საქართველოს ტექნიკური უნივერსიტეტი

რომან სამხარაძე, ნინო ჯოჯუა, ლია გაჩეჩილაძე

მეთოდური მითითებები ლაბორატორიული სამუშაოების შესასრულებლად საგანში: დაპროგრამება .NET გარემოში

I ნაწილი

საქართველოს ტექნიკური უნივერსიტეტი

რომან სამხარაძე, ნინო ჯოჯუა, ლია გაჩეჩილაძე

მეთოდური მითითებები ლაბორატორიული სამუშაოების შესასრულებლად საგანში: დაპროგრამება .NET გარემოში

I ნაწილი

დამტკიცებულია
მეთოდურ მითითებად
სტუ-ის სარედაქციო
საგამომცემლო
საბჭოს მიერ

თბილისი - 2013

სასწავლო კურსის ათვისების შემდეგ სტუდენტი შეძლებს: საინფორმაციო და მიკროპროცესორული სისტემებისთვის პროგრამული უზრუნველყოფის შემუშავებასა და განხორციელებას .NET პლატფორმაზე; სერიული პორტების, ADO.NET კლასებისა და ობიექტების შესწავლას და მათ გამოყენებას აღნიშნული ტიპის სისტემების შესაქმნელად.

რეცენზენტები: სრული პროფესორი მედეა ანდღულაძე
ასოცირებული პროფესორი ზაურ ჯოჯუა

სარჩევი

თავი 1. მოწყობილობის პროტოკოლის ანალიზი და მართვის პროგრამის შედგენა. ფაილებთან მუშაობის დაგეგმვა.

მიკროპროცესორული მოწყობილობა მუშაობს MODBUS პროტოკოლით და კომპიუტერს COM პორტის საშუალებით უკავშირდება. მოწყობილობის თითოეული რეგისტრი ოთხბაიტიანია. ამ პროტოკოლის მიხედვით კომპიუტერმა მოწყობილობას უნდა გაუგზავნოს ინფორმაცია, რომელსაც შემდეგი ფორმატი აქვს:

პირველი ბაიტი - მოწყობილობის მისამართი;

მეორე ბაიტი - წაკითხვის ბრძანება;

მესამე და მეოთხე ბაიტები - მოწყობილობის იმ რეგისტრის მისამართი, საიდანაც ინფორმაცია უნდა მივიღოთ;

მეხუთე და მეექვსე ბაიტები - რეგისტრების რაოდენობა;

მეშვიდე და მერვე ბაიტები - საკონტროლო ჯამი.

ამ ბრძანების საპასუხოდ მოწყობილობა დაახლოებით 600 მილიწამის შემდეგ კომპიუტერს უგზავნის ინფორმაციას, რომელსაც პროტოკოლის მიხედვით, შემდეგი ფორმატი აქვს:

პირველი ბაიტი - მოწყობილობის მისამართი;

მეორე ბაიტი - წაკითხვის ბრძანება;

მესამე ბაიტი - ბაიტების რაოდენობა;

მეოთხე, მეხუთე, მეექვსე და მეშვიდე ბაიტები - პირველი რეგისტრის მნიშვნელობები;

მერვე, მეცხრე, მეათე და მეთერთმეტე ბაიტები - მეორე რეგისტრის მნიშვნელობები;

მეთორმეტე და მეცამეტე ბაიტები - საკონტროლო ჯამი.

ოთხბაიტიანი რეგისტრიდან მიღებული მონაცემი მთელ რიცხვად შეგვიძლია შემდეგნაირად გარდავქმნათ:

$Y = X1 * 2563 + X2 * 2562 + X3 * 2561 + X4 * 2560$

აქ Y - გარდაქმნის შედეგად მიღებული მთელი რიცხვია, X1 - რეგისტრის უფროსი ბაიტია, X2 - რეგისტრის მომდევნო ბაიტი და ა.შ. გარდაქმნის შედეგად მიღებული მთელი რიცხვები ჩავწეროთ ფაილში.

კონკრეტულად, თუ რომელი პორტი გამოყო ოპერაციულმა სისტემამ მიკროპროცესორულ მოწყობილობასთან სამუშაოდ, შეგვიძლია ვნახოთ Device Manager ფანჯარაში. სწორედ, ამ გამოყოფილი პორტის სახელი უნდა მივუთითოთ პროგრამაში პორტის პარამეტრების განსაზღვრის დროს. ჩვენს შემთხვევაში ინფორმაციის წაკითხვა მრიცხველიდან, კერძოდ აბონენტის ნომერი და მრიცხველის ჩვენება; მონაცემები განთავსდება faili.dat ფაილში.

