JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 8-12
Received: 4 October 2021
Accepted: 26 October 2021

Comparative analysis of the time performance of database queries in C# language

Analiza porównawcza wydajności czasowej zapytań bazodanowych w języku C#

Tomasz Nowicki*, Sebastian Tomczak*, Grzegorz Kozieł

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
There are many computer applications in the world that use databases to store, process, and use data. That translates into many different ways of handling these databases. It is therefore difficult to choose a solution that meets the needs of the user. This article compares three C# solutions in terms of time efficiency: the Entity Framework Core application framework, pure SQL queries, and parameterized Prepared Statement queries. The results obtained in the course of the research has shown that the fastest solution is the use of non-parameterised SQL queries. The use of Entity Framework Core is the slowest of the three tested solutions.

Keywords: C#; Entity Framework Core; SQL Database; Prepared Statement; .NET Framework

Streszczenie
Na świecie istnieje duża liczba aplikacji komputerowych wykorzystujących bazy danych celem utrwalania, przetwarzania i wykorzystania danych, co przekłada się na wiele różnych sposobów obsługi tychże baz. Trudno jest więc wybrać rozwiązanie spełniające potrzeby użytkownika. W niniejszym artykule porównano pod kątem wydajności trzy rozwiązania dla języka C#: szkielet aplikacji Entity Framework Core, zapytania SQL przesyłane w postaci jednego łańcucha znaków, oraz sparametryzowane zapytania Prepared Statement. Uzyskane w toku badań wyniki pozwoliły określić, że najszybszym rozwiązaniem jest wykorzystanie niesparametryzowanego zapytania SQL. Wykorzystanie Entity Framework Core jest najwolniejszym z trzech badanych rozwiązań.

Słowa kluczowe: C#; Entity Framework Core; SQL Database; Prepared Statement; .NET Framework

*Corresponding author
Email address: tomasz.nowicki1@pollub.edu.pl, sebastian.tomczak@pollub.edu.pl

1. Wstęp

Tworzenie aplikacji, niezależnie od tego czy są to aplikacje mobilne, desktopowe czy internetowe, jest blisko powiązanie z użyciem baz danych aby utrwalić, przetworzyć oraz wykorzystać dane nie tylko uzyskane w czasie działania aplikacji, ale również dane niezbędne do działania tychże aplikacji. Nie jest więc zaskakujące, że wraz w upływem czasu powstały coraz to bardziej zróżnicowane sposoby na ułatwienie i usprawnienie połączenia i komunikacji pomiędzy aplikacją oraz bazą danych [1]. Sytuacja ta stawia programistę przed wyzwaniem, który ze sposobów dostępnych na rynku jest tym, który spełnia założone oczekiwania oraz wymagania.

W niniejszym artykule zdecydowano się zestawić ze sobą trzy popularne rozwiązania dostępne dla języka programistycznego C#. W zestawieniu wzięto pod uwagę czas potrzebny na wykonanie danego zapytania. W efekcie przeprowadzonego badania zidentyfikowano najwydajniejsze rozwiązanie. Tak przeprowadzone badanie oraz wnioski wyciągnięte na jego podstawie pozwolą na wybranie najszybszego rozwiązania do komunikacji i obsługi aplikacji z bazą danych. Systemem bazodanowym, na której zostały przeprowadzone badania jest system Microsoft SQL Server.

Rozwiązania przedstawione w artykule to szkielet aplikacji Entity Framework Core, zapytania SQL przesyłane w postaci łańcucha znaków z aplikacji, oraz sparametryzowane zapytania Prepared Statement. Entity Framework Core [2], jako rozszerzalna część technologii Entity Framework, wykorzystuje nowocześniejsze oraz obiektowe podejście przy pracy z bazami danych jakim jest mapowanie obiektowo-relacyjne (ang. Object-Relational Mapping, ORM) [3]. Pozostałe dwa rozwiązania są rozwiązaniami starszymi, gdzie programista komunikuje się z serwerem bazodanowym bezpośrednio za pomocą zapytań SQL, gdzie w pierwszym sposobie są to zapytania SQL przesyłane do serwera bazodanowego w postaci łańcucha znaków zawierającego komendy SQL oraz parametry i dane, natomiast drugim sposobem są zapytania sparametryzowane tzw. Prepared Statement.

