Računarstvo i informatika

Katedra za računarstvo
Elektronski fakultet u Nišu

Baze podataka (Računske vežbe)
ADO.NET (1)

Letnji semestar 2016/2017

Sadržaj

- Osnovni pojmovi
- Arhitektura
- Connection
- Command
- DataReader

Osnovni pojmovi

- Većina aplikacija ima potrebu za nekom vrstom obrade podataka.
- Relacione baze podataka predstavljaju domainantnu tehnologiju za skladištenje podataka.
- Svaki DBMS poseduje sopstveni programski interfejs (API) čijim korišćenjem je moguće iz programskog koda odnosno iz aplikacije, vršiti manipulaciju podacima u bazi podataka.
- Programski interfejs (API) predstavlja kolekciju objekata i metoda koji omogućavaju pozivanje funkcija DBMS-a iz programskog koda.
- Svaki DBMS poseduje svoj API pa je bilo neophodno razviti standarde za pristup bazama podataka kako projektanti aplikacija ne bi morali da koriste različite intefejse u zavisnosti od konkretnog DBMS-a koji koriste.

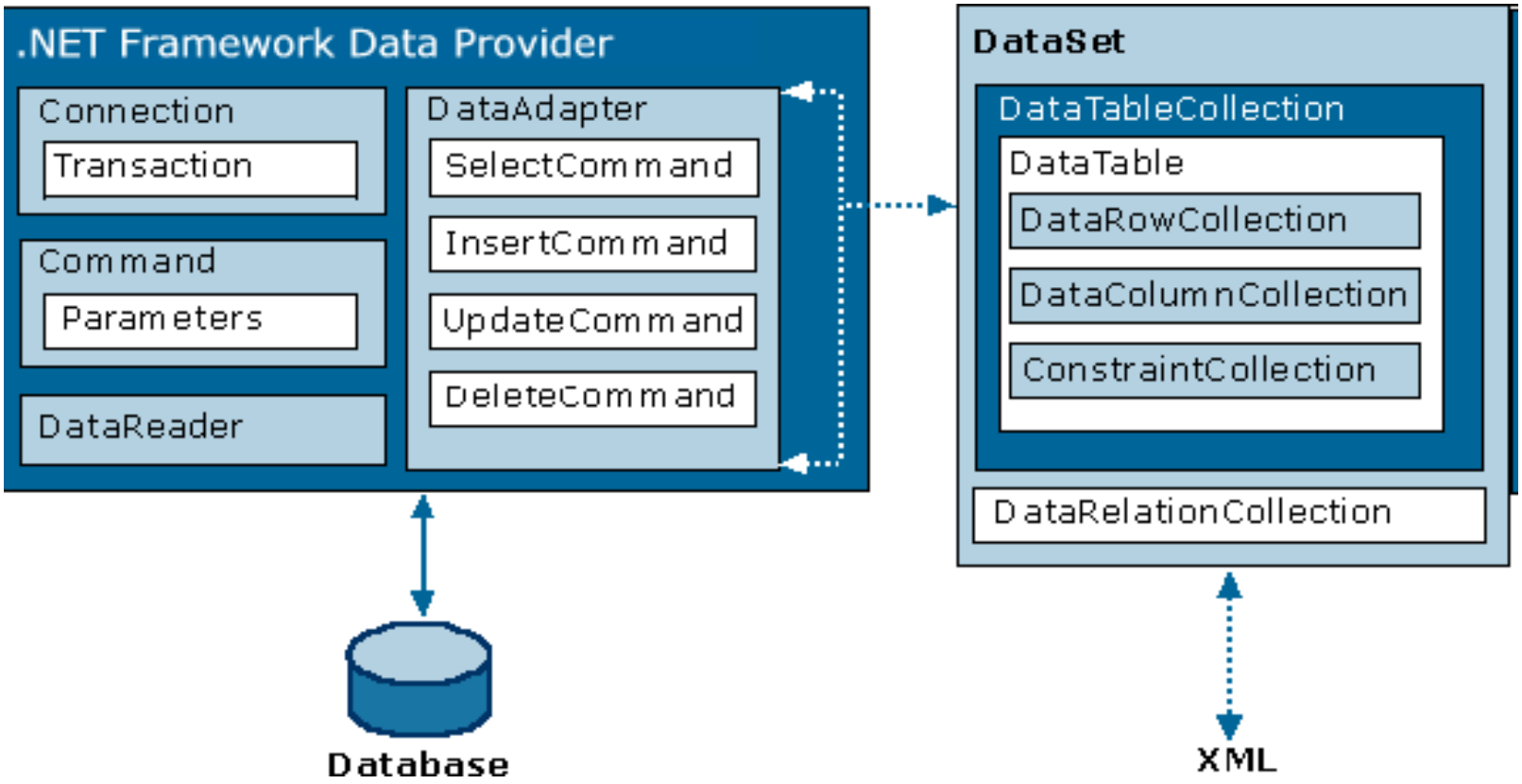
Osnovni pojmovi

- U prošlosti je razvijeno više standardnih interfejsa za pristup bazama podataka.
- **Open Database Connectivity (ODBC)** standard je razvijen sa ciljem da obezbedi načine za manipulaciju podacima u relacionim bazama podataka koji bi bili nezavisni od konkretnog DBMS-a.
- **OLE DB** je objektno-orijentisani interfejs niskog nivoa koji je implementirao Microsoft kako bi enkapsulirao funkcionalnosti DBMS-a. OLE DB je razvijen ne samo za relacione baze podataka već ima i mogućnost korišćenje drugih tipova podataka.
- OLE DB interfejs nisu mogli koristiti projektanti koji su svoje aplikacije razvijali korišćenjem Visual Basic-a i script jezika pa je Microsoft razvio **Active Data Object (ADO)** interfejs. ADO se bazira na funkcionalnosti OLE DB interfejsa i može biti korišćen iz bilo kog programskog jezika.

Osnovni pojmovi

