Access the database via C # ADO.NET

In this article, I will show you how to access the database via C # ADO.NET.

In this article, I will show you how to access the database via C # ADO.NET.

Database and object-oriented language

Before mini computers and PCs came out, the IT world was much simpler. At that time, it was thought that only very large organizations needed computer technology. However, a few years ago, we could all have mobile phones with strong computing power that could handle data for a missile firing process on the moon. And to this day, computing power, software, storage capacity and bandwidth are all interchangeable items. However, there is still something quite valuable in the computer field. The problem is, the lack of a universal platform is also a way to promote innovation and innovation in the wider field. In the form of software standards, we have an impressive list of choices, see the form of open source and very popular platforms like Linux and iPhone. If there is only one operating system and one main programming language, there will also be little competition - that is often coupled with having few choices.

For the world, relational databases also have a wide range of competing products. Program access to these database products also presents many options —Java Persistence API (JPA) / Hibernate, C ++, C #, Java,… In this article, we will investigate the public C # and ADO.NET technologies in a special case, the goal is to see what these technologies can provide for programmers. However, if you do not have much knowledge about the database, you should not be too worried because we will introduce from the most basic problems.

Note: The source code for this article is available for download here as a C # solution for Microsoft Visual C # 2008 Express Edition. If you want to run this source yourself, just extract the files into a directory, such as C: dbcode. Then open the solution file named DBConsoleApplication.sln in Microsoft Visual C # 2008 Express Edition. This will automatically build a source code and create the necessary executables.

Here we will intervene and deploy a database product.

Install SQL Server

Installing and running the program can take a long time, so we don't want to spend your time on this work. For the purpose of focusing on the use of a professional product, this article therefore chose to use SQL Server 2005 Express Edition. You can use the 2008 version if you like - the examples in this article will work on all 2008 platforms.

Let's start by downloading and installing the following three items from the Microsoft website:

- Microsoft SQL Server 2005 Express Edition

When installing SQL Server 2005 Express Edition, use the default settings; These settings will automatically create an instance for the database engine.

- Microsoft SQL Server Management Studio Express

  For SQL Server Management Studio Express, no special configuration is required - just accept the default configuration. When the product is installed, it will automatically detect the instance created by SQL Server 2005 Express Edition.

- Microsoft Visual C # 2008 Express Edition

  Code examples will be built and tested using Visual C # 2008 Express Edition.

All of these steps seem complicated at first, but they are not. Let's install new tools on and run them.

Launch SQL Server Management Studio Express

The SQL Server Management Studio Express product allows you to manage SQL Server instances. That is, you can interact with the configured database on the given SQL Server instance. Run SQL Server Management Studio Express from the Windows **Start** menu. The actual program icon for SQL Server Management Studio Express is installed as a submenu item in the Microsoft SQL Server 2005 programs group: **Start> All Programs> Microsoft SQL Server 2005** .

When starting the SQL Server Management Studio Express application, a connection is automatically created for the installed SQL Server instance, as shown in Figure 1. The server name in Figure 1 is taken from the name of the host machine (in the field combination, **LAPTOP1**) and instance of SQL Server (**SQLEXPRESS** in this example).