2. Wybrane technologie

2.1. Język C#

Język C# jest zorientowanym obiektowo, nowoczesnym językiem programowania działającym w środowisku .NET [4], umożliwiającym tworzenie typów bezpiecznych. Język ten udostępnia również konstrukcje językowe bezpośrednio obsługujące koncepcje obiektowości. Wbudowane w niego zostały również takie funkcje

jak automatyczne odzyskiwanie pamięci zajętej przez nieosiągalne i nieużywane obiekty, wyrażenia lambda obsługujące techniki programowania funkcjonalnego, ustrukturyzowana i rozszerzalna obsługa wyjątków oraz wykrywania i odzyskiwania błędów. Ponadto, jedną z wbudowanych funkcji jest składnia zapytań LANGUAGE Integrated Query (LINQ) [5], tworzące wzorzec pracy z danymi z dowolnego źródła.

2.2. Entity Framework Core

Entity Framework Core jest lekką, otwartą, międzyplatformową i rozszerzalną wersją technologii Entity Framework [6], w której nacisk położono na sprawny dostęp do danych. Entity Framework Core może służyć jako maper obiektowo-relacyjny (ang. O/RM - Object/Relational Mapper), pozwalający na pracę z bazą danych z użyciem obiektów .NET oraz zmniejszenie zapotrzebowana na większość kodu dostępu do danych, o który zadbać musiał programista. Technologia ta posiada obsługę wielu silników bazodanowych takich jak: MySQL, Oracle DB, PostreSQL, Firebird, oraz wykorzystywanym w badaniach Microsoft SQL Serverem.

2.3. Microsoft SQL Server

System zarządzania relacyjną bazą danych (ang. Relational Database Management System, RDBMS) firmy Microsoft składa się między innymi z silnika bazy danych, przez co cały system jest często postrzegany jako baza danych. SQL Server, jako system RDBMS, jest zaimplementowany i pracuje w architekturze klient/serwer, gdzie każdy komponent jest w stanie pracować samodzielnie i niezależnie od pozostałych. Cechą SQL Servera jest obsługa odbywająca się głównie z wykorzystaniem narzędzi zarządzania z wbudowanym interfejsem graficznym użytkownika, np. z wykorzystaniem narzędzia SQL Server Management Studio. Komunikacja pomiędzy użytkownikiem a silnikiem bazy danych odbywa się poprzez strukturalny język zapytań (ang. Structured Query Language, SQL), który został również rozbudowany o autorskie rozszerzenia firmy Microsoft oraz występuje w dialekcie Transact-SQL.

3. Metodyka badawcza

3.1. Środowisko badawcze

Aby przeprowadzić badania, najpierw przygotowano odpowiednio skonfigurowane środowisko badawcze z lokalnie zainstalowanym potrzebnym oprogramowaniem oraz wymaganymi narzędziami pozwalającymi na przeprowadzenie tychże badań oraz rejestrację ich wyników na maszynie wirtualnej. Wykorzystane oprogramowanie oraz parametry środowiska badawczego przedstawiono w tabelach 1 i 2.

Bazą danych wykorzystaną w badaniach jest udostępniona przez firmę Microsoft na licencji MIT baza danych Adventure Works 2019 [7] przedstawiająca fikcyjny międzynarodowy zakład produkcyjny. Ze względu na rozmiar tejże bazy, zdecydowano się na skorzystanie z zaledwie jej fragmentu, który jest przedstawiony na rysunku 1. Wykorzystane tabele wraz z ich liczbą rekordów pokazane zostały w tabeli 3.

