PROGRESS® DataDirect®

USE LINQ AND ENTITY SQL QUERIES ON SYBASE 15.5

INTRODUCTION

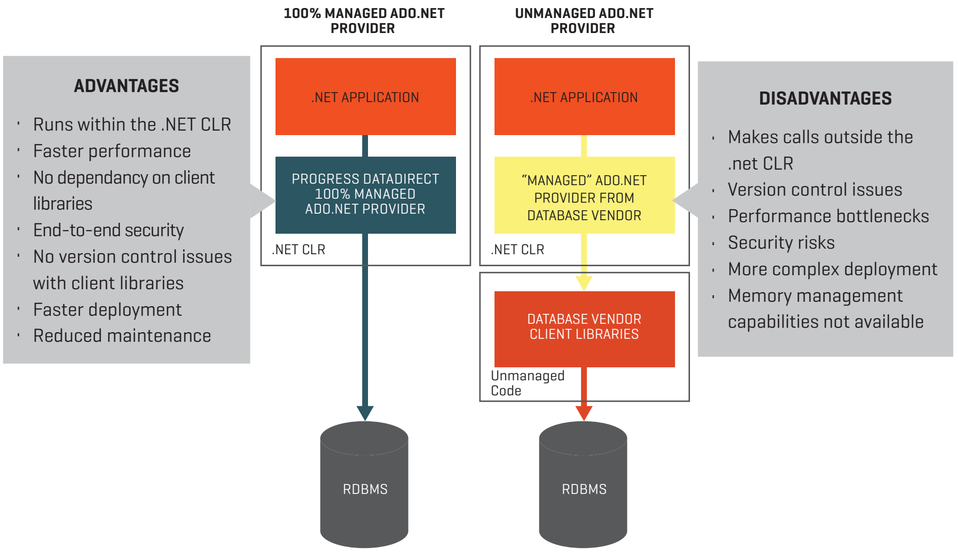
In this article, we explore how easy it is to connect Microsoft Visual Studio to a Sybase 15.5 server by using Progress® DataDirect® Connect *for* ADO.NET in conjunction with the ADO.NET Entity Framework. This configuration lets you fully exploit the power of Microsoft's ADO.NET Entity Data Framework when working with Sybase.

- **Visual Studio setup**
- **Install and test Progress DataDirect Connect *for* ADO.NET**
- **Create a new console-based project**
- **Add the ADO.NET Data Model**
- **Create a LINQ-style query**
- **Create an Entity SQL query**

While new architectures that interact with data are continually arising, one constant remains: the underlying technologies that make access to information possible remain as critical as ever. In this article, you'll see that all communication with the database is handled within the CLR (Common Language Runtime) leading to better performance, security, and reliability. What makes Progress DataDirect Connect *for* ADO.NET such a compelling choice for this task is that its 100% managed code architecture runs entirely within the CLR and eliminates the need to install, configure, or maintain any additional client-side software such as Sybase's client libraries, as shown in Figure 1.

HIGHLIGHTS:

- Use Progress® DataDirect® Connect *for* ADO.NET with the ADO.NET Entity Framework
- Increase performance, security and reliability
- Use a Language-Integrated Query (LINQ) as a cross-platform mechanism to access information
- Retrieve data from multiple data with Entity SQL


100% MANAGED ADO.NET PROVIDER
UNMANAGED ADO.NET PROVIDER
ADVANTAGES
Runs within the .NET CLR
Faster performance
No dependancy on client libraries
End-to-end security
No version control issues with client libraries
Faster deployment
Reduced maintenance
.NET APPLICATION
PROGRESS DATADIRECT 100% MANAGED ADO.NET PROVIDER
.NET CLR
.NET APPLICATION
"MANAGED" ADO.NET PROVIDER FROM DATABASE VENDOR
.NET CLR
DATABASE VENDOR CLIENT LIBRARIES
Unmanaged Code
DISADVANTAGES
Makes calls outside the .net CLR
Version control issues
Performance bottlenecks
Security risks
More complex deployment
Memory management capabilities not available
RDBMS
RDBMS

