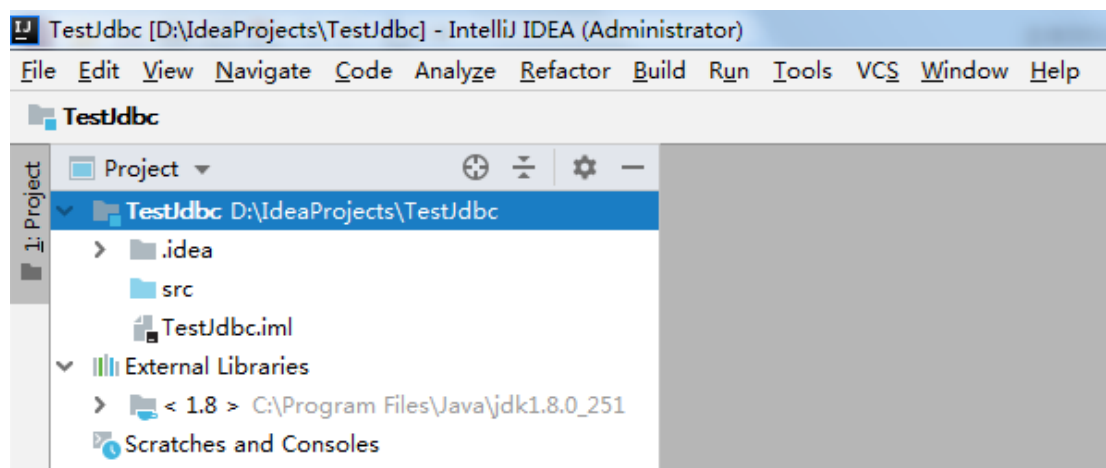
3.1.1. JDBC 原生连接

示例使用 IDEA 2018.3.6，要求 2.1 Windows 下的 JDBC 驱动安装 已经完成。

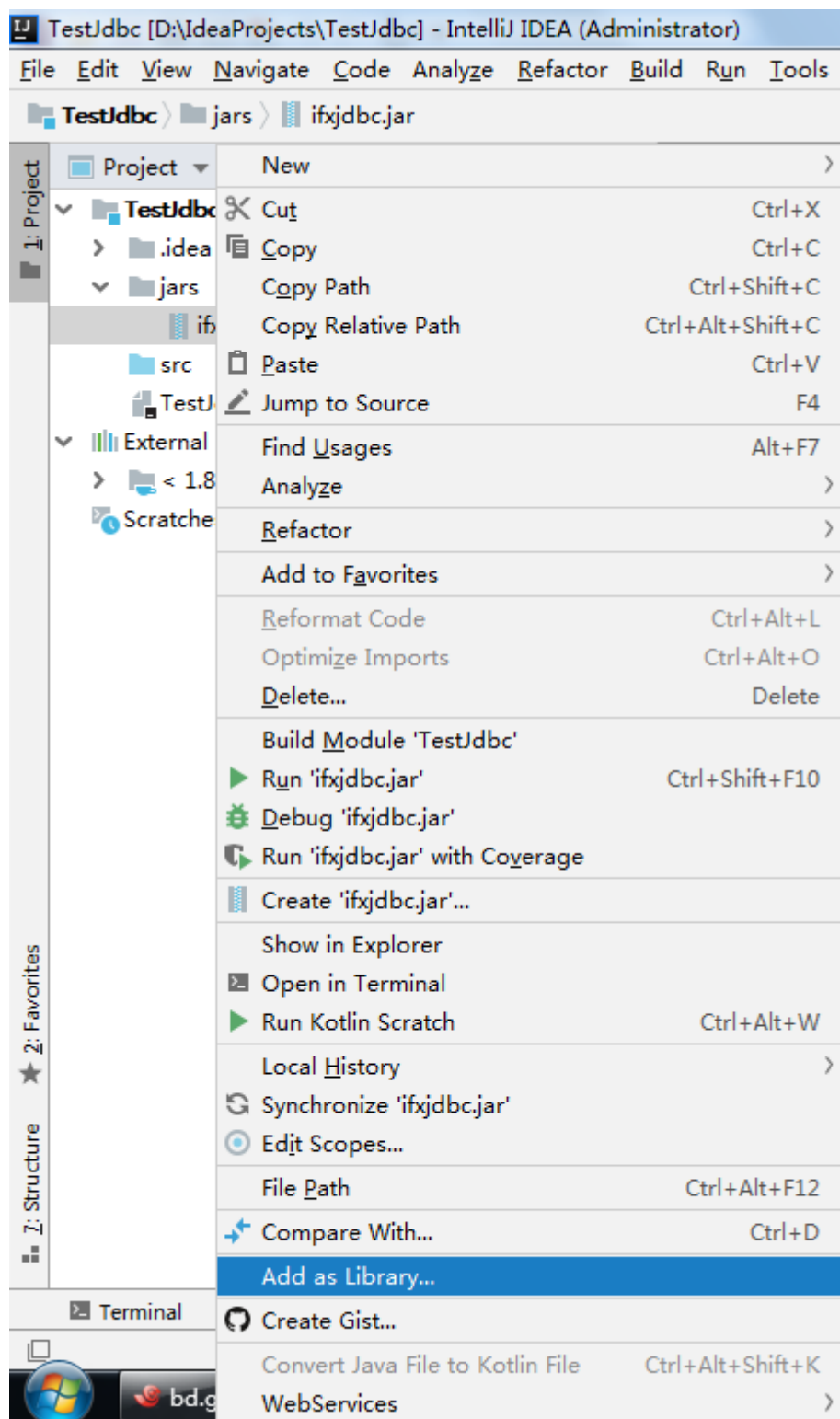
1)、打开 IDEA，File -> New -> Project，创建一个 Java 项目，项目名称为 TestJdbc。