Tabela 1: Wykorzystane oprogramowanie

Nazwa narzędzia	Wersja
SQL Server Management Studio	15.0.18384.0
Microsoft SQL Server 2019	15.0.2000.5
Microsoft Visual Studio Community 2019	16.8.4
.Net 5.0	5.0.202
VirtualBox	6.1.22 r144080 (Qt5.6.2)

Tabela 2: Parametry środowiska badawczego

System operacyjny	Windows 10 Education – Wersja 10.0.18363
System operacyjny maszyny wirtualnej	Windows 10 Education – Wersja 10.0.19041
Pamięć RAM	Pamięć Corsair Vengeance LPX, DDR4, 16 GB, 3200MHz, CL16
Pamięć RAM maszyny wirtualnej	8 GB
Procesor	Procesor AMD Ryzen 5 1600 AF, 3.2GHz, 16 MB
Procesor maszyny wirtualnej	6 rdzeni logicznych, 3.2GHz, 16 MB
Dysk	Dysk SSD GoodRam 240 GB 2.5" SATA III (SSDPR-IRIDPRO-240)
Dysk maszyny wirtualnej	65GB HDD
Pamięć wideo maszyny wirtualnej	128MB

Tabela 3: Wykorzystane tabele wraz z ich liczbą rekordów

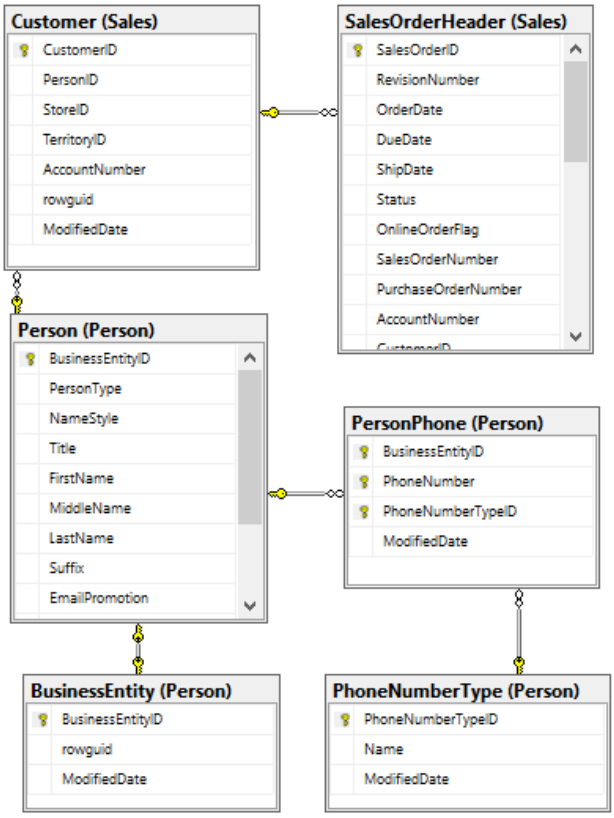
Nazwa tabeli	Liczba rekordów
SalesOrderHeader	31465
Customer	19820
Person	19972
PersonPhone	19972
PhoneNumberType	3
BusinessEntity	20777

3.2. Scenariusze badawcze

Aby przeprowadzić badania i zebrać wyniki potrzebne do porównania wydajności czasowej zapytań bazodanowych przygotowano 5 scenariuszy badawczych, różniących się złożonością zapytań SQL. Przy wyborze zapytań kierowano się przede wszystkim ich wykorzystaniem biznesowym oraz praktycznością. Każdy ze scenariuszy został uruchomiony na bazie danych Adventure Works 2019 działającej na serwerze Microsoft SQL Server 100 razy aby zebrać pulę wyników poddanej następnie analizie. Scenariusze badawcze przedstawione zostały w tabeli 4, podczas gdy ich kod pokazany został na listingach 1-5.

