SQL Anywhere と ADO.NET Entity Framework

November 2008

目次

はじめに

本書では、データベース・アプリケーション開発者が SQL Anywhere 12 および ADO.NET Entity Framework を使用してデータベース駆動型アプリケーションを構築する方法について説明します。さらに、LINQ (Language Integrated Query ：統合言語クエリ) to Entities、オブジェクト・サービス、および EntityClient プロバイダの方法論を使用して、SQL Anywhere データベースに格納されているデータにアクセスする方法についても説明します。

ADO.NET Entity Framework は、Microsoft から提供されている新しい技術です。データ認識アプリケーションの開発を簡素化し、データベース・アプリケーション開発者が記述するコード数を減らします。通常、データベースを設計するデータベース管理者は、データを取り出しやすいように最適化したスキーマにすべての情報を確実に格納します。スキーマによりデータへの効率的なアクセスが可能になりますが、それらはアプリケーションが必要とするビジネス目的を実現する最善策ではありません。たいてい、リレーショナル・システムに格納されたデータをエンタープライズ・ソリューションの不可欠部分である堅牢かつ再利用可能なオブジェクトに変換するためのメソッドをプログラマが設計して実装する必要があります。ADO.NET Entity Framework は、リレーショナル・データに基づくビジネス論理を設計するための直感的な方法を提供します。

ADO.NET Entity Framework は、EDM (Entity Data Model ：エンティティ・データ・モデル) を導入します。これは、アプリケーションで使用されるデータの概念ビューです。通常、既存のデータベース・スキーマには大量のデータが格納されていますが、クライアント・プログラムが必要なのはこのデータの一部分だけです。たとえば、オンライン注文を処理する Microsoft Windows クライアント・アプリケーションの場合、顧客、製品、および注文に関する情報だけが必要です。すべてのデータベース・テーブルを EMD に含める代わりに、開発者はアプリケーションに関係があるテーブルだけを指定します。

前提条件

本書の手順を実行するには、以下のソフトウェアが必要です。

- SQL Anywhere 12.0.0 またはそれ以降
- SQL Anywhere 11.0.1 build (EBF) 2436 またはそれ以降
- SQL Anywhere の Developer Edition は、以下の場所から無償でダウンロードできます。
  http://www.ianywhere.jp/dl/index.html
- Microsoft Visual Studio 2008 with Service Pack 1
- Microsoft .NET Framework 3.5 with Service Park 1 またはそれ以降

SQL Anywhere 統合コンポーネントのインストール確認

Visual Studio で、[Tools] をクリックします。Sybase Central および Interactive SQL がオプションとして表示されない場合、以下の手順を実行します。

- Visual Studio を閉じます。
- コマンド・プロンプトで、以下のディレクトリに移動します。

  C:¥Program Files¥SQL Anywhere 11¥Assembly¥v2
- 以下のコマンドを実行します。

  SetupVSPackage /i

SQL Anywhere データベースからエンティティ・データ・モデルを作成する

Visual Studio で定義した任意の SQL Anywhere データベース・プロファイルを使用して、新しいエンティティ・データ・モデルを作成します。この手順では、SQL Anywhere 11 のインストールに含まれている SQL Anywhere 11 Demo データベースを使用します。

1. Visual Studio 2008 を起動します。
2. SA11EntityFramework という名前の新しい Windows フォーム・アプリケーション・プロジェクト (C# または VB) を作成します。
3. [Project] メニューで、[Add New Item] - [ADO.NET Entity Data Model] を選択します。
4. [Name] フィールドに SADemo.edmx と入力し、[Add] をクリックします。
5. [Generate from database] を選択し、[Next] をクリックします。
6. [New Connection] をクリックします。
7. [Data source] リストで [SQL Anywhere] をクリックし、[OK] をクリックします。

[Data source] リストに [SQL Anywhere] が表示されない場合、SQL Anywhere 統合コンポーネントがインストールされているかどうかを確認してください。

8. [ODBC Data Source name] をクリックし、[SQL Anywhere 11 Demo] を選択します。[OK] をクリックします。
9. [Next] をクリックします。
10. [Finish] をクリックします。

Entity Designer でのエンティティ・データ・モデルの表示

エンティティ・データ・モデルを作成すると、Entity Designer にモデルのビジュアル表現が表示されます。以下の図では、生成されたプロパティと関係がデータベース・スキーマと一致しています。

この図では、データベース・スキーマに対してエンティティが適切にマップされています。

コントロールへのデータのバインド

SQL Anywhere データベースに格納されている情報を表示する DataGridView コントロールを 1 つ追加するには、以下の手順を実行します。

手順 10 で LINQ to Entities を使用することにより、Visual Studio の IntelliSense (オート・コンプリート) 機能の使用が可能になります。ツール・チップと IntelliSense を使用することで、Visual Studio を使用するプログラマがアプリケーションのランタイム・エラーを検出して回避できるようになります。