2)、打开的 Java 项目，新建个 lib 目录添加 JDBC 驱动（ifxjdbc.jar，驱动包需跟数据库版本配套）

注：3.0.0 版本之后的驱动包名称一般为 gbasedbtjdbc_3.3.0_3.jar

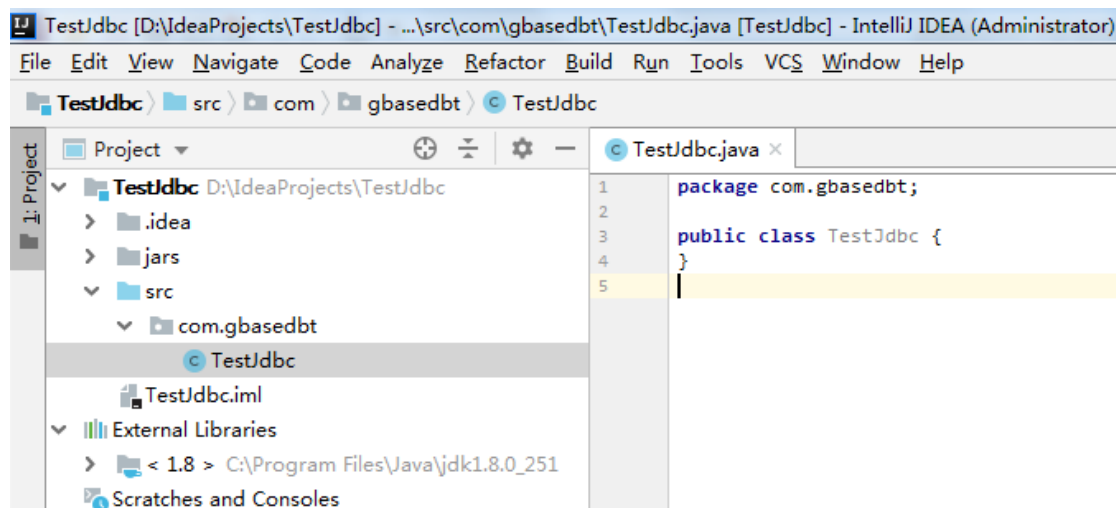


将 ifxjdbc.jar 加入到 Library 中



3)、新建 Java 类

在 src 目录下创建 package (com.gbasedbt)，然后在 package 下创建 java 类 TestJdbc.java



4)、增加 java 代码

将以下示例代码复制到 TestJdbc.java 中

```
package com.gbasedbt;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class TestJdbc {

    public static void main(String[] args){
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;
        String url = "jdbc:gbasedbt-sqli://180.76.182.233:9088/utf8:GBASEDBTSERVER=gbase01;" +
            "DB_LOCALE=zh_CN.utf8;CLIENT_LOCALE=zh_CN.utf8;IFX_LOCK_MODE_WAIT=30";
        String user = "gbasedbt";
        String pass = "GBase123";

        try{
            Class.forName("com.gbasedbt.jdbc.Driver");
            connection = DriverManager.getConnection(url, user, pass);
            statement = connection.createStatement();

            statement.execute("drop table if exists company");

            statement.execute("create table company(coid serial,coname varchar(255),coaddr varchar(255))");

            statement.execute("insert into company values (0,'南大通用','天津市海泰绿色产业基地')");
        }
    }
}
```

```

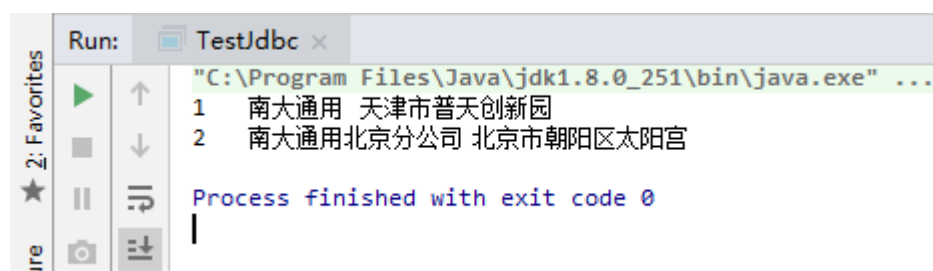
statement.execute("insert into company values (0,'南大通用北京分公司','北京市朝阳区太阳宫')");

statement.execute("update company set coaddr = '天津市普天创新园' where coid = 1");

resultSet = statement.executeQuery("select * from company");
while(resultSet.next()){
    System.out.println(resultSet.getInt("coid") + "\t" +
        resultSet.getString("coname").trim() + "\t" +
        resultSet.getString("coaddr").trim()
    );
}
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

5)、执行 run -> run(TestJdbc) 测试连接到数据库结果



3.2. C# 连接到数据库

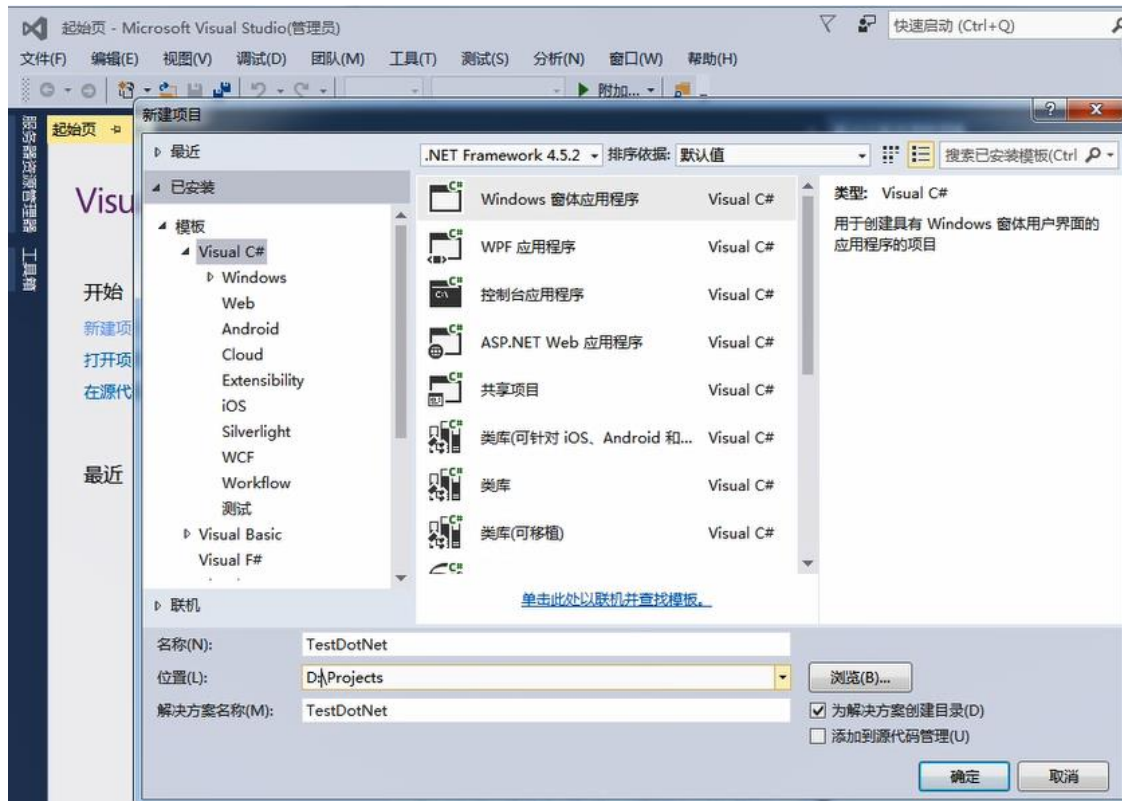
3.2.1. ADO.NET 方式 (.net framework) 连接到数据库-Windows

示例使用 Visual Studio 2015 社区版。要求 2.2.2 CSDK 的配置 已经完成。

1)、打开 Visual Studio, 文件(F) -> 新建(N) -> 项目(P)

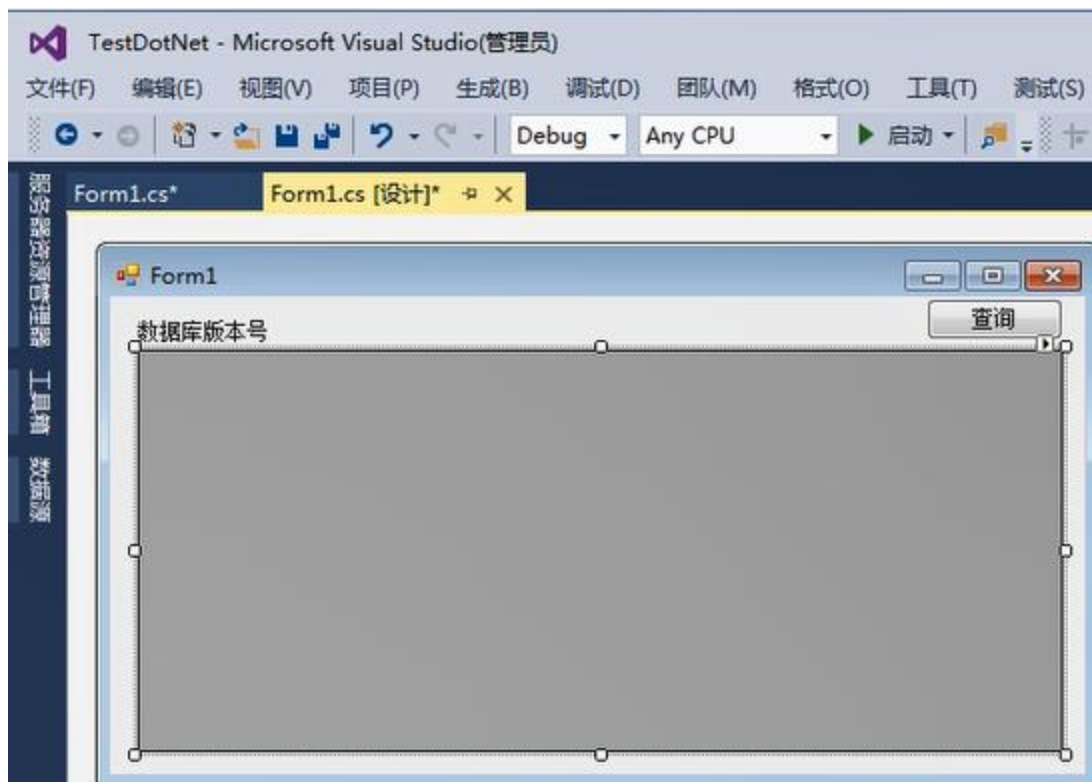
2)、指定编程语言及 .net framework 版本

使用 Visual C#, .NET Framework 4.5.2, 创建 Windows 窗体应用程序, 指定项目名称为 TestDotNet, 位置等



3)、Form1 窗体添加工具，并调整格式

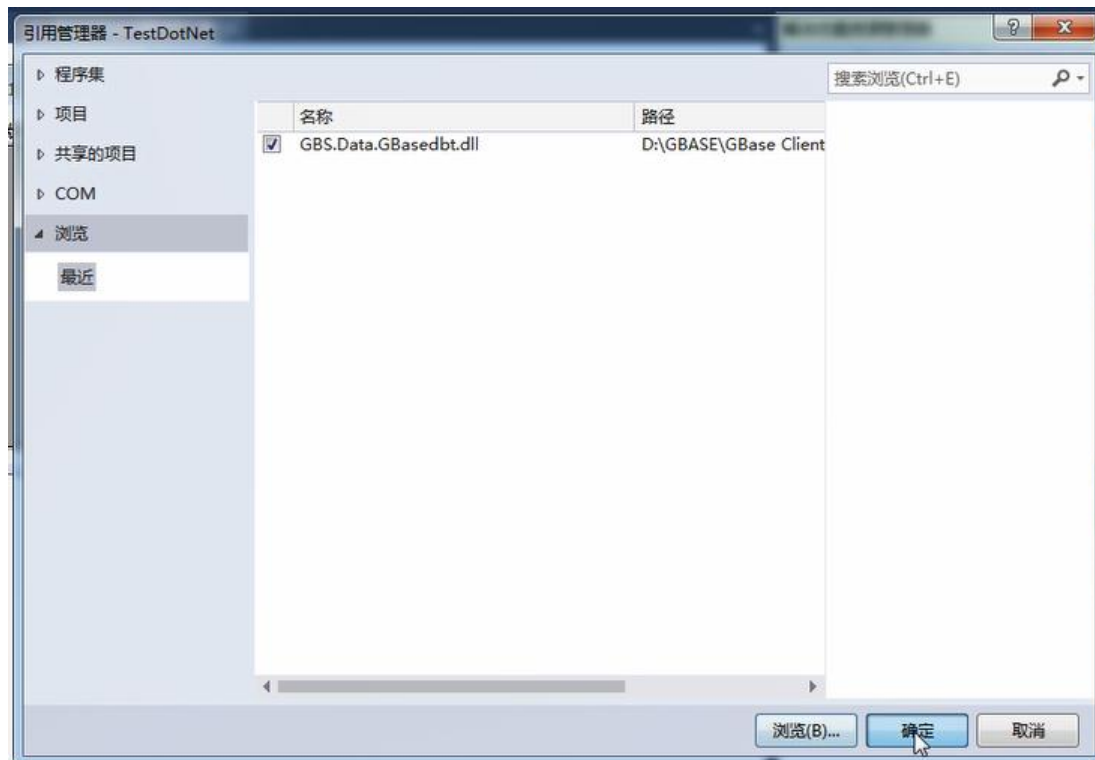
增加一个 Label, 名称为 label1; 一个 dataGridView, 名称为 dataGridView1; 一个 button, 名称为 btnSelect



4)、添加引用 GBS.Data.GBasedbt.dll

在解决方案管理器上的 引用 中 右键添加引用，浏览并增加 GBS.Data.GBasedbt.dll 文件。

示例中的路径：D:\GBASE\GBase Client-SDK\bin\netf40\GBS.Data.GBasedbt.dll



5)、增加 C#代码

将以下示例代码复制到 Form1.cs 中（视需要修改控件名称）

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using GBS.Data.GBasedbt;

namespace TestDotNet
{
    public partial class Form1 : Form
    {
        IfxConnection ifxconn;
        DataSet ds;
```

```
public Form1()
{
    InitializeComponent();

    IfxConnectionStringBuilder build = new IfxConnectionStringBuilder();
    // 以下信息写完整，可以不使用 setnet 配置 sqlhosts
    build.Host = "bd.gbasedbt.com";    // 主机名或者 IP 地址
    build.Protocol = "onsoctcp";      // 数据库使用的协议
    build.Service = "9088";           // 数据库服务器使用的端口号
    build.Server = "gbase01";         // 数据库服务名称
    build.Database = "utf8";          // 数据库名 (DBNAME)
    build.UID = "gbasedbt";           // 用户
    build.Pwd = "GBase123";           // 密码
    build.DbLocale = "zh_CN.utf8";    // 数据库字符集
    build.ClientLocale = "zh_CN.utf8"; // 客户端字符集
    build.PersistSecurityInfo = true; // 保存安全信息
    ifxconn = new IfxConnection(build.ConnectionString);
    ifxconn.Open();

    using (IfxCommand ifxcmd = ifxconn.CreateCommand())
    {
        ifxcmd.CommandText = "drop table if exists company";
        ifxcmd.ExecuteNonQuery();

        ifxcmd.CommandText = "create table company(coid serial, coname varchar(255), coaddr varchar(255))";
        ifxcmd.ExecuteNonQuery();

        ifxcmd.CommandText = "insert into company values (0, '南大通用', '天津市海泰绿色产业基地')";
        ifxcmd.ExecuteNonQuery();

        ifxcmd.CommandText = "insert into company values (0, '南大通用北京分公司', '北京市朝阳区太阳宫')";
        ifxcmd.ExecuteNonQuery();

        ifxcmd.CommandText = "update company set coaddr = '天津市普天创新园' where coid = 1";
        ifxcmd.ExecuteNonQuery();

        ifxcmd.CommandText = "select dbinfo('version', 'full') from dual";
        IfxDataReader dr = ifxcmd.ExecuteReader();
        if (dr.Read())
        {
            this.labell.Text = "数据库版本号为: " + dr[0];
        }
    }
}
```

```
}

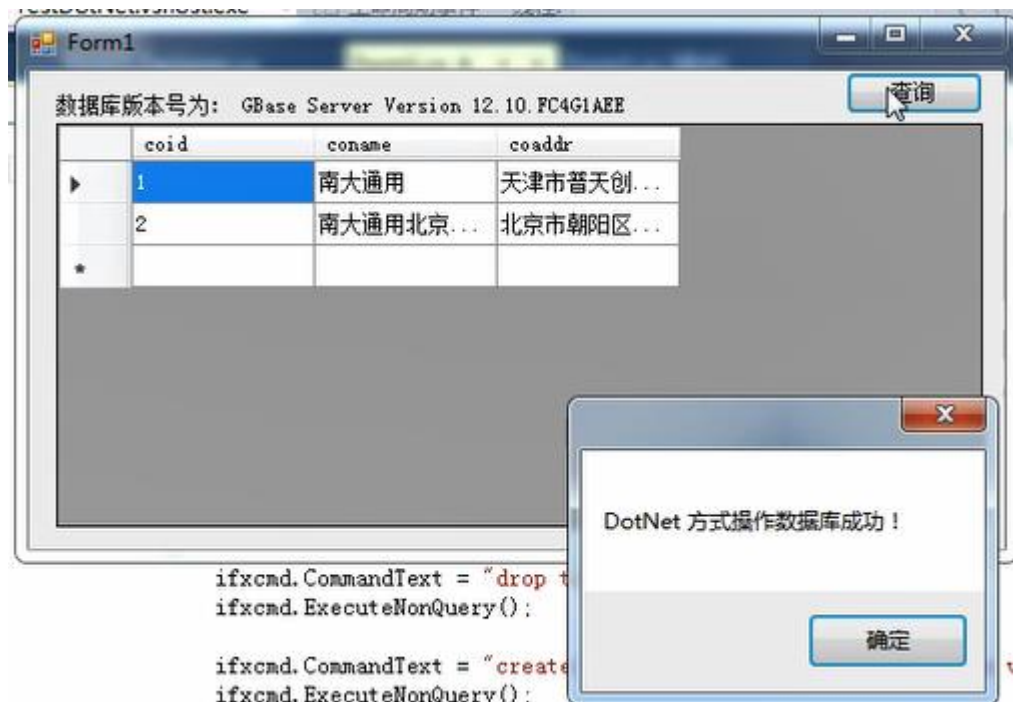
private void btnSelect_Click(object sender, EventArgs e)
{
    IfxDataAdapter ifxadpt = new IfxDataAdapter("select * from company", ifxconn);
    ds = new DataSet();
    ifxadpt.Fill(ds);
    this.dataGridView1.DataSource = ds.Tables[0];
    MessageBox.Show("DotNet 方式操作数据库成功! \n");
}
}
```

6)、执行 Debug 测试连接到数据库结果

所有的软件都使用了 64 位的，故 Debug 也使用 x64，如果使用的是 32 位的 CSDK，则选择 x86



出现 Form1 界面后，点击查询，将显示 company 表记录及弹出框提示成功。



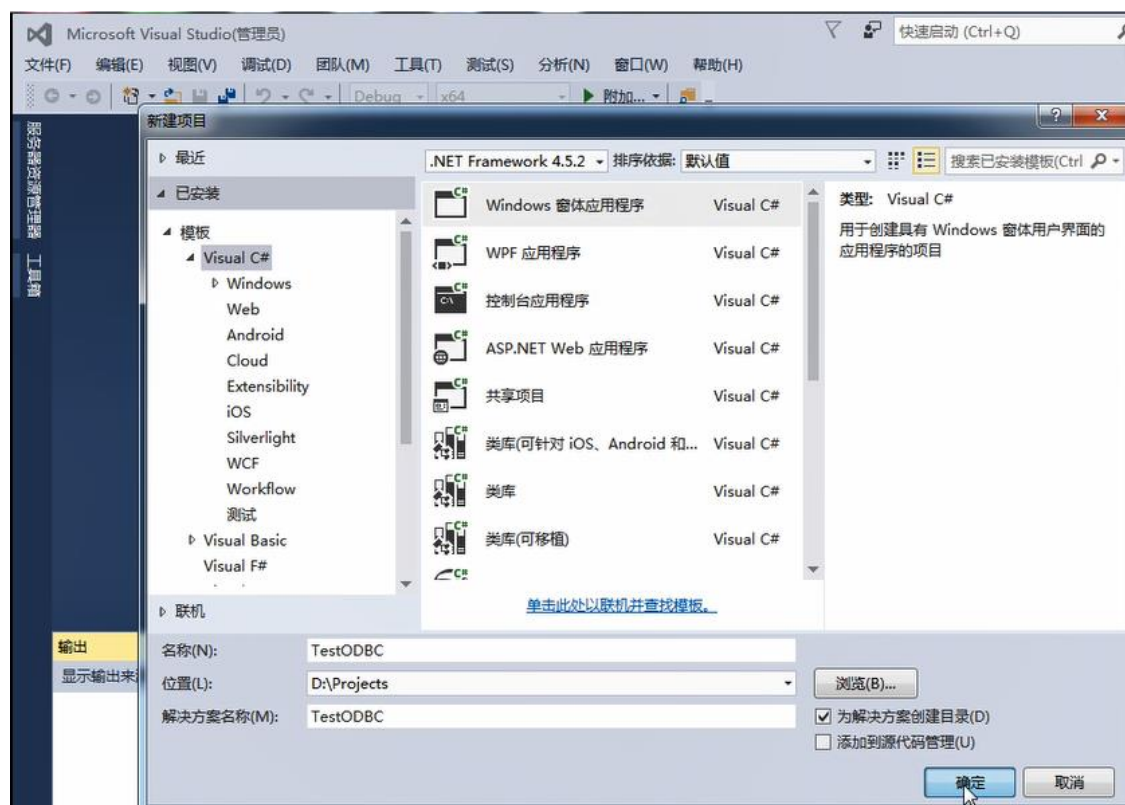
3.2.2. ODBC 方式 (.net framework) 连接到数据库-Windows

示例使用 Visual Studio 2015 社区版。要求 2.2.2 CSDK 的配置 和 2.2.3 ODBC 的配置 已经完成。

1)、打开 Visual Studio，文件(F) -> 新建(N) -> 项目(P)

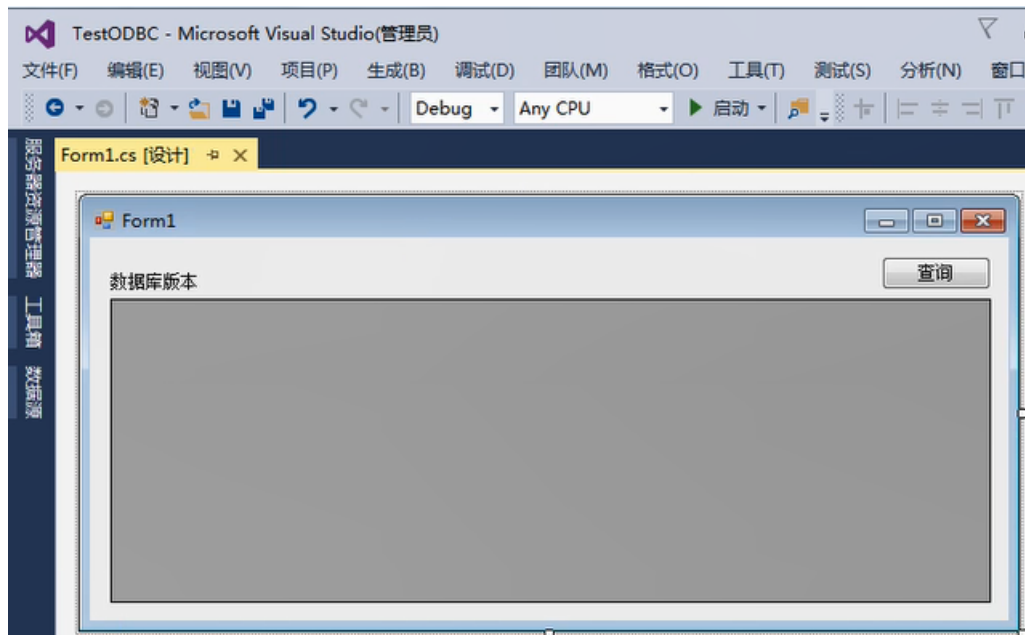
2)、指定编程语言及.net framework 版本

使用 Visual C#， .NET Framework 4.5.2， 创建 Windows 窗体应用程序，指定项目名称为 TestODBC，位置等



3)、Form1 窗体添加工具，并调整格式

增加一个 Label, 名称为 label1; 一个 dataGridView, 名称为 dataGridView1; 一个 button, 名称为 btnSelect



4)、增加 C#代码

将以下示例代码复制到 **Form1.cs** 中（视需要修改控件名称）

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Odbc;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TestODBC
{
    public partial class Form1 : Form
    {
        OdbcConnection odbconn;
        DataSet ds;
        public Form1()
        {
            InitializeComponent();
            OdbcConnectionStringBuilder build = new OdbcConnectionStringBuilder();
            build.Driver = "GBase ODBC DRIVER (64-Bit)"; // 在系统中注册的驱动名称
            build.Add("Host", "bd.gbasedbt.com"); // 主机地址或者 IP 地址
            build.Add("Service", "9088"); // 数据库服务器的使用的端口号
            build.Add("Server", "gbase01"); // 数据库服务名称
        }
    }
}
```

