

- You can use `Using()` in C# to tell that an object should be automatically `Dispose()`ed when going out of scope— That is very handy, for example, for connections and file objects that you don't want to wait for being garbage collected.

To be fair, Visual Basic .NET has some advantages over C# too:

- Visual Basic .NET differs `Inherits` from `Implements`— In C#, it's written in both cases with a colon.
- The event syntax is simple in Visual Basic .NET— It's as clean and intuitive as in VB6. In C#, you have to take care of more by hand.
- Visual Basic .NET is not case sensitive— This is a matter of taste, but I prefer non-case sensitive. Otherwise, there is always somebody that will have two methods in a single class named something like `getData()` and `GetData()`.

Note

Despite the differences between C# and Visual Basic .NET, you can use this book with either C# or Visual Basic .NET. The book is probably more “direct” if you have your background in VB6, because that is what I have too. I think and reason more as a VB programmer than as a C++ programmer.

SQL Server 2000

Compared to Visual Basic .NET, SQL Server 2000 is almost an old-timer. Still I think it's important to discuss some of its new features. Although you may think that the leap from SQL Server 7 to 2000 would be a big one, the real changes aren't all that dramatic. However, there are some interesting features for developers to consider, including XML support, user-defined functions, and distributed partitioned views, among others.

XML Support

I'm a geek, and so I love to use new technology. When XML came on the market a couple of years ago, I used it as the file format in a project for a customer of mine. (Hey, I know what you think, but my customer actually thought—and still thinks—it was a good solution!) However, I soon learned that XML—especially its tool support—was not the wonder technology I had first envisioned. The project worked out fine in the end, but I became reluctant to use XML for some time. Since then, the standards and tool support for XML have gone through tremendous development, and are sure to continue into the future.

I became re-interested in XML with SQL Server 2000. There is a lot of XML support in SQL Server 2000. The following are the most interesting features:

- You can now pass XML documents to stored procedures and open them for further processing— That gives, for example, a good solution for the problem of handing over several rows to a stored procedure in one call.
- You can fetch results from `SELECT` statements as XML documents— In some situations, it will be a good solution to create the result in XML directly at the database server instead of converting it at the middle tier or at the client.

Perhaps the XML support is the reason that SQL Server 2000 now is called a .NET server. It's not written in managed code and nothing has changed since it only was called SQL Server 2000, but hey, it can talk XML! What can I say? Marketers...

User-Defined Functions

In SQL Server 2000, we finally had User-Defined Functions (UDFs). Since we have waited so long, Microsoft was kind enough to give us three different versions of UDFs:

- *Scalar Functions*— These can be used for creating functions to be used the same way as columns in `SELECT` statements, similar to, for example, `CONVERT()`.
- *Inline Table Valued Functions (ITVFs)*— ITVFs are similar to views, but can take parameters. The result is created with one single `SELECT` statement.
- *Multistatement Table Valued Functions (MTVFs)*— In this case, the result can be created with multiple statements. MTVFs are, for example, a good solution in situations where you normally use temporary tables.

The UDF implementation in SQL Server 2000 is a typical first version, with several quirks. For example, it is not possible to use `RAND()` and `GETDATE()` in MTVFs. The same goes for touching temporary tables and calling stored procedures. Microsoft will probably fix some of the rough edges in an upcoming version.