JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 14–20
Received: 10 September 2023
Accepted: 1 October 2023

Comparative analysis of query execution speed using Entity Framework for selected database engines

Analiza porównawcza szybkości wykonywania zapytań za pomocą Entity Framework dla wybranych silników baz danych

Krzysztof Winiarczyk*, Rafał Stęgierski

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

Abstract
This article presents the comparative analysis of time efficiency while executing queries by object-relational mapping framework Entity Framework for the following database engines: Microsoft SQL Server, MySQL and PostgreSQL. Time measurements of obtaining object results from database queries were made by app created in C#. Queries referred to single or multiple tables linked by relationships (1:1, 1:n, m:n) and performed operations of reading, creating, updating and deleting data. Obtained results have been cleaned from outliers and trimmed means were given as final results. Different database engines obtained the shortest query execution times depending on record number and table structures.

Keywords: time efficiency; Entity Framework; database

Streszczenie
Artykuł przedstawia analizę porównawczą wydajności czasowej wykonywania zapytań za pomocą szkieletu mapowania obiektowo-relacyjnego Entity Framework dla następujących silników baz danych: Microsoft SQL Server, MySQL i PostgreSQL. Pomiaru czasu uzyskania obiektowych rezultatów zapytań do bazy danych dokonano przy pomocy aplikacji napisanej w języku C#. Zapytania dotyczyły jednej tabeli bądź kilku tabel połączonych relacjami (1:1, 1:n, m:n) oraz realizowały operacje odczytu, tworzenia, aktualizacji i usuwania danych. Uzyskane rezultaty oczyszczono z wartości odstających, a jako wyniki podano średnie ucinane. W zależności od liczby rekordów oraz struktury tabel różne silniki baz danych uzyskiwały najkrótsze czasy wykonania zapytań.

Słowa kluczowe: wydajność czasowa; Entity Framework; baza danych

*Corresponding author
Email address: **krzysztof.r.winiarczyk@gmail.com** (K. Winiarczyk)

1. Wstęp

Jednym z rozwiązań wykorzystywanym we współczesnych aplikacjach serwerowych do trwałego przechowywania informacji są relacyjne bazy danych. Do ich niewątpliwych zalet należą m. in.: zapewnianie bezpieczeństwa i integralności informacji oraz możliwość agregowania danych w celu uzyskania złożonych raportów.

W relacyjnych bazach danych przechowywanie informacji zorientowane jest na relacje jakie zachodzą pomiędzy konkretnymi encjami, natomiast duża część języków programowania (używanych współcześnie do tworzenia aplikacji internetowych) zorientowana jest obiektowo. Rozwiązaniem ułatwiającym przenoszenie danych pomiędzy bazą danych a aplikacją są szkielety programistyczne mapowania obiektowo-relacyjnego. Jednym z nich jest Entity Framework, który oferuje możliwość pisania zapytań do bazy danych bezpośrednio w kodzie programu, bez użycia języka SQL. Taka funkcjonalność usprawnia tworzenie aplikacji, stanowi ułatwienie dla programisty (który operuje jednym językiem) oraz poprawia przejrzystość kodu – logika biznesowa nie jest rozproszona pomiędzy kod aplikacji i skrypty SQL [1].

Szkielet programistyczny mapowania obiektowo-relacyjnego stanowi dodatkową warstwę, która pośredniczy między aplikacją i bazą danych. Zastosowanie tego rodzaju narzędzi ma negatywny wpływ na wydajność [2], która jest kluczowym elementem wytwarzanego oprogramowania. Istotną kwestią w przypadku projektowania nowych systemów informatycznych jest dobór odpowiednich dla danego zastosowania technologii, które po połączeniu pozwolą na stworzenie wydajnych rozwiązań.

Niniejsza praca została poświęcona porównaniu wydajności czasowej wykonywania zapytań przy użyciu narzędzia Entity Framework współpracującego z wybranymi silnikami baz danych.

