

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КПІ”**

**Кафедра**

**Автоматизованих систем обробки інформації та управління**

**Звіт**

**до лабораторної роботи з дисципліни**

**“** **Технології розроблення програмного забезпечення -1. Основи розробки програмного забезпечення на платформі Microsoft.NET”**

**Комп’ютерний практикум № 6.**

**Варіант - 6**

**Перевірли:**

**доц. Ліщук К. І.**

**Виконала:**

**студентка групи ІС-83**

**Прийдун М.В.**

**м. Київ-2020**

Програмне забезпечення «Діяльність фірми з розробки програмних продуктів». Підприємства, о розробляють ПЗ, зазвичай мають декілька відділів, а саме: дирекція, бухгалтерія, маркетинговий відділ, відділ розробки ПЗ, відділ тестування ПЗ, відділ супроводження тощо. ПЗ, котре поставляється Замовнику, має назву, список розробників (внутрішній список тестувальників, котрий Замовнику не надається), вартість, документацію, дистрибутив, правила використання. Замовниками можуть бути як фізичні так і юридичні особи. Кожний Замовник має можливість замовити декілька ПП, на кожний з яких він отримує ліцензію, в якій вказано назву продукту, дату продажу, вартість, терміни апгрейдів.

CREATE TABLE Departments

(

Id INT IDENTITY,

Name NVARCHAR(100),

CONSTRAINT PK\_ Departments\_Id PRIMARY KEY (Id),

)

CREATE TABLE Employees

(

Id INT IDENTITY,

DepartmentId INT,

Name NVARCHAR(100),

position NVARCHAR(100),

CONSTRAINT PK\_Employees\_Id PRIMARY KEY (Id),

CONSTRAINT FK\_Department\_Id FOREIGN KEY (DepartmentId) REFERENCES Departments (Id)

)

CREATE TABLE Softwares

(

Id INT IDENTITY,

Name NVARCHAR(100),

Price FLOAT,

Documentation NVARCHAR(100),

Distribution NVARCHAR(100),

Rules NVARCHAR(100),

CONSTRAINT PK\_Softwares\_Id PRIMARY KEY (Id),

)

CREATE TABLE Participants

(

Id INT IDENTITY,

EmployeeId INT,

SoftwareId INT,

CONSTRAINT PK\_Participant\_Id PRIMARY KEY (Id),

CONSTRAINT FK\_Software1\_Id FOREIGN KEY (SoftwareId) REFERENCES Softwares (Id),

CONSTRAINT FK\_Employee\_Id FOREIGN KEY (EmployeeId) REFERENCES Employees (Id),

)

CREATE TABLE Customers

(

Id INT IDENTITY,

Name NVARCHAR(100),

Type NVARCHAR(100),

CONSTRAINT PK\_Customers\_Id PRIMARY KEY (Id),

)

CREATE TABLE Licenses

(

Id INT IDENTITY,

SoftwareId INT,

CustomerId INT,

ProductName NVARCHAR(100),

DateSale DATETIME,

Price FLOAT,

DateUpgrade DATETIME,

CONSTRAINT PK\_Licenses\_Id PRIMARY KEY (Id),

CONSTRAINT FK\_Software\_Id FOREIGN KEY (SoftwareId) REFERENCES Softwares (Id),

CONSTRAINT FK\_Customer\_Id FOREIGN KEY (CustomerId) REFERENCES Customers (Id),

)

