Wyższa Szkoła Technologii Informatycznych w Katowicach

Temat pracy: „Badanie i analiza wydajności mechanizmów mapowań obiektowo relacyjnych”

Autor: Piotr Golli

Promotor: dr Jadwiga Bakonyi

Kategorie: porównanie wydajności

Słowa kluczowe: C#, .NET, Entity Framework, Entity Framework Core, ADO .NET, ORM, SQL, Performance

Referat pracy dyplomowej

1. Cel i podstawowe założenia

Celem pracy było dokonanie analizy wydajności oraz przeprowadzenie badań związanych z mapowaniem obiektowo relacyjnym, pomiędzy wybranym silnikiem bazy danych, a weryfikującą wydajność aplikacją utworzoną z wykorzystaniem wybranego języka programowania. W aplikacji zastosowano wyszczególnione mechanizmy mapowań obiektowo relacyjnych, były nimi:

- Mapowanie manualne - ADO .NET Data Reader.
- Entity Framework Core 2.2.
- Entity Framework 6.2.

2. Realizacja projektu

W zakresie pracy dokonano analizy problemu badawczego, wybrano silnik bazy danych do zgromadzenia danych i przeprowadzenia badań, a także zadeklarowano język programowania. Za pomocą określonego języka programowania zrealizowano badania korzystające z wybranych technologii mapujących obiektowo relacyjnie. Zastosowanym językiem programowania był język C# firmy Microsoft. Język ten wybrano, aby zachować jak największą kompatybilność technologiczną z systemem zarządzania bazami danych Microsoft SQL Server, który został wybrany do realizacji zadania dostarczania i przechowywania danych. Umieszczono w nim zbiory danych, które posłużyły przeprowadzeniu analizy wydajności podstawowych operacji przeprowadzanych przez mechanizmy mapujące obiektowo relacyjnie. Podstawowym celem pracy nie było oparcie się o określoną strukturę bazy danych, a jedynie wskazanie różnicy wydajności technologii względem samej ilości jak i stopnia złożoności danych.