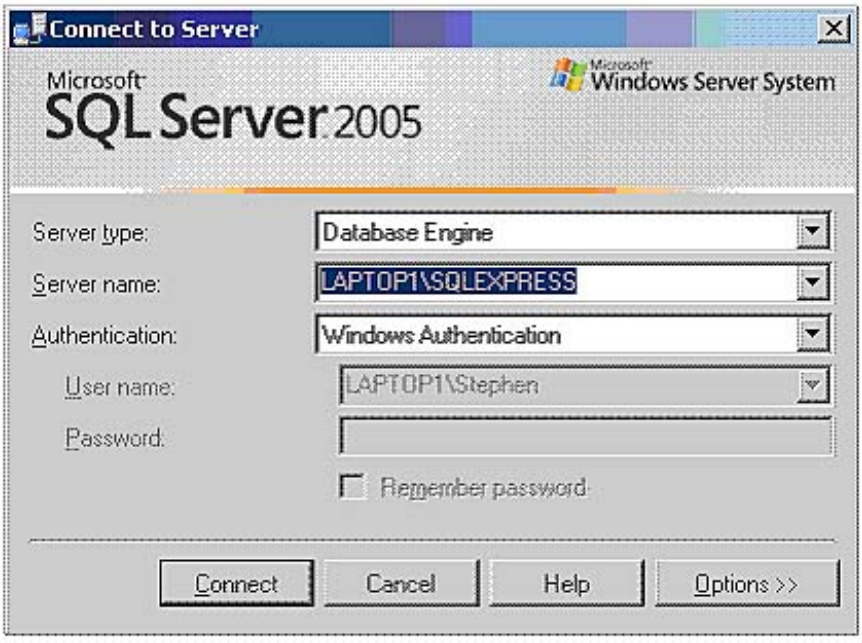

Figure 1: Connecting to an instance of SQL Server

When finished, click the **Connect** button in the dialog box. After a minute, the main **SQL Server Management Studio Express screen** will appear (see Figure 2).

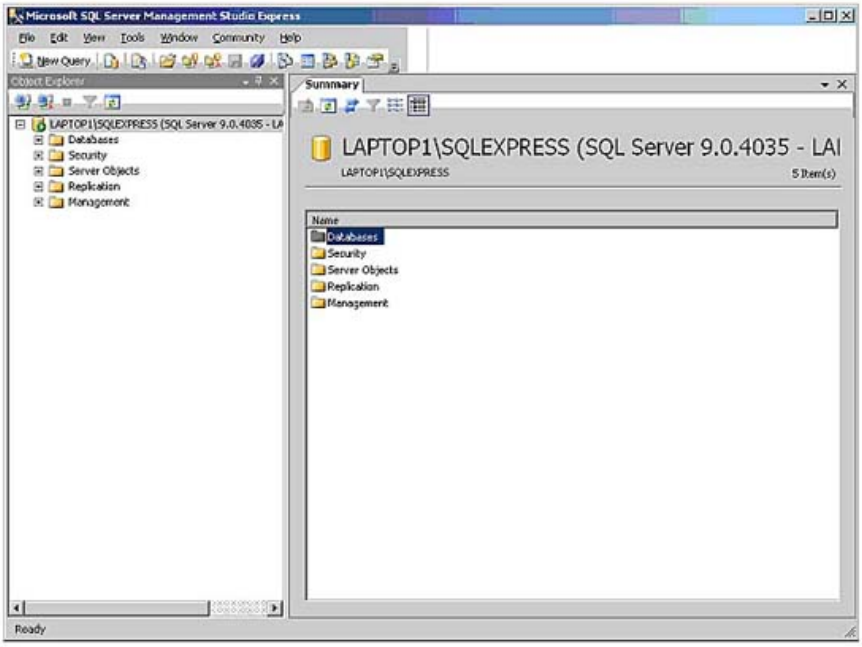

Figure 2: Successfully connecting to SQL Server instance

So what exactly is the SQL Server Management Studio Express application? In Figure 2, notice the ***Object Explorer*** shown at the top of the left pane - this program is like an explorer application type to allow you to see database engine instances. As in Windows, you can view a significant amount of information related to all database instances here. Let's learn a little bit about the Object Explorer window.

In the left pane, click the plus sign to the left of the **Databases** folder icon. The action will display all the databases hosted by the given SQL Server instance (see Figure 3).

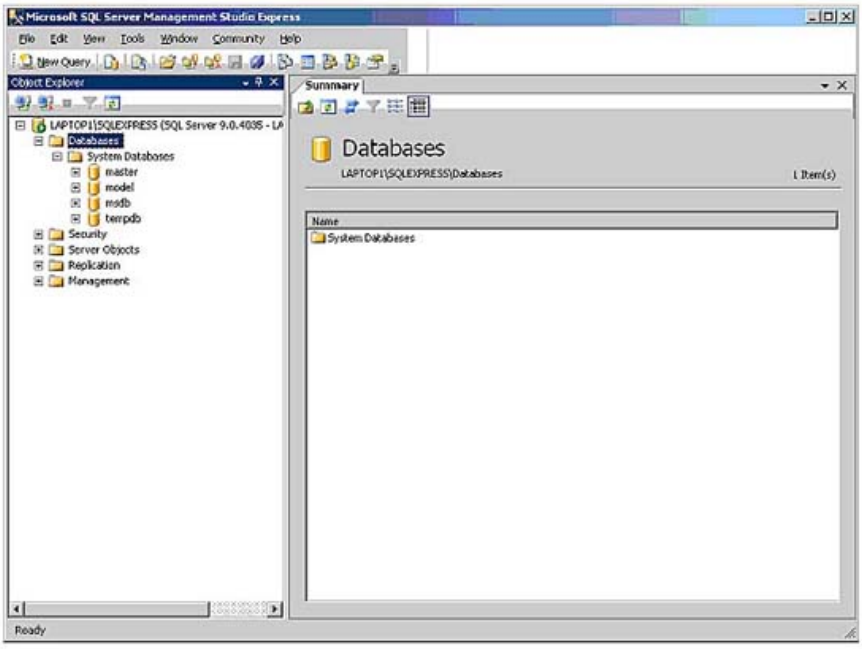

Figure 3: Database resides on SQLEXPRESS.