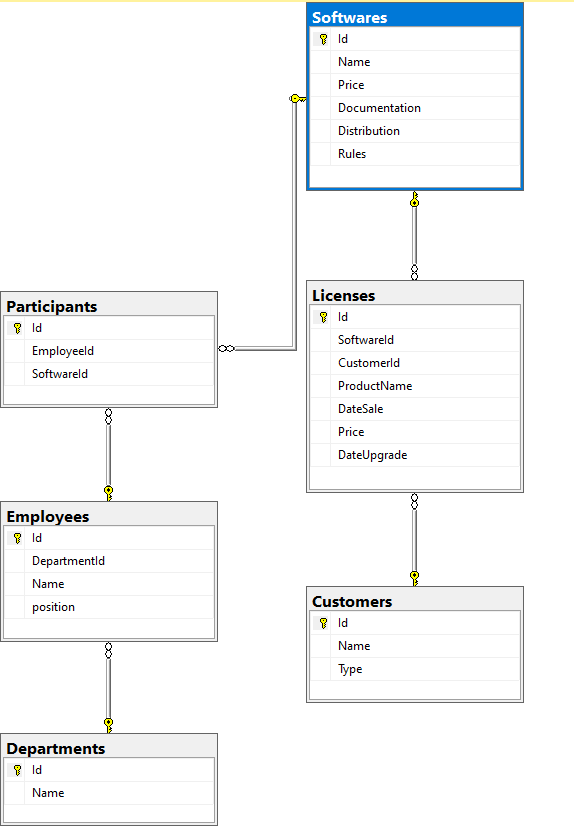


Рис.1 СхемаБД

INSERT INTO Customers (Name,Type)

VALUES

('MSK','FOP'),

('John Martinez','individual'),

('John Mayer ','individual'),

('Rhine Horwood','FOP'),

('William Smith','individual'),

('Jacob Miller',' individual ')

INSERT INTO Departments (Name)

VALUES

('Software development department'),

('Software testing department')

INSERT INTO Employees (DepartmentId, Name, position)

VALUES

(1, 'Illyenko Makar', 'Project Manager' ),

(1, 'Makorov Volodymyr', 'Architect' ),

(1, 'Korylov Mykhayil', 'Developer' ),

(1, 'Zabudko Kateryna', 'Developer' ),

(1, 'Mykhaylochyk Illya', 'Developer' ),

(1, 'Adamenko Mariya', 'UI/UX designer' ),

(2, 'Kalynovskyy Andriy', 'QA Engineer' )

INSERT INTO Softwares (Name, EmployeeId, Price, Documentation, Distribution, Rules)

VALUES

('Mobile application to support patients',1,2000,' visible', 'EP83956',' visible' ),

('Mobile application to support patients',2,2000,' visible', 'EP83956',' visible' ),

('Mobile application to support patients',3,2000,' visible', 'EP83956',' visible' ),

('Mobile application to support patients',4,2000,' visible', 'EP83956',' visible'),

('Mobile application to support patients',6,2000,' visible', 'EP83956',' visible' ),

('Mobile application to support patients',7,2000,' visible', 'EP83956',' visible'),

('Fluid',1,2500,' visible', 'GL85253',' visible' ),

('Fluid',2,2500,' visible', 'GL85253',' visible' ),

('Fluid',4,2500,' visible', 'GL85253',' visible' ),

('Fluid',5,2500,' visible', 'GL85253',' visible' ),

('Fluid',7,2500,' visible', 'GL85253',' visible' ),

('Brawhalla',1, 30000,' visible', 'US78535',' visible'),

('Brawhalla',2, 30000,' visible', 'US78535',' visible' ),

('Brawhalla',3, 30000,' visible', 'US78535',' visible' ),

('Brawhalla',4, 30000,' visible', 'US78535',' visible' ),

('Brawhalla',5, 30000,' visible', 'US78535',' visible' ),

('Brawhalla',6, 30000,' visible', 'US78535',' visible' ),

('Brawhalla',7, 30000,' visible', 'US78535',' visible' ),

('Skillorton',1, 5000,' visible', 'DA18275',' visible' ),

('Skillorton',3, 5000,' visible', 'DA18275',' visible' ),

('Skillorton',4, 5000,' visible', 'DA18275',' visible' ),

('Skillorton',7, 5000,' visible', 'DA18275',' visible' ),

('Gelato',1, 800,' visible', 'IN45026',' visible' ),

('Gelato',2, 800,' visible', 'IN45026',' visible' ),

('Gelato',3, 800,' visible', 'IN45026',' visible' ),

('Gelato',4, 800,' visible', 'IN45026',' visible' ),

('Gelato',7, 800,' visible', 'IN45026',' visible' ),

('Robo Advisor',1, 3000,' visible', 'SS84426',' visible' ),

('Robo Advisor',4, 3000,' visible', 'SS84426',' visible' ),

('Robo Advisor',5, 3000,' visible', 'SS84426',' visible' ),

('Robo Advisor',6, 3000,' visible', 'SS84426',' visible' ),

INSERT INTO Softwares (Name, Price, Documentation, Distribution, Rules)

VALUES

('Mobile application to support patients',2000,' visible', 'EP83956',' visible' ),