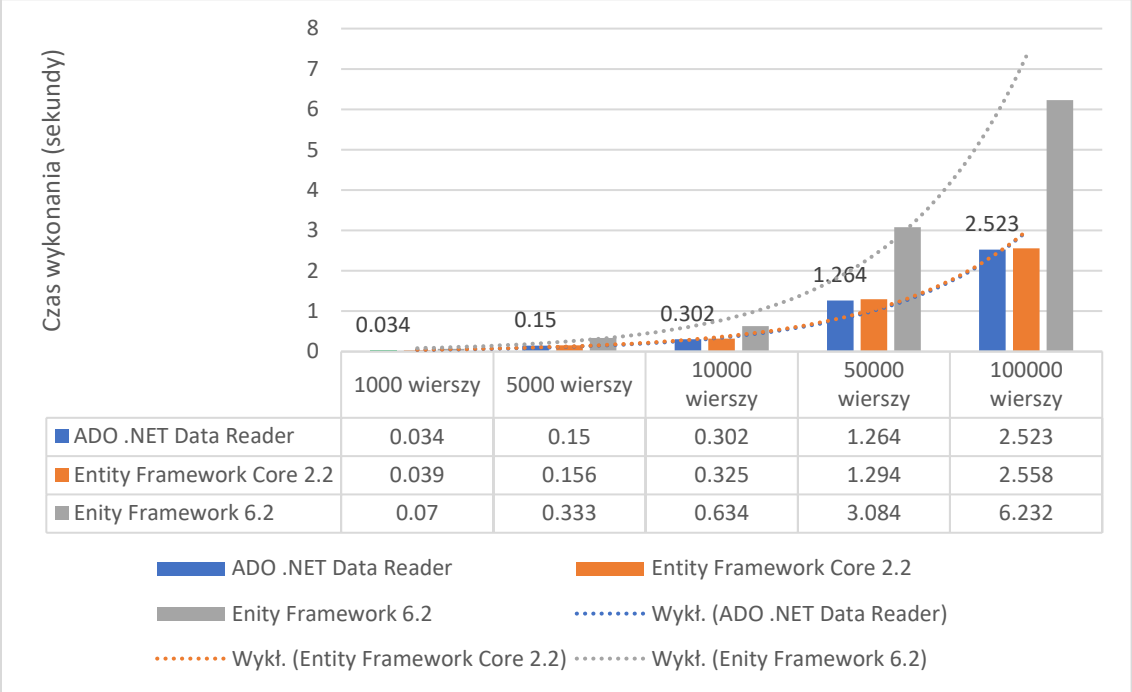
3. Realizacja projektu

Do scenariuszy badań należały podstawowe operacje wykonywane na danych przechowywanych w bazie danych. Większość scenariuszy została zrealizowana w obrębie liczby tysiąca, pięciu, dziesięciu oraz pięćdziesięciu tysięcy obiektów. Wyjątkami od tego schematu były operacje pobierania danych, które zostały przeprowadzone również na większej liczbie obiektów. Scenariusze obejmowały badania nad odwzorowaniem obiektowo relacyjnym dla operacji pobierania dużych, małych oraz zależnych obiektów, czyli posiadających obiekty innego typu wewnątrz swojej struktury. Kolejną koncepcją było dodawanie dużych oraz mniejszych obiektów. Wśród ostatnich scenariuszy znalazły się również operacje częściowej jak i całkowitej aktualizacji małych oraz dużych obiektów obejmujących różne ilości danych. Ostatnimi zaplanowanymi scenariuszami były operacje usuwania wielotysięcznych danych małych i dużych obiektów.

Autor przeprowadził rozliczne badania w przytoczonych wyżej scenariuszach. Wyniki badań zostały przedstawione w formie wykresów oraz zestawień tabelarycznych. Przykłady rezultatów badań dla określonych scenariuszy zostaną zaprezentowane poniżej.

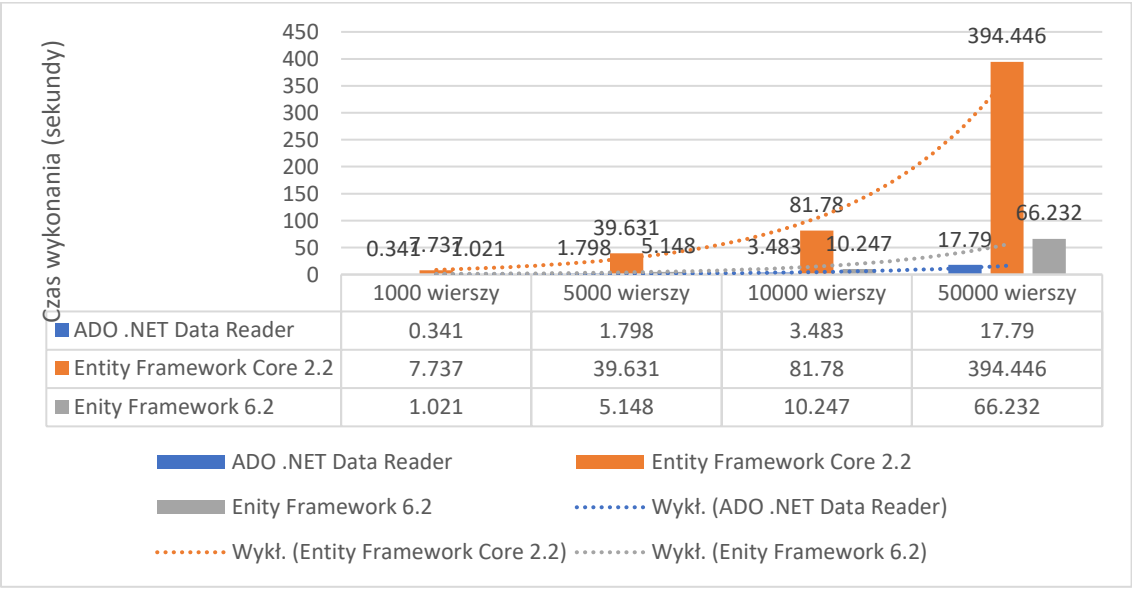
W pracy dokonano analizy wydajności mapowania obiektowo relacyjnego dla operacji DML oraz DQL względem tabel o skomplikowanej i prostej strukturze.

3.1. Pobranie dużych obiektów



Wykres powyżej przedstawia rezultaty czasu pobierania i odwzorowania obiektowego dużych obiektów w zadanych liczebnościach.

3.2. Dodawanie dużych obiektów



Wykres powyżej przedstawia czasy przetwarzania operacji dodawania nowych dużych obiektów.

3.3. Aktualizacja dużych obiektów

		ADO .NET Data Reader	Entity Framework Core 2.2	Entity Framework 6.2
Aktualizacja częściowa	**1 tys. wierszy**	0.106 s.	1.187 s.	0.778 s.
	5 tys. wierszy	0.533 s.	5.998 s.	3.741 s.
	10 tys. wierszy	1.073 s.	11.841 s.	7.528 s.
	50 tys. wierszy	5.407 s.	59.337 s.	38.264 s.
Aktualizacja całkowita	**1 tys. wierszy**	0.227 s.	7.210 s.	74.296 s.
	5 tys. wierszy	1.192 s.	36.401 s.	1858.612 s.
	10 tys. wierszy	2.364 s.	96.876 s.	7358.208 s.
	50 tys. wierszy	19.617 s.	669.234 s.	-

Aktualizację dla każdego ze scenariuszy aktualizujących dane przeprowadzono w formie częściowej jak i całkowitej. W tabeli powyżej zamieszczone są rezultaty badań.