- **ADO.NET** predstavlja skup komponenti koje obezbeđuju mehanizme za interakciju sa različitim izvorima podataka.
- ADO.NET komponente su sastavni deo Microsoft .NET Framework okruženja.
- ADO.NET komponente su implementirane u okviru **System.Data** biblioteke.
- ADO.NET se prevashodno koristi za pristupanje i izmenu podataka koji se nalaze u relacionim bazama podataka.
- ADO.NET obezbeđuje i mehanizme za pristupanje podacima iz drugih izvora podataka: tekstualne datoteke, XML datoteke, Excel datoteke i sl.

Arhitektura



Arhitektura

- ADO.NET ne poseduje jedinstveni skup objekata koji komuniciraju sa različitim izvorima podataka.
- Za interakciju sa određenim izvorom podataka zadužena je komponenta **ADO.NET data provider**.
- Svaki od data provider-a optimizovan je za interakciju sa konkretnim izvorom podataka.
- Prednosti ovakvog pristupa su:
  - ADO.NET data provider ima mogućnost da manipuliše objektima koji su specifični za konkretni izvor podataka
  - ADO.NET data provider komunicira direktno sa konkretnim izvorom podataka odnosno nema potrebe za dodatnim međuslojem koji zahteve korisnika prilagođava izvoru podataka.
  - ADO.NET data provider ka klijentima implementira jedinstveni interfejs bez obzira na konkretan izvor podataka.

Arhitektura

- .NET Framework data provider predstavlja biblioteku komponenti koje implementiraju mehanizme za interakciju sa specifičnim izvorom podataka.
- .NET Framework standardno uključuje sledeće data provider biblioteke:
  - Data Provider for SQL Server (System.Data.SqlClient).
  - Data Provider for OLEDB (System.Data.OleDb).
  - Data Provider for ODBC (System.Data.Odbc).
- Postoji veliki broj ADO.NET data provider biblioteka koje su razvijene za potrebe drugih proizvoda (Oracle, MySQL, PostgreSQL, ...).
- Dodatni ADO.NET data provider-i moraju eksplicitno da se instaliraju kako bi povezivanje sa drugim izvorima podataka bilo moguće.