('Fluid',2500,' visible', 'GL85253',' visible' ),

('Brawhalla', 30000,' visible', 'US78535',' visible'),

('Skillorton',5000,' visible', 'DA18275',' visible' ),

('Gelato', 800,' visible', 'IN45026',' visible' ),

('Robo Advisor', 3000,' visible', 'SS84426',' visible' )

(1, 'Makorov Volodymyr', 'Architect' ),

(1, 'Korylov Mykhayil', 'Developer' ),

(1, 'Zabudko Kateryna', 'Developer' ),

(1, 'Mykhaylochyk Illya', 'Developer' ),

(1, 'Adamenko Mariya', 'UI/UX designer' ),

(2, 'Kalynovskyy Andriy', 'QA Engineer' )

INSERT INTO Participants (EmployeeId, SoftwareId)

VALUES

(1,1),

(2,1),

(3,1),

(4,1),

(6,1),

(7,1),

(1,2),

(2,2),

(4,2),

(5,2),

(7,2),

(1,3),

(2,3),

(3,3),

(4,3),

(5,3),

(6,3),

(7,3),

(1,4),

(3,4),

(4,4),

(7,4),

(1,5),

(2,5),

(3,5),

(4,5),

(7,5),

(1,6),

(4,6),

(5,6),

(6,6)

INSERT INTO Licenses (SoftwareId, CustomerId,ProductName,DateSale,Price,DateUpgrade)

VALUES

(1,1,'Mobile application to support patients','20190720',2000,'20190810'),

(2,2,'Fluid','20190910',2500,'20190920'),

(3,3,'Brawhalla','20190421',2500,'20190505'),

(4,4,'Skillorton','20190320',2500,'20190330'),

(5,5,'Gelato','20190818',2500,'20190901'),

(6,1,'Robo Advisor','20191218',2500,'20200110')

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

using System.Security.Cryptography.X509Certificates;

using System.Runtime.Remoting.Services;

/\*6) Програмне забезпечення «Діяльність фірми з розробки програмних продуктів».

\* Підприємства, о розробляють ПЗ, зазвичай мають декілька відділів, а саме:

\* дирекція, бухгалтерія, маркетинговий відділ, відділ розробки ПЗ, відділ тестування ПЗ,

\* відділ супроводження тощо. ПЗ, котре поставляється Замовнику, має назву, список розробників , вартість, документацію,

\* дистрибутив, правила використання. Замовниками можуть бути як фізичні так і юридичні особи.

\* Кожний Замовник має можливість замовити декілька ПП, на кожний з яких він отримує ліцензію,

\* в якій вказано назву продукту, дату продажу, вартість, терміни апгрейдів.

\*/

namespace lab6

{

class Program

{

static void Main(string[] args)

{

//string connectionString = @"Data Source=.\SQLEXPRESS;Initial Catalog=lab6;Integrated Security=True";

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=lab6;Integrated Security=True";

string sqlExpression;

while (true)

{

Console.Clear();

Console.WriteLine("Menu");

Console.WriteLine("1. Display the table all tables(SqlDataReader,SqlDataAdapter.DataSet).");

Console.WriteLine("2. Add element to the Departments table.");

Console.WriteLine("3. Change the position of QA Engineer to QA/QC Engineer (Update Employee table).");

Console.WriteLine("4. Delete adding item from the Departments table.");

Console.WriteLine("5. Count the number of projects and display the minimum project price.");

Console.WriteLine("6. Display the project with the maximum cost.");

Console.WriteLine("7. Display project participants.");

Console.WriteLine("8. Display the license on product for some customer.");

Console.WriteLine("9. Display the numbers of each departments.");

Console.WriteLine("10. Display all developers sorted by name.");

string ch = Console.ReadLine();

switch (ch)

{

case "1":

Console.WriteLine("--------------Customers-----------------");

sqlExpression = "SELECT \* FROM Customers";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0}\t{1}\t\t{2}", reader.GetName(0), reader.GetName(1), reader.GetName(2));

while (reader.Read()) // построчно считываем данные

{

object id = reader.GetValue(0);

object name = reader.GetValue(1);

object type = reader.GetValue(2);

Console.WriteLine("{0} \t{1} \t{2}", id, name, type);

}

}

reader.Close();

Console.WriteLine("\n--------------Departments-----------------");

sqlExpression = "SELECT \* FROM Departments";

command = new SqlCommand(sqlExpression, connection);

reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0}\t\t{1}\t", reader.GetName(0), reader.GetName(1));

while (reader.Read()) // построчно считываем данные

{

object id = reader.GetValue(0);

object name = reader.GetValue(1);