The first thing to note about the databases in Figure 3 is that they all relate to SQLEXPRESS; In other words, there is no application database. Let's fix this situation by installing a new database.

Microsoft has provided many sample databases for using SQL Server products. One of these we use for this example is pubs. You can download the pubs database here and install it to use for your own experiences, or you can use your own database.

When you run the installation file, you will deploy the pubs database, but you must attach it in SQL Server Management Studio Express. This step may sound difficult, but this attachment is really easy. Just click the File button and browse to the file instpubs.sql (see Figure 4). Our file is located according to the link below:

```
C: SQL Server 2000 Sample Databasesinstpubs.sql
```

Select the file and click **Open** . When the script file is opened, click the **Execute** button in the toolbar (see Figure 5).

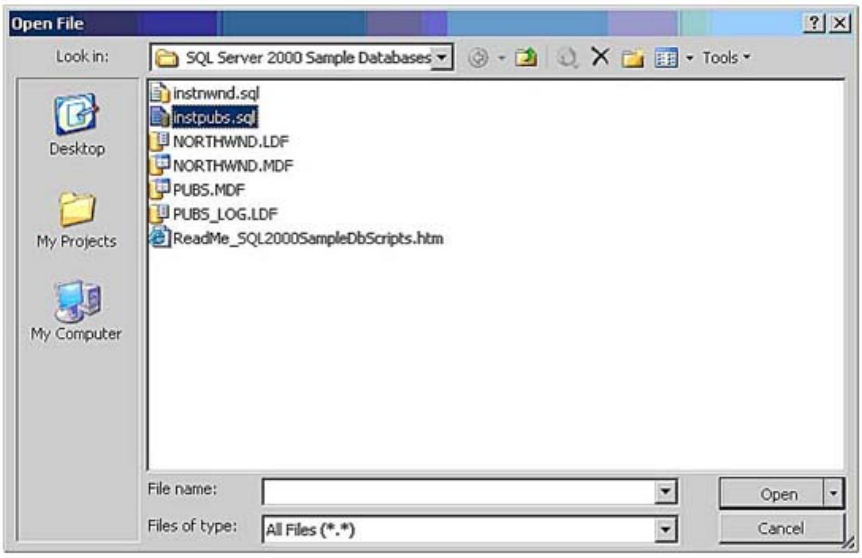

Figure 4: Running the pubs database script

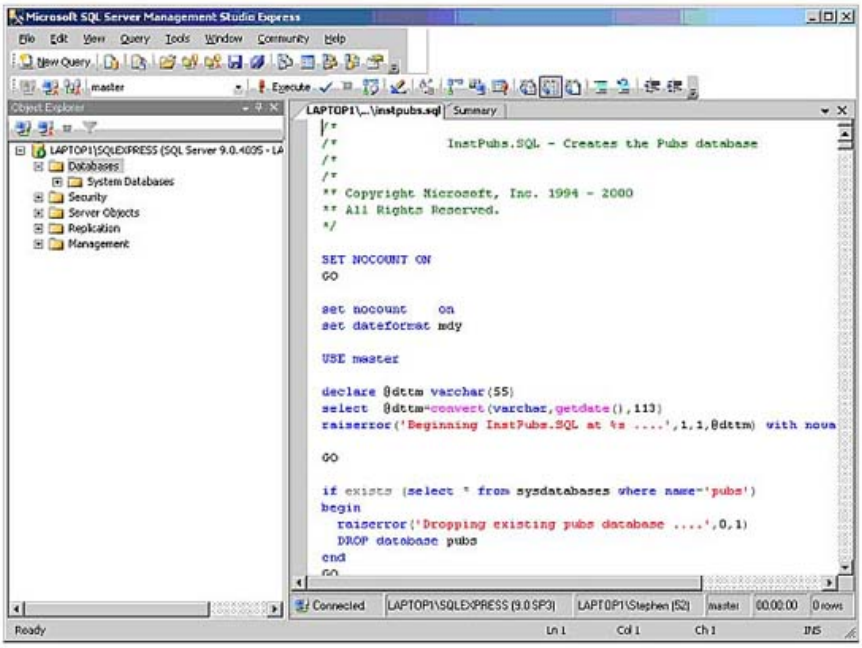

Figure 5: Before instpubs.sql file is executed

If all goes well with your installation, a new database called pubs will appear in the list to the left of the panel, as shown in Figure 6.