PROGRESS

You've probably heard a lot about the ADO.NET Entity Framework. The purpose of this article is to help you make use of it in your Sybase environment. To illustrate your options, we'll work with two examples. In the first instance, you'll use a Language-Integrated Query (LINQ) as a common, cross-platform mechanism to access information. In the second scenario, you'll see how Entity SQL (which is similar to Microsoft's Transact-SQL) can retrieve data from a multi-table relationship.

Finally, to keep things as simple and straightforward as possible, code samples of each of the examples referenced in this document are available. Click here to download the LINQ and Entity SQL to Sybase 15.5 code samples.

INSTRUCTIONS

1. Install Visual Studio Professional 2008 SP1 Visual Studio Professional 2010. If you already have Visual Studio on your computer, make sure to download the .NET Framework 3.5 SP1 or 4.0; the wizards and other key components won't work without it.
2. Install Progress DataDirect Connect *for* ADO.NET Please note that this article was written with version 3.5. If you are using a newer version, you can follow these same steps as well.

   Make sure that you have administrative privileges on this computer. In addition, before installing the DataDirect product, make sure you're running one of the following operating systems:

- Windows Server 2008 (all Editions)
- Windows Vista (all Editions)
- Windows XP (all Editions)
- Windows Server 2003 (all Editions)

   Progress DataDirect Connect *for* ADO.NET supports the 32 and 64 bit versions of these operating systems. Don't worry: the installer takes care of this for you automatically.

3. Launch Visual Studio, and create a new project named EntityFramework_ Console_LINQ_Demo, using a Visual C# Console Application.

   Make sure that you've selected .NET Framework 3.5 or 4.0 from the dropdown on the upper right side of the New Project dialog box. Also, the source code examples below assume that you used the same names for your projects as we do in this tutorial

4. Add the ADO.NET Data Model to your project by following these steps:

   a. Right-click on the name of your new project.

   b. Choose Add-> New Item.

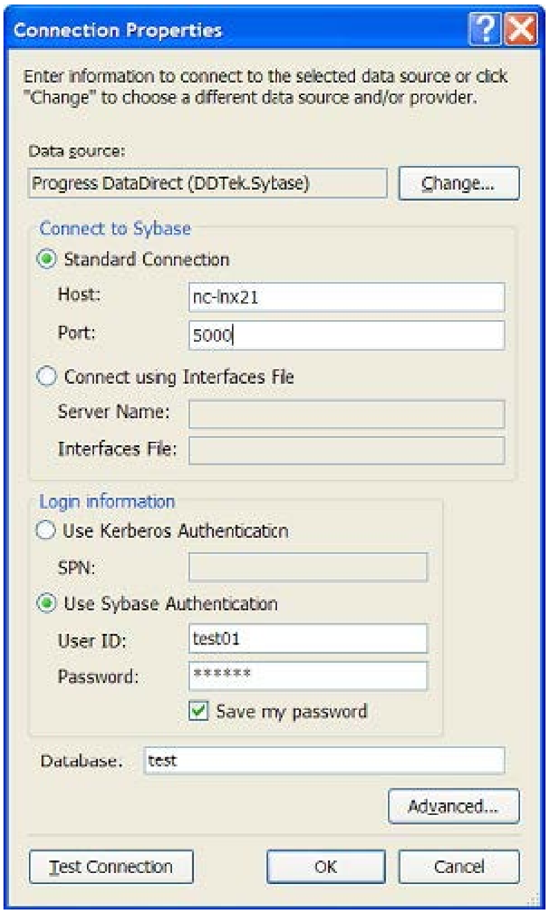
   c. Select the ADO.NET Entity Data Model template. This launches the Entity Data Model Wizard.

   d. Choose the Generate from database option.

   e. Click on the New Connection button.

   f. Fill in details about your Sybase connection as shown in figure 2. Make sure to select the Progress DataDirect Connect *for* ADO.NET Sybase Data Provider.