Console.WriteLine("{0} \t\t{1}", id, name);

}

}

reader.Close();

Console.WriteLine("\n--------------Employees-----------------");

sqlExpression = "SELECT \* FROM Employees";

// connection.Open();

command = new SqlCommand(sqlExpression, connection);

reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0}\t{1}\t{2}\t\t{3}", reader.GetName(0), reader.GetName(1), reader.GetName(2), reader.GetName(3));

int a = 0;

while (reader.Read()) // построчно считываем данные

{

object id = reader.GetValue(0);

object DepartmentId = reader.GetValue(1);

object name = reader.GetValue(2);

object position = reader.GetValue(3);

if(a==0 ||a==5)

Console.WriteLine("{0}\t{1}\t{2}\t\t{3}", id, DepartmentId, name, position);

else

Console.WriteLine("{0}\t{1}\t{2}\t{3}", id, DepartmentId, name, position);

a++;

}

}

reader.Close();

Console.WriteLine("\n--------------Softwares-----------------");

sqlExpression = "SELECT \* FROM Softwares";

// Создаем объект DataAdapter

SqlDataAdapter adapter = new SqlDataAdapter(sqlExpression, connection);

// Создаем объект Dataset

DataSet ds = new DataSet();

// Заполняем Dataset

adapter.Fill(ds);

// Отображаем данные

//Console.WriteLine(DataSource);

int i = 0;

foreach (DataRow pRow in ds.Tables[0].Rows)

{

if(i==0)

Console.WriteLine("{0} {1}\t{2}\t{3}\t{4}\t{5}", pRow["Id"], pRow["Name"], pRow["Price"], pRow["Documentation"], pRow["Distribution"],pRow["Rules"]);

else

Console.WriteLine("{0} {1}\t\t\t\t\t{2}\t{3}\t{4}\t{5}", pRow["Id"], pRow["Name"], pRow["Price"], pRow["Documentation"], pRow["Distribution"], pRow["Rules"]);

i++;

}

Console.WriteLine("\n--------------Participants-----------------");

sqlExpression = "SELECT \* FROM Participants";

// using (SqlConnection connection = new SqlConnection(connectionString))

// {

//connection.Open();

// Создаем объект DataAdapter

adapter = new SqlDataAdapter(sqlExpression, connection);

// Создаем объект Dataset

ds = new DataSet();

// Заполняем Dataset

adapter.Fill(ds);

// Отображаем данные

//Console.WriteLine(DataSource);

foreach (DataRow pRow in ds.Tables[0].Rows)

{

Console.WriteLine("{0}\t{1}\t{2}",pRow["Id"], pRow["EmployeeId"], pRow["SoftwareId"]);

}

Console.WriteLine("\n--------------Licenses-----------------");

sqlExpression = "SELECT \* FROM Licenses";

// Создаем объект DataAdapter

adapter = new SqlDataAdapter(sqlExpression, connection);

// Создаем объект Dataset

ds = new DataSet();

// Заполняем Dataset

adapter.Fill(ds);

// Отображаем данные

//Console.WriteLine(DataSource);

int k = 0;

foreach (DataRow pRow in ds.Tables[0].Rows)

{

if (k == 0)

Console.WriteLine("{0} {1}\t{2}\t{3}\t{4}\t{5}\t{6}", pRow["Id"], pRow["SoftwareId"], pRow["CustomerId"], pRow["ProductName"], Convert.ToDateTime(pRow["DateSale"]).ToShortDateString(), pRow["Price"], Convert.ToDateTime(pRow["DateUpgrade"]).ToShortDateString());

else if(k==1||k==4)

Console.WriteLine("{0} {1}\t{2}\t{3}\t\t\t\t\t{4}\t{5}\t{6}", pRow["Id"], pRow["SoftwareId"], pRow["CustomerId"], pRow["ProductName"], Convert.ToDateTime(pRow["DateSale"]).ToShortDateString(), pRow["Price"], Convert.ToDateTime(pRow["DateUpgrade"]).ToShortDateString());

else

Console.WriteLine("{0} {1}\t{2}\t{3}\t\t\t\t{4}\t{5}\t{6}", pRow["Id"], pRow["SoftwareId"], pRow["CustomerId"], pRow["ProductName"], Convert.ToDateTime(pRow["DateSale"]).ToShortDateString(), pRow["Price"], Convert.ToDateTime(pRow["DateUpgrade"]).ToShortDateString());

k++;