Note: To see the output in Figure 6, you can click on **database instance** (**LAPTOP1SQLEXPRESS**) in the left pane of the **Object Explorer** window, then click the **Refresh** button.

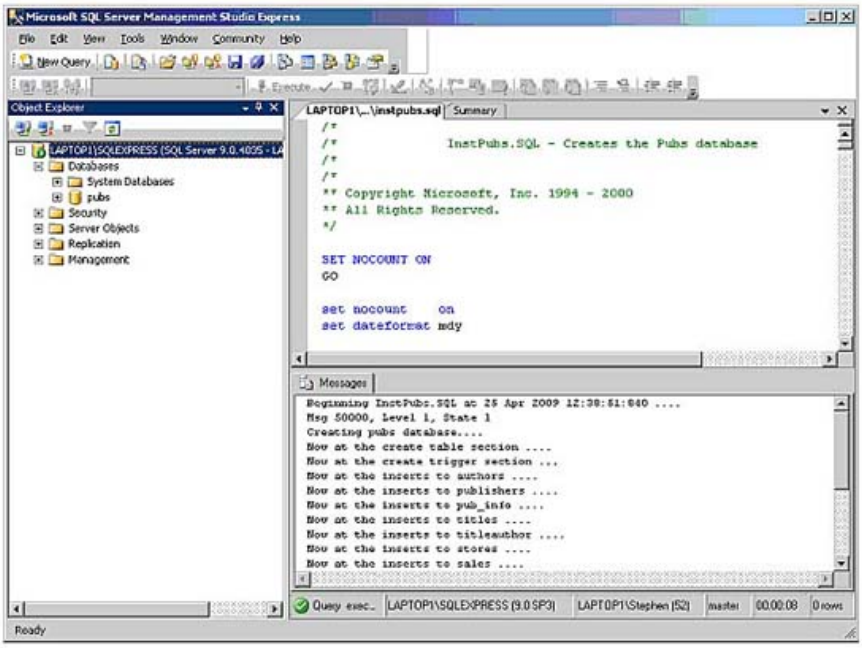

Figure 6: After attaching to the *pubs* database

If you expand the pubs database by clicking on the plus sign, you will see the component tables, as shown in Figure 7.

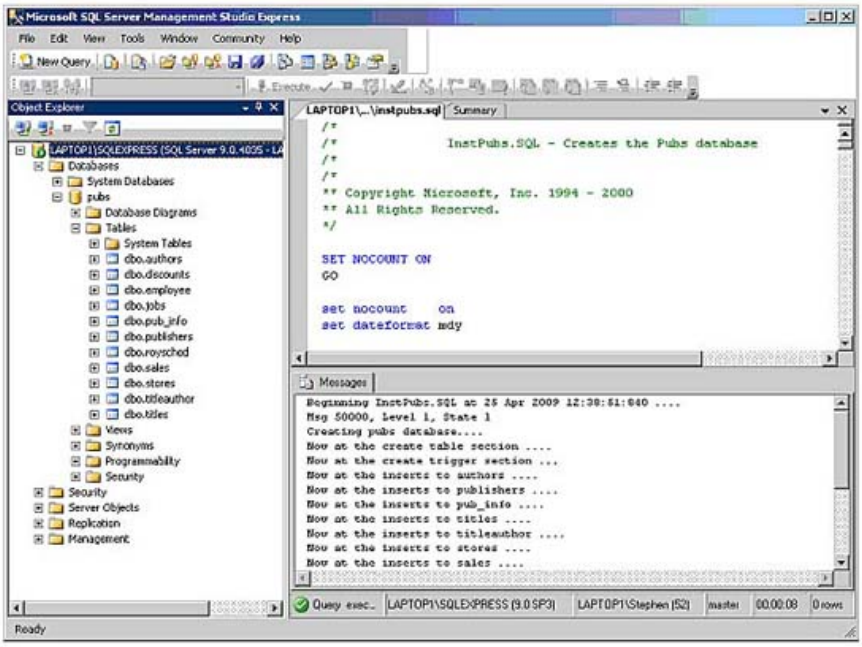

Figure 7: After successfully installing the *pubs* database

The database in this example is completely ready for access via C #. That's all we will do with SQL Server Management Studio Express in this article. As shown in the figure, this product provides a nice environment for you to administer your SQL Server databases. Now we will see how to access those databases using Visual C # 2008 Express Edition.