2. Przegląd literatury

Optymalizacja aplikacji jest ważnym elementem procesu wytwórczego oprogramowania. Jednym ze sposobów na skrócenie czasu działania poszczególnych funkcjonalności aplikacji jest zmniejszenie czasu dostępu do danych poprzez użycie optymalnych metod łączenia się z bazą danych oraz wykorzystanie silników baz danych najbardziej odpowiednich dla rozpatrywanej aplikacji. Porównanie wydajności w kontekście baz danych jest popularnym tematem prac naukowych.

Celem artykułu [2] jest porównanie wydajności wykonywania zapytań dla trzech frameworków ORM dla .NET: Entity Framework Core 2.2, nHibernate 5.2.3 i Dapper 1.50.5. Dla każdego frameworka ORM autorzy przeprowadzili testy obejmujące zapytania typu insert, select, update i delete na tej samej bazie danych. Program utworzony do testowania jest aplikacją konsolową. Jako silnika bazodanowego autorzy użyli MSSQL Server 2018. Zebrane wyniki zostały podsumowane stwierdzeniem, że nie da się jednoznacznie wskazać najbardziej wydajnego frameworka, ani pod względem czasu wykonywania zapytań, ani pod względem użytej pamięci operacyjnej. W związku z czym wybór szkieletu ORM dla aplikacji (pod kątem wydajności) zależy od tego jakie zapytanie (typ operacji) wykonywany jest najczęściej.

Artykuł [3] zawiera porównanie wydajnościowe dwóch popularnych otwarto-źródłowych silników baz danych: MySQL (baza danych SQL) i CouchDB (baza danych NoSQL). Głównym celem publikacji jest przeprowadzenie analizy porównawczej wpływu systemu zarządzania bazą danych na wydajność aplikacji podczas wykonywania operacji typu CRUD. Autorzy rozpatrzyli cztery rozwiązania bazodanowe: MySQL w podejściu relacyjnym, MySQL w podejściu opartym na dokumentach oraz dwie struktury w CouchDB. Dokument w pierwszej strukturze, w CouchDB, jest prostym obiektem JSON zawierającym jedynie pary klucz-wartość (bez zagnieżdżonych obiektów), natomiast posiadającym odniesienie do innego dokumentu poprzez id. Dokument w drugiej strukturze jest złożonym obiektem JSON – posiadającym obiekty wielokrotnie zagnieżdżone, ale nieposiadającym odniesień do innych dokumentów. Struktura relacyjna bazy MySQL składa się z 5 tabel i 5 relacji jeden do wielu. Relacje można przedstawić następująco: kontynent (1:n) kraj, kraj (1:n) miasto, miasto (1:n) restauracja, miasto (1:n) hotel, hotel (1:n) restauracja. Struktura dla dokumentowego MySQL jest identyczna jak druga struktura bazy CouchDB. Każdy rodzaj operacji na danych został wykonany dla kilku liczności rekordów (1000, 10 000, 100 000, 1 000 000). Wynik dla rozwiązania, operacji i liczby rekordów stanowi średnia arytmetyczna z pięciu pomiarów czasu wykonania. W omówieniu wyników autorzy wskazali, że MySQL w podejściu relacyjnym wykazuje duże spadki wydajności przy zwiększaniu ilości danych (liczby rekordów). Lepszym rozwiązaniem okazuje się zastosowanie bazy CouchDB, natomiast najlepszą wydajnością czasową wyróżnia się MySQL w podejściu opartym na dokumentach. W podsumowaniu autorzy wskazują, że MySQL oparty na dokumentach jest bardzo dobrą alternatywą dla aplikacji przetwarzających duże ilości danych. Użycie drugiej struktury CouchDB pozwala osiągnąć lepszą wydajność czasową niż zastosowanie relacyjnego MySQL. Autorzy zwracają również uwagę na istotną kwestię, którą należy rozważyć przy projektowaniu aplikacji: czy zysk na wydajności czasowej jest ważniejszy niż normalizacja bazy danych i reguły ACID, które zostają odrzucone przy wyborze bazy zorientowanej na dokumenty.