Arhitektura

- ADO.NET data provider standardno implementira sledeće komponente:

Objekat	Primer	Značenje
Connection	SqlConnection, OleDbConnection, OdbcConnection, OracleConnection	Omogućava otvaranje i zatvaranje veze ka izvoru podataka.
Command	SqlCommand, OleDbCommand, OdbcCommand, OracleCommand	Predstavlja komandu (najčešće SQL upit) konkretnog izvora podataka. Omogućava pristup DataReader objektu konkretnog data provider-a
DataReader	SqlDataReader, OleDbDataReader, OdbcDataReader, OracleDataReader	Omogućava čitanje podataka korišćenjem kursora na serverskoj strani
DataAdapter	SqlDataAdapter, OleDbDataAdapter, OdbcDataAdapter, OracleDataAdapter	Omogućava komunikaciju između DataSet objekta i konkretnog izvora podataka. Poseduje konekciju i skup od četiri komande koje predstavljaju osnovne operacije za selektovanje, dodavanje, izmenu i brisanje podataka u izvoru podataka
Parameter	SqlParameter, OleDbParameter, OdbcParameter, OracleParameter	Predstavlja imenovani parametar u parametrizovanoj komandi.
Transaction	SqlTransaction, OleDbTransaction, OdbcTransaction, Oracletransaction	Enkapsulira transakciju izvora podataka.

Arhitektura

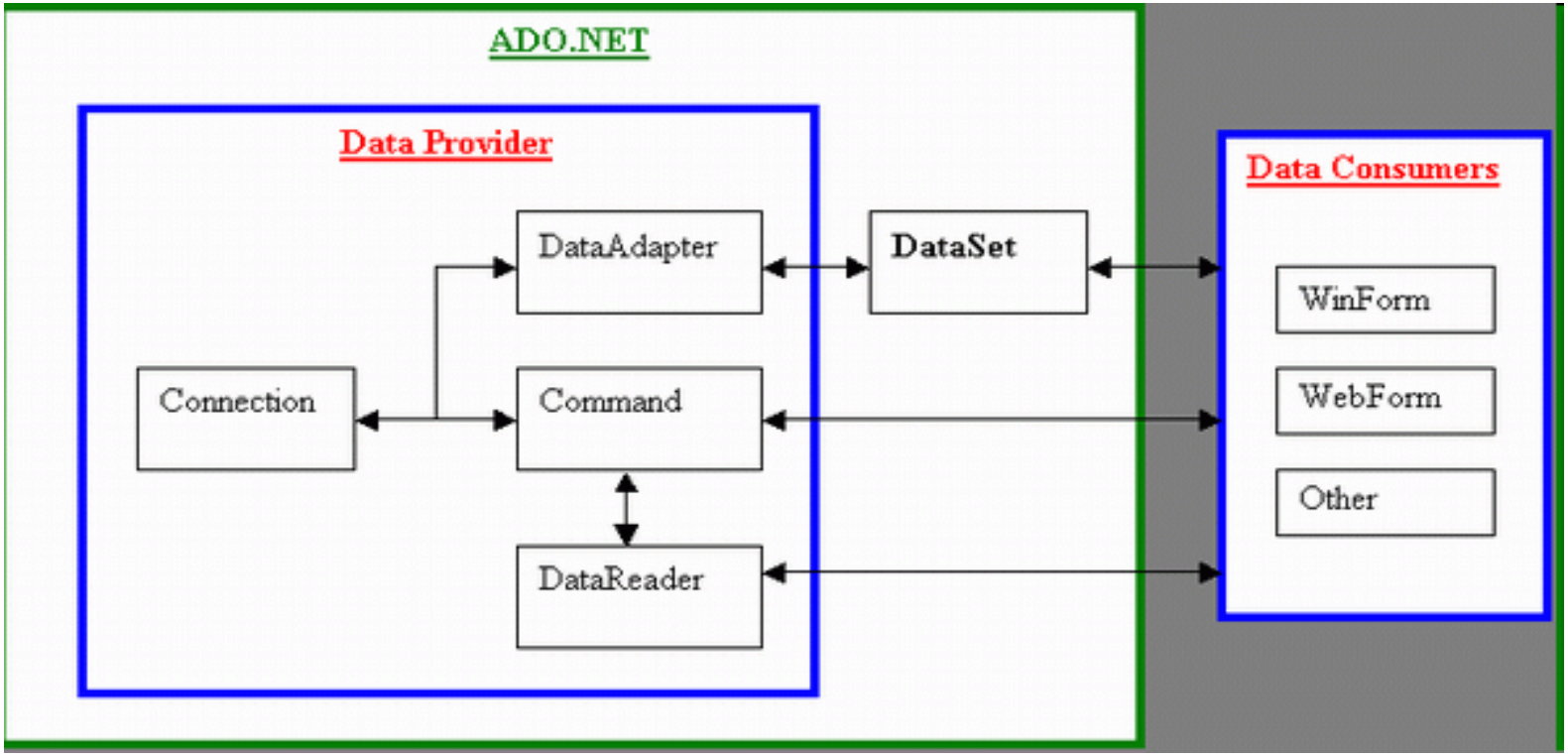
- **DataSet** predstavlja memorijsku reprezentaciju podataka pribavljenih iz jednog ili više izvora (relacionih baza podataka, XML dokumenata, lokalnih podataka aplikacija i sl.).
- DataSet je implementiran u biblioteci System.Data.
- DataSet obezbeđuje uniformne mehanizme za rad sa podacima bez obzira na njihovo poreklo (bez obzira na ADO.NET data provider koji je korišćen za njihovo pribavljanje).
- DataSet obezbeđuje nezavisnost klijenata od specifičnosti konkretnih izvora podataka.