1. [Data] - [Show Data Sources] を選択します。
2. [Add New Data Source] をクリックします。
3. [Data Source Type] リストで [Object] をクリックし、[Next] をクリックします。
4. [Model] リストを展開して [Customers] をクリックし、[Next] をクリックします。
5. [Finish] をクリックします。
6. [Data Sources] ウィンドウで、[Customers] オブジェクトをメイン・フォーム上にドラッグします。データベースに格納されているレコードを表示する [DataGridView] および [BindingControlNavigator] コントロールが生成されます。
7. メイン・フォームの [Load event] ペインで、[Customers] オブジェクトをデータ・グリッドにバインドするために以下のコードを追加します。

C#

```
var saEntities = new SA11EntityFramework.Entities();
// Retrieve the list of customers from the EDM
var customers = saEntities.Customers;
// Bind the customers object to the data grid
customersBindingSource.DataSource = customers;
```

VB.NET

```
Dim saEntities As New SA11EntityFrameworkVB.Entities
' Retrieve the list of customers from the EDM
```

```
Dim customers = saEntities.Customers
' Bind the customers object to the data grid
CustomersDataGridView.DataSource = customers
```

8. プロジェクトを実行し、データ・グリッドの顧客リストを確認します。
9. サンプル・アプリケーションを閉じます。
10. メイン・フォームの［Load event］ペインで、カナダの顧客のみをデータ・グリッドに表示するために以下のコードを追加します。

C#

```
var saEntities = new SA11EntityFramework.Entities();
// Retrieve the list of Canadian customers from the EDM
var customers = from c in saEntities.Customers
      where c.Country == "Canada"
                  select c;
// Bind the customers object to the data grid
customersBindingSource.DataSource = customers;
```

VB.NET

```
Dim saEntities As New SA11EntityFrameworkVB.Entities
' Retrieve the list of Canadian customers from the EDM
Dim customers = From c In saEntities.Customers _
                  Where c.Country Is "Canada" _
                  Select c
' Bind the customers object to the data grid
CustomersDataGridView.DataSource = customers
```

11. プロジェクトを実行し、カナダの顧客のみが表示されていることを確認します。
12. サンプル・アプリケーションを閉じます。

エンティティ・データ・モデルの拡張

エンティティ・データ・モデル用に生成されたコードは、それが部分クラス・エンティティであることを示しています。データベース・アプリケーション開発者は、部分クラス・エンティティを拡張し、ビジネス目的を完了するプロパティおよびメソッドを含めることができます。

以下の手順を実行し、情報を取り出す 2 つのメソッドを含む新しいクラス・ファイルをサンプル・アプリケーションに追加します。この手順は、開発者がデータベース・スキーマを変更できないことを前提としています。したがって、エンティティ・データ・モデルの拡張により論理を追加します。

1. プロジェクトに新しいクラス・ファイルを追加し、コードを以下のように変更します。

C#

```
namespace SA11EntityFramework
{
    public partial class Entities
    {
        public List<string> AvailableColours()
        {
            var colours = (from Products p in this.Products
                           orderby p.Color
                           select p.Color).Distinct();
            return colours.ToList();
        }
        public List<Products> GetColourProducts(string colour)
        {
            var prods = from Products p in this.Products
                        where p.Color == colour
                        select p;
            return prods.ToList();
        }
    }
}
```

VB.NET

```
Partial Public Class Entities
    Public Function AvailableColours() As List(Of String)
        Dim colours As IQueryable(Of String)

        colours = (From Products In Me.Products _
                   Order By Products.Color _
                   Select Products.Color).Distinct()
```

```
        Return colours.ToList()
    End Function
    Public Function GetColourProducts(ByVal colour As String) _
    As List(Of Products)
        Dim prods As IQueryable(Of Products)
        prods = From Products In Me.Products _
                Where Products.Color Is colour _
                Select Products
        Return prods.ToList()
    End Function
End Class
```

2. データ・グリッドの下のペインで、コンボ・ボックス、リスト・ボックス、および [Get Products] ボタンをメイン・フォームに追加します。これらのコントロールは、手順 1 で追加した 2 つのメソッドの結果を表示します。

3. メイン・フォームの [Load event] ペインで、以下のコードを追加します。

C#

```
var colours = saEntities.AvailableColours();
// Add each colour to combo box
foreach (var c in colours)
{
    comboBox1.Items.Add(c.ToString());
} // foreach
```

VB.NET

```
Dim colours = saEntities.AvailableColours()
' Add each colour to the combo box
For Each c In colours
    ComboBox1.Items.Add(c.ToString())
Next
```

4. ボタン・クリック・イベントに以下のコードを追加します。

C#

```
listBox1.Items.Clear();
string colour = comboBox1.SelectedItem.ToString();
var saEntities = new SA11EntityFramework.Entities();
var available = saEntities.GetColourProducts(colour);
// Add product to list box
foreach (var a in available)
{
    listBox1.Items.Add(a.Name.ToString());
} // foreach
```