Figure 2


Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Progress DataDirect (DDTek.Sybase)

Connect to Sybase
[✓ Standard Connection]
Host: nc-lnx21
Port: 5000
Connect using Interfaces File
Server Name:
Interfaces File:

Login information
Use Kerberos Authentication
SPN:
[✓ Use Sybase Authentication]
User ID: test01
Password: ******
[✓ Save my password]

Database: test

PROGRESS

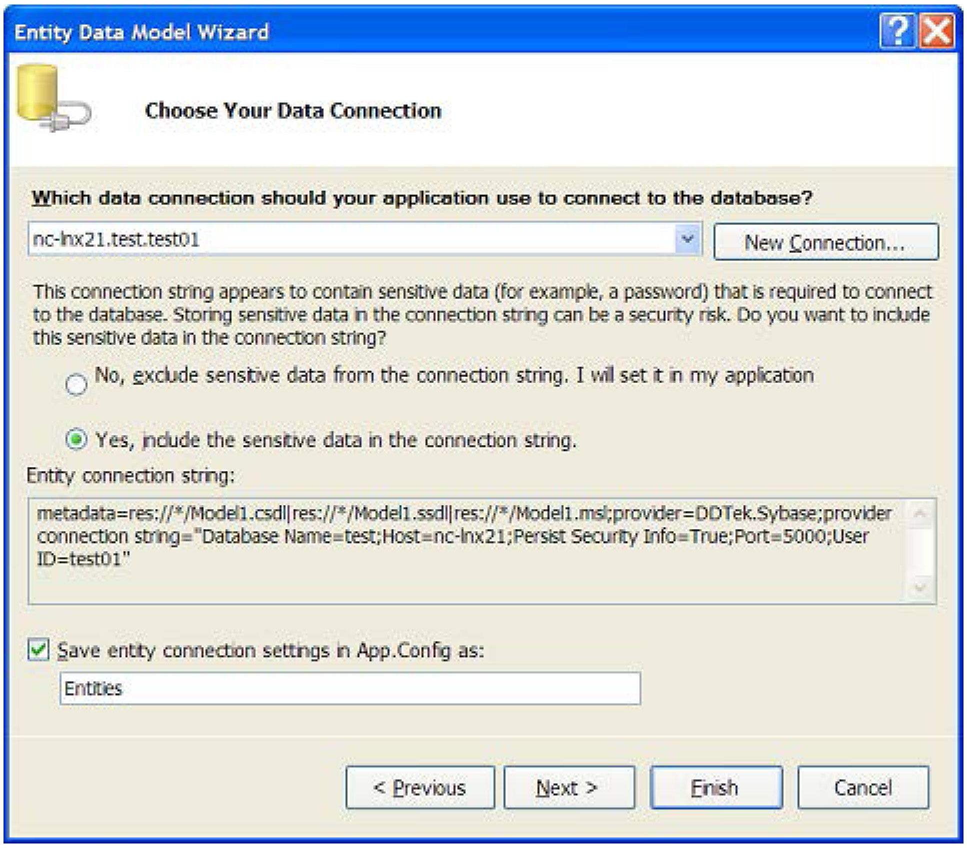
When you've finished, click on the Test Connection button to double-check that everything is configured correctly. If you receive a Test connection succeeded message, move on to these steps:

a. Click the Yes, include the sensitive data in the connection string. radio button

b. Check the Save entity connection settings in App.Config box, and enter Entities in the text box as shown in figure 3.

c. Click Next. The Wizard connects to the database and presents a list of the objects that you can include in your model.

d. Expand the Tables entry, and place check marks next to the EMPLOYEES (TEST01) and JOBS (TEST01) entries. Use the default namespace called Model.



