Super Application Development Capabilities

DB2 & .NET EDITION

.NET

DB2, Information Management Software

OLE-DB
ADO
MS Office

Open

Pick an IDE, Pick an API
And pick DB2 for data access
DB2 and the Windows Developer
By Dirk deRoos

Introduction
Relational database technology has advanced to a point where most commercially available database servers meet the needs of potential customers, at least from a data storage perspective. There are even open-source relational database solutions that are proving to be increasingly feature-rich, robust, and stable. But relational databases are never standalone installations.
They are strictly middleware, placed somewhere between end-user applications and the operating system in a software stack. While DB2 UDB is a leading database server in terms of performance, it is the following features, involving database interaction, which truly set it apart from competing solutions:

- Tight integration into a variety of operating systems, including Microsoft Windows.
- Extensive support for numerous data access programming interfaces.
- The capability to access information from various data sources, including data on different platforms, different relational databases, and non-relational data sources.

Database interaction is extremely important, and DB2 UDB for Windows excels in this regard. In addition to JDBC, SQLJ, ODBC, OLE DB, and support for open source APIs (like Python, Perl, and PHP), DB2 UDB provides a feature-rich, high performing ADO.NET database driver. DB2 UDB also includes a rich set of developer tools that integrate seamlessly in Visual Studio .NET. These tools can generate data access code for you, enabling you to develop database applications without requiring SQL skills or years of database management experience. After all, what good is a powerful database server if you can’t develop quality applications to access your data?

The IBM advantage:
- DB2 UDB integrates seamlessly with Microsoft products
- With DB2 UDB, your applications can access information from a wide variety of data sources
In this eBook, you’ll see a development perspective on how DB2 UDB offers tremendous value on Microsoft Windows. Tight integration with Visual Studio .NET, the .NET Framework, and Microsoft Windows, coupled with innovative development tools enable you to quickly develop powerful, stable, and secure database applications.

**Requirements for developing and running DB2 UDB applications**
The DB2 UDB requirements for developing and running client applications on Windows are minimal. DB2 UDB clients (for example, the Run-Time Client) and client database applications can run on most Windows flavors, including production editions on Windows XP (which is not the case for all database vendors).

In addition to the DB2 UDB client, ADO.NET applications that access DB2 UDB require the installation of the .NET Framework Version 1.1 Redistributable Package.

For the development of Windows-based applications for DB2 UDB, you will need the DB2 UDB Application Development client. In addition to providing client access to DB2 UDB servers, this software includes database administration and application development tools.

Visual Studio .NET is the most commonly used development environment for .NET development, but you can develop ADO.NET applications for DB2 UDB with other .NET development environments, such as Borland Delphi, or with the freely downloadable .NET Framework SDK (version 1.1 is recommended).


**Preparation for ADO.NET development**

To develop ADO.NET applications for DB2 UDB, your database clients and servers will need to be at Version 8.1.2 or higher. This is because the DB2 .NET Data Provider was introduced in Version 8.1.2 of DB2 UDB. The OLE DB bridge provider was introduced in DB2 UDB Version 8.1.5. The ODBC .NET bridge provider was introduced in DB2 UDB, Version 8.2.

If you have Visual Studio .NET installed on your computer, the DB2 .NET Data Provider and the DB2 Add-in for Visual Studio .NET are enabled when you install a DB2 UDB client or server. However, if you installed Visual Studio .NET after installing a DB2 UDB client or server, you will need to manually enable these two components. You can do this from the DB2 UDB section in the Windows Start menu.

**Note:** Both the DB2 UDB installer and the Register Visual Studio Add-Ins tool will enable the DB2 .NET Data
Provider, provided they can detect an installed .NET Framework SDK.

Commitment to the Windows platform
IBM has long invested heavily in DB2 UDB for Windows, providing innovative features and industry leading performance, stability, and scalability. Also, the DB2 development community continues its work in making the experience of DB2 UDB on Windows seamless and natural. These activities, combined with Windows Server certifications and partnerships with Microsoft demonstrate IBM’s ongoing commitment to making DB2 UDB the leading relational database product on Windows.

Windows Server Certification

Windows Server certification is a specification that indicates whether a server software application meets rigorous standards for security, reliability, high availability, and supportability. This certification provides official recognition of how DB2 UDB is a well-integrated software product on Microsoft Windows.
Microsoft Gold Certified Partner Program
Membership in the Microsoft Gold Certified Partner Program for Software Products is a public endorsement of IBM’s expertise in developing quality software for Microsoft Windows. This program also enables IBM to maintain the strong integration of DB2 UDB with Microsoft Windows, through special access to tools and support from Microsoft.

.NET Connected Logo Program
DB2 UDB Version 8.2 is a member of the Microsoft’s .NET Connected Logo program because of its extensive support for the .NET Framework. Important pieces of DB2 UDB for Windows are built on the .NET Framework, such as the DB2 .NET Data Provider, and the DB2 Add-in for Visual Studio .NET. In addition, DB2 UDB Version 8.2 provides these features to aid in the development of .NET database applications.

Visual Studio Integration Program
To encourage further use of the Visual Studio .NET developer environment, Microsoft created the Visual Studio Industry Partner (VSIP) program. In a nutshell, VSIP enables software vendors to provide Visual Studio .NET extensions. These extensions make it easy for users to develop powerful, highly customized .NET applications. As we will see later, IBM integration into Visual Studio .NET has greatly simplified database application development for DB2 UDB.