Run Visual C # 2008 Express Edition

You can run Visual C # 2008 Express Edition from the Windows Start menu. When the IDE opens, select **Tools> Connect to Database** from the menu. This will open the dialog box shown in Figure 8.

Note: The first time you run this command, the dialog box is called **Choose Data Source** . When you are connected to the data source, the dialog box becomes **Change Data Source** , as shown in Figure 8.

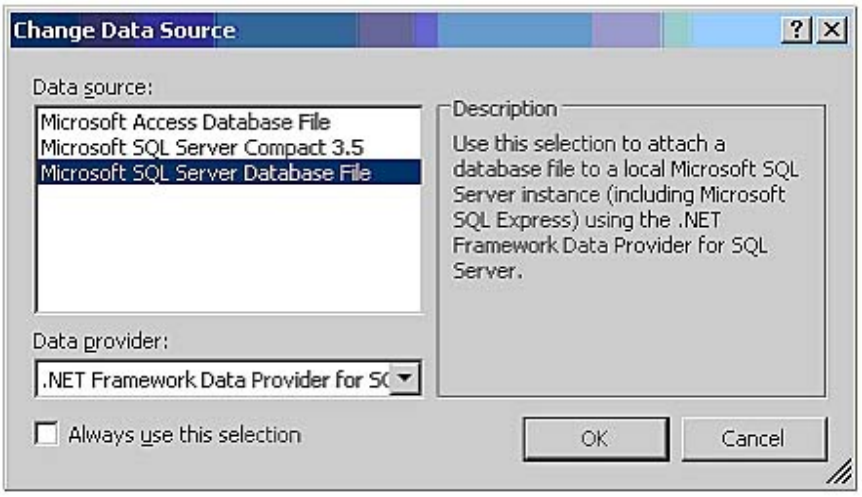

Figure 8: Connecting to the data source

Select the data source, if necessary, click **OK** . In the next dialog box (**Add Connection**), enter the path to the file pubs.mdf. On our system, the default path is shown below:

```
C: Program FilesMicrosoft SQL ServerMSSQL.1MSSQLData
```

Select the **file** , click **Open** , and then click **OK** . The pubs data source will appear, shown in Figure 9.

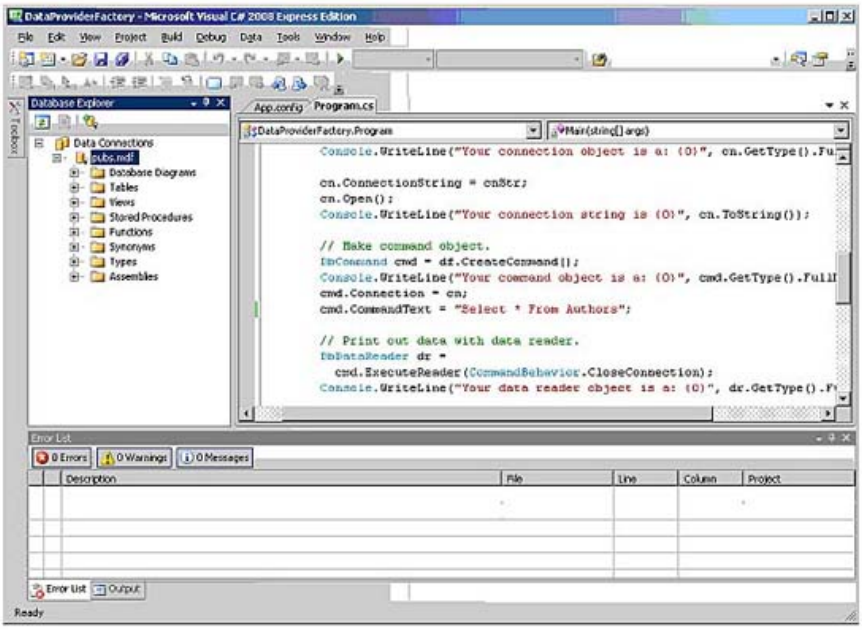

Figure 9: Successful access to the database from within Visual C # 2008 Express Edition

You can now interact with the pubs database just like you can with SQL Server Management Studio Express.