Tabela 4: Scenariusze badawcze

Nr.	Zapytanie
1	Wybranie wszystkich zamówień poszczególnych klientów.
2	Wybranie wszystkich zamówień poszczególnych klientów wraz z przypisanym do nich numerem telefonu i typem numeru telefonu.
3	Wybranie wszystkich zamówień poszczególnych klientów o danej wartości Statusu.
4	Wybranie wszystkich zamówień poszczególnych klientów wraz z przypisanym do nich numerem telefonu i typem numeru telefonu, zawężone do konkretnego typu numeru telefonu.
5	Policzenie liczby zamówień dla klientów osobowych reprezentowanych przez ich numer identyfikacyjny klienta, bądź osoby.



Rysunek 1: Fragment schematu wykorzystywanej bazy danych.

Listing 1: Kod zapytania Nr. 1

```
SELECT SalesOrderID,
SalesOrderNumber, PurchaseOrderNumber,
OrderDate, [Status], AccountNumber,
CustomerID, SubTotal, TotalDue
FROM Sales.SalesOrderHeader
```

Listing 2: Kod zapytania Nr. 2

```
SELECT SOH.SalesOrderID, SOH.CustomerID,
C.AccountNumber, C.PersonID,
P.Title, P.FirstName, P.LastName,
PP.PhoneNumber, PNT.PhoneNumberTypeID, PNT.Name
FROM Sales.SalesOrderHeader SOH
JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
JOIN Person.PersonPhone PP
ON P.BusinessEntityID = PP.BusinessEntityID
JOIN Person.PhoneNumberType PNT
ON PP.PhoneNumberTypeID = PNT.PhoneNumberTypeID
```

Listing 3: Kod zapytania Nr. 3

```
SELECT SalesOrderID, SalesOrderNumber,
PurchaseOrderNumber, OrderDate,
[Status], AccountNumber, CustomerID,
SubTotal, TotalDue
FROM Sales.SalesOrderHeader
WHERE Status = 5
```

Listing 4: Kod zapytania Nr. 4

```
SELECT SOH.SalesOrderID, SOH.CustomerID,
C.AccountNumber, C.PersonID,
P.Title, P.FirstName, P.LastName,
PP.PhoneNumber, PNT.PhoneNumberTypeID, PNT.Name
FROM Sales.SalesOrderHeader SOH
JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
JOIN Person.PersonPhone PP
ON P.BusinessEntityID = PP.BusinessEntityID
JOIN Person.PhoneNumberType PNT
ON PP.PhoneNumberTypeID = PNT.PhoneNumberTypeID
WHERE PNT.PhoneNumberTypeID = 1
```

Listing 5: Kod zapytania Nr. 5

```
SELECT SOH.CustomerID, C.PersonID,
P.Title, P.FirstName, P.LastName,
COUNT(SOH.SalesOrderID) AS OrderCount
FROM Sales.SalesOrderHeader SOH
JOIN Sales.Customer C
ON SOH.CustomerID = C.CustomerID
JOIN Person.Person P
ON C.PersonID = P.BusinessEntityID
GROUP BY SOH.CustomerID, C.PersonID,
P.Title, P.FirstName, P.LastName
```

3.3. Aplikacja testowa

Do przeprowadzenia serii badań z wykorzystaniem przedstawionych w tabeli 3 scenariuszy badawczych przygotowana została aplikacja napisana w języku C# składająca się z 3 projektów, każdy z nich związany odpowiednio z danym badanym rozwiązaniem. W każdym projekcie zawarty jest ten sam mechanizm pomiaru czasu, powielony celem odseparowania bibliotek od siebie, aby potencjalne referencje między nimi nie wpływały negatywnie na wyniki badań.