Which data connection should your application use to connect to the database?

nc-lnx21.test.test01

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

No, exclude sensitive data from the connection string. I will set it in my application [✓ Yes, include the sensitive data in the connection string.]

Entity connection string:

metadata=res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;provider=DDTek.Sybase;provider connection string="Database Name=test;Host=nc-lnx21;Persist Security Info=True;Port=5000;User ID=test01"

✓ Save entity connection settings in App.Config as:

Entities

Figure 3

e. Click Finish.

The Wizard now connects to Sybase, deciphers the relationships between these two tables, and then creates a data model based on what it learned. There are a few important points to bear in mind about the interaction between the Entity Data Model Wizard and Sybase 15.5:

- It may take a few minutes for the Wizard to complete its work.
- The Wizard must be able to infer primary and foreign keys from your Sybase schema.
- Be on the lookout for any diagnostic messages from Entity Framework and related Wizard regarding your database structure.

- If your Sybase tables use niche data types, such as UDT (user-defined types), you may receive some messages from the Entity Framework.
- The Entity Framework mandates that your stored procedure parameters need to be comprised of supported data types.

When the Wizard completes its work, click the Show all Files icon at the top of the Solution Explorer. Notice that the Wizard has created numerous references, an App.Config file, and a data model.

Figure 4 shows the relationship mapping between the two tables as visualized in the ADO.NET Entity Data Model Designer, the references and other files that were created, as well the Model Browser:

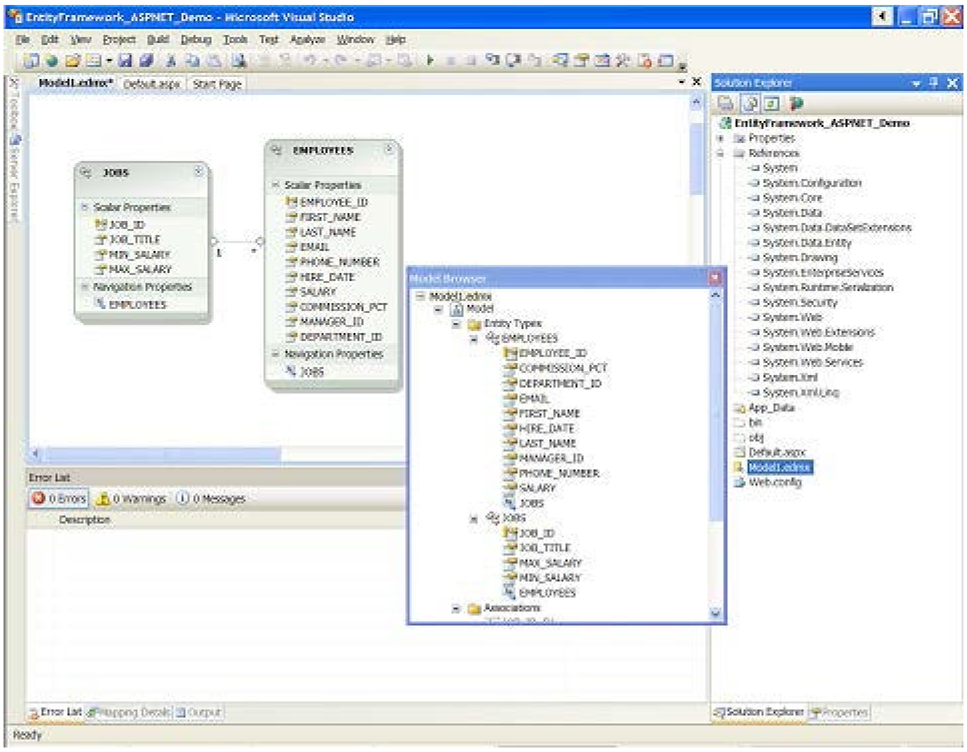


Figure 4

5. Save your project.

   In the upcoming steps, you'll see how to issue LINQ and Entity SQL queries that use this model.