IBM became a VSIP Premier Partner (the highest VSIP membership level) as of February 12, 2002, shortly after the
program’s inception. Well before VSIP existed, IBM was collaborating with Microsoft for the integration of DB2 UDB with developer tools. In contrast, Oracle and Sybase only joined VSIP in May 2004 and December 2004 respectively.

As a result of the VSIP partnership, IBM delivered DB2 UDB Version 8.1.2 (and subsequent versions) with industry leading Visual Studio .NET integration and support for the .NET Framework. As of the writing of this eBook, no other generally available database product (including Microsoft SQL Server) has as high a degree of Visual Studio .NET integration and as rich a set of .NET-specific features as DB2 UDB. This integration includes a native high performance ADO.NET data provider, DB2 database object creation wizards, integrated help content, support for CLR routines, an interface for browsing DB2 database objects, and more.

.NET integration with DB2 UDB is so successful that the DB2 Add-in for Visual Studio .NET was demonstrated at Microsoft’s VSLive! 2003 Visual Studio Developer Conference in New York. (See the transcript[here].) In addition to the Add-in, IBM representatives demonstrated DB2 UDB support for CLR procedures (released in 2004, with Version 8.2). The integration of the CLR into DB2 UDB was a major achievement, which is all the more significant, given that Microsoft SQL Server 2005 will feature CLR integration over a year after DB2 UDB.

IBM’s embrace of VSIP now enables developers familiar with Visual Studio .NET and the .NET Framework to quickly and easily develop DB2 database applications. You’ll never need
to work outside Visual Studio .NET to build a DB2 UDB application, and you don’t need to learn many DB2 UDB concepts in order to develop DB2 database applications. If you understand the major concepts about .NET development and how to use Visual Studio .NET features, you’ll have no problems developing DB2 database applications.

**DB2, ADO.NET, and you**

In recent years, Microsoft has been promoting a new software development platform for Windows, known as the .NET Framework. The .NET Framework is Microsoft’s replacement for Component Object Model (COM) technology. The following points highlight the key .NET Framework features:

- You can code .NET applications in over forty different programming languages. The most popular languages for .NET development are C# and Visual Basic .NET.
- The .NET Framework class library provides the building blocks with which you build .NET applications. This class library is language agnostic and provides interfaces to operating system and application services.
- Your .NET application (regardless of language) compiles into Intermediate Language (IL), a type of bytecode.
- The Common Language Runtime (CLR) is the heart of the .NET Framework, compiling the IL code on the fly, and then running it. In running the compiled IL code, the CLR activates objects, verifies their security clearance, allocates their memory, executes them, and cleans up their memory once execution is finished.
Through these features, the .NET Framework facilitates a wide variety of application implementations (for instance, Windows forms, Web forms, and Web services), rapid application development, and secure application deployment. COM and COM+ proved to be inadequate or cumbersome for all the aforementioned features.

The .NET Framework provides extensive data access support through ADO.NET. ADO.NET supports both connected and disconnected access. The key component of disconnected data access in ADO.NET is the DataSet class, instances of which act as a database cache that resides in your application’s memory. Connected access in ADO.NET requires no additional classes.

For both connected and disconnected access, your applications use databases through what’s known as a .NET data provider. Various database products include their own .NET data providers, including DB2 UDB for Windows.

A .NET data provider features implementations of the base classes shown in Figure 1:

- **Connection:** Establishes and manages a database connection.
- **Command:** Executes an SQL statement against a database.
- **DataReader:** Reads and returns result set data from a database.
- **DataAdapter:** Links a `DataSet` instance to a database.
Through a `DataAdapter` instance, the `DataSet` can read and write database table data.

Figure 1: The key ADO.NET classes

DB2 UDB for Windows includes three .NET data providers, which are shown in Figure 2, and are described as follows:
• **DB2 .NET Data Provider:** A high performance, managed ADO.NET data provider. This is the recommended .NET data provider for use with DB2 family databases. ADO.NET database access using the DB2 .NET Data Provider delivers significantly better performance than the OLE DB and ODBC .NET bridge providers.

• **OLE DB .NET Data Provider:** A bridge provider that feeds ADO.NET requests to the native IBM OLE DB Provider (by way of the COM interop module). The DB2 .NET Data Provider is faster and more feature-rich than this OLE DB bridge provider. IBM does not recommend using the OLE DB .NET Data Provider unless you are enabling an application to connect to multiple vendors’ databases and you don’t want to change any code.

• **ODBC .NET Data Provider:** A bridge provider that feeds ADO.NET requests to the IBM ODBC Driver. The DB2 .NET Data Provider is faster and more feature-rich than this ODBC bridge provider. IBM does not recommend using the ODBC .NET Data Provider unless you are enabling an application to connect to multiple vendors’ databases and you don’t want to change any code.
DB2 Runtime Client on Windows

Figure 2: DB2 UDB and .NET data access

DB2 .NET Data Provider
The DB2 .NET Data Provider (IBM.Data.DB2.dll) is a common ADO.NET interface to information in DB2 family databases. As such, the DB2 .NET Data Provider gives your
.NET applications managed, secure, high performance access to DB2 data.