```

        build.Add("Database", "utf8");           // 数据库名 (DBNAME)
        build.Add("Protocol", "onsoctcp");       // 网络协议名称
        build.Add("Uid", "gbasedbt");           // 用户
        build.Add("Pwd", "GBase123");           // 密码
        build.Add("Db_locale", "zh_CN.utf8");    // 数据库字符集
        build.Add("Client_locale", "zh_CN.utf8"); // 客户端字符集
        odbconn = new OdbcConnection(build.ConnectionString);
        odbconn.Open();
        using (OdbcCommand odbccmd = odbconn.CreateCommand())
        {
            odbccmd.CommandText = "drop table if exists company";
            odbccmd.ExecuteNonQuery();

            odbccmd.CommandText = "create table company(coid serial, coname varchar(255), coaddr varchar(255))";
            odbccmd.ExecuteNonQuery();

            odbccmd.CommandText = "insert into company values (0,'南大通用','天津市海泰绿色产业基地')";
            odbccmd.ExecuteNonQuery();

            odbccmd.CommandText = "insert into company values (0,'南大通用北京分公司','北京市朝阳区太阳宫')";
            odbccmd.ExecuteNonQuery();

            odbccmd.CommandText = "update company set coaddr = '天津市普天创新园' where coid = 1";
            odbccmd.ExecuteNonQuery();

            odbccmd.CommandText = "select dbinfo('version','full') from dual";
            OdbcDataReader dr = odbccmd.ExecuteReader();
            if (dr.Read())
            {
                this.labell.Text = "数据库版本号为: " + dr[0];
            }
        }

    private void btnSelect_Click(object sender, EventArgs e)
    {
        OdbcDataAdapter odbcadpt = new OdbcDataAdapter("select * from company", odbconn);
        ds = new DataSet();
        odbcadpt.Fill(ds);
        this.dataGridView1.DataSource = ds.Tables[0];
        MessageBox.Show("ODBC 方式操作数据库成功! \n");
    }

```

```
}  
}
```

5)、执行 Debug 测试连接到数据库结果

所有的软件都使用了 64 位的，故 Debug 也使用 x64，如果使用的是 32 位的 CSDK，则选择 x86



出现 Form1 界面后，点击查询，将显示 company 表记录及弹出框提示成功。



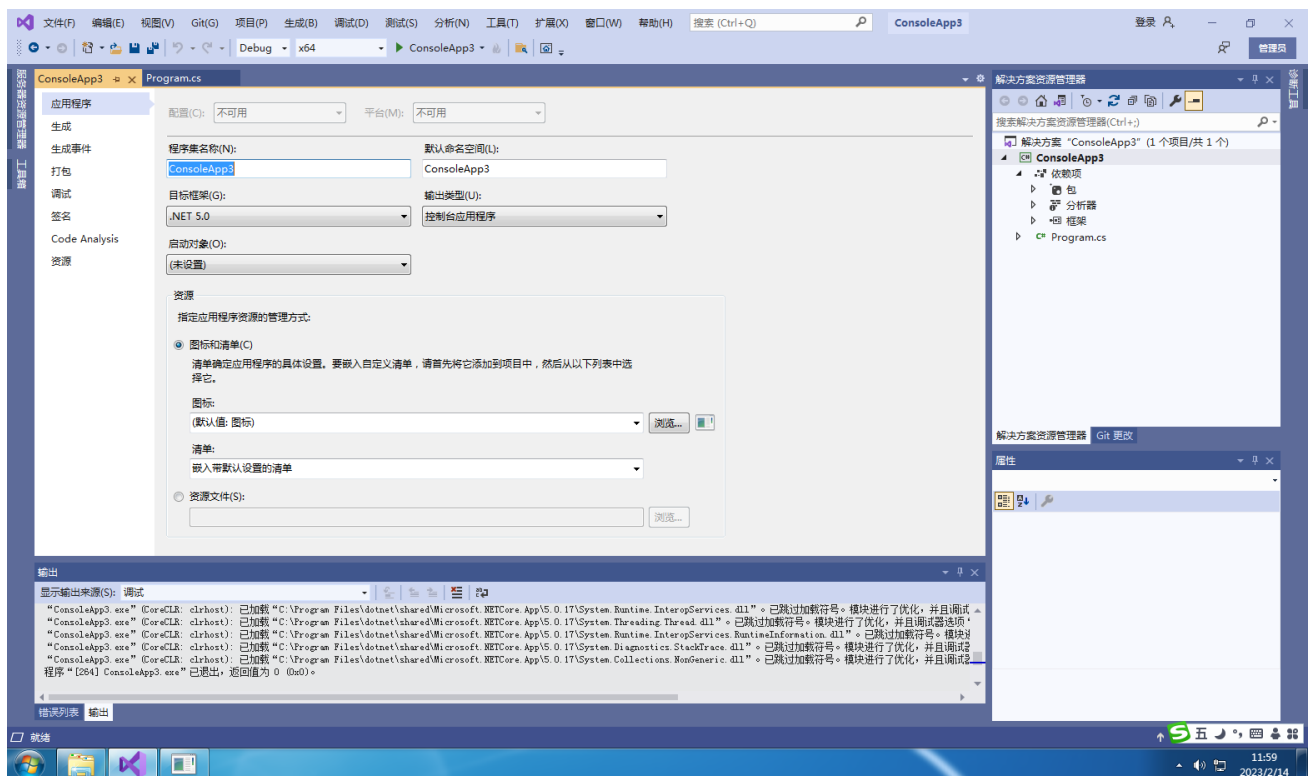
3.2.3. .NET Core 方式（net core）连接到数据库-Windows

示例使用 Visual Studio 2019 社区版。要求 2.2.2 CSDK 的配置 已经完成。

1)、打开 Visual Studio，文件(F) -> 新建(N) -> 项目(P)

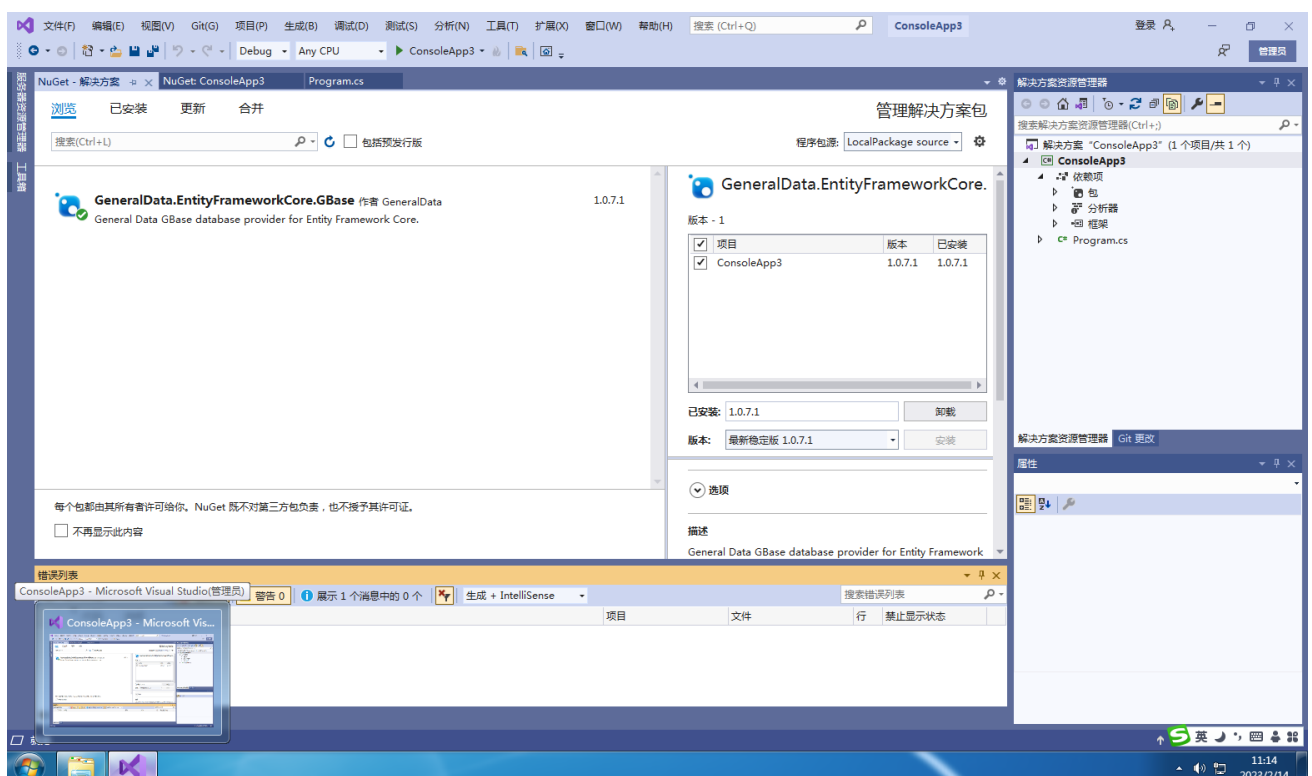
2)、指定编程语言及.net 5.0

使用 Visual C#，.NET Core，创建控制台应用程序，指定项目名称为 ConsoleApp3，位置等



3)、在 nuget 中增加本地源, GeneralData.EntityFrameworkCore.GBase.1.0.7.1.nupkg, 安装到项目中

下载地址: <https://gbasedbt.com/dl/EFcore/GeneralData.EntityFrameworkCore.GBase.1.0.7.1.nupkg>



4)、在 Program.cs 的 Main()中增加以下代码 (需按实际修改数据库连接方式)

```
using GBS.Data.GBasedbt;
using System;

namespace ConsoleApp3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");

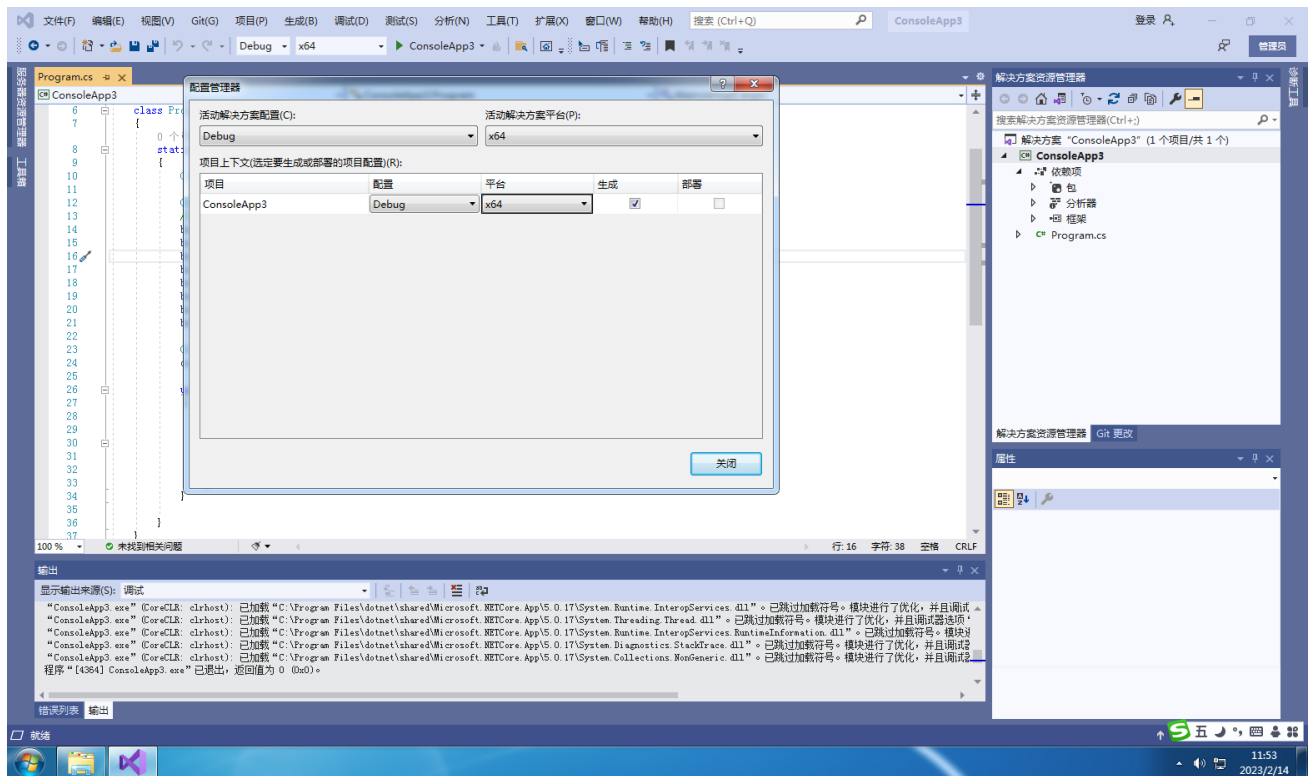
            GbsConnectionStringBuilder build = new GbsConnectionStringBuilder();
            // 远程数据库连接方式
            build.Host = "192.168.80.104";
            build.Service = "9088";
            build.Server = "gbase01";
            build.Database = "testdb";
            build.UID = "gbasedbt";
            build.Pwd = "GBase123";
            build.DbLocale = "zh_CN.utf8";
            build.ClientLocale = "zh_CN.utf8";

            GbsConnection conn = new GbsConnection(build.ConnectionString);
            conn.Open();

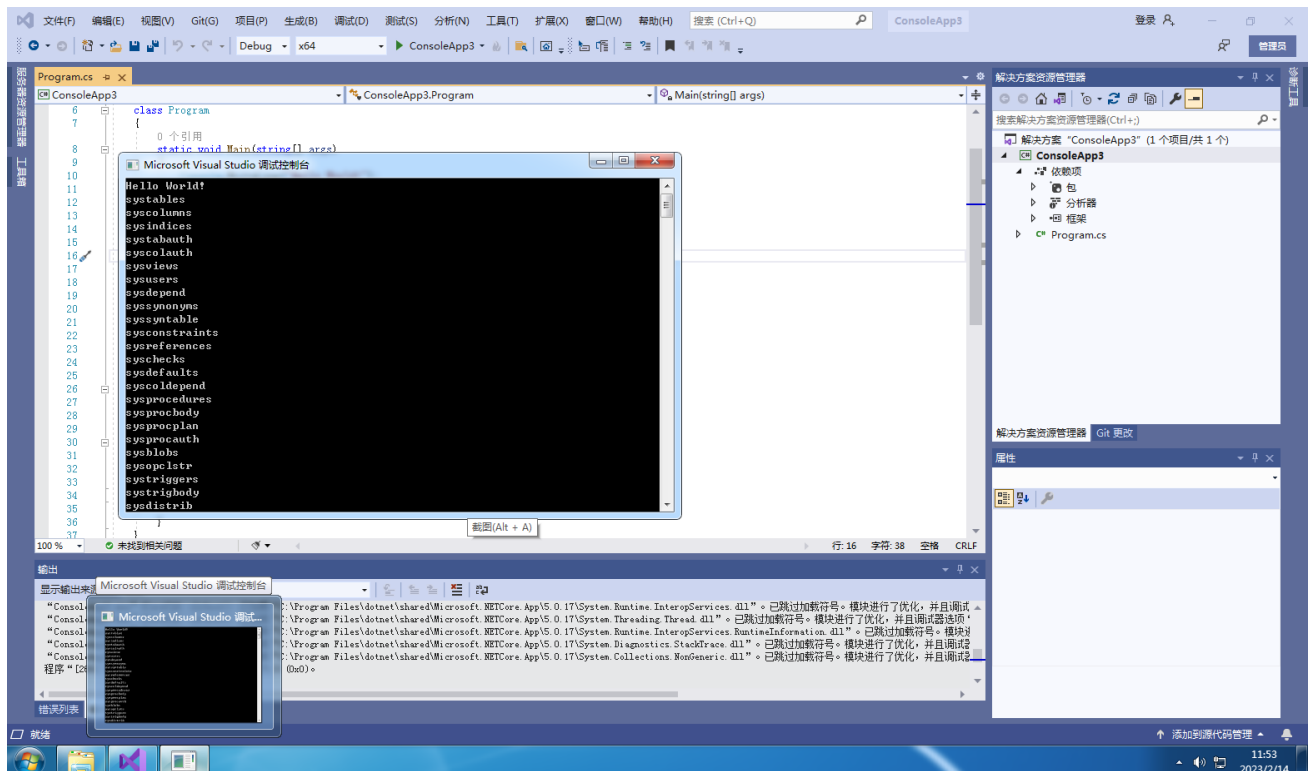
            using (GbsCommand cmd = conn.CreateCommand())
            {
                cmd.CommandText = "select * from systables";
                GbsDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    Console.WriteLine(dr["tabname"]);
                }
            }
        }
    }
}
```

5)、执行 Debug 测试连接到数据库结果

所有的软件都使用了 64 位的，故 Debug 也使用 x64，如果使用的是 32 位的 CSDK，则选择 x86



执行测试



3.2.4. ODBC 方式（netcore3）连接到数据库-Linux

要求 2.4.2 CSDK 的配置 和 2.4.3 ODBC 的配置已经完成，通过 iusql 确认通过 unixODBC 能访问数据库，如果不能，则需要升级 unixODBC 版本。示例为 CentOS 7 下 dotnet 3.1.418 环境，使用命令行方式。

注：升级 unixODBC 操作参考：<https://liaosnet.com/index.php/archives/113/>

1)、确认 dotnet 已经安装，如未安装建议使用 yum 源方式。

```
[root@h01 ~]# dotnet --version
3.1.418
```

2)、使用普通用户创建目录、创建项目

创建目录

```
[gbasedbt@h01 ~]$ mkdir -p Projects/TestCsODBC
[gbasedbt@h01 ~]$ cd Projects/TestCsODBC/
```

创建 console 项目

```
[gbasedbt@h01 TestCsODBC]$ dotnet new console

Welcome to .NET Core 3.1!