Materiał konferencyjny [4] zawiera porównanie wydajności wykonywania pojedynczych poleceń SQL, poleceń multi-SQL oraz przygotowanych instrukcji SQL. Do celów publikacji zalicza się wskazanie optymalnych warunków do użycia przygotowanych zapytań. Po przeprowadzeniu analizy otrzymanych wyników autorzy wskazali, że pod względem wydajności, przygotowane instrukcje SQL nie powinny być używane do prostych zapytań. Zalecane jest natomiast wykorzystanie zapytań standardowych oraz zapytań multi-SQL. Autorzy wskazali również, na konieczność porównania korzyści dla bezpieczeństwa aplikacji z kosztami wydajnościowymi związanymi z użyciem preapred statements.

Materiał konferencyjny [5] zawiera porównanie wydajności Entity Framework i NHibernate – dwóch najpopularniejszych szkieletów mapowania obiektowo-relacyjnego dla platformy .NET Framework. Autorzy wykonali testy dla dwóch silników bazodanowych (MS SQL Server i PostgreSQL) oraz różnych języków zapytań (wyrażenia lambda i LINQ dla Enity Framework oraz HQL i Criteria API dla NHibernate). Uzyskane wyniki zostały porównane z zapytaniami wykonani za pomocą standardowego sposobu wykorzystującego SqlClient. W podsumowaniu autorzy wskazują, że w ogólności lepszą wydajnością cechowały się zapytania wykonywane przy użyciu SqlClient, jakkolwiek różnice wydajnościowe pomiędzy dobrze zaprojektowanym ORM a standardowym sposobem wykonywania zapytań nie były znaczące. Uzyskane rezultaty badań zostały podsumowane stwierdzeniem, że nie da się na ich podstawie jednoznacznie wskazać rekomendowanego pod względem wydajności szkieletu ORM, a wybór powinien być dokonywany uwzględniając wiele innych kryteriów (takich jak wspierane silniki baz danych, wsparcie grupowania i funkcji agregujących).

3. Cel badań

Celem badań jest porównanie wydajności czasowej wykonywania zapytań za pomocą Entity Framework dla wybranych silników baz danych.

4. Uwzględniane technologie

Przy analizie porównawczej uwzględniono następujące silniki baz danych: Microsoft SQL Server, MySQL oraz PostgreSQL. Program testowy został napisany przy użyciu języka C#, na platformie .NET.

4.1. Język C#, platforma .NET

C# jest wysoko-poziomowym językiem programowania ogólnego przeznaczenia. Jest silnie typowany i zorientowany obiektowo [6].

Platforma .NET to bezpłatne i otwartoźródłowe oprogramowanie pozwalające tworzyć i uruchamiać aplikacje napisane m. in. w języku C# [7].

Kod w C# jest kompilowany do kodu pośredniego (Intermediate Language – IL). Podczas wykonywania programu napisanego w języku C# środowisko uruchomieniowe (Common Language Runtime – CLR) prze-

prowadza kompilację Just-In-Time (JIT), aby przekonwertować kod IL do języka maszynowego [6].

4.2. Szkielet programistyczny Entity Framework Core

Entity Framework Core jest otwartoźródłowym szkieletem programistycznym mapowania obiektowo-relacyjnego dla platformy .NET. Narzędzie pozwala programiście pracować z bazą danych za pomocą obiektów i eliminuje konieczność pisania większości kodu odpowiadającego za dostęp do danych [8].

4.3. System zarządzania bazą danych Microsoft SQL Server Express

Microsoft SQL Server jest systemem zarządzania relacyjnymi bazami danych rozwijanym i dostarczanym przez firmę Microsoft. Wersją wolną do pobierania i rozpowszechniania jest SQL Server Express [9].