The DB2 .NET Data Provider is a prime example of the tight Windows integration in DB2 UDB and the DB2 development team’s close involvement with Microsoft. This data provider is coded in C#, and testing is done with the Microsoft Conformance Suite. (This test suite consists of over 10,000 test cases, written by Microsoft to exhaustively test each ADO.NET API.)

The four key classes in the DB2 .NET Data Provider are: `DB2Connection`, `DB2Command`, `DB2DataReader`, and `DB2DataAdapter`. Reference documentation about the members of these and other related classes is integrated in the Visual Studio .NET help, and in the .NET Framework SDK help. You can also find this documentation online.[here](#).

The DB2 .NET Data Provider allows your .NET applications to access the following database management systems:

- DB2 Universal Database Version 8 for Windows, UNIX, and Linux-based computers
- DB2 Universal Database Version 6 (or later) for OS/390 and z/OS, through DB2 Connect
- DB2 Universal Database Version 5, Release 1 (or later) for AS/400 and iSeries, through DB2 Connect
- DB2 Universal Database Version 7.3 (or later) for VSE & VM, through DB2 Connect
When developing an ADO.NET application that makes use of the DB2 .NET Data Provider, you will need to refer the application to the IBM.Data.DB2.dll assembly. From a Visual Studio .NET project, you can do this from the Add Reference dialog. (You can access this dialog by right-clicking the References node in the Solution Explorer, or by selecting the Add Reference menu item in the Project menu.) For an application developed outside Visual Studio .NET, you can compile your application directly from the command line or a batch file, and reference the IBM.Data.DB2.dll assembly from the compile command. The IBM.Data.DB2.dll assembly for the .NET Framework, version 1.1 is located at this path:

```
“%DB2PATH%\bin\netf11”
```

Figure 3 and Figure 4 show code samples in C# and Visual Basic .NET, respectively, that demonstrate how to connect to a DB2 family database, issue an SQL statement, direct the result set to a DataSet object, and handle any database-related errors. This source code assumes that the IBM.DB2.Data namespace has been declared at the beginning of the source file.
DB2Connection conn = new DB2Connection("Database=SAMPLE");
DB2DataAdapter adapter = new DB2DataAdapter();
DataSet dataset = new DataSet();

try {
    conn.Open();
    String query = "SELECT name, salary FROM STAFF";
    adapter.SelectCommand = new DB2Command(query, conn);
    adapter.Fill(dataset);
}

catch (DB2Exception ex)
{
    for (int i=0; i < ex.Errors.Count; i++)
    {
        MessageBox.Show("Index #" + i + "\n" +
                        "Message: " + ex.Errors[i].Message + "\n" +
                        "Native: " + ex.Errors[i].NativeError.ToString() + "\n" +
                        "Source: " + ex.Errors[i].Source + "\n" +
                        "SQL: " + ex.Errors[i].SQLState + "\n");
    }
}

finally{
    conn.Close();
}

Figure 3: C# example code demonstrating a database connection, SQL statement, data adapter, and error handling
Dim conn As DB2Connection = New DB2Connection("Database=SAMPLE")
Dim adapter As DB2DataAdapter = New DB2DataAdapter()
Dim dataset As DataSet = New DataSet()

Try
    conn.Open()

    Dim query As String = "SELECT name, salary FROM STAFF"
    adapter.SelectCommand = New DB2Command(query, conn)
    adapter.Fill(dataset)
Catch ex As DB2Exception
    Dim i As Integer
    For i = 0 To ex.Errors.Count - 1
        MessageBox.Show("Index "+i.ToString() + ControlChars.Cr _
            + "Native: " + ex.Errors(i).NativeError.ToString() +
            ControlChars.Cr _
            + "Source: " + ex.Errors(i).Source + ControlChars.Cr _
            + "SQL: " + ex.Errors(i).SqlState + ControlChars.Cr)
    Next i
Finally
    conn.Close()
End Try

Figure 4: Visual Basic .NET example code demonstrating a database connection, SQL statement, data adapter, and error handling

Statement chaining
A significant resource expense in issuing statements to a database server is network transmission. If you need to issue a large number of INSERT, UPDATE, and DELETE SQL statements, a sensible resource saving action would be to group all the statements together and submit them to the database server as a batch job. The DB2 .NET Data Provider enables this with statement chaining functionality.

To activate statement chaining, call the DB2Connection.BeginChain method from a DB2Connection object with an active database
connection. After you call the `DB2Connection.BeginChain` method, the INSERT, UPDATE, and DELETE statements that you execute against the `DB2Connection` will be queued on the client until the `DB2Connection.EndChain` method is called. After you call the `DB2Connection.EndChain` method, the DB2 .NET Data Provider will send the chained statements to the database server for execution.

If a server does not support statement chaining, the DB2 .NET Data Provider will ignore the `DB2Connection.BeginChain` and `DB2Connection.EndChain` requests and submit all statements to the database server individually. This reduces the need to change your applications with statement chaining if they connect to various DB2 family databases.

**Concurrent active data readers**
As of DB2 UDB for Windows, Version 8.2.2, the DB2 .NET Data Provider supports the use of concurrent active data readers. With this functionality, your applications can simultaneously process multiple commands and result sets, using the same database connection. This can simplify your application logic since you don’t need to juggle database connections. Fewer database connections can also result in improved performance and scalability.