}

connection.Close();

}

Console.Read();

Console.Clear();

break;

case "2":

Console.WriteLine("Enter the name of department:");

string cname = Console.ReadLine();

string start = "INSERT INTO Departments (Name) VALUES ('";

string end = "')";

sqlExpression = start + cname + end;

Console.WriteLine("Choose the metod of adding");

Console.WriteLine("1.SqlCommand");

Console.WriteLine("2.SqlCommandBuilder,DataSet");

string ch2 = Console.ReadLine();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

if (ch2 == "1")

{

SqlCommand command = new SqlCommand(sqlExpression, connection);

int number = command.ExecuteNonQuery();

Console.WriteLine("Added object: {0}", number);

}

else if (ch2 =="2")

{

sqlExpression= "SELECT \* FROM Departments";

SqlDataAdapter adapter = new SqlDataAdapter(sqlExpression, connection);

DataSet ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

// добавим новую строку

DataRow newRow = dt.NewRow();

newRow["Name"] = cname;

dt.Rows.Add(newRow);

// создаем объект SqlCommandBuilder

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(ds);

// альтернативный способ - обновление только одной таблицы

//adapter.Update(dt);

// заново получаем данные из бд

// очищаем полностью DataSet

ds.Clear();

// перезагружаем данные

adapter.Fill(ds);

foreach (DataColumn column in dt.Columns)

Console.Write("\t{0}", column.ColumnName);

Console.WriteLine();

// перебор всех строк таблицы

foreach (DataRow row in dt.Rows)

{

// получаем все ячейки строки

var cells = row.ItemArray;

foreach (object cell in cells)

Console.Write("\t{0}", cell);

Console.WriteLine();

}

}

connection.Close();

}

Console.Read();

Console.Clear();

break;

case "3":

sqlExpression = "UPDATE Employees SET position='QA/QC Engineer' WHERE position=' QA Engineer' ";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

int number = command.ExecuteNonQuery();

Console.WriteLine("Updated object: {0}", number);

connection.Close();

}

Console.Read();

Console.Clear();

break;

case "4":

sqlExpression = "DELETE FROM Departments WHERE Id>1000";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

int number = command.ExecuteNonQuery();

Console.WriteLine("Deleted object: {0}", number);

connection.Close();

}

Console.Read();

Console.Clear();

break;

case "5":

sqlExpression = "SELECT COUNT(\*) FROM Softwares";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

object count = command.ExecuteScalar();

command.CommandText = "SELECT MIN(Price) FROM Softwares";

object minAge = command.ExecuteScalar();

Console.WriteLine("There are {0} projects", count);

Console.WriteLine("Min price: {0}", minAge);

connection.Close();

}

Console.Read();

Console.Clear();

break;

case "6":

sqlExpression = "SELECT \* FROM Softwares WHERE price = (SELECT MAX(Price) FROM Softwares)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0} {1}\t{2}\t{3}\t{4}\t{5}", reader.GetName(0), reader.GetName(1), reader.GetName(2), reader.GetName(3), reader.GetName(4), reader.GetName(5));

while (reader.Read()) // построчно считываем данные

{

object id = reader.GetValue(0);

object name = reader.GetValue(1);

object price = reader.GetValue(2);

object documentation = reader.GetValue(3);

object distribution= reader.GetValue(4);

object rules = reader.GetValue(5);

Console.WriteLine("{0} {1}\t{2}\t{3}\t{4}\t\t{5}", id, name, price,documentation,distribution,rules);

}

}

/\*SqlDataAdapter adapter = new SqlDataAdapter(sqlExpression, connection);

// Создаем объект Dataset

DataSet ds = new DataSet();

// Заполняем Dataset

adapter.Fill(ds);

// Отображаем данные

//Console.WriteLine(DataSource);

int i = 0;

foreach (DataRow pRow in ds.Tables[0].Rows)

{

if (i == 0)

Console.WriteLine("{0} {1}\t{2}\t{3}\t{4}\t{5}", pRow["Id"], pRow["Name"], pRow["Price"], pRow["Documentation"], pRow["Distribution"], pRow["Rules"]);

else

Console.WriteLine("{0} {1}\t\t\t\t\t{2}\t{3}\t{4}\t{5}", pRow["Id"], pRow["Name"], pRow["Price"], pRow["Documentation"], pRow["Distribution"], pRow["Rules"]);

i++;