-----
SDK Version: 3.1.418

Telemetry
-----

The .NET Core tools collect usage data in order to help us improve your experience. It is collected by Microsoft
and shared with the community. You can opt-out of telemetry by setting the DOTNET_CLI_TELEMETRY_OPTOUT environment
variable to '1' or 'true' using your favorite shell.

Read more about .NET Core CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry

-----

Explore documentation: https://aka.ms/dotnet-docs
Report issues and find source on GitHub: https://github.com/dotnet/core
Find out what's new: https://aka.ms/dotnet-whats-new
Learn about the installed HTTPS developer cert: https://aka.ms/aspnet-core-https
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli-docs
Write your first app: https://aka.ms/first-net-core-app

-----

Getting ready...
The template "Console Application" was created successfully.
```

```
Processing post-creation actions...
Running 'dotnet restore' on /home/gbase/Projects/TestCsODBC/TestCsODBC.csproj...
    Determining projects to restore...
    Restored /home/gbase/Projects/TestCsODBC/TestCsODBC.csproj (in 172 ms).

Restore succeeded.
```

在当前目录下生成相应的工程文件，默认生成 Hello World.

```
[gbasedbt@h01 TestCsODBC]$ ls -al
total 20
drwxrwxr-x 3 gbasedbt gbasedbt 4096 May  7 09:54 .
drwxrwxr-x 3 gbasedbt gbasedbt 4096 May  7 09:53 ..
drwxrwxr-x 2 gbasedbt gbasedbt 4096 May  7 09:54 obj
-rw-rw-r-- 1 gbasedbt gbasedbt 192 May  7 09:54 Program.cs
-rw-rw-r-- 1 gbasedbt gbasedbt 178 May  7 09:54 TestCsODBC.csproj
```

3)、项目中增加 System.Data.ODBC，版本为 6.0.0

```
[gbasedbt@h01 TestCsODBC]$ dotnet add package System.Data.ODBC -v 6.0.0
    Determining projects to restore...
    Writing /tmp/tmpzqjnt1.tmp
info : Adding PackageReference for package 'System.Data.ODBC' into project
'/home/gbase/Projects/TestCsODBC/TestCsODBC.csproj'.
info : Restoring packages for /home/gbase/Projects/TestCsODBC/TestCsODBC.csproj...
info :   GET https://api.nuget.org/v3-flatcontainer/system.data.odbc/index.json
info :   OK https://api.nuget.org/v3-flatcontainer/system.data.odbc/index.json 250ms
info :   GET https://api.nuget.org/v3-flatcontainer/system.data.odbc/6.0.0/system.data.odbc.6.0.0.nupkg
info :   OK https://api.nuget.org/v3-flatcontainer/system.data.odbc/6.0.0/system.data.odbc.6.0.0.nupkg 123ms
info :   GET https://api.nuget.org/v3-flatcontainer/system.text.encoding.codepages/index.json
info :   OK https://api.nuget.org/v3-flatcontainer/system.text.encoding.codepages/index.json 257ms
info :   GET
https://api.nuget.org/v3-flatcontainer/system.text.encoding.codepages/6.0.0/system.text.encoding.codepages.6
.0.0.nupkg
info :   OK
https://api.nuget.org/v3-flatcontainer/system.text.encoding.codepages/6.0.0/system.text.encoding.codepages.6
.0.0.nupkg 60ms
info :   GET https://api.nuget.org/v3-flatcontainer/system.runtime.compilerservices.unsafe/index.json
info :   OK https://api.nuget.org/v3-flatcontainer/system.runtime.compilerservices.unsafe/index.json 278ms
info :   GET
https://api.nuget.org/v3-flatcontainer/system.runtime.compilerservices.unsafe/6.0.0/system.runtime.compilers
ervices.unsafe.6.0.0.nupkg
info :   OK
https://api.nuget.org/v3-flatcontainer/system.runtime.compilerservices.unsafe/6.0.0/system.runtime.compilers
ervices.unsafe.6.0.0.nupkg 58ms
info : Installed System.Runtime.CompilerServices.Unsafe 6.0.0 from https://api.nuget.org/v3/index.json with
content hash /iUeP3tq1S0XdNNoMz5C9twLSrM/TH+qElHkXWapvuN0t+99G75NrV00S2EqHx5wMN7popYjpc8oTjC1y16DLg==.
```

```

info : Installed System.Data.Odbc 6.0.0 from https://api.nuget.org/v3/index.json with content hash
pnZjwe0QwrlRnp7NExd5zz4YwXJrYuAbWNKjEQpTzCEg6f/L5DYJS7w3hG3vgSjlt/r79UL390YzXIklf1VuQQ==.
info : Installed System.Text.Encoding.CodePages 6.0.0 from https://api.nuget.org/v3/index.json with content hash
ZFCILZu0vtKPauZ/j/swhvw68ZRI9ATCfvGbk1QfydmcXBkIWecWKn/250UH7rahZ50oDBaiAudJtPvLwzw85A==.
info : Package 'System.Data.ODBC' is compatible with all the specified frameworks in project
'/home/gbase/Projects/TestCsODBC/TestCsODBC.csproj'.
info : PackageReference for package 'System.Data.ODBC' version '6.0.0' added to file
'/home/gbase/Projects/TestCsODBC/TestCsODBC.csproj'.
info : Committing restore...
info : Writing assets file to disk. Path: /home/gbase/Projects/TestCsODBC/obj/project.assets.json
log : Restored /home/gbase/Projects/TestCsODBC/TestCsODBC.csproj (in 3.86 sec).

```

完成后，TestCsODBC.csproj 中增加包的引用

```

[gbasedbt@h01 TestCsODBC]$ more TestCsODBC.csproj
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="System.Data.ODBC" Version="6.0.0" />
  </ItemGroup>

</Project>

```

4)、修改编辑 Program.cs 文件，内容如下：

```

using System;
using System.Data;
using System.Data.Odbc;

namespace TestCsODBC
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\n 测试 Linux 下 dotnet 通过 ODBC 方式连接到数据库\n");
            // odbinst.ini 里的 Driver 名称, DSN 名称
            String connStr = "Driver={GBase ODBC DRIVER};DSN=utf8";
            /*
            // 如果不使用 DSN 方式，以下参数均需要加上，以下参数为缩写格式。
            String connStr = "Driver=/opt/gbase/lib/cli/iclit09b.so;" +
                "Host=192.168.0.57;SERV=9088;PROT=onsoctcp;SRVR=gbase01;" +

```

```

        "DB=utf8;DLOC=zh_CN.utf8;CLOC=zh_CN.utf8;" +
        "UID=gbasedbt;PWD=GBase123;" +
        "TDLL=/opt/gbase/lib/esql/igo4a304.so";

    */

    OdbcConnection odbconn = new OdbcConnection(connStr);
    try
    {
        odbconn.Open();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    using (OdbcCommand odbccmd = odbconn.CreateCommand())
    {
        // drop table
        odbccmd.CommandText = "drop table if exists company";
        odbccmd.ExecuteNonQuery();

        // create table
        odbccmd.CommandText = "create table company(coid serial, coname varchar(255), coaddr " +
        "varchar(255))";
        odbccmd.ExecuteNonQuery();

        // insert rows
        odbccmd.CommandText = "insert into company values (0, '南大通用', '天津市海泰绿色产业基地')";
        odbccmd.ExecuteNonQuery();

        odbccmd.CommandText = "insert into company values (0, '南大通用北京分公司', '北京市朝阳区太阳宫 " +
        "')";
        odbccmd.ExecuteNonQuery();

        // update row
        odbccmd.CommandText = "update company set coaddr = '天津市普天创新园' where coid = 1";
        odbccmd.ExecuteNonQuery();

        // select rows
        odbccmd.CommandText = "select * from company";
        OdbcDataReader dr = odbccmd.ExecuteReader();
        while (dr.Read())
        {
            Console.WriteLine(" " + dr[0] + "\t" + dr[1] + "\t" + dr[2]);
        }
    }

```

```
    }  
    }  
  
    Console.WriteLine("\n 测试 Linux 下 dotnet 通过 ODBC 方式连接到数据库 结束\n");  
    }  
}  
}
```

5)、编译运行

build

```
[gbasedbt@h01 TestCsODBC]$ dotnet build  
Microsoft (R) Build Engine version 16.7.2+b60ddb6f4 for .NET  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Determining projects to restore...  
All projects are up-to-date for restore.  
TestCsODBC -> /home/gbase/Projects/TestCsODBC/bin/Debug/netcoreapp3.1/TestCsODBC.dll  
  
Build succeeded.  
    0 Warning(s)  
    0 Error(s)  
  
Time Elapsed 00:00:01.13
```

运行

```
[gbasedbt@h01 TestCsODBC]$ dotnet run  
  
测试 Linux 下 dotnet 通过 ODBC 方式连接到数据库  
  
1   南大通用      天津市普天创新园  
2   南大通用北京分公司  北京市朝阳区太阳宫  
  
测试 Linux 下 dotnet 通过 ODBC 方式连接到数据库 结束
```

3.3. C 连接到数据库

3.3.1. ESQLC 方式（嵌入式 C）连接到数据库-Linux

要求 2.4.2 CSDK 的配置已经完成。Linux 下还需要编译器 gcc 和 gcc-c++均已经安装。

1)、确认 gcc 和 gcc-c++均已经安装。

```
[root@localhost esqlc]# rpm -qa gcc gcc-c++
gcc-4.8.5-39.el7.x86_64
gcc-c++-4.8.5-39.el7.x86_64
```

2)、设置系统网络连接（可选，也可配置为信任方式）

在用户目录下创建`.netrc` 配置文件，权限为 600，内容如下：

```
[root@localhost esqlc]# more ~/.netrc
# machine 目标主机名或 IP 地址 login 用户名 password 密码
machine bd.gbasedbt.com login gbasedbt password GBase123
```

3)、编写测试程序 TestESQLC.ec，内容如下：

```
#include <stdio.h>

main()
{
EXEC SQL BEGIN DECLARE SECTION;
    int v_tabid;
    char v_tabname[ 128 ];
EXEC SQL END DECLARE SECTION;

    printf( "\nESQLC 测试程序开始运行.\n\n");
EXEC SQL WHENEVER ERROR STOP;
EXEC SQL connect to 'utf8';

EXEC SQL declare democursor cursor for
select tabid,tabname
    into :v_tabid, :v_tabname
    from systables where tabid < 10;

EXEC SQL open democursor;
for (;;)
{
EXEC SQL fetch democursor;
if (strcmp(SQLSTATE, "00", 2) != 0)
    break;

    printf("%d\t%s\n", v_tabid, v_tabname);
}

if (strcmp(SQLSTATE, "02", 2) != 0)
    printf("SQLSTATE after fetch is %s\n", SQLSTATE);

EXEC SQL close democursor;
EXEC SQL free democursor;
```

```
EXEC SQL disconnect current;
printf("\nESQLC 测试程序结束运行.\n\n");

exit(0);
}
```

4)、编译成可执行文件，并执行测试

```
[root@localhost esqlc]# esql -o TestESQLC -glu TestESQLC.ec
[root@localhost esqlc]# ./TestESQLC
```

ESQLC 测试程序开始运行.