DB2 UDB support of concurrent active data readers is similar to Microsoft’s upcoming support of Multiple Active Result Sets (MARS) in SQL Server 2005. Unlike concurrent active data readers in DB2 UDB, MARS does not provide the same performance benefits. SQL Server architecture does not easily
lend itself to concurrent retrieval of multiple result sets over a single connection. As a result, MARS support requires some performance taxing work-arounds (for more details see this Microsoft developer’s blog entry).

**CLR procedures and user-defined functions**

With the release of DB2 UDB Version 8.2 in September 2004, IBM introduced support for CLR procedures and user-defined functions to DB2 UDB for Windows (stored procedures and user-defined functions are collectively referred to as routines). As of the writing of this eBook, no other generally available database product supports this feature. Microsoft SQL Server 2005 will feature CLR procedures by late 2005, but by then DB2 UDB will have had this functionality for over a year.

CLR routines are .NET assemblies that are stored and executed on the database server. You can develop routines from within Visual Studio .NET, or by using a text editor and a command line. The CLR procedure wizard integrated in Visual Studio .NET is described in the section, “Rapid application development with DB2 and Visual Studio .NET”.

Regardless if you develop a CLR routine from within Visual Studio .NET or from a command line, the fundamental steps are the same:

1. Compile a .NET application into a .dll assembly.
2. Copy this assembly to the SQLLIB\function folder.
3. Using a CREATE PROCEDURE or CREATE
FUNCTION statement, register this assembly with the database server as a CLR procedure or UDF.

Figure 5 and Figure 6 show CLR procedure code samples in C# and Visual Basic .NET, respectively, that accept a single input parameter, execute an SQL query to retrieve sales data, and return the sales data as a result set. This source code assumes that the IBM.DB2.Data namespace has been declared at the beginning of the source file.

```csharp
class MyProc
{
    public static void EmpSales(String inRegion)
    {
        DB2Datareader reader = null;
        DB2Parameter param;
        DB2Command cmd = DB2Context.GetCommand();

        cmd.CommandText = 
            "SELECT emono, firstname, lastname, salary, bonus, sales " 
            + "FROM employee, (SELECT sales_person, SUM(sales) AS sales " 
            + "FROM sales WHERE region = ?" 
            + "GROUP BY sales_person) AS tot_sales " 
            + "WHERE lastname=tot_sales.sales_person";

        param = cmd.Parameters.Add("@inRegion", DB2Type.VarChar, 15);
        param.Direction = ParameterDirection.Input;
        param.Value = inRegion;

        reader = cmd.ExecuteReader();
    }
}
```

**Figure 5: C# example code demonstrating a CLR procedure**
Figure 6: Visual Basic .NET example code demonstrating a CLR procedure

Compile the CLR procedure source code in Figures 5 and 6 into a .dll assembly and copy the assembly to the SQLLIB\function folder. Figure 7 shows the SQL statement that registers your newly created assembly (called MyProc.dll) as a stored procedure.
CREATE PROCEDURE EMPSALES (IN region VARCHAR(15))
  DYNAMIC RESULT SETS 1
  LANGUAGE CLR
  PARAMETER STYLE GENERAL
  READS SQL DATA
  EXTERNAL NAME 'MyProc.dll:MyProcEmpSales'

**Figure 7: SQL statement that catalogs the C# or Visual Basic .NET CLR procedures**

**Rapid application development with DB2 and Visual Studio .NET**

The integration of DB2 UDB for Windows into Visual Studio .NET makes it easy for you to quickly develop DB2 applications. Included in the clients and servers for DB2 UDB for Windows and DB2 Connect for Windows are utilities that automatically embed tooling and help into your Visual Studio .NET environment, which will aid you in your DB2 development work.

The design goal behind the DB2 Add-in for Visual Studio .NET is to make the DB2 development experience as natural as possible for .NET programmers. With this in mind, you can perform the following tasks, all from within Visual Studio .NET:

- Create databases
- Configure database connections and connect to DB2 family databases.
- Browse database objects with object-level filtering and caching capabilities. For example, browse tables, their metadata, and their data.
- Create various database objects, such as tables, views, indexes, triggers.
- Create CLR procedures and SQL procedures/functions.
• Create data adapters and create corresponding data set objects.
• Expose data adapters as IIS Web methods, or WebSphere Application Server Web services (also known as DB2 Web services).
• Read documentation about the DB2 .NET Data Provider classes and the DB2 Add-in for Visual Studio.

You can complete all of the above tasks either by dragging and dropping the appropriate objects, or by navigating through wizards. This makes for quick and simple interaction with DB2 UDB, which also does not require you to have great deal of DB2 UDB knowledge.

The DB2 Add-in for Visual Studio .NET supports the following DB2 family servers:

• DB2 Universal Database Version 8 for Windows, UNIX, and Linux-based computers.
• DB2 Universal Database Versions 6, 7, and 8 for OS/390 and z/OS, through DB2 Connect.
• DB2 Universal Database Version 5, Releases 1 and 2 for AS/400 and iSeries, through DB2 Connect.

**Getting help**
For an introduction to DB2 .NET development, or for in-depth reference documentation, you don’t need to leave Visual Studio .NET. During the DB2 UDB installation, two help collections are registered with Visual Studio .NET, with the following filter names:
• **IBM DB2 Development Add-in Help**: This collection describes the DB2 Add-in for Visual Studio .NET, and provides instructions on how to use its features.