{

```
    BinaryWriter file_out = new BinaryWriter(new
FileStream("faili.dat", FileMode.Create));
    long ricxvi1, ricxvi2;
    int wakitxuli_baitebis_raodenoba;
    byte[] buf8 = new byte[8];  // ბუფერი მონაცემების
გასაგზავნად
    byte[] buf13 = new byte[13];  // ბუფერი მონაცემების
მისაღებად
    // მოწყობილობისთვის გასაგზავნი მონაცემების
მომზადება
    buf8[0] = 2; // მოწყობილობის მისამართი
    buf8[1] = 3; // მონაცემების წაკითხვის ბრძანება
    buf8[2] = 29; // რეგისტრის მისამართი
    buf8[3] = 79;
    buf8[4] = 0; // რეგისტრების რაოდენობა
    buf8[5] = 2;
    buf8[6] = 243; // საკონტროლო ჯამი
    buf8[7] = 131;
    SerialPort COM_Porti_1 = new SerialPort();
    // პორტის პარამეტრების მომზადება
    COM_Porti_1.BaudRate = 38400; // პორტის
სწრაფქმედება
    COM_Porti_1.DataBits = 8; // ბაიტში მონაცემების
ბიტების რაოდენობა
    COM_Porti_1.Parity = Parity.None; // პარიტეტი არ
გვაქვს
```

```
    COM_Porti_1.PortName = textBox1.Text; // პორტის
სახელი
    COM_Porti_1.Open(); // პორტის გახსნა
    // პორტში მონაცემების გაგზავნა
    COM_Porti_1.Write(buf8, 0, 8);
    Thread.Sleep(600); // დაყოვნება 600 მილიწამი
    // პორტიდან მონაცემების მიღება
    wakitxuli_baitebis_raodenoba =
COM_Porti_1.Read(buf13, 0, 13);
    // რეგისტრებიდან მიღებული მონაცემების
გარდაქმნა
    ricxvi1 = buf13[3] * 256 * 256 * 256 + buf13[4] * 256 * 256 +
buf13[5] * 256 + buf13[6];
    ricxvi2 = buf13[7] * 256 * 256 * 256 + buf13[8] * 256 * 256 +
buf13[9] * 256 + buf13[10];
    // მონაცემების ჩაწერა ფაილში
    file_out.Write(ricxvi1);
    file_out.Write(ricxvi2);
    file_out.Close();
    //
    label1.Text = COM_Porti_1.PortName; // პორტის
სახელის ეკრანზე გამოტანა
    label2.Text = wakitxuli_baitebis_raodenoba.ToString();
    label3.Text = "წაკითხული ბაიტების რაოდენობა = " +
COM_Porti_1.BytesToRead.ToString() +
```

```
"\nჩაწერილი ბაიტების რაოდენობა = " +
COM_Porti_1.BytesToWrite.ToString();
// გარდაქმნილი მონაცემების ეკრანზე გამოტანა
label4.Text = ricxvi1.ToString();
label5.Text = ricxvi2.ToString();
//
COM_Porti_1.Close(); // პორტის დახურვა
}
```

ფაილის შექმნისა და მასში უკანასკნელად ჩაწერის თარიღის მისაღებად button ინსტრუმენტს მივაბათ შემდეგი კოდი:

```
{
FileInfo obieqti = new FileInfo("faili.dat");
label1.Text += obieqti.CreationTime.ToString() + '\n';
Label2.Text += obieqti.LastWriteTime.ToString() + '\n';
}
```

ავაგოთ გრაფიკი სადაც ასახული იქნება აბონენტის მიერ წლის განმავლობაში მოხმარებული გაზის ოდენობა და გადახდილი თანხა რაიონის მიხედვით

```
using System;
```

ფორმაზე მოვათავსოთ menuStrip არავიზუალური კომპონენტი, განვათავსოთ მასზე ჩამოშლადი მენიუები ”ფაილი” მასში ”დახურვა” პუნქტით და ”შეტყობინება” ”დიაგრამის აგება” და ”ჰისტოგრამის აგება” პუნქტებით. აგრეთვე მოვათავსოთ ოთხი Label, ორი

textBox და ორი Button კომპონენტი. როგორც პროგრამიდან ჩანს, პროგრამა დიაგრამას აგებს FillPie ფუნქციის საშუალებით, ხოლო ჰისტოგრამას – FillRectangle ფუნქციის საშუალებით. სექტორების პარამეტრების (გრადუსი და ფერი) შესატანად გამოიყენება შვილობილი ფანჯარა. ფერი ენიჭება PictureBox -ის Click მოვლენით.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;

public partial class Form1 : Form
 {
     ArrayList diagrammaSectors = new ArrayList();
     ArrayList diagrammaColors = new ArrayList();
     ArrayList gistogrammaRectangles = new ArrayList();

     ArrayList gistogrammaColors = new ArrayList();
     public Form1()
```

```
    }
     {
            InitializeComponent();
        }
    private void button1_Click(object sender, EventArgs e)
        {
            Graphics g = this.CreateGraphics();
            DiagramCreator dc = null;
            if (((Button)sender).Name == "button1")
            {
                if (textBox1.Text != "" &&
Convert.ToInt32(textBox1.Text) < 361)
                    dc = new
DiagramCreator(Convert.ToInt32(textBox1.Text),"დიაგრამ
ა");
            }
            else
            {
                if (textBox2.Text != "")
                    dc = new
DiagramCreator(Convert.ToInt32(textBox2.Text),"ჰისტოგრ
ამა");
            }
            if (dc != null)
            {
                DialogResult r = dc.ShowDialog();
```