```
1  systables
2  syscolumns
3  sysindices
4  systabauth
5  syscolauth
6  sysviews
7  sysusers
8  sysdepend
9  syssynonyms
```

ESQLC 测试程序结束运行.

3.4. PHP 连接到数据库

3.4.1. ODBC 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置 和 2.4.3 ODBC 的配置已经完成。PHP 通过 ODBC 方式连接到数据库，需要 php 及 php-odbc 均已经安装。

1)、确认 php 及 php-odbc 已经安装

```
[root@localhost php-odbc]# rpm -qa php php-odbc
php-5.4.16-48.el7.x86_64
php-odbc-5.4.16-48.el7.x86_64
```

2)、编写测试程序 TestPhpOdbc.php

```
<?php
# 连接到 DSN: utf8
echo "PHP ODBC 测试程序开始运行.\n\n";
$conn=odbc_connect('utf8','','');
if (!$conn)
```

```
{exit("数据库连接失败: " . $conn);}

$sql="SELECT tabid,tabname FROM systables where tabid < 10";

$rs=odbc_exec($conn,$sql);
if (!$rs)
    {exit("SQL 错误");}

while (odbc_fetch_row($rs))
{
    $col1=odbc_result($rs,"tabid");
    $col2=odbc_result($rs,"tabname");
    echo "$col1\t$col2\n";
}

odbc_close($conn);
echo "\nPHP ODBC 测试程序结束运行.\n";
```

3)、执行测试

```
[root@localhost php-odbc]# php -e TestPhpOdbc.php
PHP ODBC 测试程序开始运行.
```

```
1  systables
2  syscolumns
3  sysindices
4  systabauth
5  syscolauth
6  sysviews
7  sysusers
8  sysdepend
9  syssynonyms
```

```
PHP ODBC 测试程序结束运行.
```

3.4.2. PDO_GBASEDBT 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置已经完成。PHP 通过 POD_GBASEDBT 连接到数据库，需要 php、php-devel 和 php-pdo 均已经安装。建议使用 yum 方式安装，该方式将自动安装依赖包（比如 automake, autoconf 等）。

1)、确认 php、php-devel 和 php-pdo 已经安装

```
[root@localhost php-pdo_gbasedbt]# rpm -qa php php-devel php-pdo
php-pdo-5.4.16-48.el7.x86_64
```

```
php-devel-5.4.16-48.el7.x86_64
php-5.4.16-48.el7.x86_64
```

2)、通过 php-config 获取配置两个信息：php-config 路径，extension-dir 目录

```
[root@localhost php-pdo_gbasedbt]# php-config
Usage: /usr/bin/php-config [OPTION]

Options:
  --prefix                [/usr]
  --includes               [-I/usr/include/php -I/usr/include/php/main -I/usr/include/php/TSRM
-I/usr/include/php/Zend -I/usr/include/php/ext -I/usr/include/php/ext/date/lib]
  --ldflags                []
  --libs                  [-lcrypt -lresolv -lcrypt -ledit -lncurses -lstdc++ -lgmp -lbz2 -lz -lpcres -lrt -lm -ldl
-lns1 -lxml2 -lz -lm -ldl -lgssapi_krb5 -lkrb5 -lk5crypto -lcom_err -lssl -lcrypto -lssl -lcrypto -lxml2 -lz
-lm -ldl -lcrypt -lxml2 -lz -lm -ldl -lcrypt ]
  --extension-dir         [/usr/lib64/php/modules]
  --include-dir            [/usr/include/php]
  --man-dir                [/usr/share/man]
  --php-binary             [/usr/bin/php]
  --php-sapis              [ cli cgi]
```

3)、下载 PDO_GBASEDBT，并上传解压

下载地址为：https://gbasedbt.com/dl/PDO_GBAsedbt/

PDO_GBAsEDBT 基于 PDO_INFORMIX 改写而来

```
[root@localhost php-pdo_gbasedbt]# ll
总用量 76
-rw-r--r--. 1 root root 74679 10月 28 23:01 PDO_GBAsEDBT-1.3.3.tgz
[root@localhost php-pdo_gbasedbt]# tar -zxvf PDO_GBAsEDBT-1.3.3.tgz
```

4)、使用 phize 扩展 php 动态模块 PDO_GBAsEDBT

```
[root@localhost PDO_GBAsEDBT-1.3.3]# phize
Configuring for:
PHP Api Version:      20100412
Zend Module Api No:   20100525
Zend Extension Api No: 220100525
```

5)、编译 PDO_GBAsEDBT

其中 prefix 指向前面获取的 extension-dir 目录

with-php-config 指向前面获取的 php-config 路径

with-pdo-gbasedbt 指向 GBAsEDBTDIR

```
[root@localhost PDO_GBAsEDBT-1.3.3]# ./configure --prefix=/usr/lib64/php/modules \
--with-php-config=/usr/bin/php-config \
--with-pdo-gbasedbt=/opt/gbase
```

```
[root@localhost PDO_GBASEDBT-1.3.3]# make

[root@localhost PDO_GBASEDBT-1.3.3]# make install
Installing shared extensions:      /usr/lib64/php/modules/
```

6)、为 php 增加扩展

在/etc/php.d/目录下增加 pdo_gbasedbt.ini 配置文件，内容如下：

```
; Enable pdo_gbasedbt extension module
extension=pdo_gbasedbt.so
```

7)、编写测试程序 TestPhpPdoGBasedbt.php，内容如下：

```
<?php
    echo "PHP PDO_GBAsedbt 测试程序开始运行.\n\n";
    # 数据库连接
    $dbh = new PDO("gbasedbt:host=bd.gbasedbt.com; service=9088; database=utf8; server=gbase01;
protocol=onsoctcp","gbasedbt","GBase123");
    # 连接到 CM 的话,需要在 ODBCINI 中 DSN 的 Servername 指定为 CM/DB 的 group, 在 SQLHOSTS 中定义
    # $dbh = new PDO("gbasedbt:DSN=utf8","gbasedbt","GBase123");

    $sth1 = $dbh->prepare("SELECT tabid,tabname FROM systables where tabid < 10");
    $sth1->execute();

    while( $row = $sth1->fetch() )
    {
        $col1=$row["TABID"];
        $col2=$row["TABNAME"];
        echo "$col1\t$col2\n";
    }

    $dbh = null;
    echo "\nPHP PDO_GBAsedbt 测试程序结束运行.\n\n";
```

8)、执行测试程序

```
[root@localhost php-pdo_gbasedbt]# php -e TestPhpPdoGBasedbt.php
PHP PDO_GBAsedbt 测试程序开始运行.

1  systables
2  syscolumns
3  sysindices
4  systabauth
5  syscolauth
6  sysviews
7  sysusers
8  sysdepend
9  syssynonyms
```

PHP PDO GBasedbt 测试程序结束运行.

3.5. Python3 连接到数据库

3.5.1. Pyodbc 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置 和 2.4.3 ODBC 的配置已经完成。Python3 通过 pyodbc 连接到数据库，需要 python3、python3-devel 和 pyodbc。

1)、确认 python3 和 python3-devel 已经安装

```
[root@localhost python-pyodbc]# rpm -qa python3 python3-devel
python3-devel-3.6.8-13.el7.x86_64
python3-3.6.8-13.el7.x86_64
```

2)、确认安装 pyodbc, 如果没有, 则安装之

```
[root@localhost python-pyodbc]# pip3 list --format=columns
Package      Version
-----
pip          9.0.3
setuptools   39.2.0

[root@localhost python-pyodbc]# pip3 install pyodbc
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting pyodbc
  Downloading
https://files.pythonhosted.org/packages/81/0d/bb08bb16c97765244791c73e49de9fd4c24bb3ef00313aed82e5640dee5d/p
yodbc-4.0.30.tar.gz (266kB)
  100% |████████████████████████████████████████████████████████████████████████████████| 276kB 71kB/s
Installing collected packages: pyodbc
  Running setup.py install for pyodbc ... done
Successfully installed pyodbc-4.0.30
```

3)、编写测试程序 TestPython3Pyodbc.py, 内容如下:

```
#!/usr/bin/python3
# filename: TestPython3Pyodbc.py

import sys
import pyodbc

print("Python Pyodbc 测试程序开始运行.\n")
# use DSN, need PWD key word.
```

```
conn = pyodbc.connect("DSN=utf8;PWD=GBase123")
# set connection encoding
conn.setencoding(encoding='UTF-8')

mycursor = conn.cursor()
mycursor.execute("drop table if exists company")

mycursor.execute("create table company(coid serial,coname varchar(255),coaddr varchar(255))")

mycursor.execute("insert into company(coname,coaddr) values (?,?)", '南大通用', '天津市海泰绿色产业基地')
mycursor.execute("insert into company(coname,coaddr) values (?,?)", '南大通用北京分公司', '北京市朝阳区太阳宫')
mycursor.execute("update company set coaddr = ? where coid = 1", '天津市普天创新园')
conn.commit()

cursor1 = conn.cursor()
cursor1.execute("select * from company")
rows = cursor1.fetchall()

for i, (coid, coname, coaddr) in enumerate(rows):
    print("公司 ID: %d \t 公司名称: %s\t 公司地址: %s" % (coid, str(coname), str(coaddr)))

conn.close()
print("\nPython Pyodbc 测试程序结束运行.")
sys.exit(0)
```

4)、执行测试程序

```
[root@localhost python-pyodbc]# python3 TestPython3Pyodbc.py
Python Pyodbc 测试程序开始运行.

公司 ID: 1      公司名称: 南大通用 公司地址: 天津市普天创新园
公司 ID: 2      公司名称: 南大通用北京分公司    公司地址: 北京市朝阳区太阳宫

Python Pyodbc 测试程序结束运行.
```

3.5.2. Pyodbc 方式连接到数据库-Windows

示例使用 python-3.8.7。要求 2.2.2 CSDK 的配置 和 2.2.3 ODBC 的配置 已经完成。

1)、确认 python3 已经安装

```
D:\Projects> python -V
Python 3.8.7
```

2)、确认安装 pyodbc，如果没有，则安装之

```
D:\Projects> pip3 list
```

```

Package      Version
-----
pip          20.2.3
setuptools  49.2.1

D:\Projects> pip3 install pyodbc
Collecting pyodbc
  Using cached pyodbc-4.0.30-cp38-cp38-win_amd64.whl (68 kB)
Installing collected packages: pyodbc
Successfully installed pyodbc-4.0.30

D:\Projects> pip3 list
Package      Version
-----
pip          20.2.3
pyodbc       4.0.30
setuptools  49.2.1

```

3)、编写测试程序 TestPython3Pyodbc.py，内容如下：

```

#!/usr/bin/python3
# filename: TestPython3Pyodbc.py

import sys
import pyodbc

print("Python Pyodbc 测试程序开始运行.\n")
# use DSN, need PWD key word.
conn = pyodbc.connect("DSN=utf8;PWD=GBase123")
# conn = pyodbc.connect("Driver={GBase ODBC DRIVER (64-bit)};Host=47.106.34.174;" +
# "Service=9088;Protocol=onsocket;Server=gbase01;Database=testdb;Uid=gbasedbt;" +
# "Pwd=GBase123;DB_LOCALE=zh_CN.utf8;CLIENT_LOCALE=zh_CN.utf8")
# conn = pyodbc.connect("Driver={GBase ODBC DRIVER (64-bit)};HOST=47.106.34.174;" +
# "SERV=9088;PROT=onsocket;SRVR=gbase01;DB=testdb;UID=gbasedbt;PWD=GBase123;" +
# "DLOC=zh_CN.utf8;CLOC=zh_CN.utf8")

# set connection encoding
conn.setencoding(encoding='UTF-8')

mycursor = conn.cursor()
mycursor.execute("drop table if exists company")

mycursor.execute("create table company(coid serial,coname varchar(255),coaddr varchar(255))")

mycursor.execute("insert into company(coname,coaddr) values (?,?)", '南大通用', '天津市海泰绿色产业基地')
mycursor.execute("insert into company(coname,coaddr) values (?,?)", '南大通用北京分公司', '北京市朝阳区太阳宫')

```

```

mycursor.execute("update company set coaddr = ? where coid = 1", '天津市普天创新园')
conn.commit()

cursor1 = conn.cursor()
cursor1.execute("select * from company")
rows = cursor1.fetchall()

for i, (coid, coname, coaddr) in enumerate(rows):
    print("公司 ID: %d \t 公司名称: %s\t 公司地址: %s" % (coid, str(coname), str(coaddr)))

conn.close()
print("\nPython Pyodbc 测试程序结束运行.")
sys.exit(0)

```

4)、执行测试程序

```

D:\Python3> python TestPython3Pyodbc.py
Python Pyodbc 测试程序开始运行.

公司 ID: 1      公司名称: 南大通用      公司地址: 天津市普天创新园
公司 ID: 2      公司名称: 南大通用北京分公司  公司地址: 北京市朝阳区太阳宫

Python Pyodbc 测试程序结束运行.

```

3.5.3. DbtPy 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置已经完成。需要 python3、python3-devel 和 DbtPy。建议使用 yum 源安装 python3，该方式将自动安装依赖包。

1)、确认 python3 已经完成

```

[root@gbasehost01 ~]# python3 -V
Python 3.6.8

```

2)、配置、安装 DbtPy

设置环境变量，在/etc/profile 配置文件里增加 DbtPy 需要的环境变量 CSDK_HOME，GBASEBTDIR，LD_LIBRARY_PATH

```

# /etc/profile
# Add/Modify for DbtPy
export GBASEBTDIR=/opt/gbase
export CSDK_HOME=/opt/gbase
export LD_LIBRARY_PATH=$GBASEBTDIR/lib:$GBASEBTDIR/lib/cli:$GBASEBTDIR/lib/esql:$LD_LIBRARY_PATH

```

检查 DbtPy 是否安装

```
[root@gbasehost01 ~]# pip3 list | grep DbtPy
```

使用 pip3 在线安装

```
[root@gbasehost01 ~]# pip3 install DbtPy
Collecting DbtPy
  Downloading
https://pypi.tuna.tsinghua.edu.cn/packages/a7/12/82faa077c97d3891fadc4566d402259dba059ea79d546db2197a68b64910/DbtPy-3.0.5.2.tar.gz (171kB)
    100% |██████████████████████████████████████████████████████████████| 174kB 351kB/s
Installing collected packages: DbtPy
  Running setup.py install for DbtPy ... done
Successfully installed DbtPy-3.0.5.2
```

检查 DbtPy 信息

```
[root@gbasehost01 ~]# pip3 show DbtPy
Name: DbtPy
Version: 3.0.5.2
Summary: GBase 8s native Python driver
Home-page: https://gbasedbt.com/DbtPy
Author: Informix Application Development Team & GBase 8s Support Team
Author-email: None
License: Apache License 2.0
Location: /usr/local/lib64/python3.6/site-packages
Requires:
```

3) 编写测试程序 TestP3DbtPy.py, 内容如下:

```
#!/usr/bin/python3
# filename: TestP3DbtPy.py

import sys
import DbtPy

print("Python DbtPy 测试程序开始运行.\n")

connectStr="PROTOCOL=onsocket;HOST=a02.gbasedbt.com;SERV=9088;PROT=onsocket;SRVR=gbase01;DB=testdb;UID=gbase
dbt;PWD=GBase123;DLOC=zh_CN.utf8;CLOC=zh_CN.utf8"

conn=DbtPy.connect(connectStr, "gbasedbt", "GBase123")

stmt=DbtPy.exec_immediate(conn, "drop table if exists company")

stmt=DbtPy.exec_immediate(conn, "create table company(coid serial,coname varchar(255),coaddr varchar(255))")

stmt=DbtPy.prepare(conn,"insert into company(coname,coaddr) values(?,?)")