Aplikacja składa się na zbiór testów oznaczonych uprzednio zaprogramowanym atrybutem “Performance”. Głównym elementem każdego z testów jest część wykonawcza, która zależnie od typu testu zawiera inne zapytanie. W opisywanym rozwiązaniu znajduje się 5 testów - po jednym na każde zapytanie w każdym z projektów. W zależności od danej implementacji, “SQL”, “Prepared Statement” czy “EFC” kod wykonawczy jest różny. W pierwszym z nim znajduje się zapytanie w postaci kodu SQL oraz mapowania zwróconych danych do obiektów.

Kolejne z rozwiązań rozszerza wcześniejsze o dodanie do niektórych zapytań w postaci kodu SQL argumentów, które są w odpowiedni sposób przekazywane do zapytania. Ponownie zwrócony rezultat jest mapowany do pamięci. Ostatni projekt obejmuje wykorzystanie Entity Framework Core, zapytania tym razem realizowane są za pomocą LINQ wykorzystywanego na specjalnym obiekcie Context, całość za sprawą wykorzystania ORM obejmuje pracę na obiektach, wynik mapowany jest po stronie EFC, a jego wynik zapisywany jest do zmiennej analogicznie do poprzednich przypadków.

Atrybut “Performance” odpowiada za programowe zliczanie czasu trwania danych akcji - zapytań, we wcześniej przygotowanej bazie danych, a więc ostatecznie każde wykonanie dowolnego testu z dowolnego projektu skutkuje wykonaniem zapytania za pomocą danej technologii i utrwalenie jego czasu realizacji w celu umożliwienia dalszej analizy.

4. Wyniki badań

Przeprowadzenie serii badań przy wykorzystaniu aplikacji testowej opisanej w rozdziale 3.3 zgodnie ze scenariuszami badawczymi ukazanymi w rozdziale 3.2 poskutkowało następującymi wynikami przedstawionymi w tabelach 5-9, oraz na rysunkach 2-6. Wartościami, na których zdecydowano się skupić porównując je ze sobą, są mediana oraz czas średni wykonania zapytań w milisekundach. Natomiast w celu zobrazowania

uzyskanych wyników posłużono się wykresami pudełkowymi z wąsem.

Tabela 5: Czas wykonania zapytania Nr. 1

Metoda	Mediana [ms]	Czas średni [ms]
SQL	122,19	122,89
Prepared Statement	121,01	125,25
Entity Framework Core	1258,51	1354,19

Tabela 6: Czas wykonania zapytania Nr. 2

Metoda	Mediana [ms]	Czas średni [ms]
SQL	270,83	275,84
Prepared Statement	271,55	752,88
Entity Framework Core	1540,15	2004,88

Tabela 7: Czas wykonania zapytania Nr. 3

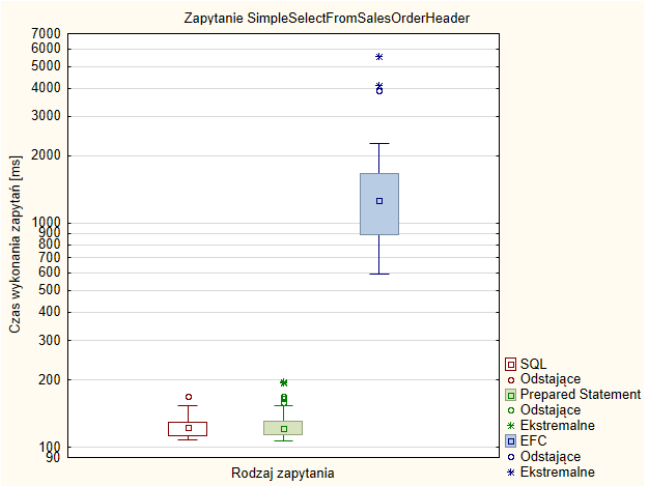
Metoda	Mediana [ms]	Czas średni [ms]
SQL	118,39	120,74
Prepared Statement	141,42	328,32
Entity Framework Core	1252,05	1325,35