```
            if (r == DialogResult.OK)
            {
                if (((Button)sender).Name == "button1")
                {
                    int grad = 0;
                    for (int i = 0; i < dc.procents.Count; i++)
                    {
    g.FillPie(new SolidBrush(((Color)dc.colors[i])), new
Rectangle(0, 150, 200, 200), grad, ((int)dc.procents[i]));

diagrammaSectors.Add(((int)dc.procents[i]));

diagrammaColors.Add(((Color)dc.colors[i]));
                        grad += ((int)dc.procents[i]);
                    }
                }
                else
                {
    for (int i = 0, addW = 0; i < dc.procents.Count; i++, addW
+= 20)
                    {
                        if (i == 1)
                            addW -= 20;
                        g.FillRectangle(new
SolidBrush(((Color)dc.colors[i])), new Rectangle(292 +
addW, 400 - ((int)dc.procents[i]), 20, ((int)dc.procents[i])));
```

```
gistogrammaColors.Add(((Color)dc.colors[i]));

gistogrammaRectangles.Add(((int)dc.procents[i]));
                    }
                }
            }
        }
    }

        private void
diagramaToolStripMenuItem_Click(object sender,
EventArgs e)
        {
      MessageBox.Show("დიაგრამა შედგება
სექტორებისაგან, ჯამში ყველა სექტორის გრადუსული
ზომა უნდა იყოს 360-ის ტოლი. სექტორებს
სიცხადისთვის სხვადასხვა ფერი ენიჭება", "
ინფორმაცია დიაგრამის აგების შესახებ");
        }
    private void histogramaToolStripMenuItem_Click(object
sender, EventArgs e)
        {
      MessageBox.Show("ჰისტოგრამა
მართკუთხედებისგან აგებული გრაფიკის
საშუალებით ასახავს ცხრილის მონაცემებს. ხშირ
```

შემთხვევაში ამ მართკუთხედების სიგანე ერთნაირია, ხოლო სიმაღლე განისაზღვრება გადაცემული პარამეტრებით.", "ინფორმაცია ჰისტოგრამის აგების შესახებ");
        }
        private void daxurvaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics;
            if (gistogrammaColors.Count != 0)
            {
                for (int i = 0, addW = 0; i < gistogrammaRectangles.Count; i++, addW += 20)
                {
                    if (i == 1)
                        addW -= 20;
                    g.FillRectangle(new SolidBrush(((Color)gistogrammaColors[i])), new Rectangle(292 + addW, 400 -

```
((int)gistogrammaRectangles[i]), 20,
((int)gistogrammaRectangles[i])));
                }
            }
            if (diagrammaColors.Count != 0)
            {
                int grad = 0;
                for (int i = 0; i < diagrammaSectors.Count; i++)
                {
                    g.FillPie(new
SolidBrush(((Color)diagrammaColors[i])), new Rectangle(0,
150, 200, 200), grad, ((int)diagrammaSectors[i]));
                    grad += ((int)diagrammaSectors[i]);
                }
            }
        }
    }
}

    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
```

```
using System.Windows.Forms;
using System.Collections;

public partial class DiagramCreator : Form
  {
    int sectors;
    int x = 4, y = 8;
    int x2 = 114, y2 = 8;
    int ind = 2;
    public ArrayList colors = new ArrayList();
    public ArrayList procents = new ArrayList();
    string type = "";
    public DiagramCreator(int s, string t)
    {
      InitializeComponent();
      sectors = s;
      for (int i = 0; i < sectors ; i++, ind++)
      {
        TextBox temp = new TextBox();
        PictureBox temp2 = new PictureBox();
        temp.Name = "textBox" + ind.ToString();
        temp.Location = new System.Drawing.Point(x,
y);
        temp.Size = new System.Drawing.Size(90, 20);
        temp2.Name = "pictureBox" + ind.ToString();
```

```
                temp2.Location = new
System.Drawing.Point(x2, y2);
                temp2.Size = new System.Drawing.Size(114, 20);
                temp2.Click += new
System.EventHandler(this.pictureBox1_Click);
                temp2.BackColor = Color.Green;
                panel1.Controls.Add(temp);
                panel1.Controls.Add(temp2);
                y += 35;
                y2 = y;
                type = t;
            }
        }
    private void pictureBox1_Click(object sender, EventArgs
e)
        {
            ColorDialog c = new ColorDialog();
            DialogResult r = c.ShowDialog();
            if (r == DialogResult.OK)
                ((PictureBox)sender).BackColor = c.Color;
        }
    private void DiagramCreator_Scroll(object sender,
ScrollEventArgs e)
        {
            Invalidate();
        }
```