• **IBM DB2 .NET Data Provider Help**: This collection contains an overview of the DB2 .NET Data Provider, detailed reference information about all the classes and their members, and C# and Visual Basic .NET code samples.

![DB2 UDB help integrated in Visual Studio .NET](image)

**Figure 8: DB2 UDB help integrated in Visual Studio .NET**

**Browsing databases and their objects**

When writing a database application, it’s important to know the nature of the database schema, its tables, and various other objects. The DB2 Add-in for Visual Studio .NET features an
IBM Explorer that makes exploring DB2 family databases a breeze.

Connecting to a DB2 family database
From the IBM Explorer you can connect to any DB2 database that is accessible from your DB2 UDB client. The IBM Explorer makes use of the DB2 UDB discovery service, which searches both local and remote servers for any DB2 family databases.

Figure 9 shows the main panel of the Add DB2 Data Connection wizard, which enables you to configure access to a DB2 family database from within Visual Studio .NET. You can create multiple connections to the same database, so that each connection can have a different user login, default schema, or schema filtering properties. This is a powerful feature, which enables you to create a customized perspective of a single database, specific to the application that you’re developing.
Filtering database objects by schema

When creating your database connection, you can configure the filtering of database objects. Figure 10 shows the Filter tab of the Add DB2 Data Connection wizard, and a view of the IBM Explorer window, which demonstrates the effects of schema filtering. In this figure, the second connection to the SAMPLE database only shows tables belonging to the DDEROOS schema. For databases with numerous tables, schema filtering makes finding the table you’re looking for much easier.
Database catalog data caching
By default, when you create your database connection, the option to prefetch database catalog data is enabled. This feature is especially useful for scenarios where you’re developing applications for remote databases. With this option enabled, when you first connect to a remote database, Visual Studio .NET will automatically retrieve catalog data about the database objects associated with this database. For future development sessions when you’re not connected to the remote database, you can still browse the properties of database objects.
In a disconnected state, each database object has a timestamp indicating the most recent refresh date.

**Viewing and altering data**

From the database connections you created in the IBM Explorer, you can not only see the database objects, but also the data associated with them. In the case of procedures and functions, the IBM Explorer will run them after prompting you for values for their input parameters, and then return any output parameters and result sets. In the case of tables and views, if your connection user ID has the appropriate privileges (and if the view definition allows you to update the underlying tables), you can also update data, delete rows, and insert rows. Figure 11 shows a view of data from the STAFF table in the SAMPLE database. After the table data was retrieved, an update was made to the first row, where the SALARY value was changed to 30,000.00. From the IBM DB2 Output Message Pane, you can see that this update was successful.
Creating database objects

A significant component of database application development is the design and creation of database objects. Traditionally, management of database objects falls under the domain of database administrators. However, for a database system to run effectively, the design of the database and its applications must be done in tandem. Also, in smaller firms the administrator and the database application developer are often the same person. With all this in mind, IBM has provided tooling to easily create database objects from within Visual Studio .NET.

Figure 11: Viewing data in the STAFF table
You can launch any of the DB2 UDB graphical administration tools, including the Control Center, from the Visual Studio .NET Tools menu.

You can launch graphical wizards from the IBM Explorer or from a DB2 Database Project’s Solution Explorer for easy creation of the following database objects:

- Tables
- Views
- Indexes
- Triggers
- SQL procedures
- SQL functions (scalar functions and table functions)
- CLR Procedures

**Note:** Unless otherwise mentioned, you can use the wizards for the creation of the above objects for any of the supported DB2 family database servers.

For DB2 family databases, you create database objects by using SQL data definition language (DDL) statements (for example, the CREATE TABLE statement). These statements can sometimes be intricate and difficult to understand. By selecting options and entering values in the wizards, you are providing the IBM Explorer the information it needs to generate SQL statements that will create your database object. This means that you can be productive in creating and manipulating DB2 family database objects with little or no DB2 SQL skills.
The IBM Explorer provides an additional means of aiding you with database object creation: if you see a database object that is similar to an object that you need to create, you can generate the SQL required to create such an object. Simply right-click on the applicable database object and select *Generate Create Script*. This opens a menu from where you can either direct the creation script to a DB2 Database Project, or into a new edit window.

For some of the database object creation wizards (views, triggers, SQL procedures, and SQL functions) you will require some DB2 SQL skills. However, even editing SQL does not need to be a daunting task, since the Visual Studio .NET editor has been extended with a DB2 SQL mode. The DB2 SQL editing mode comes complete with syntax highlighting, DB2 schema Intellisense, and the ability to insert SQL statement fragment templates, which you can flesh out.

**Example: Creating an index**
To demonstrate how easy it is to create a database object for a DB2 family database from within Visual Studio .NET, let’s use the DB2 Add-in index creation wizard.

To launch the index creation wizard, open the IBM Explorer and right-click the DB2 table to which the index will belong. From the table’s context menu, select *New*, and then *Index*. Figure 12 shows the first step to create an index. In this wizard panel, the only information we entered is in the *Comment* field. The wizard provides all other values for us. For the schema name and table name fields, the wizard provides drop-down
menus listing all the possible values, based on the state of the database. The panels in all the other IBM Explorer wizards provide similar interaction with the target database.