Arhitektura

- Biblioteka System.Data implementira sledeće komponente za rad sa podacima:

Klasa	Značenje
Constraint	Predstavlja ograničenje primenjeno na DataColumn objekat
DataColumn	Predstavlja jednu kolonu DataTable objekta
DataRelation	Predstavlja roditelj/dete odnos između dva DataTable objekta
DataRow	Predstavlja jedan red u DataTable objektu
DataSet	Predstavlja lokalnu kopiju podataka u memoriji klijentskog računara koji se sastoji od niza povezanih DataTable objekata
DataTable	Predstavlja lokalnu kopiju tabele baze podataka
DataTableReader	Omogućava čitanje podataka iz DataTable objekta red po red
DataView	Predstavlja pogled na tabelu baze podataka i koirsti se za sortiranje, filtriranje, pretraživanje i izmenu podataka

Arhitektura



Connection

- Objekat **Connection** enkapsulira komunikaciju sa izvorom podataka.
- Objekat Connection ima zadatak da uspostavi i omogući komunikaciju sa izvorom podataka.
- Prilikom kreiranja instance klase Connection potrebno je definisati **ConnectionString** atribut ove klase.
- ConnectionString atribut predstavlja niz **ime=vrednost** parova koji su međusobno odvojeni karakterom ';'.
- Ovaj atribut sadrži informaciju o lokaciji i nazivu izvora podataka kome pristupam, načinu autentifikacije korisnika i sl.
- Broj raspoloživih konekcija je ograničen pa je potrebno držati konekciju otvorenom što je kraće moguće.

Connection

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//uključuje se Oracle data provider
using Oracle.DataAccess.Client;


namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {

            //definicija ConnectionString-a
            string conString = "Data Source=vezbe;User Id=S1010;Password=S1010;";

            //Kreiranje i inicijalizacija objekta Connection
            OracleConnection con = new OracleConnection(conString);

            // uspostavljanje konekcije sa izvorom podataka na osnovu
            //parametara prosleđenih u ConnectionString-u
            con.Open();

        }
    }
}
```

Connection

- Svaku uspostavljenu konekciju je potrebno zatvoriti (raskinuti) pre završetka rada.
- Za zatvaranje konekcije za izvorom podataka koriste se metoda **Close()** ili metoda **Dispose()** objekta Connection.
- Treba izbegavati konekcije koje se uspostavljaju na početku rada aplikacije i raskidaju tek prilikom zatvaranja aplikacije.
- Preporuka za savremene aplikacije je da se konekcije kreiraju i uspostavljaju samo po potrebi. Po završetku operacije sa podacima konekcija se raskida.

Connection

```
static void Main(string[] args)
{
    string conString;
    OracleConnection con = null;

    try
    {
        conString = "Data Source=160.99.9.139/gislab;User Id=S1010;Password=S1010;";
        con = new OracleConnection(conString);
        con.Open();
        //neka obrada podataka
    }
    catch (Exception ec)
    {
        Console.WriteLine("Greska pri konekciji: " + ec.Message);
    }
    finally
    {
        if (con != null && con.State == System.Data.ConnectionState.Open)
            con.Close();
    }