Next, we will access the database using C # code.

Access the database with C # code

Before running C # code, you must disconnect from the pubs database. To do this in Visual C # 2008 Express Edition, right-click the pubs database and select **Detach Database** . If you do not do this, when you run the C # code you can get an error message as shown in Figure 10.

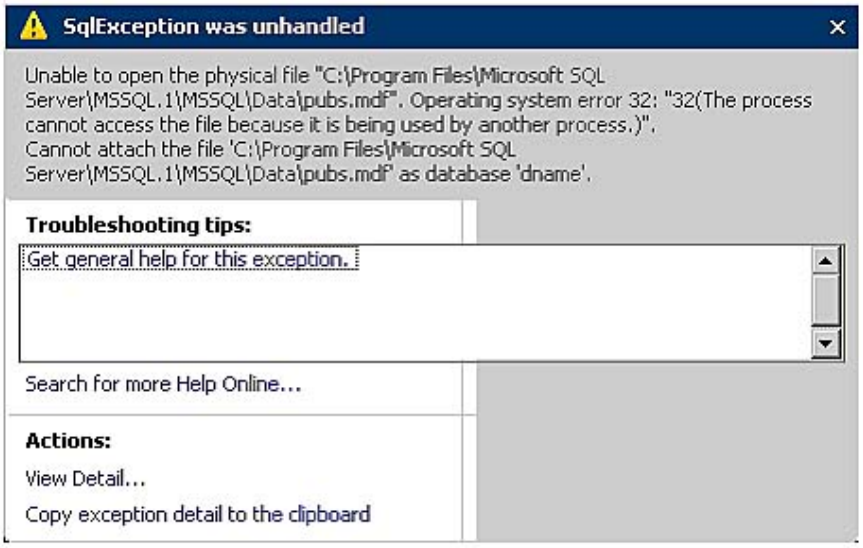

Figure 10: A common problem - too many connections

Now that we have removed all connections to the pubs database, run the C # code you downloaded to get the output shown in Figure 11.

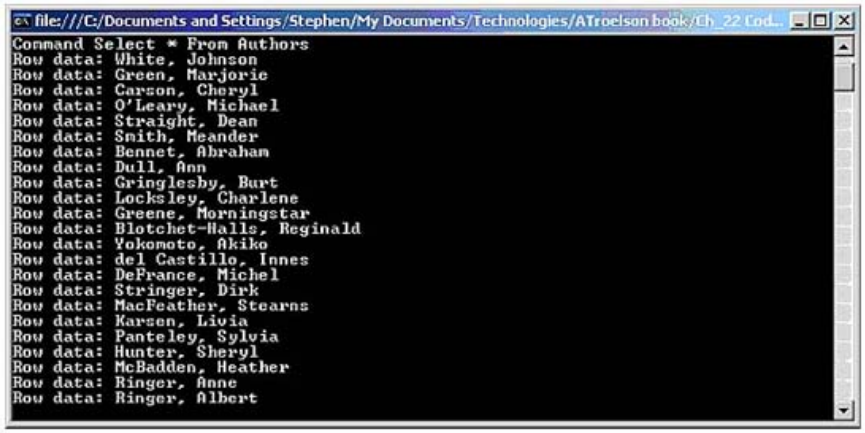

Figure 11: Successful data retrieval

Figure 11 shows its complete database access code. It looks very complicated, but it really isn't. In a nutshell, some parameters are read from a file named App.config then a connection will be made to the database. The requested data will be retrieved from the database and displayed in a row.