6. Create a LINQ-style query

   LINQ lets you write queries directly within your application logic, rather than needing to write SQL itself. This approach is especially compelling when you're faced with diverse dialects of SQL, typically provided by different database vendors. For example, you could use LINQ to write platform-independent queries that will work with disparate databases such as SQL Server, MySQL, DB2, and Oracle; the Entity Framework will handle all of the database-specific logic.

This sample application conducts a simple query of the EMPLOYEES entity that was discovered by the Entity Data Model Wizard. To run the example, all that's required is to open the Program.cs file that was generated for you when you created your project. Delete all of the code in that file, and then just copy and paste the following code as a replacement:

```
using System;
using System.Collections.Generic;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.Objects; namespace EntityFramework _
Console _ LINQ _ Demo
{
class Program
{
  static void Main(string[] args)
  {
    using (Entities EmpEntities = new Entities())
    {
      ObjectQuery<EMPLOYEES> OQemployees = EmpEntities.
EMPLOYEES;
      IQueryable<EMPLOYEES> employeesQuery = from EMPLOYEE
in OQemployees select EMPLOYEE;
      Console.WriteLine("Employees:");
      foreach (var emp in employeesQuery)
      {
        Console.WriteLine(emp.LAST _ NAME + ", " + emp.
FIRST _ NAME);
      }
      Console.ReadLine();
    }
  }
}
}
```

Once you've finished, save your project and then click the green Start Debugging icon at the top of the main Visual Studio window. This launches the sample application, which presents a console window with a list of all employees.

7. Create an Entity SQL query.

The second scenario showcases Entity SQL. This is similar in nature to Microsoft's Transact-SQL, with some important enhancements, such as inheritance/relationship support as well as being able to work with collections.

In this example, you'll explore two additional capabilities: filtering a query, and retrieving data from related tables. When the Entity Data Model Wizard evaluated your Sybase database, it determined that there was a relationship between the EMPLOYEES and JOBS tables. These relationships are displayed graphically, as well as presented to you via IntelliSense tips when you write your application logic.

To run this sample application, create a new project and data model. Call your project EntityFramework_Console_ESQL_Demo and follow steps 3 through 5 above.

Once you've completed those steps, replace all of the code in the Program.cs file with the following:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.Common;
using System.Data.Objects;
using System.Data.Objects.DataClasses; namespace
EntityFramework _
Console _ ESQL _ Demo
{
class Program
{
  static void Main(string[] args)
  {
    using (Entities jobEntities = new Entities())
    {
      string esqlQuery = @"SELECT j.job _ title, j.EMPLOYEES
FROM Entities.jobs AS j where j.job _ id = 'IT _ PROG'";
      try
      {
        foreach (DbDataRecord result in new
ObjectQuery<DbDataRecord> (esqlQuery, jobEntities))
        {
          Console.WriteLine("Job title {0}: ", result[0]);
          List<EMPLOYEES> list = result[1] as
List<EMPLOYEES>;
          foreach (EMPLOYEES emp in list)
          {
            Console.WriteLine(" Name {0}, {1} Phone {2}",
emp.LAST _ NAME, emp.FIRST _ NAME, emp.PHONE _ NUMBER);
          }
        }
      }
      catch (EntityException trouble)
      {
        Console.WriteLine(trouble.ToString());
      }
      catch (InvalidOperationException trouble)
      {
        Console.WriteLine(trouble.ToString());
      }
    }
    Console.ReadLine();
  }
}
}
```

Once you've finished, save your project and then click the green Start Debugging icon at the top of the main Visual Studio window. This launches the sample application, which retrieves a list of all employees who have a job identifier of IT_PROG.

AUTHOR

Robert Schneider is a Silicon Valley-based technology consultant. He has written five books and numerous articles on advanced technical topics such as Service Oriented Architecture (SOA), open source, and relational database design/optimization.

He can be reached at Robert.Schneider@think88.com.