4.4. System zarządzania bazą danych MySQL

MySQL jest otwartoźródłowym systemem zarządzania bazami danych rozwijanym, dostarczanym i wspieranym przez Oracle Corporation [10]. Oprogramowanie jest dostępne na zasadach licencji GNU General Public License oraz płatnej licencji komercyjnej [10].

4.5. System zarządzania bazą danych PostreSQL

PostgreSQL jest darmowym i otwartoźródłowym systemem zarządzania bazami danych [11]. Początki systemu sięgają 1986 roku i projektu POSTGRES na Uniwersytecie Kolifornijskim w Berkeley. Oprogramowanie jest aktywnie rozwijane od ponad 35 lat [11].

5. Badania

W celu zapewnienia wiarygodnych wyników, po każdym przeprowadzeniu testu baza danych była usuwana, tworzona i wypełniana rekordami na nowo.

5.1. Środowisko testowe

Parametry komputera osobistego, na którym przeprowadzono eksperyment przedstawiono w Tabeli 1.

Tabela 1: Parametry komputera użytego do przeprowadzonego eksperymentu

Procesor	AMD Ryzen 7 4700U
Pamięć RAM	32GB 2666MHz
System operacyjny	Windows 11 Home 64-bitowy

W Tabeli 2 przedstawiono wersje i edycje systemów zarządzania relacyjnymi bazami danych użytych do przeprowadzenia eksperymentu.

Tabela 2: Wersje i edycje systemów zarządzania bazami danych użytych do przeprowadzonego eksperymentu

System zarządzania relacyjną bazą danych	Edycja	Wersja
Microsoft SQL Server 2022	Express Edition	16.0.1050.5
MySQL	Community Server - GPL	8.0.32
PostgreSQL	-	15.2

Wersje użytego oprogramowania, narzędzi i wybranych pakietów zostały przedstawione w Tabeli 3.

Tabela 3: Pozostałe wersje wykorzystanego oprogramowania

Program/narzędzie/pakiet	Wersja
C#	10.0
.NET	6.0
Microsoft.EntityFrameworkCore.Tools	7.0.5
Microsoft.EntityFrameworkCore.SqlServer	7.0.5
MySql.EntityFrameworkCore	7.0.2
Npgsql.EntityFrameworkCore.PostgreSQL	7.0.4

5.2. Scenariusze

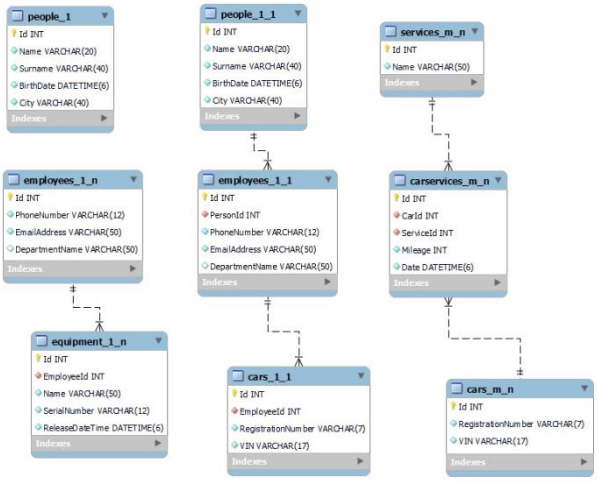
Scenariusze testowe obejmowały wykonanie zapytań realizujących operację odczytu, zapisu, aktualizacji i usuwania dla 1 000, 5 000, 10 000 i 100 000 rekordów w następujących przypadkach:

1. Zapytanie dotyczące jednej tabeli.
2. Zapytanie dotyczące trzech tabel (połączonych relacjami 1:1).
3. Zapytanie dotyczące dwóch tabel (połączonych relacjami 1:n).
4. Zapytanie dotyczące trzech tabel (połączonych relacjami 1:n w taki sposób, że rozbijają one relację m:n).

