Zadanie domowe

0 - dokończyć bazę z laboratorium o nowe tabele:

-tabela Seller - sprzedający

-tabela rozwiązująca problem wiele do wielu w przypadku sprzedających i kupujących, gdy kupujący wystawia opinię, nazwa tabeli SellerCustomer z polami: CustomerId, SellerId, Opinion

-tabela rozwiązująca problem wiele do wielu w przypadku sprzedających i przedmiotów, które wystawiają, SellerProduct, z polami: SellerId, ProductId,

Do którejś z tabel dodać kolumnę typu datetime (np data rejestracji sprzedającego/jego przedmiotu, wystawienia opini) + jakieś inne, im więcej tym lepiej.  Zaproponowano money, ale mogę być inne.

1 - Stworzyć bazę danych zawierającą 10 tabel, posiadającą również rozwiązania relacji wiele do wielu.

Przykłady: wielu studentów ma wiele prowadzących,  wielu studentów ma wiele przedmiotów, wielu pracowników uczestniczy w wielu projektach, wiele uprawnień do wielu zasobów,  autorzy wielu książek i książki z wieloma autorami, wielu aktorów i wiele filmów itp.

Rozwiązanie problemu wiele do wielu to utworzenie tabeli pośredniej zawierającą Id jednej tabeli i Id drugiej tabeli.

Zapis na laboratorium 2xjoin oznacza prośbę o wykonanie dwóch przykładowych zapytań join i zapisanie ich w pliku tekstowym, aby sprawdzając zadanie domowe sprawdzający skopiował zapytanie i również mógł zobaczyć rezultaty.

Część programistyczna dotyczy wykonania aplikacji MVC. Może być rozszerzeniem z laboratorium lub nowej.

Aplikacja ma zawierać:

-Przycisk Display do wyświetlenia zawartości całej tabeli

-Przycisk filtrujący + textbox na podstawie którego filtrowana jest zawartość tabeli

Filtrowanie to po prostu wykonanie zapytania z dodatkiem Where. Może to być operacja '=', '<','>', lub Like.

Np. Select * from customer where username Like '%a%'

Czyli w aplikacji będzie to sklejenie komendy:

string sqlCommand = "Select * from customer where username Like '%"+textBoxFilter.Text+"%'";

2* (na 5,5!)

Wykonanie Insert/Update/Delete

W controlerze:

```
public static void ExecuteSql(string command, SqlConnection sqlConnection)
{
   SqlCommand sqlCommand = new SqlCommand(command);
   sqlCommand.Connection = sqlConnection;
   sqlConnection.Open();
   sqlCommand.ExecuteNonQuery();
   sqlConnection.Close();
}
```

W widoku:

```
string insertCommand = "Insert into Persons (Name, Surname) values ('" + textBoxName.Text + "','" + textBoxSurname.Text + "')";
ModelWorker.ExecuteSql(insertCommand, sqlConnection);
labelIfAddedWorker.Visible = true;
labelIfAddedWorker.Text += textBoxName.Text;
```

Zasada działania:

- w stringu należy zdefiniować zapytanie SQL

- utworzenie obiektu SqlCommand

-zdefiniowanie połączenia dla obiektu SqlCommand poprzez sqlConnection założone jak podczas laboratorium

-nawiązanie połączenia poleceniem Open()

-wykonanie zapytania poleceniem ExecuteNonQuery()

-zakończenie połączenia