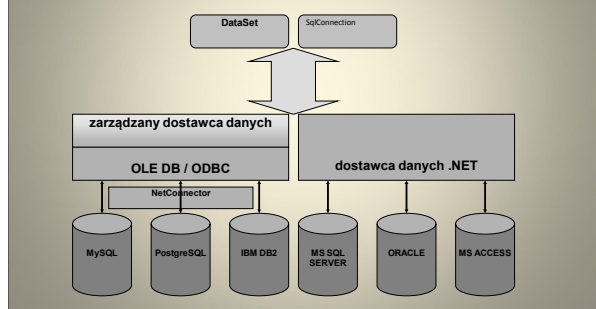


Metody obsługi relacyjnych baz danych (SQL)



Dostęp do bazy danych

- IDbConnection – obsługuje połączenie z bazą danych, bazowy dla klas dostawców, np.:
 - SqlConnection,
 - OracleConnection,
 - OleDbConnection,
 - OdbcConnection.

Dostęp do bazy danych

- MS SQL

```
using System.Data.Common;
using System.Data.SqlClient;
using System.Data.SqlTypes;

// fragment kodu
SqlConnection scon = new
    SqlConnection("Server=localhost\\SQLEXPRESS;
        database=NazwaMojejBazy; User Id=uzytownik;
        Password=haslo;");

scon.Open();
// operacje na bazie danych
scon.Close();
```

Dostęp do bazy danych

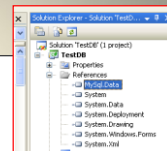
- MySQL

```
using System.Data.Common;
using System.Data.SqlClient;
using System.Data.SqlTypes;

using MySql.Data.MySqlClient;

// fragment kodu
MySqlConnection scon = new MySqlConnection(
    „server=127.0.0.1; database=bazadanych; user id=root;
    password=bazy2”);

scon.Open();
// operacje na bazie danych
scon.Close();
```



MySql.Data.dll

Metody obsługi relacyjnych baz danych (SQL)

- Metoda połączeniowa
- Metoda bezpołączeniowa

Metoda połączeniowa

- w trakcie połączenia wymieniane są dane między bazą a klientem wiersz po wierszu
- opiera się na transmisji sieciowej,
- mało wydajne w przypadku dużych zbiorów danych,
- wydajne w przypadku prostych komend i niewielkiego transferu danych, np. UPDATE, DELETE.

Metoda połączeniowa

```
// przykład wykonania komendy (połączeniowo)

SqlConnection scon = new
SqlConnection("Server=localhost\\SQLEXPRESS;
             database=NazwaMojejBazy; User Id=uzytkownik;
             Password=haslo;");

scon.Open();
SqlCommand cmd = new SqlCommand("DELETE FROM Dane
WHERE NrKolejny = 0;", scon);
int ldel = cmd.ExecuteNonQuery();
scon.Close();
```

ldel – liczba krotek objętych działaniem (tu: usuniętych). Przy replikacji typu merge x2 (triggery)
Odpytanie kolejnych krotek (wczytywanie dużych tabel) - nieefektywne

Metoda bezpołączeniowa

- Dane są kopiowane z bazy do pamięci klienta (blokowo), odczyty i zmiany odbywają się na bloku pamięci, istnieje możliwość naniesienia zmian na bazę.
- + szybkie operacje w przypadku dużych bloków danych,
- + łatwa obsługa tabel,
- przy ekstremalnie dużych blokach – duże zużycie pamięci -> model połączeniowy,
- nanoszenie drobnych zmian – nieefektywne.

Metoda bezpołączeniowa

```
// przykładowy odczyt danych z tabeli do obiektu w pamięci
(DataSet/DataTable)

SqlConnection scon = new
SqlConnection("Server=localhost\\SQLEXPRESS;
             database=NazwaMojejBazy; User Id=uzytkownik;Password=haslo;");

scon.Open();
string sqlcom = " SELECT * FROM Dane WHERE NrKolejny > 0 ";
SqlDataAdapter da = new SqlDataAdapter(sqlcom, scon);
SqlCommandBuilder sqcmb = new SqlCommandBuilder(da);

DataSet ds = new DataSet("Dane");
da.Fill(ds, "Dane");
DataTable dt = ds.Tables["Dane"];
string tekst = (string)ds.Tables["Dane"].Rows[7]["kolumnatxt"];
```

Metoda bezpołączeniowa

```
// przykład: zapis do tabeli / modyfikacja danych
{..}
da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
DataSet ds = new DataSet("Dane");
da.Fill(ds, "Dane");

byte[] imgrow = new byte[1024]; // dane binarne
DataRow dro; // dodanie nowej wiersza
dro = ds.Tables["Dane"].NewRow();
dro["NrKolejny"] = 501;
dro["kolumnatxt"] = "tekst do wpisania";
dro["daneBLOB"] = imgrow;
ds.Tables["Dane"].Rows.Add(dro);

ds.Tables["Dane"].Rows[0]["daneBLOB"] = imgrow; // modyfikacja BLOB w wierszu nr 0
da.Update(ds, "Dane"); // zapisanie zmian
scon.Close();
```

Metoda bezpołączeniowa

```
da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
```

Add Adds the necessary columns to complete the schema.

Adds the necessary columns and primary key information to complete the schema. For more information about how primary key information is added to a [DataTable](#), see [FillSchema](#). To function properly with the .NET Framework Data Provider for OLE DB, `AddWithKey` requires that the native OLE DB provider obtains necessary primary key information by setting the `DBPROP_UNIQUEROWS` property, and then determines which columns are primary key columns by examining `DBCOLUMN_KEYCOLUMN` in the `ColumnsRowset`. As an alternative, the user may explicitly set the primary key constraints on each [DataTable](#). This ensures that incoming records that match existing records are updated instead of appended. When using `AddWithKey`, the .NET Framework Data Provider for SQL Server appends a `FOR BROWSE` clause to the statement being executed. The user should be aware of potential side effects, such as interference with the use of `SET FMTONLY ON` statements. See SQL Server Books Online for more information.