}\*/

reader.Close();

connection.Close();

}

Console.Read();

Console.Clear();

break;

case "7":

sqlExpression = "SELECT Id,Name FROM Softwares";

object Id = null;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0} {1}", reader.GetName(0), reader.GetName(1));

while (reader.Read()) // построчно считываем данные

{

Id= reader.GetValue(0);

object name = reader.GetValue(1);

Console.WriteLine("{0} {1}", Id, name);

}

}

reader.Close();

connection.Close();

}

Console.WriteLine("Enter id of project in which you want to see partisipants");

string n = Console.ReadLine();

if(Convert.ToInt32(n)<=Convert.ToInt32(Id))

{

sqlExpression = "SELECT Employees.Name, Employees.position" +

" FROM Participants JOIN " +

" Softwares ON Participants.SoftwareId = Softwares.Id JOIN " +

" Employees ON Participants.EmployeeId = Employees.Id" +

" WHERE Softwares.Id="+n;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0}\t\t{1}", reader.GetName(0), reader.GetName(1));

while (reader.Read()) // построчно считываем данные

{

object name = reader.GetValue(0);

object position = reader.GetValue(1);

Console.WriteLine("{0}\t\t{1}", name, position);

}

}

reader.Close();

connection.Close();

}

}

Console.Read();

Console.Clear();

break;

case "8":

sqlExpression = "SELECT Id, Name FROM Customers";

object ID = null;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0} {1}", reader.GetName(0), reader.GetName(1));

while (reader.Read()) // построчно считываем данные

{

ID = reader.GetValue(0);

object name = reader.GetValue(1);

Console.WriteLine("{0} {1}", ID, name);

}

}

reader.Close();

connection.Close();

}

Console.WriteLine("Enter id of customer in which you want to see liecence");

string C = Console.ReadLine();

if (Convert.ToInt32(C) <= Convert.ToInt32(ID))

{

sqlExpression = "SELECT ProductName, DateSale, Price, DateUpgrade FROM Licenses WHERE CustomerId=" + C;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0}\t{1}\t{2}\t{3}", reader.GetName(0), reader.GetName(1), reader.GetName(2), reader.GetName(3));

while (reader.Read()) // построчно считываем данные

{

object ProductName = reader.GetValue(0);

object DateSale = reader.GetValue(1);

object Price = reader.GetValue(2);

object DateUpgrade = reader.GetValue(3);

Console.WriteLine("{0}\t\t{1}\t{2}\t{3}", ProductName, Convert.ToDateTime(DateSale).ToShortDateString(), Price,Convert.ToDateTime( DateUpgrade).ToShortDateString());

}

}

reader.Close();

connection.Close();

}

}

Console.Read();

Console.Clear();

break;

case "9":

sqlExpression = "SELECT Departments.Name, COUNT(\*) AS Number FROM Employees JOIN Departments ON Employees.DepartmentId = Departments.Id GROUP BY Departments.Name";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0}\t\t\t\t\t{1}", reader.GetName(0), reader.GetName(1));

while (reader.Read()) // построчно считываем данные

{

object name = reader.GetValue(0);

object number = reader.GetValue(1);

Console.WriteLine("{0}\t\t{1}", name, number);

}

}

reader.Close();

connection.Close();

}

Console.Read();

Console.Clear();

break;

case "10":

sqlExpression = "SELECT \* FROM Employees WHERE position='Developer' ORDER BY Name";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть данные

{

// выводим названия столбцов

Console.WriteLine("{0}\t{1}\t{2}\t\t{3}", reader.GetName(0), reader.GetName(1), reader.GetName(2), reader.GetName(3));

while (reader.Read()) // построчно считываем данные

{

object id = reader.GetValue(0);

object DepartmentId = reader.GetValue(1);

object name = reader.GetValue(2);

object position = reader.GetValue(3);

Console.WriteLine("{0}\t{1}\t{2}\t{3}", id, DepartmentId, name, position);

}

}

reader.Close();

connection.Close();

}

Console.Read();

Console.Clear();

break;

//default:

// {

// Console.WriteLine("Error! Enter an existing menu item.");

// Console.ReadLine();

// break;

// }

}

}

}

}

}