DbtPy.bind_param(stmt,1,"南大通用")

DbtPy.bind_param(stmt,2,"天津市普天创新园")

DbtPy.execute(stmt)
```

```

print("插入表 生效的行数: ", DbtPy.num_rows(stmt))

param="南大通用北京分公司","北京市朝阳区太阳宫",
DbtPy.execute(stmt,param)
print("插入表 生效的行数: ", DbtPy.num_rows(stmt))

'''
bool  fetch_row    : 判断是否获取到行
dict  fetch_assoc  : 结果集用字段名称编号
dict  fetch_both   : 结果集使用序号和字段名称同时编号(两份数据)
tuple fetch_tuple  : 获取的结果为元组
'''

# 使用 fetch_tuple
stmt=DbtPy.exec_immediate(conn, "select * from company")
tuple=DbtPy.fetch_tuple(stmt)
while tuple != False:
    print(tuple)
    tuple = DbtPy.fetch_tuple(stmt)

# 使用 fetch_both/fetch_assoc
stmt=DbtPy.exec_immediate(conn, "select * from company")
dict=DbtPy.fetch_both(stmt)
while dict != False:
    print(dict)
    print("COID: ", dict[0], " CONAME: ", dict[1], " COADDR: ", dict[2])
    dict = DbtPy.fetch_both(stmt)

# 使用 fetch_row
stmt=DbtPy.exec_immediate(conn, "select * from company")
while DbtPy.fetch_row(stmt) != False:
    print("COLD : ", DbtPy.result(stmt,0), " CONAME : ", DbtPy.result(stmt,"coname"), " COADDR : ",
    DbtPy.result(stmt,"coaddr"))

DbtPy.free_result(stmt)
DbtPy.free_stmt(stmt)
DbtPy.close(conn)

print("\nPython DbtPy 测试程序结束运行.")
sys.exit(0)

```

4) 执行测试程序

```

[root@gbasehost01 ~]# python3 TestP3DbtPy.py
Python DbtPy 测试程序开始运行.

```

```

插入表 生效的行数: 1

```

插入表 生效的行数: 1

(1, '南大通用', '天津市普天创新园')

(2, '南大通用北京分公司', '北京市朝阳区太阳宫')

{'coid': 1, 0: 1, 'coname': '南大通用', 1: '南大通用', 'coaddr': '天津市普天创新园', 2: '天津市普天创新园'}

COID: 1 CONAME: 南大通用 COADDR: 天津市普天创新园

{'coid': 2, 0: 2, 'coname': '南大通用北京分公司', 1: '南大通用北京分公司', 'coaddr': '北京市朝阳区太阳宫', 2: '北京市朝阳区太阳宫'}

COID: 2 CONAME: 南大通用北京分公司 COADDR: 北京市朝阳区太阳宫

COLD: 1 CONAME: 南大通用 COADDR: 天津市普天创新园

COLD: 2 CONAME: 南大通用北京分公司 COADDR: 北京市朝阳区太阳宫

Python DbtPy 测试程序结束运行.

3.5.4. DbtPy 方式连接到数据库-Windows

示例使用 python-3.7.9。要求 2.2.2 CSDK 的配置 已经完成。

1)、确认 python3 已经安装

```
D:\Project> python -V
Python 3.7.9
```

2)、确认安装 DbtPy, 如果没有, 则安装之

Python3.4-3.8 将会自动下载对应版本的包进行安装, 其他 python 版本需要手动编译安装。

```
D:\Project> pip3 list
Package Version
-----
pip      20.1.1
setuptools 47.1.0

D:\Project> pip3 install DbtPy
Collecting DbtPy
  Downloading DbtPy-3.0.5.2-cp37-cp37m-win_amd64.whl (69 kB)
    |████████████████████████████████████████| 30 kB 960 kB/s eta 0:00:01
    |████████████████████████████████████████| 40 kB 238 kB/s eta 0:
    |████████████████████████████████████████| 51 kB 272 kB/s e
    |████████████████████████████████████████| 61 kB 302 k
    |████████████████████████████████████████| 69 kB 35
3 kB/s
Installing collected packages: DbtPy
Successfully installed DbtPy-3.0.5.2
```

```
D:\Project> pip3 show DbtPy
Name: DbtPy
Version: 3.0.5.2
Summary: GBase 8s native Python driver
Home-page: https://gbasedbt.com/DbtPy
Author: Informix Application Development Team & GBase 8s Support Team
Author-email: None
License: Apache License 2.0
Location: c:\users\administrator\appdata\local\programs\python\python37\lib\site
-packages
Requires:
Required-by:
```

3) 编写测试程序 TestP3DbtPy.py, 内容如下:

```
#!/usr/bin/python3
# filename: TestP3DbtPy.py

import sys
import DbtPy

print("Python DbtPy 测试程序开始运行.\n")
connectStr="PROTOCOL=onsoctcp;HOST=a02.gbasedbt.com;SERV=9088;PROT=onsoctcp;SRVR=gbase01;DB=testdb;UID=gbase
dbt;PWD=GBase123;DLOC=zh_CN.utf8;CLOC=zh_CN.utf8"
conn=DbtPy.connect(connectStr, "gbasedbt", "GBase123")

stmt=DbtPy.exec_immediate(conn, "drop table if exists company")

stmt=DbtPy.exec_immediate(conn, "create table company(coid serial,coname varchar(255),coaddr varchar(255))")

stmt=DbtPy.prepare(conn,"insert into company(coname,coaddr) values(?,?)")
DbtPy.bind_param(stmt,1,"南大通用")
DbtPy.bind_param(stmt,2,"天津市普天创新园")
DbtPy.execute(stmt)
print("插入表 生效的行数: ", DbtPy.num_rows(stmt))

param="南大通用北京分公司","北京市朝阳区太阳宫",
DbtPy.execute(stmt,param)
print("插入表 生效的行数: ", DbtPy.num_rows(stmt))

'''
bool  fetch_row    : 判断是否获取到行
dict  fetch_assoc  : 结果集用字段名称编号
dict  fetch_both   : 结果集使用序号和字段名称同时编号(两份数据)
tuple fetch_tuple  : 获取的结果为元组
```

```

'''
# 使用 fetch_tuple
stmt=DbtPy.exec_immediate(conn, "select * from company")
tuple=DbtPy.fetch_tuple(stmt)
while tuple != False:
    print(tuple)
    tuple = DbtPy.fetch_tuple(stmt)

# 使用 fetch_both/fetch_assoc
stmt=DbtPy.exec_immediate(conn, "select * from company")
dict=DbtPy.fetch_both(stmt)
while dict != False:
    print(dict)
    print("COID: ", dict[0], " CONAME: ", dict[1], " COADDR: ", dict[2])
    dict = DbtPy.fetch_both(stmt)

# 使用 fetch_row
stmt=DbtPy.exec_immediate(conn, "select * from company")
while DbtPy.fetch_row(stmt) != False:
    print("COLD : ", DbtPy.result(stmt, 0), " CONAME : ", DbtPy.result(stmt, "coname"), " COADDR : ",
    DbtPy.result(stmt, "coaddr"))

DbtPy.free_result(stmt)
DbtPy.free_stmt(stmt)
DbtPy.close(conn)

print("\nPython DbtPy 测试程序结束运行.")
sys.exit(0)
'''

```

4) 执行测试程序

```

D:\Project> python TestP3DbtPy.py
Python DbtPy 测试程序开始运行.

插入表 生效的行数: 1
插入表 生效的行数: 1
(1, '南大通用', '天津市普天创新园')
(2, '南大通用北京分公司', '北京市朝阳区太阳宫')
{'coid': 1, 0: 1, 'coname': '南大通用', 1: '南大通用', 'coaddr': '天津市普天创新园', 2: '天津市普天创新园'}
COID: 1 CONAME: 南大通用 COADDR: 天津市普天创新园
{'coid': 2, 0: 2, 'coname': '南大通用北京分公司', 1: '南大通用北京分公司', 'coaddr': '北京市朝阳区太阳宫', 2:
'北京市朝阳区太阳宫'}
COID: 2 CONAME: 南大通用北京分公司 COADDR: 北京市朝阳区太阳宫
COLD: 1 CONAME: 南大通用 COADDR: 天津市普天创新园
COLD: 2 CONAME: 南大通用北京分公司 COADDR: 北京市朝阳区太阳宫

```

Python DbtPy 测试程序结束运行。

3.5.5. JayDeBeApi (JDBC) 方式连接到数据库

要求 2.1 Windows 下的 JDBC 驱动安装或者 2.3 Linux 下的 JDBC 安装 已经完成。

1)、确认 JayDeBeApi 已经安装

```
[root@a02 ~]# pip3 list | grep JayDeBeApi

[root@a02 ~]# pip3 install JayDeBeApi
Looking in indexes: http://mirrors.cloud.aliyuncs.com/pypi/simple/
Collecting JayDeBeApi
  Using cached
http://mirrors.cloud.aliyuncs.com/pypi/packages/ff/1f/6a627c9bd7deal3235b65fce0fff987507269d41f957c578031796
f70319/JayDeBeApi-1.2.3-py3-none-any.whl (26 kB)
Requirement already satisfied: JPype1 in /usr/local/lib64/python3.6/site-packages (from JayDeBeApi) (1.3.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.6/site-packages (from
JPype1->JayDeBeApi) (4.0.1)
Installing collected packages: JayDeBeApi
Successfully installed JayDeBeApi-1.2.3
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system
package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

2)、确认 java 环境已经安装

```
[root@a02 ~]# java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
```

3)、复制 JDBC 驱动 (gbasedbtjdbc_3.3.0_2.jar) 到指定目录, 如/root

```
[root@a02 ~]# ls -al gbasedbtjdbc_3.3.0_2.jar
-rw-r--r-- 1 root root 2506640 Mar 31 2021 gbasedbtjdbc_3.3.0_2.jar
```

4)、编写测试程序 TestJayDeBeApi.py, 内容如下:

```
#!/usr/bin/python3
# filename: TestJayDeBeApi.py
import sys
import jaydebeapi

print("\nPython JayDeBeApi JDBC 测试程序开始运行.\n")
conn = jaydebeapi.connect("com.gbasedbt.jdbc.Driver",

"jdbc:gsbasebt-sqli://a02.gbasedbt.com:9088/testdb:GBASEDBTSERVER=gsbase01;DB_LOCALE=zh_CN.utf8;CLIENT_LOCALE
```

```
=zh_CN.utf8;IFX_LOCK_MODE_WAIT=60",
                ["gbasedbt", "GBase123"],
                "/root/gbasedbtjdbc_3.3.0_2.jar")

mycursor = conn.cursor()
mycursor.execute("drop table if exists company")

mycursor.execute("create table company(coid serial,coname varchar(255),coaddr text)")

mycursor.execute("insert into company(coname,coaddr) values (?,?)", ('南大通用', '天津市海泰绿色产业基地'))
mycursor.execute("insert into company(coname,coaddr) values (?,?)", ('南大通用北京分公司', '北京市朝阳区太阳宫'))

mycursor.execute("update company set coaddr = ? where coid = 1", ('天津市普天创新园',))

mycursor.execute("select * from company")
rows = mycursor.fetchall()

for i, (coid, coname, coaddr) in enumerate(rows):
    print("公司 ID: %d \t 公司名称: %s\t 公司地址: %s" % (coid, str(coname), coaddr))

mycursor.close()
conn.close()

print("\nPython JayDeBeApi JDBC 测试程序结束运行.\n")
sys.exit(0)
```

5)、执行测试

```
[root@a02 ~]# python3 TestJayDeBeApi.py

Python JayDeBeApi JDBC 测试程序开始运行.

公司 ID: 1          公司名称: 南大通用          公司地址: 天津市普天创新园
公司 ID: 2          公司名称: 南大通用北京分公司      公司地址: 北京市朝阳区太阳宫

Python JayDeBeApi JDBC 测试程序结束运行.
```

3.6. Perl 连接到数据库

3.6.1. DBI::GBASEDBT 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置已经完成。Perl 通过 DBI::GBASEDBT 连接到数据库，需要 perl、

perl-DBI、perl-Test-Pod、perl-Test-Pod-Coverage 和 perl-ExtUtils-MakeMaker 均已经安装。建议使用 yum 方式安装，该方式将自动安装依赖包。

1)、确认 perl 和 perl-DBI 等已经安装

```
[root@localhost perl-dbi]# rpm -qa perl perl-DBI \
perl-Test-Pod perl-Test-Pod-Coverage perl-ExtUtils-MakeMaker
perl-5.16.3-295.el7.x86_64
perl-DBI-1.627-4.el7.x86_64
perl-Test-Pod-Coverage-1.08-21.el7.noarch
perl-ExtUtils-MakeMaker-6.68-3.el7.noarch
perl-Test-Pod-1.48-3.el7.noarch
```

2)、下载 DBD-GBasedbt-2018.1031，并上传解压

下载地址为：<https://gbasedbt.com/dl/DBD-GBasedbt/>

DBD-GBasedbt-2018.1031 基于 DBD-Informix-2018.1031 改写而来

```
[root@localhost perl-dbi]# ll
总用量 300
drwxr-xr-x. 8 501 games 4096 10月 29 00:22 DBD-GBasedbt-2018.1031
-rw-r--r--. 1 root root 299983 10月 29 00:19 DBD-GBasedbt-2018.1031.tar.gz
```

3)、设置临时环境变量

以下 4 个环境变量仅用于编译时的测试，utf8 库应当存在

```
[root@localhost perl-dbi]# export DBI_DBNAME=utf8
[root@localhost perl-dbi]# export DBD_GBASEDBT_DATABASE=utf8
[root@localhost perl-dbi]# export DBD_GBASEDBT_USERNAME=gbasedbt
[root@localhost perl-dbi]# export DBD_GBASEDBT_PASSWORD=GBase123
```

4)、编译 DBI::GBASEDBT，并确认已经安装

```
[root@localhost DBD-GBasedbt-2018.1031]# perl Makefile.PL

[root@localhost DBD-GBasedbt-2018.1031]# make

[root@localhost DBD-GBasedbt-2018.1031]# make install

[root@localhost ~]# perl -MDBD::GBasedbt -le 'print DBD::GBasedbt->VERSION'
2018.1031
```

5)、编写测试程序脚本 TestPerlDbi.pl，内容如下：

```
#!/usr/bin/perl

use warnings;
use strict;
use DBI;
```

```
print "Perl DBI 测试程序开始运行.\n\n";
# 连接到数据库 utf8
my $dbh = DBI->connect
(
    "DBI:GBase:dbt:utf8",
    'gbasedbt',
    'GBase123',
    { PrintError => 0, RaiseError => 1, AutoCommit => 0 }
);

# 开始事务
$dbh->do("begin");

# 删除表及创建表
my $sth_d_tab = $dbh->prepare("drop table if exists company");
$sth_d_tab->execute();

my $sth_c_tab = $dbh->prepare("create table company(coid serial,coname varchar(255),coaddr varchar(255))");
$sth_c_tab->execute();

# 插入数据
my $sth_i_tab = $dbh->prepare("insert into company values (0,'南大通用','天津市海泰绿色产业基地')");
$sth_i_tab->execute();

$sth_i_tab = $dbh->prepare("insert into company values (0,'南大通用北京分公司','北京市朝阳区太阳宫')");
$sth_i_tab->execute();

# 更新数据
my $sth_u_tab = $dbh->prepare("update company set coaddr = '天津市普天创新园' where coid = 1");
$sth_u_tab->execute();

# 查询数据
my $sth_s_tab = $dbh->prepare("select * from company");
$sth_s_tab->execute();

while (my $row = $sth_s_tab->fetchrow_hashref())
{
    print sprintf("%d\t%s\t%s\n", $row->{coid}, $row->{coname}, $row->{coaddr});
}

# 提交事务，关闭数据库连接
$dbh->do("commit");
$dbh->disconnect();
```

```
print "\nPerl DBI 测试程序结束运行.\n";

exit 0;
```

6)、执行测试程序

```
[root@localhost perl-dbi]# perl TestPerlDbi.pl
Perl DBI 测试程序开始运行.

1   南大通用 天津市普天创新园
2   南大通用北京分公司 北京市朝阳区太阳宫

Perl DBI 测试程序结束运行.
```

3.6.2. DBI::ODBC 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置 和 2.4.3 ODBC 的配置已经完成。Perl 通过 DBI::ODBC 方式连接到数据库，需要 perl 及 perl-devel 均已经安装。

1)、确认 perl 及 perl-devel 已经安装

```
[root@localhost perl-odbc]# rpm -qa perl perl-devel
perl-5.16.3-295.el7.x86_64
perl-devel-5.16.3-295.el7.x86_64
```

2)、下载 DBD-ODBC-1.60，并上传解压

下载地址：<https://metacpan.org/pod/DBD%3A%3AODBC>

```
[root@localhost perl-odbc]# ll
总用量 284
drwxr-xr-x. 4 gbasedbt gbasedbt 4096 11月 1 2018 DBD-ODBC-1.60
-rw-r--r--. 1 root      root      282900 10月 29 01:26 DBD-ODBC-1.60.tar.gz
```

3)、编译 DBD-ODBC，并确认安装

需要设置 LANG=C，若设置为 zh_CN.UTF-8 会有个警告。make 操作的过程中有警告，可以忽略。

```
[root@localhost DBD-ODBC-1.60]# export LANG=C
[root@localhost DBD-ODBC-1.60]# perl Makefile.PL

[root@localhost DBD-ODBC-1.60]# make

[root@localhost DBD-ODBC-1.60]# make install

[root@localhost ~]# perl -MDBD::ODBC -le 'print DBD::ODBC->VERSION'
1.60
```