![DB2 Create Index Wizard - Step 1](image)

**Figure 12: Creating an index: step 1**

Step 2 of the index creation wizard, shown in Figure 13, asks for the columns we want for the new index key. Again, to simplify database object creation, the wizard provides all the applicable existing database catalog values. In this case, we can select our index columns from a list of all the columns for the table we selected in the previous step.
Figure 13: Creating an index: step 2

Step 3 of the index creation wizard, shown in Figure 14, features an option to add your own SQL statements as a header and a footer to the index creation script to be generated by the wizard. This feature is present in all of the other DB2 Add-in wizards, and enables great flexibility in database object creation for DB2 family databases.
Figure 14: Creating an index: step 3

Step 4 of the index creation wizard, shown in Figure 15, provides us with the option to create the index immediately, or have the index creation SQL script added to a DB2 Database Project, where the script can be executed in the future. This feature is also present in all of the other DB2 Add-in wizards.
Step 5 of the index creation wizard, shown in Figure 16, presents us with a summary of all the options selected for the new index.

**Figure 15: Creating an index: step 4**
Figure 16: Creating an index: step 5

In Figure 17, you can see the SQL statements generated by this wizard for the creation of an index.
For all the database activities performed from the DB2 Add-in components, output messages are visible in the IBM DB2 Output Message Pane. The messages relating to the completion of the wizard-driven index creation can be seen in Figure 18. Also visible in Figure 18 is a listing of the indexes for the SALES table, which shows the newly created index.
Figure 18: Creating an index: verify index creation

The DB2 Database Project
Deploying database applications often includes the creation of related database objects. For these scenarios, the DB2 Add-in for Visual Studio .NET includes the DB2 Database Project, which is a special project type especially geared to manage the creation of database objects.

DB2 Database Projects contain the SQL scripts by which DB2 database objects are created and dropped. Figure 19 shows an example of a DB2 Database Project. This project contains two object creation scripts: one for an index, and one for a view.
You can generate SQL scripts from the Solution Explorer, either by filling in a script template, or by launching one of the previously described wizards.

The DB2 Database Project features the same code management features as other Visual Studio .NET projects. This is a great benefit to DB2 developers, as it accommodates many powerful script management tools applicable to DB2 family databases:

- **Script build order:** You can determine the order in which your SQL scripts are executed. This is an indispensable feature if you are running scripts that have dependencies on other scripts.
• **Build configurations:** You can define multiple build scenarios, where the properties for the DB2 Database Project and individual scripts can be customized for a particular build configuration. For instance, for a deployment build, you can execute all the SQL scripts using a different database connection than for a debugging build. You can also exclude certain scripts from being built in one build configuration, while they are still active in another build configuration.

• **Source code control:** You can check your script files into a source control management application such as Rational ClearCase or Microsoft SourceSafe.

• **SQL debugger for SQL procedures:** The DB2 SQL debugger seamlessly integrates into the Visual Studio .NET debugging tool, enabling you to step through your SQL procedures that reside on DB2 UDB for Linux, UNIX, or Windows servers. You can debug your SQL procedures from a DB2 Database Project, with the name of your SQL procedure identified in the project’s *Startup Procedure* property.

**Developing database applications**

With the DB2 Add-in for Visual Studio .NET, enabling your applications to access DB2 family databases has never been easier. Connecting to a database or accessing data involves little more than clicking and dragging icons from the IBM Explorer to your Windows or Web form.
Example: Creating a simple database application
To see how easy it is to add DB2 interaction to your .NET applications, let’s create a simple Windows form application. The form design for this application is shown in Figure 20.

![Figure 20: Creating a simple database application: the form design](image)

The Refresh button will pull data from the SAMPLE database’s SALES table and put it in the DataGrid object. The Update button will propagate any changes made to information in the DataGrid object back to the SALES table.

Step 1: From the IBM Explorer, find the SALES table, and drag it onto the form designer. This action automatically adds a
reference to the DB2 .NET Data Provider (IBM.Data.DB2.dll) to your project. And as you can see in Figure 21, the Db2Connection and Db2DataAdapter objects are automatically added to your application. The Db2Connection object is automatically configured with the same properties that you used to create the database connection in the IBM Explorer. Also, the Db2DataAdapter object is automatically configured to reflect the columns in the SALES table.

Figure 21: Creating a simple database application: automatic generation of Db2Connection and Db2DataAdapter objects
**Step 2:** In order to link the *DataGrid* object to the SALES table, your application requires two intermediary ADO.NET objects: a *DataSet* and a data adapter. Specifically, the *DataGrid* interacts with a *DataSet*, a *DataSet* interacts with a data adapter from the database’s .NET data provider, and the data adapter communicates directly with the database. We already have a *DB2DataAdapter* object, so all we need now is a *DataSet* object.

From the *Db2DataAdapter* object’s context menu (available by right-clicking the object), select *Generate Data Set*. Figure 22 shows the dialog where you configure your *DataSet*. Many other database vendors do not provide this time-saving feature.
Figure 22: Creating a simple database application: generating a DataSet object

Once you click OK, you’ll see a DataSet object in the component tray, as in Figure 23.
Figure 23: Creating a simple database application: automatic generation of a DataSet object

**Step 3:** Link the DataGrid object to the newly created DataSet object. From the Properties window of the DataGrid object, select the DataSource property, and select the newly created DataSet object with the “.SALES” suffix, as shown in Figure 24.
Step 4: Add a few lines of code to the Refresh and Update button event handlers. By double-clicking on a form button, an empty event handler method for that button is created for you to add your code. Figure 25 shows the code to be added.