Każda operacja została wykonana 10 razy dla danego przypadku i liczby rekordów.

5.3. Struktura bazy danych

Do testów wykorzystano bazę danych przechowującą wygenerowane informacje na temat hipotetycznego przedsiębiorstwa. Diagram związków encji (ERD) przedstawiony jest na Rysunku 1.



Rysunek 1: Schemat ERD bazy danych.

Tabele Cars_1_1 i Cars_m_n są przykładem zduplikowania tabel dotyczących tych samych informacji. Specyficzna struktura bazy danych ma swoje uzasadnienie w obecności kluczy obcych. Gdyby tabela przechowująca informacje o samochodach (Cars) była połączona jednocześnie z tabelą przechowującą informacje o przeprowadzonych serwisach (CarServices) oraz tabelą przechowującą informacje o pracownikach (Em-

ployees) usunięcie rekordu z tabeli Cars spowodowałoby kaskadowe usunięcie rekordów z tabeli CarsServices i usunięcie rekordów albo zastąpienie wartościami null pól tych rekordów w tabeli Employees. Taki stan rzeczy powodowałby, że wyniki nie byłyby miarodajne.

Początkową liczbę rekordów dla poszczególnych tabel przedstawiono w Tabeli 4.

Tabela 4: Początkowe liczby rekordów w poszczególnych tabelach bazy danych

Tabela	Początkowa liczba rekordów
People_1	5 000 000
Cars_1_1	5 000 000
Employees_1_1	5 000 000
People_1_1	5 000 000
Employees_1_n	5 000 000
Equipment_1_n	15 000 000
Cars_m_n	5 000 000
CarServices_m_n	19 998 419
Services_m_n	17

6. Wyniki

Na przypadek testowy składają się 4 parametry: silnik bazy danych, struktura w bazie danych, liczba rekordów oraz rodzaj operacji (odczyt, zapis, aktualizacja, usuwanie). Dla każdego przypadku testowego uzyskano 10 wyników, które następnie zostały odfiltrowane za pomocą metody opierającej się na rozstępie międzykwartylowym. Oczyszczenie danych polegało na odrzuceniu wartości odstających, to znaczy takich, które nie spełniają warunku przedstawionego we wzorze (1)

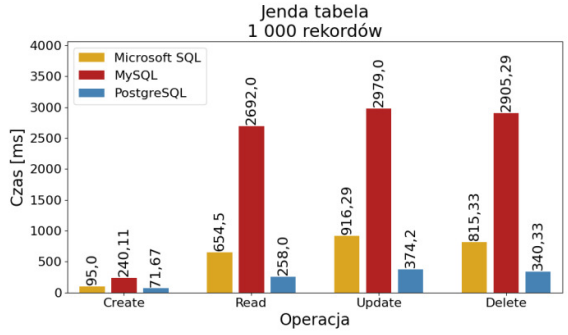
$$Q_1 - 1{,}5 * IQR \leq x \leq Q_3 + 1{,}5 * IQR \qquad (1)$$

gdzie x jest pojedynczym wynikiem, Q_1 jest pierwszym kwartylem obliczonym na podstawie wyników dla danego przypadku testowego, Q_3 jest trzecim kwartylem, a $IQR = Q_3 - Q_1$ jest rozstępem międzykwartylowym.

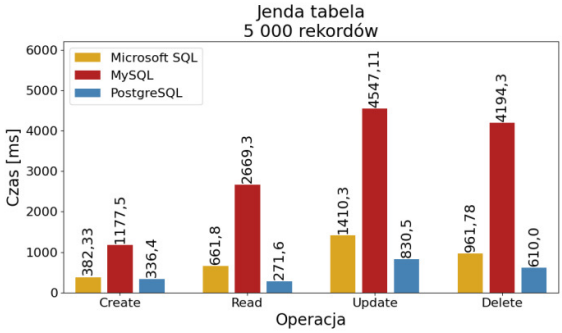
W celu obrazowego przedstawienia rozbieżności w czasach wykonania różnych operacji dla 3 silników baz danych utworzono wykresy słupkowe. Zaprezentowane na nich wyniki są średnimi ucinanymi z wyników jednostkowych dla danego przypadku testowego.