3.4. Usuwanie dużych obiektów

	ADO .NET Data Reader	Entity Framework Core 2.2	Entity Framework 6.2
1 tys. wierszy	0.06 s.	0.300 s.	37.781 s.
5 tys. wierszy	0.359 s.	1.495 s.	928.072 s.
10 tys. wierszy	1.815 s.	2.993 s.	3652.757 s.
50 tys. wierszy	8.845 s.	14.499 s.	91074.068 s.

Tabela powyżej przedstawia rezultaty czasu usuwania i odwzorowania obiektowego akcji usunięcia dużych obiektów w zadanych liczebnościach.

4. Podsumowanie wyników

Po dokonanej analizie wyników przeprowadzonych badań, pojawiło się kilka wniosków. Podczas badań zastosowano trzy technologie mapujące obiektowo relacyjnie, Entity Framework Core, ADO .NET Data Reader, oraz Entity Framework. Dwie z nich sprawdziły się we wszystkich zadaniach, tylko Entity Framewrok 6.2 nie potrafił przeprowadzić niektórych scenariuszy badań z powodzeniem. Mechanizmy mapujące obiektowo relacyjnie są licznie wykorzystywane przez programistów na całym świecie, najczęściej za pomocą wysokopoziomowych języków programowania. Bardziej zautomatyzowane mechanizmy mapujące, umożliwiają zdecydowanie szybszą pracę programistów podczas procesu wytwarzania oprogramowania. Zaawansowane rozwiązania przyczyniają się również do tego, że do zrealizowania podstawowych zastosowań operacji „CRUD", to jest dodawania, pobierania, aktualizacji oraz usuwania danych, nie jest potrzebna znajomość języka SQL. Mechanizmy te udostępniają podstawowe jak i bardzo zaawansowane możliwości, tworząc zwykle ponad domyślnymi technologiami dla języka pewną logiczną abstrakcję.

Zdecydowanie warto zapamiętać, iż wybór technologii zależy od oczekiwań i potrzeb biznesowych. W rezultacie należy dostosować użyte technologie do określonego projektu. Jaki jest cel użycia ogromnej biblioteki, kiedy jej możliwości nie zostaną wykorzystane nawet w jednym procencie? Jeśli potrzebne jest jednokrotne dostanie się do bazy danych i posłanie pojedynczego zapytania skopiowanego z sieci, nie ma sensu używać bardziej zaawansowanych narzędzi ponad te domyślne, które wymuszają mapowanie manualnie. Jeśli tworzony system posiada skomplikowaną domenę, olbrzymią bazę danych, wiele tabel, procedur, funkcji, widoków, wtedy lepiej skorzystać z narzędzia, które po części zautomatyzuje proces dostępu do danych. Z kolei, jeśli rzeczywiście system tworzony jest przez ekspertów i oczekują oni od funkcjonowania systemu największej elastyczności, szybkości wykonania, optymalizacji prowadzonych procesów, wtedy warto świadomie zrezygnować z tych automatycznych mechanizmów mapujących na rzecz mapowania manualnego. Podejście to, zapewnia największe możliwości pod względem integracji z relacyjną bazą danych. Jednak użycie takiego rodzaju technologii jest obarczone największym nakładem pracy programisty. Zadanie, które dla zautomatyzowanego mechanizmu mapującego obiektowo relacyjnie może zająć jedną linijkę kodu, przy mapowaniu

manualnym może wymagać ich aż kilkadziesiąt, kilkaset, a przy większych problemach nawet wiele więcej.

Obecnie większość aktualnych technologii zrealizowanych w języku C# zezwala na asynchroniczne użycie wybranego mechanizmu mapującego. To z kolei pozwala na przeprowadzenie wielu operacji równolegle, ze wspólnym oczekiwaniem na wynik, co znacznie może przyspieszyć czas wykonania wielu operacji. Dzisiejsze bazy danych są wysokowydajne, zdolne do przetwarzania olbrzymich ilości danych oraz realizacji wielu zapytań na zdalnym serwerze. Umożliwia to niezwykle szybkie rozwiązywanie problemów biznesowych oraz pozwala programistom na większe skupienie się wokół tematu optymalizacji wydajnościowej. Czas realizacji zadań jest niezwykle istotny z punktu widzenia celów strategicznych, przykładowo na skomplikowany raport nie można czekać cały dzień. Współcześni programiści, tworzący programy w języku C#, często wybierają technologię Entity Framework Core. Dla małych i średnich potrzeb biznesowych, znacznie przyśpiesza ona czas realizacji tworzonego oprogramowania. Z automatyzacji i technologii wyręczających programistę, powinno korzystać się świadomie, by uniknąć możliwych poważnych błędów.