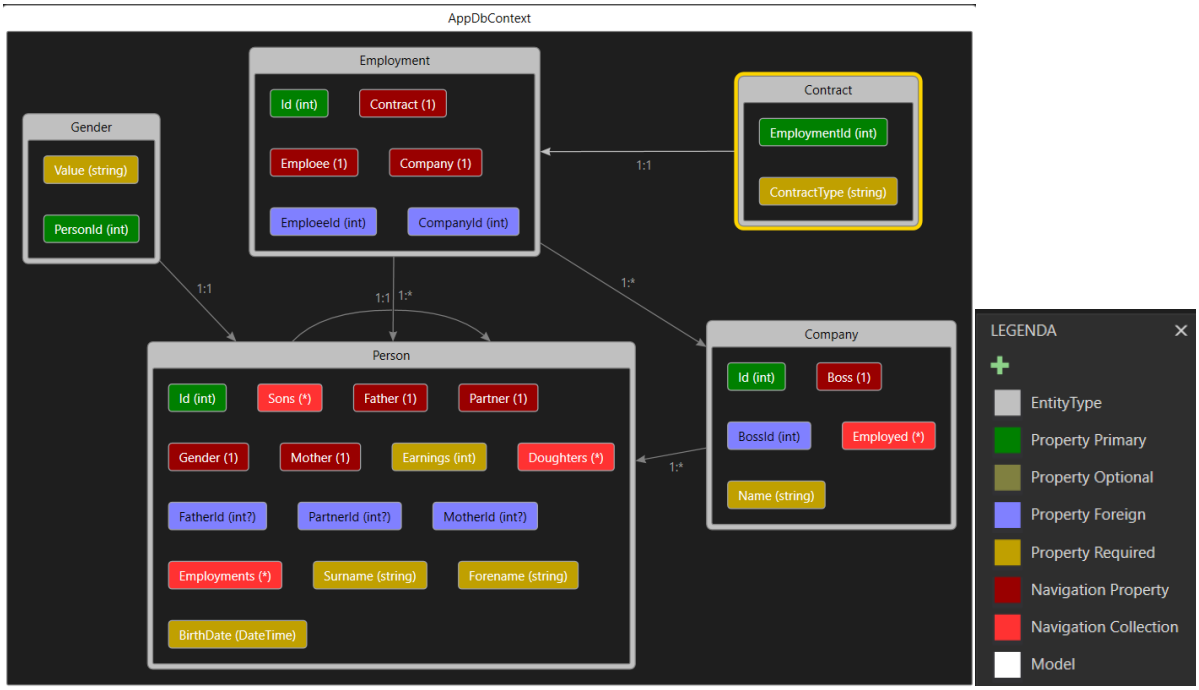
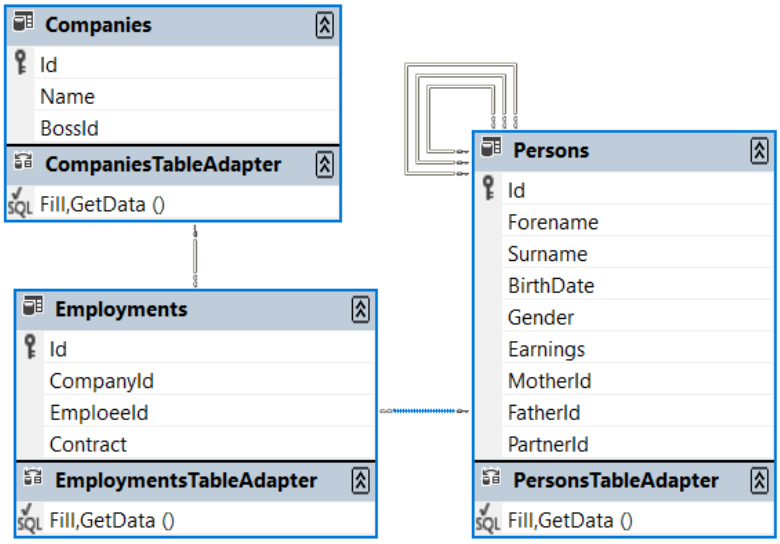
SQL/ERD – zadanie 2 – odpowiedź

Kod aplikacji dostępny jest tutaj: https://github.com/goluch/PolesDB

Wykorzystałem podejście code first i zamiast projektowania bazy danych zaprojektowałem następujący model danych:



Można wyodrębnić 3 encje: Person, Company oraz Employment i 2 value object: Gender oraz Contract. Wykorzystując narzędzia Entity Framework wygenerowałem następującą bazę danych (w tym przypadku MicrosoftSQL):



Model został tak skonfigurowany, aby objects values były przechowywane w tabelach zawierających je encji a nie w osobnych tabelach. W celu sprawdzenia poprawności

oraz wydajności bazy danych zaimplementowałem klasę DataGenerator udostępniającą funkcję GenerateFakeData pozwalającą na wypełnienie bazy fake'owymi danymi w oparciu o bibliotekę Bogus.