6.1. Operacje dla jednej tabeli

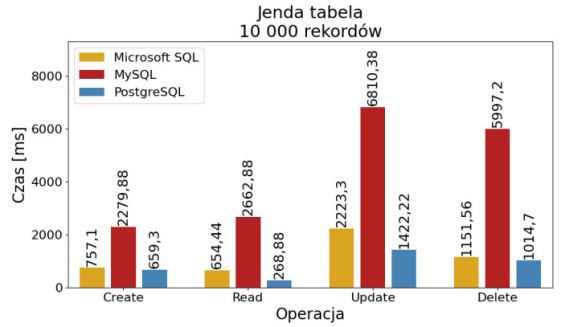
Średnie czasy wykonania operacji CRUD dotyczących jednej tabeli, dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 2-5.



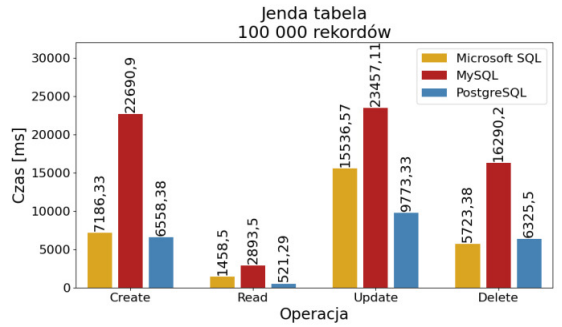
Rysunek 2: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z jednej tabeli.



Rysunek 3: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z jednej tabeli.



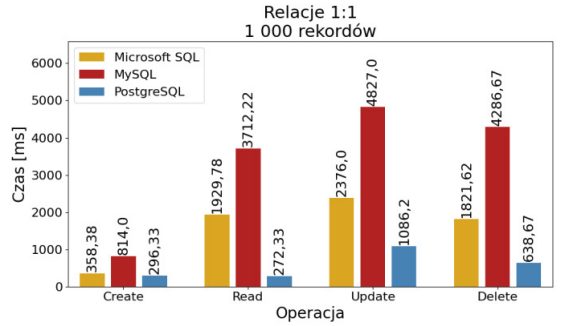
Rysunek 4: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z jednej tabeli.



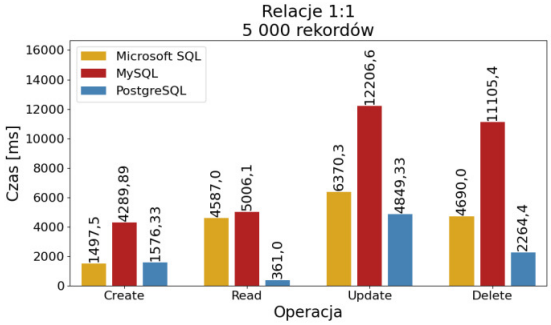
Rysunek 5: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z jednej tabeli.

6.2. Operacje dla trzech tabel połączonych relacjami 1:1

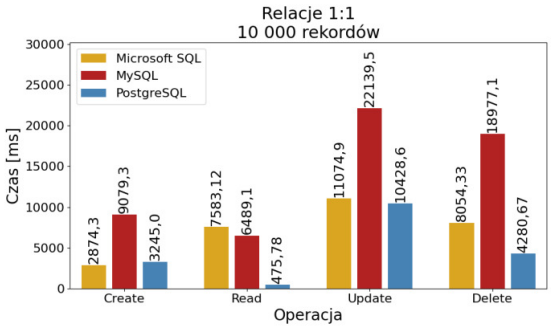
Średnie czasy wykonania operacji CRUD dotyczących trzech tabel połączonych relacjami 1:1, dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 6-9.