4)、编写测试程序脚本 TestPerlOdbc.pl，内容如下：

```
#!/usr/bin/perl

use warnings;
use strict;
use DBI;

print "Perl ODBC 测试程序开始运行.\n\n";
# 通过 ODBC 连接到数据库 utf8
my $dbh = DBI->connect
(
    "DBI:ODBC:utf8",
    'gbasedbt',
    'GBase123'
);

# 开始事务
$dbh->do("begin");

# 删除表及创建表
my $sth_d_tab = $dbh->prepare("drop table if exists company");
$sth_d_tab->execute();

my $sth_c_tab = $dbh->prepare("create table company(coid serial,coname varchar(255),coaddr varchar(255))");
$sth_c_tab->execute();

# 插入数据
my $sth_i_tab = $dbh->prepare("insert into company values (0,'南大通用','天津市海泰绿色产业基地')");
$sth_i_tab->execute();

$sth_i_tab = $dbh->prepare("insert into company values (0,'南大通用北京分公司','北京市朝阳区太阳宫')");
$sth_i_tab->execute();

# 更新数据
my $sth_u_tab = $dbh->prepare("update company set coaddr = '天津市普天创新园' where coid = 1");
$sth_u_tab->execute();

# 查询数据
my $sth_s_tab = $dbh->prepare("select * from company");
$sth_s_tab->execute();

while (my $row = $sth_s_tab->fetchrow_hashref())
{
    print sprintf("%d\t%s\t%s\n", $row->{coid}, $row->{coname}, $row->{coaddr});
}
```

```
}

# 提交事务，关闭数据库连接
$dbh->do("commit");
$dbh->disconnect();

print "\nPerl ODBC 测试程序结束运行.\n";

exit 0;
```

5)、执行测试程序

```
[root@localhost perl-odbc]# perl TestPerlOdbc.pl
Perl ODBC 测试程序开始运行.

1    南大通用 天津市普天创新园
2    南大通用北京分公司 北京市朝阳区太阳宫

Perl ODBC 测试程序结束运行.
```

3.7. GO 连接到数据库

3.7.1. ODBC 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置 和 2.4.3 ODBC 的配置已经完成。GOLANG 通过 ODBC 连接到数据库时，需要安装 GOLANG 和 alexbrainman/odbc 或者 GO-ODBC（两者选一即可，推荐使用 alexbrainman/odbc）。

1)、安装 GO

安装 GO 可通过码方式，或者是注册 epel 库后使用 yum 源方式安装。

下载地址：<https://golang.google.cn/dl/>

```
[root@localhost ~]# wget https://dl.google.com/go/go1.15.3.linux-amd64.tar.gz

[root@localhost ~]# tar -C /usr/local -xzf go1.15.3.linux-amd64.tar.gz
```

设置环境变量文件（也可以写入到/etc/profile 中）

```
# go 的目录是 GOROOT, GOPATH 是工作空间
export GOROOT=/usr/local/go
export PATH=$GOROOT/bin:$PATH
export GOPATH=/root/go-odbc
```

2)、安装 alexbrainman/odbc（选项一，推荐）

通过 go get 自动安装 ODBC，将会安装在\$GOPATH/pkg 目录下。

```
[root@localhost ~]# go get github.com/alexbrainman/odbc
```

3)、安装 GO-ODBC (选项二)

在\$GOROOT/src 目录下安装 GO-ODBC

GO-ODBC 的下载地址：[git://github.com/weigj/go-odbc.git](https://github.com/weigj/go-odbc.git)

```
[root@localhost ~] cd $GOROOT/src
[root@localhost src]# git clone git://github.com/weigj/go-odbc.git odbc

[root@localhost src]# cd odbc
[root@localhost odbc]# go install

[root@localhost odbc]# go list
odbc
```

4)、在 GOPATH 目录下编写测试程序脚本 TestGoOdbc.go

```
package main

import (
    "database/sql"
    "fmt"
    // 使用 go-odbc
    // _ "odbc/driver"
    // 使用 alexbrainman/odbc
    _ "github.com/alexbrainman/odbc"
)

func main() {
    fmt.Printf("%s\n\n", "GO ODBC 测试程序开始运行.")
    // 对于 CM 和直连，使用 DSN 方式；若是直连，可使用 DSN-LESS 方式
    conn, err := sql.Open("odbc", "DSN=utf8;UID=gbasedbt;PWD=GBase123;")
    if err != nil {
        fmt.Println(err)
        fmt.Println("链接错误")
        return
    }
    defer conn.Close()
    fmt.Printf("%s\n\n", "构建查询")
    stmt, err := conn.Prepare("select tabid,tabname from systables where tabid < ?")
    if err != nil {
        fmt.Println("查询异常: ", err)
        return
    }
}
```

```
defer stmt.Close()
row, err := stmt.Query(11)
if err != nil {
    fmt.Println("查询错误: ", err)
}
defer row.Close()
fmt.Printf("%s\n", "数据集显示")
for row.Next() {
    var tabid int
    var tabname string
    if err := row.Scan(&tabid, &tabname); err == nil {
        fmt.Printf("%d\t%s\n", tabid, tabname)
    }
}
fmt.Printf("\n%s\n", "GO ODBC 测试程序结束运行.")
return
}
```

5)、执行测试程序

```
[root@localhost go-odbc]# go run TestGoOdbc.go
```

GO ODBC 测试程序开始运行.

构建查询

数据集显示

```
1  systables
2  syscolumns
3  sysindices
4  systabauth
5  syscolauth
6  sysviews
7  sysusers
8  sysdepend
9  syssynonyms
10 syssyntable
11 sysconstraints
12 sysreferences
```

GO ODBC 测试程序结束运行.

3.8. C++ 连接到数据库

3.8.1. SOCI 方式连接到数据库-Linux

要求 2.4.2 CSDK 的配置 和 2.4.3 ODBC 的配置已经完成。

1)、安装 SOCI

SOCI 需要预先安装 cmake 2.8+ (建议 cmake3), gcc-c++,

下载地址: <https://github.com/SOCI/soci.git> 或者

<https://sourceforge.net/projects/soci/files/soci/> (建议)

```
[root@localhost ~]# wget https://netactuate.dl.sourceforge.net/project/soci/soci/soci-4.0.1/soci-4.0.1.tar.gz
[root@localhost ~]# tar -zxvf soci-4.0.1.tar.gz && cd soci-4.0.1
```

设置环境变量文件 (也可以写入到/etc/profile 中)

```
# 增加 LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/usr/local/lib64:$LD_LIBRARY_PATH
```

编译安装 SOCI, 建议使用 cmake3 (EPEL 源可直接安装)

```
[root@localhost soci-4.0.1]# cmake3 -DWITH_BOOST=ON
[root@localhost soci-4.0.1]# make
[root@localhost soci-4.0.1]# make install
```

2)、编写测试程序脚本 TestSOCIodbc.cpp

```
#include <soci/soci.h>
#include <soci/odbc/soci-odbc.h>
#include<iostream>
#include<istream>
#include<ostream>
#include<string>
#include<exception>

using namespace std;
using namespace soci;

int main() {
    cout<<"SOCI 测试程序开始运行.\n\n";
    try {
        session
        sql(odbc,"Driver=/opt/gbase8s-odbc-driver/lib/cli/iclit09b.so;HOST=a02.gbasedbt.com;SERV=9088;PROT=onsocketcp;
```

```
SRVR=gbase01;DB=testdb;UID=gbasedbt;PWD=GBase123;DLOC=zh_CN.utf8;CLOC=zh_CN.utf8");  
  
// 删除和创建表  
sql<< "drop table if exists company";  
sql<< "create table company(coid serial,coname varchar(255),coaddr varchar(255))";  
  
// 插入记录  
string cn = "南大通用";  
string ca = "天津市海泰绿色产业基地";  
sql<< "insert into company(coname,coaddr) values (:cn, :ca)", use(cn), use(ca);  
  
cn = "南大通用北京分公司";  
ca = "北京市朝阳区太阳宫";  
sql<< "insert into company(coname,coaddr) values (:cn, :ca)", use(cn), use(ca);  
  
// 查询记录  
rowset<row> rs = (sql.prepare << "select * from company");  
for (rowset<row>::iterator it = rs.begin(); it != rs.end(); ++it)  
{  
    const row& row = *it;  
    cout << "coid:" << row.get<int>(0)  
        << " coname:" << row.get<string>(1)  
        << " coaddr:" << row.get<string>(2) << endl;  
}  
} catch (exception const &e) {  
    cerr<<"Error:" <<e.what()<<endl;  
}  
cout<<"\nSOCI 测试程序结束运行.\n";  
}
```

3)、编译程序，执行测试程序

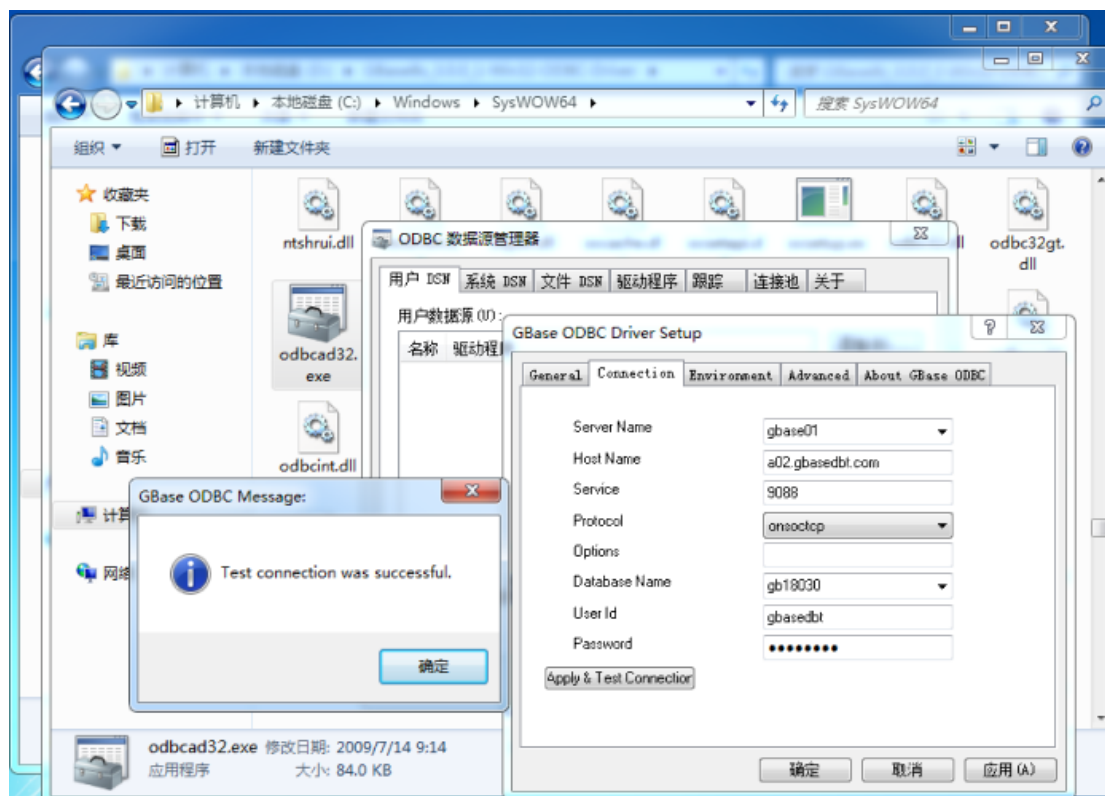
```
[root@localhost ~]# g++ -Wall -g -o TestSOCI0dbc TestSOCI0dbc.cpp -lsoci_core -lsoci_odbc  
[root@localhost ~]# ./TestSOCI0dbc  
SOCI 测试程序开始运行.  
  
coid:1 coname:南大通用 coaddr:天津市海泰绿色产业基地  
coid:2 coname:南大通用北京分公司 coaddr:北京市朝阳区太阳宫  
  
SOCI 测试程序结束运行.
```