```
        private void button1_Click(object sender, EventArgs
e)
        {
            int tso = 0;
            foreach (Control c in panel1.Controls)
            {
                if(c is PictureBox)
                    colors.Add(((PictureBox)c).BackColor);
                try
                {
                    if (c is TextBox)
                    {

procents.Add(Convert.ToInt32(((TextBox)c).Text));
                        tso += Convert.ToInt32(((TextBox)c).Text);
                    }
                }
                catch
                {
                    procents.Add(0);
                }
            }
            DialogResult r = DialogResult.Yes;
            if (tso != 360 && type == "დიაგრამა")
```

```
                r = MessageBox.Show("სექტორების
გრადუსების ჯამი არასწორია,
გაგრძელება?","გაფრხილება",MessageBoxButtons.YesNo);
            if(r==DialogResult.Yes)
                this.DialogResult = DialogResult.OK;
        }
        private void button2_Click(object sender, EventArgs
e)
        {
            this.DialogResult = DialogResult.Cancel;
        }
    }
}
```

თავი 2. მონაცემთა ბაზის სტრუქტურის შემუშავება, ცხრილების დაპროექტება

დავალება:

- შეადგინეთ მონაცემთა ბაზა გაზის კომპანიისთვის. ბაზა შეცავს ორ ცხრილს.
- პირველი ცხრილი შეიცავს ინფორმაციას აბონენტის შესახებ (სახელი და გვარი, მისამართი, აბონენტის ნიმერი, მრიცხველის ნომერი, დავალიანება, გადახდა და ა. შ). ცხრილის სახელია abonenti.
- მეორე ცხრილი შეიცავს ინფორმაციას ქვითრის შესახებ (ნომერი, გაცემის თარიღი, მიმდინარე გადასახადი, დავალიანება, ავანსი, დაფარვის დრო და ა. შ)
- შექმენით ბმა ამ ცხრილებს შორის.
- პირველ ცხრილში ინფორმაცია დაახარისხეთ გვარის ზრდის და დავალიანების კლების მიხედვით, შეასრულეთ პირველი ცხრილის ფილტრაცია იმ აბონენტების მიხედვით რომელთა დავალიანება აღემატება 100 ლარს. შექმენით ფუნქცია, რომელიც დავალიანების ჯამურ თანხას
- შექმენით შენახული პროცედურა, რომელიც გასცემს დავალიანების მქონე აბონენტების სიას მითითებული რაიონის მიხედვით
- აღნიშნულ ცხრილები მოათავსეთ მონაცემთა company ბაზაში.

- შექმენით მონაცემთა ბაზის სარეზერვო და ასლი და მოახდინეთ ბაზის აღდგენა ამ ასლიდან.

ამოხსნა:

ცხრილი 1. aboneneti ცხრილის შექმნა

```
USE company
CREATE TABLE aboneneti
(
abID int,
ab_nomeri nvarchar(8)primary key,
gvari nvarchar(30),
raioni  nvarchar(20)
misamarTi nvarchar(50),
telefoni nvarchar(8)
mricxvelis_nomeri nvarchar(10)
davalianeba float null,
```

ცხრილი 2. qviTari ცხრილის შექმნა

```
USE company
CREATE TABLE qviTari
(
qvID int primary key,
qv_nomeri nvarchar(12)
ab_nomeri nvarchar(8),
davalianeba float null,