Error An `InvalidOperationException` is generated if the specified column mapping is missing.

Ignore Ignores the extra columns.

Obiekt DataSet a XML

```
MemoryStream ms = new MemoryStream();
dt.WriteXml(ms, XmlWriteMode.WriteSchema);
```

```
MemoryStream ms = new MemoryStream();
// w strumieniu sa dane XML
DataSet dsn = new DataSet();

dsn.ReadXml(ms, XmlReadMode.ReadSchema);
```

Obiekt DataSet a XML

```
XmlWriteMode;
```

DiffGram Zapisuje cały [DataSet](#) jako element w formacie DiffGram, łącznie z formatem oryginalnym oraz aktualne wartości. Aby wygenerować w formacie DiffGram zawierające tylko zmienione wartości, należy wywołać `GetChanges`, a następnie wywołać `WriteXml` jako element w formacie DiffGram do elementu typu [DataSet](#).

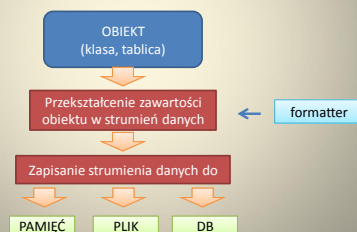
IgnoreSchema Zapisuje bieżącą zawartość [DataSet](#) jako dane XML bez schematu XSD. Jeśli dane nie są ładowane do [DataSet](#), nic nie jest zapisywane.

WriteSchema Zapisuje bieżącą zawartość [DataSet](#) jako dane XML ze strukturą relacyjną jako wbudowanego schematu XSD. Jeśli [DataSet](#) ma tylko w schemacie bez danych są zapisywane tylko wbudowanego schematu. Jeśli [DataSet](#) jest nie ma bieżącego schematu, nic nie jest zapisywane.

[http://msdn.microsoft.com/pl-pl/library/system.data.xmlwritemode\(v=vs.110\).aspx](http://msdn.microsoft.com/pl-pl/library/system.data.xmlwritemode(v=vs.110).aspx)
Tłumaczenie – robot MS ©, delikatna poprawa

SERIALIZACJA

Serializacja to operacja przekształcania obiektu danych (klasa, tablica, zmienna) w aktualnym stanie do postaci binarnej (lub innej), pozwalającej na późniejsze odтворzenie stanu obiektu (deserializacja).



SERIALIZACJA

Rodzaje serializacji w .NET:

Serializacja binarna – zapisanie stanu klasy jako strumienia jej danych odpowiednio do ich reprezentacji w pamięci. Jest to najszybsza i najefektywniejsza serializacja, niestety wadą jest zależność formatu od użytej wersji .NET – szczególnie np. między x86 a AMD64.

Serializacja XML – metoda zapisu obiektów w formacie XML, z uwzględnieniem schematu danych. Zapewnia to większą przenośność między środowiskami, jednakże daje większy zestaw wyjściowy, a czas przetwarzania jest dłuższy.

Serializacja SOAP – metoda zapisu obiektów w formacie XML zgodnie ze standardem SOAP.

SERIALIZACJA

```
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO.Compression;

namespace MojProjekt
{
    [Serializable]
    public class DaneStanu
    {
        public Bitmap mapabmp = null;

        public string path_bmp = "C:\\\\";

        public int wartosc_max = 255;

        public byte[] tabela = null;

        //...
    }
}
```

SERIALIZACJA

Podstawowe operacje na dowolnym obiekcie

```
IFormatter formatter = new BinaryFormatter();
Stream strumien = new FileStream(sciezka_pliku, FileMode.Create,
                                FileAccess.Write, FileShare.None);
formatter.Serialize(strumien, obiekt_do_serializacji);
strumien.Close();
```

```
IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream(filen, FileMode.Open, FileAccess.Read,
                               FileShare.Read);
DaneStanu danest = (DaneStanu) formatter.Deserialize(stream);
stream.Close();
```

SERIALIZACJA

np. z kompresją

```
IFormatter formatter = new BinaryFormatter();
Stream stream = new MemoryStream();
formatter.Serialize(stream, obj);
stream.Position = 0;

FileStream fstr = new FileStream(plik, FileMode.Create,
                                FileAccess.Write, FileShare.None);

GZipStream gzs = new GZipStream(fstr, CompressionMode.Compress, false);
//...
gzs.Write(bufor, 0, (int)stream.Length);
//...

IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream(filen, FileMode.Open, FileAccess.Read,
                               FileShare.Read);

GZipStream gZipStream = new GZipStream(stream,
                                       CompressionMode.Decompress);
obj = formatter.Deserialize(gZipStream);
stream.Close();
gZipStream.Close();
```

SERIALIZACJA

custom serialization

```
[Serializable]
public class MojaKlasa : ISerializable, IDeserializationCallback
{
    public MojaKlasa (SerializationInfo info, StreamingContext context)
    {
        wartosc_max = info.GetInt32("wartosc_maksymalna");
        byte[] tabela = null;
        tabela = (byte[])info.GetValue("danetabeli", typeof(byte[]));
    }
    //...

    [SecurityPermissionAttribute(SecurityAction.Demand, SerializationFormatter = true)]
    #region ISerializable Members
    public void GetObjectData(SerializationInfo info, StreamingContext ctx)
    {
        info.AddValue("wartosc_maksymalna", wartosc_max);
        info.AddValue("danetabeli", tabela);
    }
    //...
```