For Button1 (the Refresh button), we need to clear the DataSet before each refresh, and then populate this DataSet using the Db2DataAdapter.Fill method. For Button2, we need to pull the changes made to the DataSet to the database using the Db2DataAdapter.Update method.

Figure 24: Creating a simple database application: linking the DataGrid to the DataSet
Figure 25: Creating a simple database application: adding a few lines of code

With these four easy steps, we are now ready to build and run this application. In Figure 26 we can see this application in action, with sales data being displayed in the DataGrid.
Figure 26: Creating a simple database application: running the application

This application enables you to view the data and also make any changes to the SALES table data (assuming you are authorized to do so). You can delete rows, insert rows, and change data in individual columns. Simply clicking the Update button after you have made any changes will propagate them to the SALES table.

This may be a simple application, but the benefits of DB2 UDB integration in Visual Studio .NET are equally applicable for more complex applications. This menu-driven approach to database programming enables developers with little database experience (DB2 or otherwise) to build DB2 interaction into their applications. Connections to DB2 family databases, SQL
queries, and INSERT, UPDATE, and DELETE statements can all be automatically generated for you.

**Data adapters in the IBM Explorer**

One of the keys to rapid application development is the ease with which you can reuse existing code. The IBM Explorer enables you to design data adapters that you can reuse across multiple projects. To recap, data adapters are the ADO.NET interface between your application and the database, defining how data exchange takes place. So, the work you invest in refining a data adapter’s result set shape and INSERT, UPDATE, and DELETE statements can easily be used for other applications.

From the perspective of the IBM Explorer, a data adapter is quite similar to a database connection, in that both objects are visible in your IBM Explorer tree (see Figure 27), and they both convert into actual ADO.NET objects when you drag them onto an application’s form designer (*DB2Connection* and *DB2DataAdapter*, respectively).
You can create a data adapter in the IBM Explorer by the following ways:

- Directly from the Data Adapters node, using the Configure Data Adapter wizard.
- From a table, view, procedure, or function, using the Configure Data Adapter wizard.
- From an existing Db2DataAdapter object in an application designer’s component tray, by selecting Add to Connection.

Newly generated data adapters are quite basic, with default SELECT, INSERT, UPDATE, and DELETE statements. Using the Configure Data Adapter wizard, you can easily change
a data adapter’s result set shape by removing or reordering data columns, or even adding columns from other tables. You can edit the associated SELECT, INSERT, UPDATE, and DELETE SQL statements with the powerful Visual Studio .NET code editor, extended for DB2 SQL. You can access the data adapter statements in this DB2 SQL editor through either the Configure Data Adapter wizard, or directly from the IBM Explorer.

Like the database objects listed in the IBM Explorer, you can see and edit the associated data for a data adapter. This helps ensure that you are retrieving or updating the correct data much more quickly than by testing the data adapter by calling it in an application.

Once your data adapter is ready for deployment, you can drag it onto your form applications, much like we did with the STAFF table in the earlier ADO.NET application example. By dragging the data adapter onto the form, an icon representing the data adapter and a database connection will appear in the form designer’s component tray.

In order to share your data adapter with another Visual Studio .NET instance, you can export your data adapter as an XML file. From the IBM Explorer, you can import this file to generate a data adapter. You can also take this XML file, add it to a project and then check the XML file into a source control management application like Rational ClearCase, or Microsoft SourceSafe.
Developing CLR procedures
In addition to providing wizards to create non-programmatic database objects, the DB2 Add-in for Visual Studio .NET also includes a wizard and special tooling to aid in the development of CLR procedures.

From the earlier discussion on CLR procedures, remember that there are multiple steps involved in their creation. First, you program the .NET source code for your CLR procedure, and compile it into a .dll assembly. Second, you deploy the assembly on the database server. And third, you issue a CREATE PROCEDURE statement against the database to register your CLR procedure. The DB2 Add-in takes care of the DB2-specific details in these steps, leaving you to focus on writing the .NET source code for your CLR procedure.

Creating CLR procedures
To develop and deploy a CLR procedure from within Visual Studio, begin by creating a DB2 Database Project (or by opening an existing solution with a DB2 Database Project). From your solution, add a DB2 Class Library project (see Figure 28). You can choose either Visual Basic .NET or C#. It is in this project that you will write your CLR procedure code. To help you get started, a template of a DB2 class and procedure is provided for you in the language you chose. Once you are finished coding the CLR procedure, build the assembly.
Now that the CLR procedure itself has been written, you are ready to register it with the DB2 database. To prepare for registering the CLR procedure, you need to complete two small steps. First, since this is a solution with multiple projects, the build order needs to be established: do this by setting the DB2 Database Project to be dependant on the DB2 Class Library Project. Second, from the DB2 Database project, open the Assemblies context menu, and select the .dll file that you have just compiled above (see Figure 29).
Figure 29: Developing CLR procedures: selecting the CLR assembly

From your DB2 Database Project you can launch the Add CLR procedure wizard and begin the process of registering your CLR procedure with the database. This wizard examines the solution for assemblies with DB2 class libraries and enables you to choose the CLR method you want to deploy as a CLR procedure. The wizard reads the code in the CLR method you have chosen and populates the fields with recommended values (see Figure 30).
Once you complete the wizard, you can build the entire solution. The Add-in will automatically deploy your CLR procedure to the database server, and then register it with the database indicated in your DB2 Database Project.