VB.NET

```
ListBox1.Items.Clear()
Dim colour = ComboBox1.SelectedItem.ToString()
Dim saEntities As New SA11EntityFrameworkVB.Entities
Dim available = saEntities.GetColourProducts(colour)
' Add product to list box
For Each a In available
    ListBox1.Items.Add(a.Name.ToString())
Next
```

5. プロジェクトを実行します。

AvailableColours メソッドが Products テーブルのユニークなカラー・リストを取り出し、コンボ・ボックスに出力します。GetColourProducts メソッドは、LINQ クエリに一致する Products エンティティ・タイプのリストを取り出します。

その他のデータ・アクセス・メソッド

開発者は、LINQ to Entities の他にも、オブジェクト・サービスや EntityClient プロバイダを使用して SQL Anywhere データにアクセスできます。

オブジェクト・サービス

このメソッドの場合、Entity SQL を使用してエンティティ・データ・モデルを照会します。オブジェクト・サービスを使用すると、データ・アクセス・モジュールの記述に必要なコードを最小限にできるうえ、“for each” ループを使用してオブジェクトのコンテンツを簡単に取り出せるようになります。以下に例を示します。

C#

```
var saEntities = new SA11EntityFramework.Entities();
// Query EDM using the Object Services
var contacts =
      saEntities.CreateQuery<System.Data.Common.DbDataRecord>(
      @"select c.GivenName, c.Surname
        from Entities.Contacts as c");
// Add the contact's first name and last name to the list box
foreach (var record in contacts)
{
      listBox2.Items.Add(
      record.GetString(0) + " " + record.GetString(1));
} // foreach
```

```
VB.NET
Dim saEntities As New SA11EntityFrameworkVB.Entities
' Query EDM using Object Services
Dim contacts = saEntities.CreateQuery( _
Of System.Data.Common.DbDataRecord) _
("select c.GivenName, c.Surname from Entities.Contacts as c")

' Add the contact's first name and last name to the list box
For Each record In contacts
      ListBox2.Items.Add(record.GetString(0) + " " _
                              + record.GetString(1))
Next
```

EntityClient プロバイダ

このメソッドの場合、ADO.NET プロバイダを使用してデータにアクセスする場合と同様の方法で Entity SQL を使用します。Entity Framework が開発される以前は、プログラマは、ADO.NET を使用して接続、コマンド、およびデータ・リーダ・オブジェクトを持つ SQL Anywhere アプリケーションを開発していました。EntityClient プロバイダは同様のデータ・アクセス方法を提供しますが、データベースの代わりにエンティティ・データ・モデルが必要です。EntityClient プロバイダを使用する場合、プログラマは “EntityConnections” の形式で接続し、“EntityCommands” の形式でコマンドを使用できます。以下に例を示します。

```
C#
```

```
var saEntities = new SA11EntityFramework.Entities();
// Query EDM using the EntityClient provider and Entiry SQL
var saConn = new
                System.Data.EntityClient.EntityConnection("Name=Entities");
var saCmd = new System.Data.EntityClient.EntityCommand(
    @"select distinct p.Name
      from Entities.Products as p", saConn);
saConn.Open();
var saReader = saCmd.ExecuteReader(
    System.Data.CommandBehavior.SequentialAccess);
// Loop through result set and add product name to the list box
while (saReader.Read())
{
    listBox3.Items.Add(saReader.GetString(0));
} // while
```

VB.NET

```
Dim saEntities As New SA11EntityFrameworkVB.Entities
' Query EDM using the EntityClient provider and Entity SQL
Dim saConn As New _
    System.Data.EntityClient.EntityConnection("Name=Entities")
Dim saCmd = New _
    System.Data.EntityClient.EntityCommand( _
    "select distinct p.Name from Entities.Products as p", saConn)
saConn.Open()
Dim saReader = saCmd.ExecuteReader _
    (System.Data.CommandBehavior.SequentialAccess)
' Loop through result set and add product name to the list box
While saReader.Read()
    ListBox3.Items.Add(saReader.GetString(0))
End While
```

まとめ

データベース・アプリケーション開発者は、SQL Anywhere 12 を使用することにより ADO.NET Entity Framework 技術を利用できます。Visual Studio 用の SQL Anywhere 統合コンポーネント

を使用して、SQL Anywhere データベースからエンティティ・データ・モデルを生成したり、レコードを操作したりすることができます。コントロール・データ・バインディング、オブジェクト・サービス、EntityClient プロバイダ、および LINQ to Entities を使用してレコードを操作できます。また、生成されたエンティティ・データ・モデルを拡張することにより、今日のエンタープライズ・ソリューションが必要とする堅牢なビジネス論理を実装することもできます。

法的注意