W kolejnym kroku zaimplementowałem 3 zapytania realizujące treść zadania. Zapytania są wykonywane za pomocą kodu oraz poprzez wykonanie zapytań LINQ co pozwala sprawdzić ich poprawność. Zgodnie z konwencją indeksy są tworzone dla każdej właściwości/kolumny używanej jako klucz obcy. Poniżej umieszczam zapytania SQL wygenerowane przez LINQ:

A. Znajdź imię i nazwisko osoby posiadającej największą liczbę wnucząt płci żeńskiej.

```
SELECT TOP(1) [p].[Id], [p].[Forename], [p].[Surname]
FROM [Persons] AS [p]
ORDER BY (
    SELECT COUNT(*)
    FROM [Persons] AS [p0]
    WHERE [p].[Id] = [p0].[PartnerId] AND [p0].[Gender] = N'Female')
DESC
```

B. Przedstaw średnią ilość pracowników zatrudnionych na umowę zlecenie i średnią ilość pracowników zatrudnionych na umowę o pracę we wszystkich firmach oraz średnią pensję dla tych umów.

```
SELECT COUNT(*) FROM Employments
SELECT COUNT(*)
              FROM Employments AS e
              WHERE e.Contract = N'Employment Contract'
```

Tutaj można by założyć indeks na kolumnie Contract.

C. Znajdź rodzinę (co najwyżej 2 pokoleniową) najmniej zarabiającą. Przedstaw imię i nazwisko dowolnej osoby z tej rodziny.

```
SELECT [p].[Id], [p].[Forename], [p].[Surname]
FROM [Persons] AS [p]
LEFT JOIN [Persons] AS [p0] ON [p].[PartnerId] = [p0].[Id]
ORDER BY [p].[Earnings] + CASE
    WHEN [p0].[Id] IS NULL THEN 0
    ELSE [p0].[Earnings]
END + (
    SELECT COALESCE(SUM([p1].[Earnings] + CASE
        WHEN [p2].[Id] IS NULL THEN 0
        ELSE [p2].[Earnings]
```

```
    END + CASE
        WHEN [p3].[Id] IS NULL THEN 0
        ELSE [p3].[Earnings]
    END), 0)
    FROM [Persons] AS [p1]
    LEFT JOIN [Persons] AS [p2] ON [p1].[MotherId] = [p2].[Id]
    LEFT JOIN [Persons] AS [p3] ON [p1].[FatherId] = [p3].[Id]
    WHERE [p].[Id] = [p1].[MotherId]) + (
    SELECT COALESCE(SUM([p4].[Earnings] + CASE
        WHEN [p5].[Id] IS NULL THEN 0
        ELSE [p5].[Earnings]
    END + CASE
        WHEN [p6].[Id] IS NULL THEN 0
        ELSE [p6].[Earnings]
    END), 0)
    FROM [Persons] AS [p4]
    LEFT JOIN [Persons] AS [p5] ON [p4].[MotherId] = [p5].[Id]
    LEFT JOIN [Persons] AS [p6] ON [p4].[FatherId] = [p6].[Id]
    WHERE [p].[Id] = [p4].[FatherId])
```

To zapytanie można by zoptymalizować wyszukując najpierw najmniej zarabiającą bezdzietną osobę co daje nam górne oszacowanie – rodziny z dziećmi będą już tylko sumaryczne więcej zarabiać. Następnie należy sprawdzić wszystkie rodziny (co najwyżej 2 pokoleniowe) dla osób zarabiających mniej od znalezionej. Jeśli pod pojęciem najmniej zarabiającej rodziny kryje się średnia zarobków na osobę to oczywiście ta optymalizacja nie będzie działa. Natomiast aktualny wynik należy podzielić przez liczbę osób i wyszukać minimum.