3.9. Delphi 连接到数据库

3.9.1. ODBC 方式连接到数据库-Windows

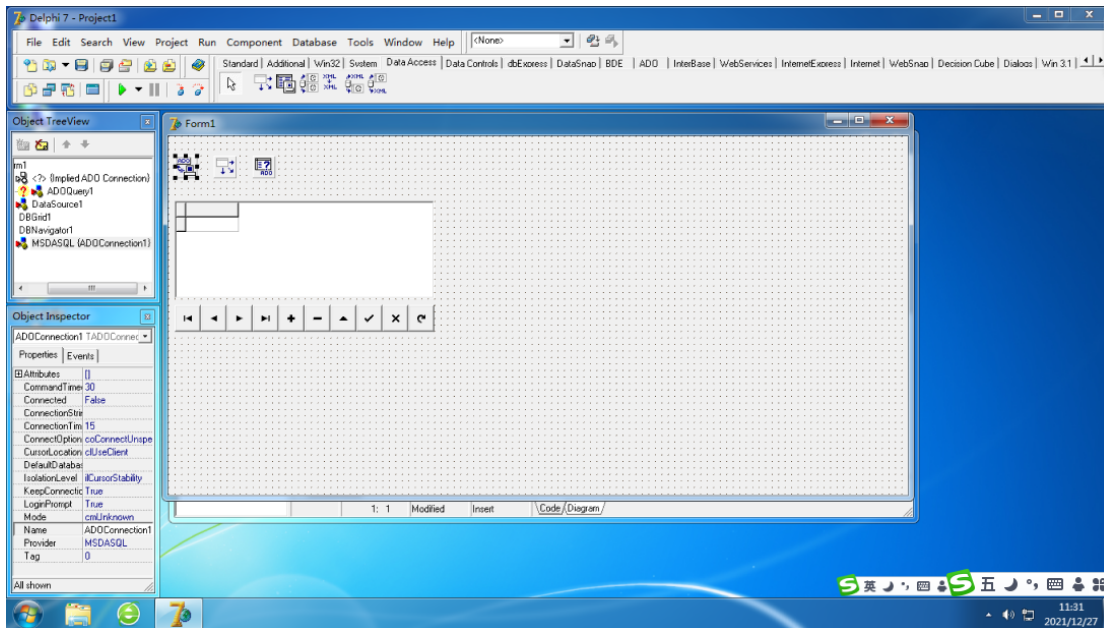
要求 2.2.2 CSDK 的配置 和 2.2.3 ODBC 的配置已经完成。

1)、确认 ODBC 能正常连接到数据库

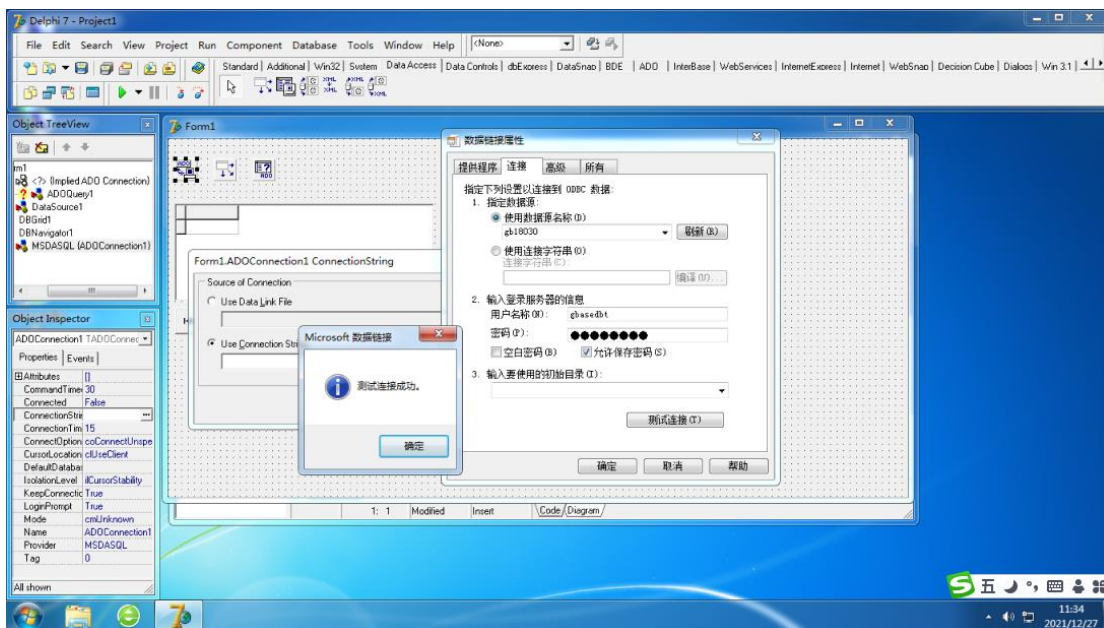


2)、在 Delphi 中新建工程

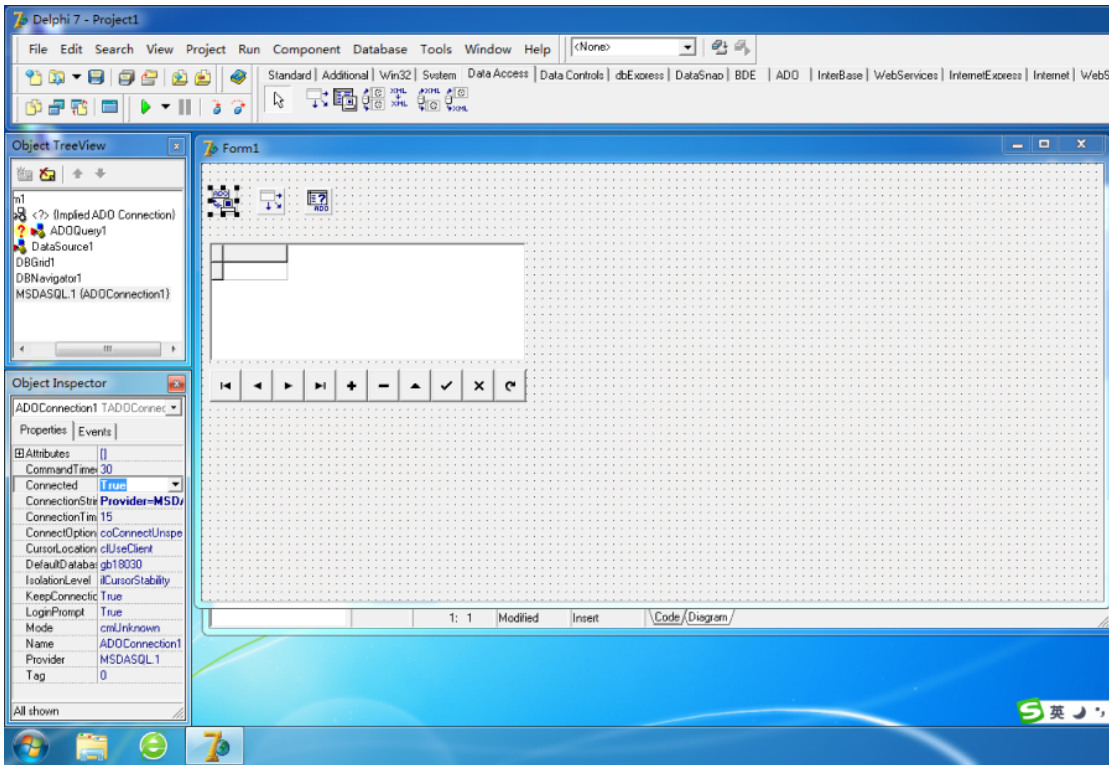
增加 ADOConnection, DBSource, ADOQuery, DBGrid 和 DBNavigator 控件到 Form 上



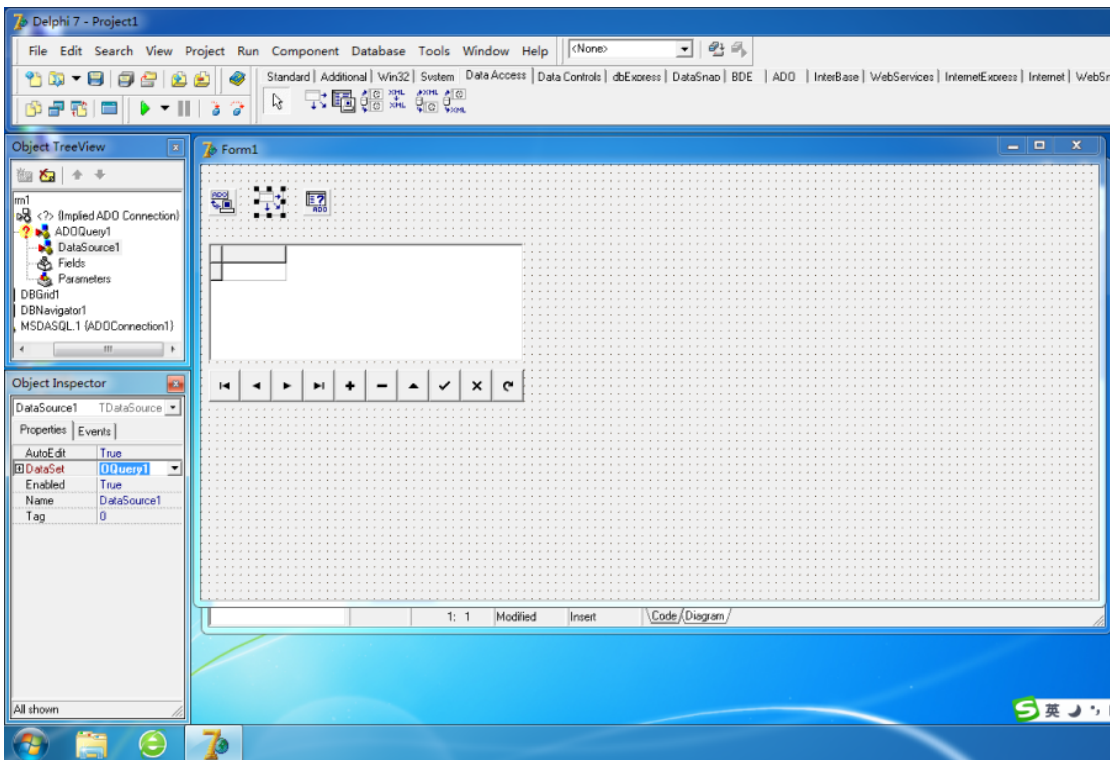
在 ADOConnection 控件 ADOConnection1 的 ConnectionString 中使用连接字符串 (Microsoft OLE DB Provider for ODBC Drivers), 选择 ODBC 中测试的连接, 测试连接成功表示可以正常连接。



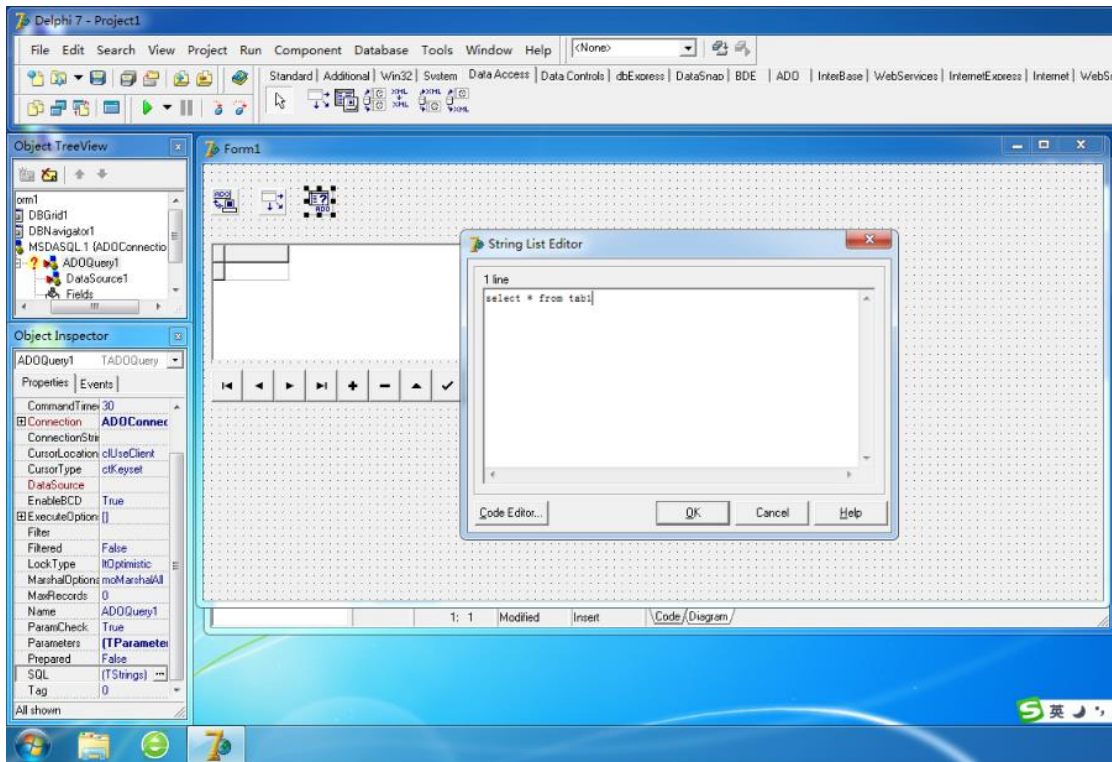
在 Connected 中选上 True, 表示连接上数据库



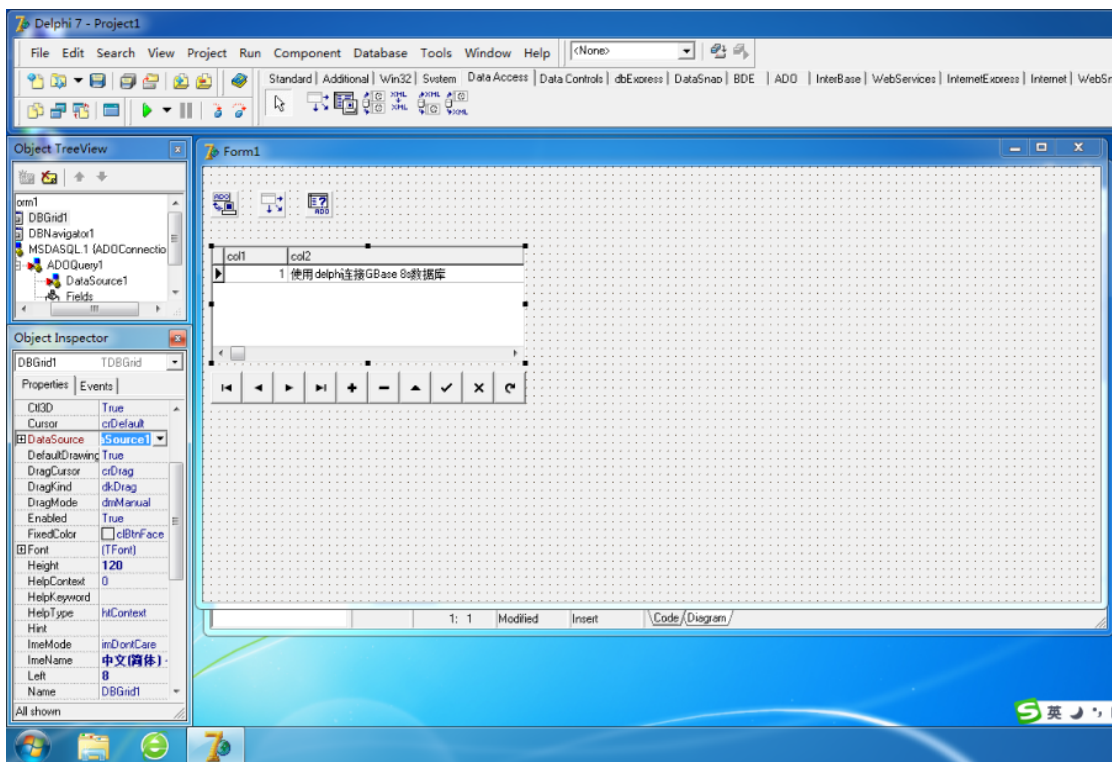
在 DBSource 控件 DBSource1 中，设置 DataSet 的值为 ADOQuery1



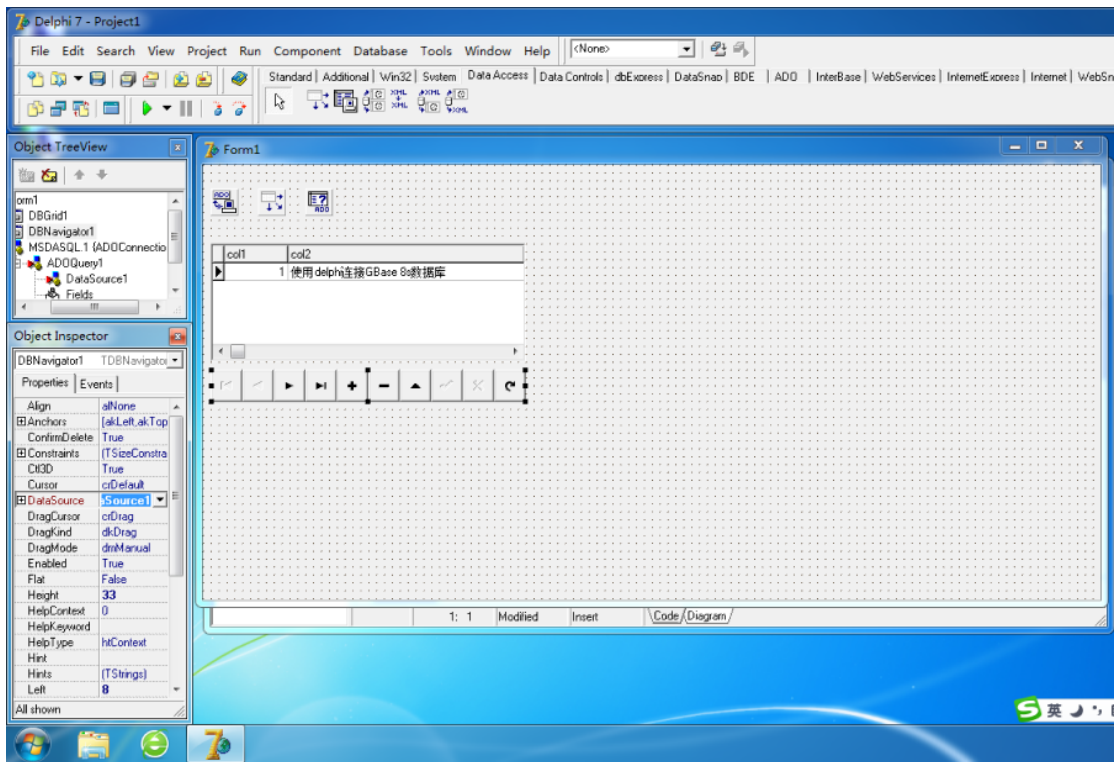
在 ADOQuery 控件 ADOQuery1 中，选择 Connection 为 ADOConnection1，SQL 为需要操作的数据库表查询语句，并把 Active 设置为 True



将 DBGrid 控件 DBGrid1 的 DataSource 的值设置为 DataSource1



将 DBNavigator 控件 DBNavigator1 的 DataSource 的值也设置为 DataSource1



完成操作

3)、运行工程，执行测试