Code 1 - Code to access the database

```
// Get metadata from App.config file
string dbProvider =
ConfigurationManager.AppSettings ["dbProvider"];
string connectionString =
ConfigurationManager.ConnectionStrings ["SqlConnStringPubs"].
ConnectionString;
// Create a factory provider
DbProviderFactory dbProviderFactory =
DbProviderFactories.GetFactory (dbProvider);

// Create a connection object
DbConnection dbConnection = dbProviderFactory.CreateConnection ();
```

```
Console.WriteLine ("Connection object: {0}",
dbConnection.GetType (). FullName);

dbConnection.ConnectionString = connectionString;
dbConnection.Open ();
Console.WriteLine ("Connection string: {0}", dbConnection.ToString ());

// Create a command object.
DbCommand cmd = dbProviderFactory.CreateCommand ();
Console.WriteLine ("Command object: {0}", cmd.GetType (). FullName);
cmd.Connection = dbConnection;
cmd.CommandText = "Select * From Authors";

// Create a data reader.
DbDataReader dbDataReader =
cmd.ExecuteReader (CommandBehavior.CloseConnection);
Console.WriteLine ("Data reader object: {0}",
dbDataReader.GetType (). FullName);

Console.WriteLine ("Command" + cmd.CommandText);

while (dbDataReader.Read ())
Console.WriteLine ("Row data: {0}, {1}",
dbDataReader ["au_lname"], dbDataReader ["au_fname"]);
dbDataReader.Close ();
```

Note that at the end of code 1, the code creates references to database columns ["au_lname"] and ["au_fname"]. To understand why that is the case, observe the authors table columns listed on the left of the panel of Figure 12. The C # code displayed is a small set of data from this table.

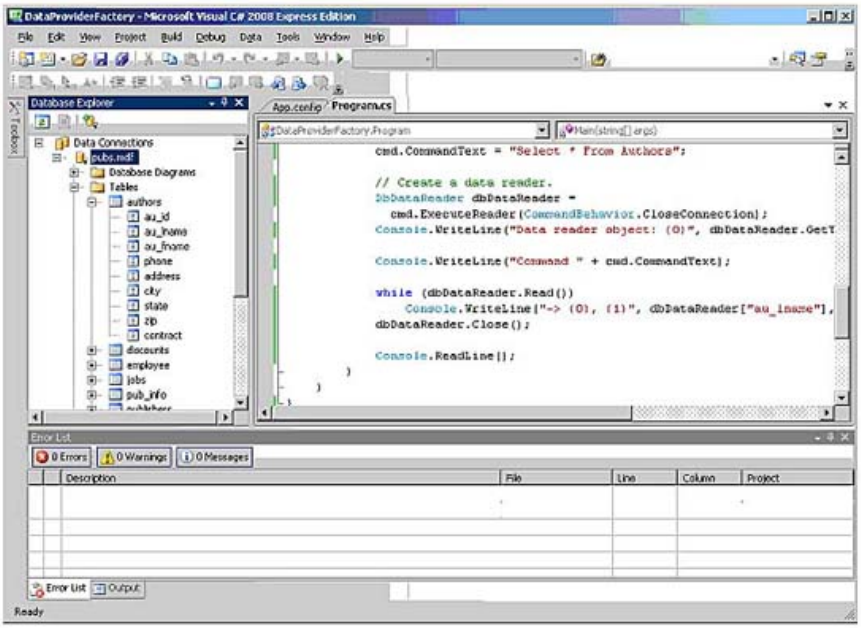

Figure 12: Columns in the authors table

Listing 2 shows the contents of the App.config configuration file.

Code 2 - External metadata file

```
connectionString = "Server = .SQLExpress;
AttachDbFilename = C: Program FilesMicrosoft SQL
ServerMSSQL.1MSSQLDatapubs.mdf; Database = dname;
Trusted_Connection = Yes; "/>
```

The settings in code 2 show the database vendor details and connection information required to access SQLEXPRESS instance. Notice the parts with the word add. These data items are referenced directly in C # code during the run, review in code 1.
At this point our process has completed a tour of ADO.NET.

Conclude

Microsoft provides a lot of rich tools for central database development on the .NET platform. SQL Server 2005 Express Edition is designed for managing database instances at a high level. If you want to interfere with the inner workings of a particular database, you can use the SQL Server Management Studio Express and Visual C # 2008 Express Edition IDE applications with additional features. By using this combination of tools, you can effectively manage and develop centralized C # database solutions.

With these useful database tools, the sample database can be downloaded easily. You can use these databases to develop your database quickly. Besides, using C # code to access an ADO.NET database is quite simple. Just need a C # file and a configuration file, the rest of the code hidden under the background is so complex that you don't need to know it. The abstraction of the technical details of database connectivity is what makes ADO.NET so powerful.

In this article, we have focused most on setting up these tools properly. Database access code installed is very simple; everything is more complicated when you want to change the database. However, the simplification of ADO.NET will help you a lot in this area.

Stephen B. Morris