}
```

Command

- Objekat Command omogućava da se nakon uspostavljanja konekcije sa izvorom podataka, izvoru proslede komande na izvršavanje i da se prihvate rezultati izvršavanja tih komandi.
- Objekat Command se može kreirati samo ako postoji objekat Connection koji je kreiran i povezan sa nekim izvorom podataka.
- U slučaju da je izvor podataka relaciona baza podataka na izvršavanje se najčešće prosleđuje SQL komanda.
- Tip komande specificiran je **CommandType** atributom i može imati neku od vrednosti iz ComandType enumeracije.

```
public enum CommandType
{
    StoredProcedure, //uskladistena procedura
    TableDirect,//direktan pristup tabeli
    Text //tekstualna komanda (npr. SQL naredba)
}
```

Command

- Objekat Command omogućava **direktan pristup podacima** koji se nalaze u izvoru podataka.
- Kod direktnog pristupa nema kreiranja lokalnih kopija podataka.
- Direktan pristup zahteva stalno postojanje otvorene konekcije ka izvoru podataka.
- Direktan pristup podrazumeva sledeći niz koraka:
  - Uspostavljanje konekcije ka izvoru podataka (kreiranje i otvaranje objekta Connection)
  - Kreiranje i pripremanje objekta Command
  - Izvršavanje komande i prihvatanje rezultata obrade (lokalne promenljive ili objekta DataReader)
  - Obrada podataka
  - Zatvaranje konekcije

Command

```
conString = "Data Source=160.99.9.139/gislab;User Id=S1010;Password=S1010;";
con = new OracleConnection(conString);
con.Open();

//1. način, komanda izvršava upit
OracleCommand cmd1 = new OracleCommand("SELECT * FROM RADNIK", con);

//2. način, komanda izvršava upit
OracleCommand cmd2 = new OracleCommand();
cmd2.Connection = con;
cmd2.CommandText = "SELECT * FROM SEKTOR";
cmd2.CommandType = System.Data.CommandType.Text;

//3. način, komanda učitava sve vrste iz tabele
OracleCommand cmd3 = con.CreateCommand();
cmd3.CommandText = "RADNIK";
cmd3.CommandType = System.Data.CommandType.TableDirect;
```

Command

- Nakon kreiranja objekat Command je pripremljen za upotrebu ali komanda nije izvršena.
- Za izvršenje se koristi jedna od metoda **Execute** koje objekat Command implementira.
- U zavisnosti od tipa rezultata koje komanda vraća koristi se odgovarajući oblik metode Exeecute:
  - **ExecuteNonQuery()** – za izvršavanje komandi koje ne vraćaju rezultat (SQL naredbe INSERT, UPDATE, DELETE i druge)
  - **ExecuteReader()** – za izvršavanje naredbi koje kao rezultat vraćaju relaciji odnosno stream podataka (SQL naredba SELECT)
  - **ExecuteScalar()** – za izvršavanje naredbi koje kao rezultat vraćaju skalarnu vrednost (pojedini oblici SQL naredbe SELECT)

Command

```
conString = "Data Source=160.99.9.139/gislab;User Id=S1010;Password=S1010;";
con = new OracleConnection(conString);
con.Open();

//izvršavanje komande koja ne vraća rezultate
OracleCommand cmdNonQuery = new OracleCommand("DELETE FROM RADNIK", con);
cmdNonQuery.ExecuteNonQuery();

cmdNonQuery.CommandText = "INSERT INTO RADNIK VALUES ('123456789','Marko', "
                          + "'J', 'Petrovic', '1/9/1965','Obilićev Venac', "
                          + " 'M', '30000')";
cmdNonQuery.ExecuteNonQuery();

cmdNonQuery.CommandText = "UPDATE RADNIK SET PLATA = 50000 "
                         + " WHERE MATBR = 123456789";
cmdNonQuery.ExecuteNonQuery();