**Altering CLR procedures**

Often times, it becomes necessary to update, repair, or otherwise change the code in a CLR procedure. To do this manually is tedious and error-prone. The DB2 Add-in greatly simplifies this process.
In the case of a code change where the signature of the CLR procedure is to remain constant, make the code change, and then simply re-build the entire solution. The DB2 Add-in re-deploys the assembly to the database server, and re-catalogs the CLR procedure so to refer to the updated version of the assembly.

In the case of a code change where the signature of the CLR procedure will change, an additional step is needed. Once you complete the code change and re-build the solution, you need to re-run the CLR procedure wizard. From the Solution Explorer, find the Procedures folder in the DB2 Database Project, and choose the relevant CLR procedure. In the source editor, right-click on the CLR procedure code block, and you will be able to launch the CLR procedure wizard from the open context menu. From this wizard, the change to the signature will automatically appear. Complete the wizard, re-build the solution, and the CLR procedure’s new signature will be updated at the database server.

**Deploying DB2 Web services**

In addition to client applications and CLR procedures, the DB2 Add-in for Visual Studio .NET enables you to create Web services. With the DB2 Add-in, Web services are a simple, easy to create interface to your DB2 data. On the one hand, client applications and stored procedures are suitable for local area networks, where applications from client computers require access to a central database server. Web services, on the other hand, are useful in scenarios where the applications requesting data are not closely associated with the database server. For
instance, a requesting application might be on a different network, in a different country, or using a different operating system and hardware.

Using the IBM Explorer, you can expose DB2 data through a Web service without writing a single line of code. Configuring how your Web services access a database is as simple as creating a data adapter. From the IBM Explorer, you can either deploy a data adapter as a DB2 Web service, or you can generate ADO.NET code based on the data adapter. DB2 Web services are hosted using the embedded DB2 application server (which is actually the WebSphere Application Server). If you choose an ASP.NET project as the target for the generated ADO.NET code, the data adapter commands will be exposed as ASP.NET Web service methods, or rather, Web methods. With either method, you can convert a data adapter with SELECT, INSERT, UPDATE, and DELETE commands into a Web service with multiple access methods.

**Generating ASP.NET Web methods from data adapters**

To generate an ASP.NET Web service from a data adapter, open the IBM Explorer, choose the data adapter, and from the context menu, choose *Generate ADO.NET code*. The IBM Explorer will then prompt you to enter the name of the CLR project to which you will add the data adapter code (see this menu in Figure 31, below). The generated code from the data adapter will depend on the language being used for the project, and if it is an ASP.NET Web project or not. If the project you choose is, for instance, a class library or a Windows form application, then the data adapter commands will generate
methods that access DB2 data. If you choose an ASP.NET Web project, the data adapter commands will generate Web methods for a Web service.

![Figure 31: Generating a Web service from a data adapter](image)

The generated code is ready to build, run, and deploy. When deploying ASP.NET Web services that access DB2 data, ensure that the ASP.NET Windows account (typically called **ASPNET**) has proper user privileges to connect to your DB2 database and access the data objects referenced by your Web service. To enable an ASP.NET Web service to access a DB2 database, make the ASPNET account a member of the **DB2USERS** group.

**Generating DB2 Web services from DB2DataAdapters**

To generate a DB2 Web service from a data adapter, open the IBM Explorer, choose the data adapter you want to use, and from its context menu, choose *Deploy DB2 Web Services*. 
From the Generate Web Service designer, you can customize the names of your Web service methods, which correspond to the individual data adapter commands. When you complete the wizard, your Web service will be deployed to the DB2 embedded application server. This wizard also provides the option of building a test client for the various Web service operations.

**The Future**

At the time of writing this article, the .NET developer community is eagerly awaiting the formal release of Visual Studio .NET 2005. Currently, DB2 UDB developers are working on even closer integration into Visual Studio for this new version. IBM will also update the DB2 .NET Data Provider to support and extend the .NET Framework, version 2.0.

IBM has recently announced that extensive, industry leading support for XML will arrive in an upcoming version of DB2 UDB. Work is currently underway that will see DB2 UDB provide support for true native XML storage and data access (what some database vendors refer to as native isn’t native), with full support for XML schema and XQuery. This effectively will make DB2 UDB both an SQL and XML database management system. The accompanying DB2 .NET Data Provider and DB2 Add-in for Visual Studio will both be fully XML-enabled.

With the growth of the .NET development community, IBM is keen on ensuring that DB2 UDB integrates seamlessly with Windows and with Visual Studio .NET. Extensive investment
and development is currently happening to ensure that DB2 UDB not only keeps pace with Visual Studio and Windows changes, but also provides additional productivity boosting features.

About the author
Dirk deRoos (BA, BCS) is a technical writer on the DB2 Information Development team. He has coauthored The Official Guide to DB2 Version 8.1.2 (Prentice-Hall, 2003), and written chapters for DB2: The Complete Reference (Osborne/ McGraw-Hill, 2001). Dirk specializes in DB2 application development, with a focus on .NET.