zveli_chveneba int ,
```

```
    axali_chveneba int ,
    aRricxvis_TariRi datetime,
    zveli_davalianeba float null,
    gadasaxdelia as (axali_chveneba-
zveli_chveneba)*0.5090+zveli_davalianeba
    )
```

ქვევით ნაჩვენებია თუ როგორ უნდა დავამყაროთ კავშირი SQL სერვერზე მოთავსებულ company მონაცემთა ბაზასთან და როგორ ავსახოთ ცხრილების სტრიქონები ფორმაზე მოთავსებულ კომპონენტებში.

Form1 კომპონენტზე ორჯერ სწრაფად დავაწკაპუნოთ და using დირექტივების ბლოკში ჩავუმატოთ using System.Data.SqlClient; დირექტივა. შემდეგ, ისევ გავხსნათ Form1.cs [Designe] განყოფილება და ToolBox ფანჯრის AllWindowsForms განყოფილებიდან BindingSource კომპონენტი ფორმაზე მოვათავსოთ. მას დაერქმევა bindingSource1 სახელი და გამოჩნდება ფორმის ქვედა ზონაში, რადგან ის არავიზუალური კომპონენტია. მოვნიშნოთ ის. გავხსნათ Properties ფანჯრის DataSource სია და დავაჭიროთ Add Project Data Source მიმართვას.

გაიხსნება ფანჯარა, რომელშიც ვირჩევთ Database მონაცემთა წყაროს და ვაჭერთ Next კლავიშს. მომდევნო ფანჯარაში ვხსნით Which data connection ... სიას და ვირჩევთ მონაცემთა ბაზას. თუ ამ სიაში არ არის ჩვენთვის საჭირო ბაზა, მაშინ ვაჭერთ New Connection კლავიშს da გახსნილი ფანჯრის Server name სვეტში შევიტანოთ სერვერის (კომპიუტერის) სახელი. ჩავრთოთ Use SQL Server

Authentication გადამრთველი. User name და Password სვეტებში შევიტანოთ მომხმარებლის სახელი და პაროლი. Select or enter a database name სიიდან ავირჩიოთ საჭირო მონაცემთა ბაზის სახელი და დავაჭიროთ OK კლავიშს.

ვუბრუნდებით წინა ფანჯარას დავაჭიროთ Connection string სტრიქონის მარცხნივ მოთავსებულ + ნიშანს. გამოჩნდება მონაცემთა ბაზასთან შეერთების სტრიქონი, რომელშიც სამი პარამეტრი ჩანს: სერვერის (კომპიუტერის) სახელი - Roma, მონაცემთა ბაზის სახელი - company და მომხმარებლის სახელი - sa. ჩავრთოთ Yes, include sensitive data in the connection string გადამრთველი. შედეგად, შეერთების სტრიქონს დაემატება მეოთხე პარამტერი - პაროლი. ვაჭერთ Next კლავიშს. გახსნილ ფანჯარაში გამოჩნდება შეერთების სახელი. სურვილის შემთხვევაში შეგვიძლია მისი შეცვლა. თუ Yes, save the connection as გადამრთველს ჩართულს დავტოვებთ, მაშინ შეერთების სტრიქონი შეინახება ჩვენი პროგრამის საკონფიგურაციო ფაილში. ვაჭერთ Next კლავიშს.

მომდევნო ფანჯარაში ვქმნით companyDataSet მონაცემთა ნაკრებს, რომელიც მოთავსებული იქნება ჩვენს ლოკალურ კომპიუტერზე. მასში შეგვიძლია მოვათავსოთ ისეთი ობიექტები, როგორიცაა ცხრილები, წარმოდგენები, შენახული პროცედურები და ფუნქციები. მათ გახსნილი ფანჯრიდან ვირჩევთ. ავირჩიოთ abonenti, qviTari და davalianebis_mqone ობიექტები. Finish კლავიშზე დაჭერის შემდეგ, ფორმის ქვეშ გამოჩნდება companyDataSet არავიზუალური კომპონენტი. შემდეგ, ისევ მოვნიშნოთ bindingSource1 კომპონენტი. Properties ფანჯარაში გავხსნათ DataMember სია და ავირჩიოთ abonenti ცხრილი. შედეგად, შეიქმნება PersonaliTable Adapter არავიზუალური კომპონენტი .

abonenti და qviTari ცხრილებს შორის კავშირის დასამყარებლად მოვნიშნოთ companiDataSet კომპონენტი, გავხსნათ კონტექსტური მენიუ და შევასრულოთ Edit in DataSet Designer ბრძანება. გაიხსნება compani1DataSet.xsd განყოფილება. ცარიელ ადგილზე მოვათავსოთ კურსორი, გავხსნათ კონტექსტური მენიუ, Add ქვემენიუ და შევასრულოთ Relation ბრძანება. გახსნილ ფანჯარაში, რომლის Name სვეტში ჩანს კავშირის სახელი - abonenti_qviTari. სურვილის შემთხვევაში შეგვიძლია მისი შეცვლა. Parent Table სიიდან ვირჩევთ მთავარი (მშობელი) ცხრილის სახელს - abonenti , ხოლო Childe Table სიიდან კი - დამოკიდებული (შვილობილი) ცხრილის სახელს - qviTari . Key Columns სიიდან ვირჩევთ მთავარი ცხრილის პირველად გასაღებს. ესაა abonenti ცხრილის abID სვეტი. Foreign Key Columns სიიდან ვირჩევთ დამოკიდებული ცხრილის გარე გასაღებს. ესაა qviTari ცხრილის qvID სვეტი. ჩავრთოთ Relation Only გადამრთველი და დავაჭიროთ OK კლავიშს. companiDataSet.xsd განყოფილებაში გამოჩნდება ცხრილებს შორის კავშირი.

თავი 3. INSERT ბრძანების ფორმირება ცხრილებში მონაცემების ჩასამატებლად.

ცხრილში ახალი სტრიქონის დასამატებლად და მათში მონაცემების ჩასაწერად გამოიყენება INSERT ბრძანება. ამის განხორციელება კი შესაძლებელია SqlCommand ობიექტის ExecuteNonQuery() მეთოდის საშუალებით.

დავალება: company მონაცემთა ბაზის abonenti ცხრილს დავუმატოთ ახალი სტრიქონი .

ამოხსნა:

```
{
SqlConnection myConnection = new SqlConnection
("server=Serveri_1; database=company; uid=sa; pwd=123");
            // შეერთების გახსნა
myConnection.Open();
            // mySqlCommand ობიექტის შექმნა
SqlCommand mySqlCommand =
myConnection.CreateCommand();
      // INSERT ბრძანების მომზადება
შესასრულებლად
mySqlCommand.CommandText = "INSERT INTO
abonenti (abID, ab_nomeri, gvari, misamarTi, telefoni,
davalianeba, raioni) " +
     "VALUES (@abID, @ab_nomeri, @gvari, @misamarTi,
@telefoni, @davalianeba, @raioni)";
            // პარამეტრების დამატება
```

```
    mySqlCommand.Parameters.Add("@abID",
SqlDbType.Int, 4);
    mySqlCommand.Parameters.Add("@ab_nomeri",
SqlDbType.NVarChar, 8);
    mySqlCommand.Parameters.Add("@gvari",
SqlDbType.NVarChar, 30);
    mySqlCommand.Parameters.Add("@misamarTi",
SqlDbType.NVarChar, 50);
    mySqlCommand.Parameters.Add("@telefoni",
SqlDbType.NVarChar, 10);
    mySqlCommand.Parameters.Add("@davalianeba",
SqlDbType.Float, 8);
    mySqlCommand.Parameters.Add("@raioni",
SqlDbType.NVarChar, 20);
                // პარამეტრებისთვის მნიშვნელობების
მინიჭება  mySqlCommand.Parameters["@gvari"].Value =
textBox1.Text;
mySqlCommand.Parameters["@piradi_nomeri"].Value =
textBox2.Text;
    mySqlCommand.Parameters["@abID"].Value =
Convert.ToInt32(textBox1.Text);
    mySqlCommand.Parameters["@ab_nomeri"].Value =
textBox2.Text;
    mySqlCommand.Parameters["@gvari"].Value =
textBox3.Text;
    mySqlCommand.Parameters["@misamarTi"].Value =
textBox4.Text;
    mySqlCommand.Parameters["@telefoni"].Value
=textBox5.Text;
```

```
    mySqlCommand.Parameters["@davalianeba"].Value =
Convert.ToDouble(textBox6.Text);
    mySqlCommand.Parameters["@raioni"].Value =
textBox7.Text;
                //  INSERT ბრძანების შესრულება
    int rowsAffected = mySqlCommand.ExecuteNonQuery();
    label8.Text = rowsAffected.ToString();
                //  შეერთების დახურვა
                myConnection.Close();
            }
```

თავი 4. მონაცემების შეცვლა ცხრილებში

დავალება: ცხრილიდან წაშალეთ რაიმე მონაცემი. საძებნი მნიშვნელობა შეიყვანეთ textBox კომპონენტიდან. (მთავარი ცხრილიდან წაშლით ავტომატურად წაიშალოს დამოკიდებულ ცხრილში)

ამოხსნა:

```
{
Label1.Text = "";
// find_key ცვლადში უნდა მოვათავსოთ საძებნი
მნიშვნელობა
int find_key = Convert.ToInt32(textBox1.Text);
// შეერთების ობიექტის შექმნა
SqlConnection myConnection =
new SqlConnection("server= Serveri_1;
database=company; uid=sa; pwd=123");
// შეერთების გახსნა
myConnection.Open();
// ადაპტერის შექმნა
SqlDataAdapter myAdapter = new SqlDataAdapter(
"SELECT  abID, ab_nomeri, gvari, misamarTi,telefoni,
davalianeba, raioni FROM abonenti", myConnection);
// SqlCommandBuilder ობიექტის შექმნა SQL
ბრძანების ასაგებად
SqlCommandBuilder myBuilder = new
SqlCommandBuilder(myAdapter);
// DataSet ობიექტის შექმნა ცხრილების,
სტრიქონებისა და სვეტების შესანახად
DataSet myDataset = new DataSet();
```

```
    //  myDataset ობიექტის შევსება abonenti ცხრილის
სტრიქონებით
    myAdapter.Fill(myDataset, "abonenti");
    //  პირველადი გასაღების ფორმირება
    DataColumn[] keys = new DataColumn[1];
    keys[0] = myDataset.Tables["abonenti"].Columns["abID"];
    myDataset.Tables["abonenti"].PrimaryKey = keys;
    //  findRow ობიექტს ენიჭება ნაპოვნი სტრიქონი
    DataRow findRow =
myDataset.Tables["abonenti"].Rows.Find(find_key);
    //  მოწმდება სტრიქონის არსებობა
    if ( findRow != null )
    {
    //  სტრიქონის წაშლა
    findRow.Delete();
    myAdapter.Update(myDataset, "abonenti");
    label1.Text = "სტრიქონი წაიშალა";
    }
    else label9.Text = "წასაშლელი სტრიქონი ვერ
მოიძებნა";
    //  შეერთების დახურვა
    myConnection.Close();
    }
```

თავი 5. ცხრილების დახარისხება და გაფილტვრა

დავალება: abonenti ცხრილის სტრიქონების დახარისხება gvari სვეტის ზრდის და davalianeba სვეტის მნიშვნელობების კლების მიხედვით:

ამოხსნა:

```
{
SqlConnection myConnection =
new SqlConnection ("server=Serveri_1;
database=company; uid=sa; pwd=123");
// შეერთების გახსნა
myConnection.Open();
// DataAdapter ობიექტის შექმნა
SqlDataAdapter myAdapter = new
SqlDataAdapter("SELECT * FROM abonenti",
myConnection);
// DataSet ობიექტის შექმნა ცხრილების,
სტრიქონებისა და სვეტების შესანახად
DataSet myDataset = new DataSet();
// myDataset ობიექტის შევსება abonenti ცხრილის
სტრიქონებით
myAdapter.Fill(myDataset, "abonenti");
// შეერთების დახურვა
myConnection.Close();
// dataView1 ობიექტის დაკავშირება abonenti
ცხრილთან
DataView dataView1 = new
DataView(myDataset.Tables["abonenti"]);
```

```
// სტრიქონების დახარისხება
dataView1.Sort = "gvari, davalianeba DESC";
// დახარისხებული სტრიქონების ასახვა ეკრანზე
dataGridView1.DataSource = dataView1;
}
```

დავალება: ცხრილის ფილტრაცია იმ აბონენტების მიხედვით რომელთა დავალიანება აღემატება 100 ლარს.

ამოხსნა:

```
{
SqlConnection myConnection =
new SqlConnection ("server=Serveri_1;
database=company; uid=sa; pwd=123");
// შეერთების გახსნა
myConnection.Open();
// DataAdapter ობიექტის შექმნა
SqlDataAdapter myAdapter = new
SqlDataAdapter("SELECT * FROM abonenti",
myConnection);
// DataSet ობიექტის შექმნა ცხრილების,
სტრიქონებისა და სვეტების შესანახად
DataSet myDataset = new DataSet();
// myDataset ობიექტის შევსება abonenti ცხრილის
სტრიქონებით
myAdapter.Fill(myDataset, "abonenti");
// შეერთების დახურვა
myConnection.Close();
```

```
    // dataView1 ობიექტის დაკავშირება abonenti
ცხრილთან
    DataView dataView1 = new
DataView(myDataset.Tables["abonenti"]);
    // სტრიქონების გაფილტვრა
    dataView1.RowFilter ="davalianeba>100";
    // გაფილტრული სტრიქონების ასახვა ეკრანზე
    dataGridView1.DataSource = dataView1;
    }
```

თავი 6. ფუნქციები

დავალება: შექმენით ფუნქცია, რომელიც ა) დავალიანების ჯამურ თანხას

```
use company
go
CREATE FUNCTION Vali()
returns float
as
begin
declare @davalianeba int
set @davalianeba=
(
select sum(davalianeba) from abonenti
)
return @davalianeba
end

use company
SELECT dbo.Vali() as [სულ დავალიანება]
```

ეს ფუნქცია შევასრულოთ .NET კლასების გამოყენებით:

```
private void button4_Click(object sender, EventArgs e)
      {
 SqlConnection mySqlConnection = new SqlConnection
("server=Serveri_1; database=company; uid=sa; pwd=123");
```

```
    SqlCommand mySqlCommand =
mySqlConnection.CreateCommand();
    mySqlCommand.CommandText = "SELECT
sum(davalianeba) FROM abonenti";
    SqlDataAdapter mySqlDataAdapter = new
SqlDataAdapter();
    mySqlDataAdapter.SelectCommand = mySqlCommand;
    DataSet myDataSet1 = new DataSet();
    //
    mySqlConnection.Open();
    myDataSet1.Clear();
    mySqlDataAdapter.Fill(myDataSet1, "abonenti");
    dataGridView1.DataSource = myDataSet1;
    dataGridView1.DataMember = "abonenti";
    mySqlConnection.Close();
    }
```

თავი 7. შენახული პროცედურების შედგენა

დავალება: შექმენით შენახული პროცედურა ა) რომელიც გასცემს დავალიანების მქონე აბონენტების სიას მითითებული რაიონის მიხედვით

ა)

```
use company
go
if OBJECT_ID('[dbo].[davalianebis_mqone]','P') is not
null
drop procedure[dbo].[davalianebis_mqone];
go
create procedure davalianebis_mqone @raioni nvarchar
(20)
as
select * from abonenti where raioni=@raioni and
davalianeba>0
go
exec [dbo].[davalianebis_mqone ]N'ვაკე'
```

ეს პროცედურა შევასრულოთ .NET კლასების გამოყენებით:

```
{
SqlConnection mySqlConnection = new
SqlConnection("server=Serveri_1; database=company;
uid=sa; pwd=123");
        string procedureString =
"dbo.davalianebis_mqone";
```