//izvršavanje komande koja vraća skalarni rezultat
OracleCommand cmdScalar = new OracleCommand("SELECT COUNT(*) FROM RADNIK", con);
decimal count = (decimal)cmdScalar.ExecuteScalar();

//izvršavanje komande čiji je rezultat nova relacija
OracleCommand cmdQuery = new OracleCommand("SELECT * FROM RADNIK", con);
OracleDataReader dr = cmdQuery.ExecuteReader();
```

DataReader

- Objekat **DataReader** omogućava prihvatanje rezultata izvršavanja komande (SQL SELECT komanda koja vraća novu relaciju).
- Podaci koje prihvata DataReader se **ne mogu menjati (read only)** i može im se pristupati samo **strogo sekvencijalno – od prvog ka poslednjem (forward only).**
- Objekat DataReader je izuzetno pogodan kada se obrađuje **velika količina podataka** jer se podaci ne čuvaju u memoriji.
- U svakom trenutku objekat DataReader u memoriji baferuje samo jednu vrstu rezultujuće relacije.

DataReader

- Za prihvatanje vrste rezultujuće tabele koristi se metoda **Read()**.
- Metoda Read() pribavlja iz izvora podataka jedan slog iz rezultujuće tabele i smešta ga u lokalnu memoriju.
- Za pristupanje podacima u pojedinim kolonama mogu se koristiti funkcije specijalizovane za pojedine tipove podataka:
  - GetInt32
  - GetString
  - GetFloat
  - GetDouble
  - GetDecimal
  - GetDateTime
  - ...

DataReader

- Po završetku obrade podataka potrebno je zatvoriti objekat DataReader pozivom metode **Close()** ili metode **Dispose()**.
- Za vreme svog postojanja objekat DataReader **ekskluzivno koristi** otvorenu konekciju.
- Za vreme njegovog postojanja konekcija se **ne može iskoristiti** za kreiranje novog objekta DataReader ili izvršavanje druge komande.

DataReader

```
conString = "Data Source=160.99.9.139/GISLAB;User Id=S1010;Password=S1010;";
con = new OracleConnection(conString);
con.Open();

//maticni brojevi, imena i prezimena svih radnika
string strSQL = "SELECT MATBR, LIME, PREZIME FROM RADNIK";
OracleCommand cmdQuery = new OracleCommand(strSQL, con);

//izvrsava se SQL komanda koja vraca rezultujucu tabelu
OracleDataReader dr = cmdQuery.ExecuteReader();

//proverava se da li rezultujuca tabela ima vrste
if (!dr.HasRows)
    return;

//cita se vrsta po vrsta rezultujuce tabele
//kada se dodje do kraja metoda Read vraca FALSE
while (dr.Read())
{
    //vrednosti kolona iz tekuce vrste
    //poziva se odgovarajuca metoda u zavisnosti od tipa kolone
    long matbr = dr.GetInt64(0);
    string ime = dr.GetString(1);
    string prezime = dr.GetString(2);

    Console.WriteLine(matbr.ToString() + " " + ime + " " + prezime);
}

dr.Close();
```

DataReader

```
conString = "Data Source=160.99.9.139/GISLAB;User Id=S1010;Password=S1010;";
con = new OracleConnection(conString);
con.Open();

//nazivi sektora i broj radnika po sektoru
string strSQL = "SELECT NAZIV, COUNT(*) "
                + " FROM SEKTOR, RADNIK "
                + " WHERE BRSEK = SBROJ "
                + " GROUP BY BRSEK, NAZIV ";
OracleCommand cmdQuery = new OracleCommand(strSQL, con);

//izvrsava se SQL komanda koja vraca rezultujucu tabelu
OracleDataReader dr = cmdQuery.ExecuteReader();

//proverava se da li rezultujuca tabela ima vrste
if (!dr.HasRows)
    return;

//cita se vrsta po vrsta rezultujuce tabele
//kada se dodje do kraja metoda Read vraca FALSE
while (dr.Read())
{
    //vrednosti kolona iz tekuce vrste
    //poziva se odgovarajuca metoda u zavisnosti od tipa kolone
    string naziv = dr.GetString(0);
    decimal broj = dr.GetDecimal(1);

    Console.WriteLine(naziv + " " + broj.ToString());
}

dr.Close();
```