Tabela 8: Czas wykonania zapytania Nr. 4

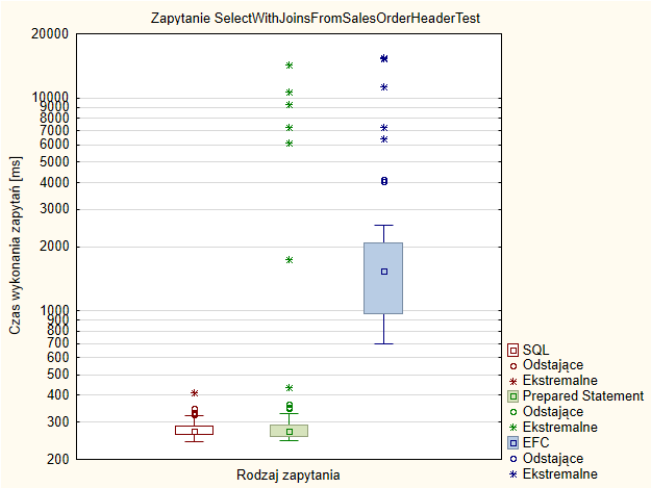
Metoda	Mediana [ms]	Czas średni [ms]
SQL	189,15	192,75
Prepared Statement	191,19	542,15
Entity Framework Core	1588,49	2228,27

Tabela 9: Czas wykonania zapytania Nr. 5

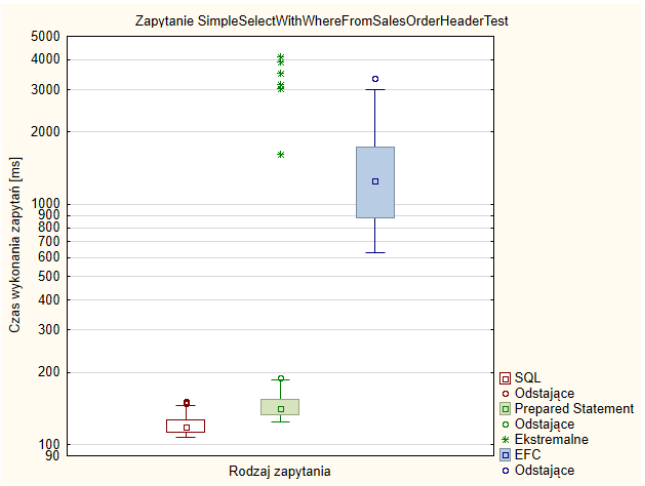
Metoda	Mediana [ms]	Czas średni [ms]
SQL	80,38	83,65
Prepared Statement	84,91	528,24
Entity Framework Core	1440,96	2125,78



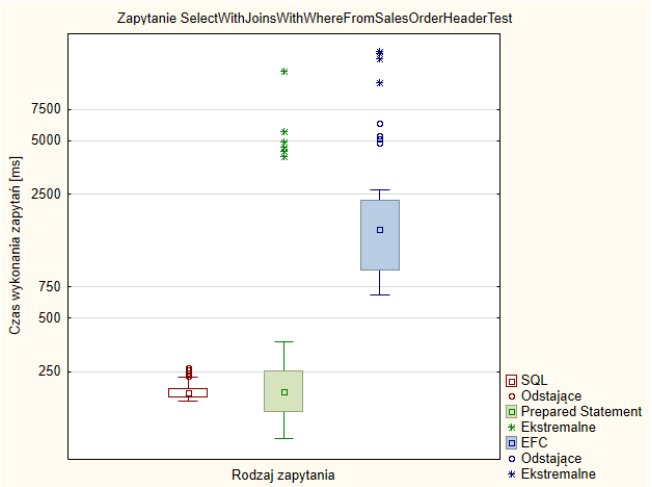
Rysunek 2: Zestawienie wyników badań zapytania Nr. 1.