```
            SqlCommand mySqlCommand =
mySqlConnection.CreateCommand();
            mySqlCommand.Parameters.Add("@raioni",
SqlDbType.NVarChar, 30);
            mySqlCommand.Parameters["@raioni"].Value =
textBox7.Text;
            mySqlCommand.CommandText =
procedureString;
            mySqlCommand.CommandType =
CommandType.StoredProcedure;
            //
            mySqlConnection.Open();
            mySqlCommand.ExecuteNonQuery();
            mySqlConnection.Close();
            SqlDataAdapter mySqlDataAdapter = new
SqlDataAdapter();
            mySqlCommand.CommandText =
"davalianebis_mqone";
            mySqlDataAdapter.SelectCommand =
mySqlCommand;
            DataSet myDataSet = new DataSet();

davalianebis_mqoneTableAdapter.Fill(this.companyDataSet.
davalianebis_mqone, textBox7.Text);
            dataGridView1.DataSource =
this.companyDataSet.davalianebis_mqone;
    }
```

თავი 8. მონაცემთა ბაზის სარეზერვო ასლის შექმნა.

სარეზერვო ასლი წარმოადგენს ერთ ან მეტ ფაილს, რომელშიც მონაცემთა ბაზაა მოთავსებული მთლიანად ან ნაწილობრივ. სარეზერვო ასლის შექმნა საჭიროა მონაცემების აღსადგენად კომპიუტერის, დისკის ან მონაცემების დაზიანების შემთხვევაში და ა.შ.

სარეზერვო ასლის შექმნისათვის სერვერზე შექმნილი ლოგიკური მოწყობილობების სია ინახება master მონაცემთა ბაზის sysdevices სისტემურ წარმოდგენაში. ახალი მოწყობილობის დასამატებლად გამოიყენება sp_addumpdevice სისტემური შენახული პროცედურა. მისი სინტაქსია:

sp_addumpdevice [@devtype =] 'მოწყობილობის_ტიპი' ,

[@logicalname =] 'ლოგიკური_სახელი' , [@physicalname =] 'ფიზიკური_სახელი'

სადაც,

- 'მოწყობილობის_ტიპი'. მოწყობილობის ტიპია და იღებს შემდეგ მნიშვნელობებს: TAPE (ლენტი), DISK (დისკი) და PIPE (სახელდებული არხი);
- 'ლოგიკური_სახელი' მოწყობილობის ლოგიკური სახელია;
- 'ფიზიკური_სახელი' მოწყობილობის ფიზიკური სახელია.

დავალება: შევქმნათ company მონაცემთა ბაზის სრული ასლი.

ამოხსნა: ვიყენებთ შენახულ პროცედურას

```
USE company
-- თუ შენახული პროცედურა არსებობს, მაშინ ის წაიშლება
IF OBJECT_ID ( 'sarezervo_asli', 'P' ) IS NOT NULL
DROP PROCEDURE sarezervo_asli;
GO
-- შენახული პროცედურის შექმნა
CREATE PROCEDURE sarezervo_asli
AS
EXEC sp_addumpdevice 'DISK', ' company_asli',
'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup\ company_asli.bak'

BACKUP DATABASE company TO company _asli WITH NAME = ' company _asli'
GO
```

მონაცემთა ბაზის სარეზერვო ასლის შექმნა შევასრულოთ .NET კლასების გამოყენებით:

```
{
SqlConnection mySqlConnection = new SqlConnection("server=Serveri_1; database=company; uid=sa; pwd=123");
        string procedureString = "dbo.sarezervo_asli";
```

```
            SqlCommand mySqlCommand =
mySqlConnection.CreateCommand();
            mySqlCommand.CommandText =
procedureString;
            mySqlCommand.CommandType =
CommandType.StoredProcedure;
            //
            mySqlConnection.Open();
            mySqlCommand.ExecuteNonQuery();
            mySqlConnection.Close();
            SqlDataAdapter mySqlDataAdapter = new
SqlDataAdapter();
            mySqlCommand.CommandText = "sarezervo_asli";
            mySqlDataAdapter.SelectCommand =
mySqlCommand;
            DataSet myDataSet = new DataSet();
            dataGridView1.DataSource =
this.companyDataSet.sarezervo_asli;
    }
```

ამოცანები

ლიტერატურა

1. SQL სერვერი. რომან სამხარაძე. საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი‖ , თბილისი.2009. ISBN 978-9941-14-190-4 . http://www.gtu.ge/publishinghouse

2. V i s u a l C#.N E T. რომან სამხარაძე. საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი“, თბილისი.2009. ISBN978-9941-14-593. http://www.gtu.ge/publishinghouse

რედაქტორი
ტექნიკური რედაქტორი
კორექტორი
კომპიუტერული უზრუნველყოფა ნ. ჯოჯუასი

წარმოებას გადაეცა . ხელმოწერილია დასაბეჭდად . ქაღალდის ზომა პირობითი ნაბეჭდი თაბახი . სააღრიცხვო-საგამომცემლო თაბახი . ტირაჟი 20 ეგზ.

გამომცემლობა "ტექნიკური უნივერსიტეტი", თბილისი კოსტავას 77