.NET Frameworkで作る Windowsサーバー

作ればわかる アプリケーションの動作とメカニズム

秋月 巌 AKIZUKI, Iwao〈秋月巌ソリューション事務所〉▶http://www.akizuki.co.jp/

第19回

SQL Serverがクライアントとやり取りするメッセージのデータ構造

SQL Serverのデータ構造

今回はSQL ServerがADO.NETと通信するときのメッセージ構造について説明する。前回、紹介したJetデータベースサーバーの実装の続きなのだが、単純にSQL Serverのメッセージ内容に興味がある人の参考にもなるだろう。またSQL Serverのデータ構造についての学習にもなるだろう。

いずれにしても、今回の内容はSQL ServerとADO.NETのやり取りをキャプチャしそれを私が解釈したものなので、その解釈がどこまで正確なのかはわからない。私が公表されていないインサイド情報をもっていて、それを公開しているわけではない。調査し、試した結果、どうやらちゃんと動いているらしいというレベルだと考えてほしい。

サンプルプログラムは、Jetデータベースエンジンをデータベースサーバーとして使用するJetデータベースサーバーである（図1・2）。本誌2006年8月号から作成しているSQLプロキシサーバーのカスタマイズ機能として実装されている。ちなみにSQLプロキシサーバー自体は、本連載の第12回（2006年4月号）で紹介したサーバープログラム開発用テンプレートであるASTサーバーを使用している。できれば、このあたりの関係を正しく理解してからソースコードを見てほしい。

また、今回提供するサンプルは前号のサンプルとはいくつかの点で異なっている。日付型データをサポートしたからである。SQLプロキシサーバーのカスタマイズ用プロシージャである「AppSQLPrxyReceiveFromClient」プロシージャの実装は前号で説明したが、このプロシージャのデータ型を識別する条件分岐に、次のような条件とプロシージャの呼び出しが追加されている。

レベル >>> Level

1 2 3 4 5

言語 >>> Language

- Visual Basic

ツール >>> Tool

- Visual Studio 2005 Professional
- SQL Server 2000
- SQL Server 2005
- Access 2003

サンプル >>> Sample

この記事で取り上げたソースコードおよびサンプルプログラムは、
http://www.shoeisha.com/mag/windev/
からダウンロード可能です。

図1：SQLプロキシサーバーをカスタマイズして作ったJetデータベースサーバー

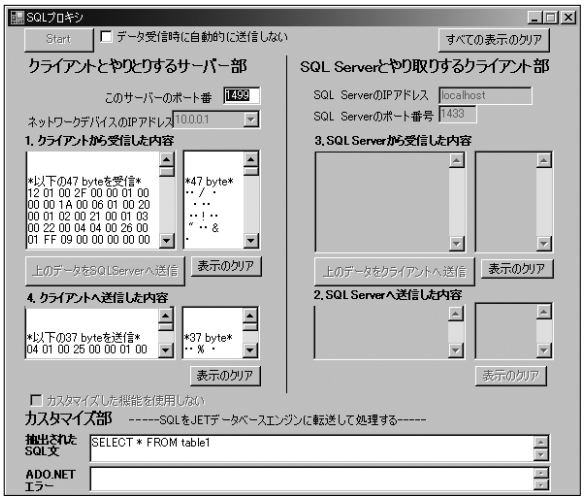

図2：クライアントプログラムの例


```
Case "System.DateTime"
   cOffset = JetDBSvrDateTimetoBinary( _
    byteSendData, row(column), cOffset)
```

他にはメッセージデータをバイナリ配列に書き込むときに、配列の要素数を確認し、不足している場合には要素を追加する「CheckByteArraySize」プロシージャを各所で呼び出している。

今回はカスタマイズ用の「AppSQLPrxyReceiveFromClient」プロシージャから呼び出される次の2つのサブプロージャの実装について説明する。

ひとつは上で引用したコードで呼び出されている「JetDBSvrDateTimetoBinary」プロシージャである。このプロシージャは、ADO.NETから読み取った日付データをバイナリに変換するプロシージャである。

もうひとつは「JetDBSvrCreateMsgHeader」プロシージャで、ADO.NETに返信する結果セットを含むメッセージのヘッダーを作成する。

数値型データに関しては前回説明したようにBitConverterクラスで変換が可能なのでサブプロシージャ化していない。だが、BitConverterクラスがどのように数値をバイナリデータに変換しているかについても後で説明する。文字列型データは前回Encodingクラスを使って1文字ずつ変換していたが、これは一括して変換するように修正した。これは単にシフトJIS漢字コードを用いて変換しているだけなので説明はしない。文字コードの変換に関しては、本連載の第15回（2006年7月号）でカスタマイズ可能なWeb用プロキシサーバーを作ったときにも説明しているので参考にしてほしい。

今まで、この連載ではWebサーバーやメールサーバーなど一般的なサーバーアプリケーションを作成してきた。Jetデータベースサーバーの場合は、SQL Serverを作成していると考えてほしい。汎用的なデータベースサーバーというよりは、SQL Serverという特定のデータベースサーバーなので、プロトコルの実装はあくまでもSQL Serverに固有のものである。だから、Jetデータベースサーバーにアクセスするには、ADO.NETの接続文字列でSQL Serverを指定する必要がある。HTTPやSMTPと違い、SQL Serverのプロトコルは、仕様が公開されているわけではない。だから、まず初めにSQL ServerとADO.NETのやり取りをキャプチャしてプロトコルを解析する必要がある。SQLプロキシサーバーを使えば、このキャプチャが容易に行なえる。もちろん、TCP/IP用のキャプチャソフトでも同じことは可能だが、SQLプロキシサーバーならばSQL Serverとやり取りされるデータだけが見れるので、ずっとわかりやすい。とはいえ、HTTPやSMTPと違って、やり取りされる内容はバイナリデータである。決して人間に読みやすいようにできているわけではない。

SQL Serverが結果セットを返すときのメッセージ構造

さて、今回のテーマは、SELECT文のような結果を持つクエリが要求されたときに、SQL ServerがADO.NETに返すメッセージのデータ構造である。図3のような構造を持っていることが、今の時点で判明している。

これがすべてである。①から⑧までがヘッダーで、⑨