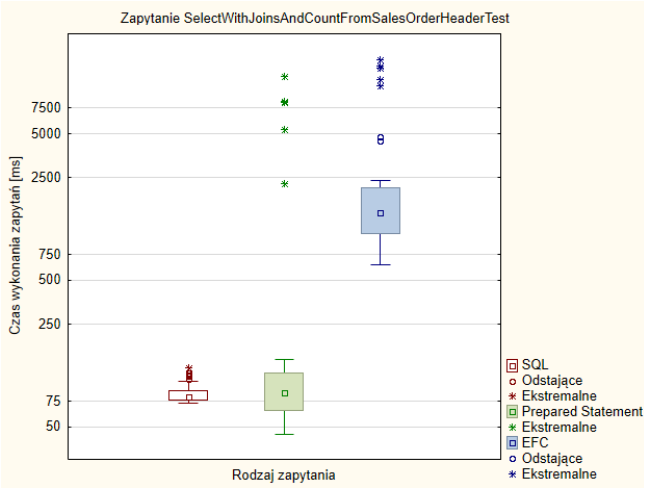
Rysunek 3: Zestawienie wyników badań zapytania Nr. 2.



Rysunek 4: Zestawienie wyników badań zapytania Nr. 3.



Rysunek 5: Zestawienie wyników badań zapytania Nr. 4.



Rysunek 6: Zestawienie wyników badań zapytania Nr. 5.

5. Wnioski

W przeprowadzonych badaniach z wykorzystaniem rozwiązań SQL, Prepared Statement oraz Entity Framework Core realizowanych za pomocą aplikacji opartej na testach napisanej w języku C# można jednoznacznie stwierdzić, że na tle dwóch pozostałych rozwiązań, Entity Framework Core jest rozwiązaniem najwolniejszym. Wykorzystanie tej technologii odciąża programistę od znajomości języka SQL z racji bazowania na obiektowo-relacyjnym mapowaniu. co może okazać się dużą zaletą.

Wartym uwagi jest fakt, że Entity Framework Core wykonując proces translacji zapisu programistycznego (LINQ) do kodu SQL dokonał koniecznych operacji dzięki któremu ostateczne zapytania SQL miały zbliżoną postać do zapytań wykorzystanych w przypadku SQL oraz Prepared Statement.

Drugim w kolejności rozwiązaniem jest Prepared Statement. Warto zauważyć, że w zapytaniach, w których występował warunek “Where”, z racji wykorzystania parametru SQL, czyli Prepared Statement, cały proces trwał dłużej niż w przypadku zwykłych zapytań SQL.

Podejście związane z wykorzystaniem zwykłych zapytań SQL cechuje najmniejszy rozrzut czasowy całego procesu ze względu na prosty sposób implementacji rozwiązania, dodatkowo bezpośrednio wskazując na fakt, że ta metoda była najszybsza na tle pozostałych badanych.

Literatura

[1] S. Cvetković, D. Janković, A comparative study of the features and performance of orm tools in a. net environment, International Conference on Object and Databases, Springer, Berlin, Heidelberg, (2010) 147-158.

[2] H. Schwichtenberg, Modern Data Access with Entity Framework Core, Apress, Essen, Germany, 2018.

[3] D. Bowers, C. Ireland, M. Newton, K. Waugh, Understanding object-relational mapping: A framework based approach, International lournal On Advances in Software 2.2, (2009) 202-216.

[4] B. Meyer, .NET is coming [Microsoft Web services platform], Computer 34.8 (2001) 92-97.

[5] E. Meijer, The world according to LINQ, Communications of the ACM 54.10 (2011) 45-51.

[6] A. Adya, J. A. Blakeley, S. Melnik, S. Muralidhar, Anatomy of the ado. net entity framework, Proceedings of the 2007 ACM SIGMOD international conference on Management of data (2007) 877-888.

[7] M. Mitri, Teaching Tip: Active Learning via a Sample Database: The Case of Microsoft's Adventure Works, Journal of Information Systems Education 26.3 (2015) 177-185.