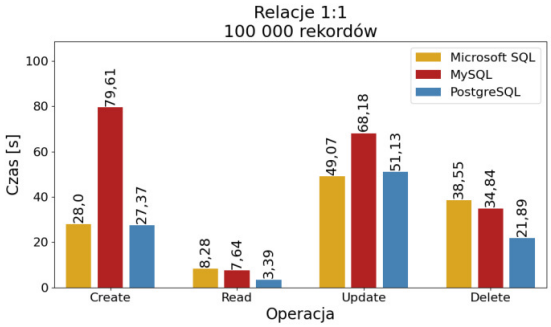
Rysunek 6: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z trzech tabel połączonych relacjami 1:1.



Rysunek 7: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z trzech tabel połączonych relacjami 1:1.



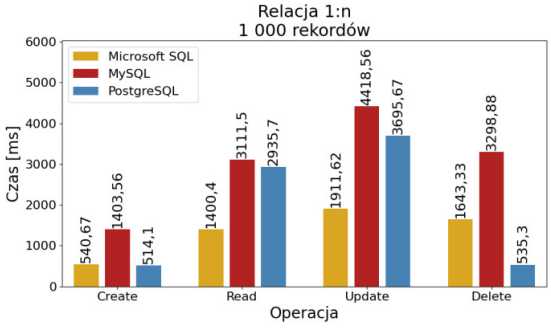
Rysunek 8: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z trzech tabel połączonych relacjami 1:1.



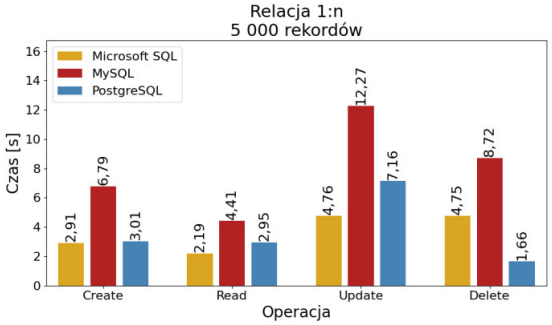
Rysunek 9: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z trzech tabel połączonych relacjami 1:1.

6.3. Operacje dla dwóch tabel połączonych relacjami 1:n

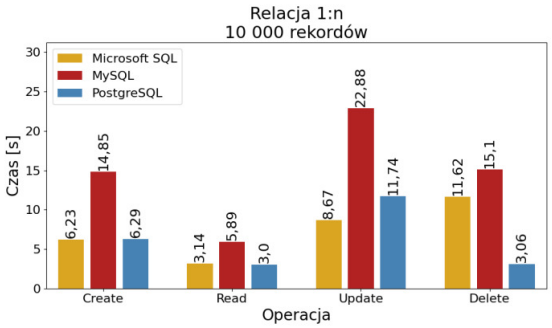
Średnie czasy wykonania operacji CRUD dotyczących dwóch tabel połączonych relacją 1:n, dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 10-13.



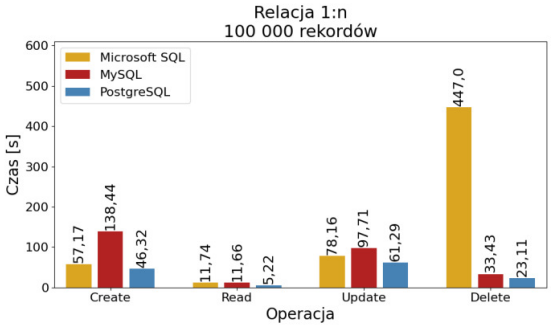
Rysunek 10: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z dwóch tabel połączonych relacją 1:n.



Rysunek 11: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z dwóch tabel połączonych relacją 1:n.



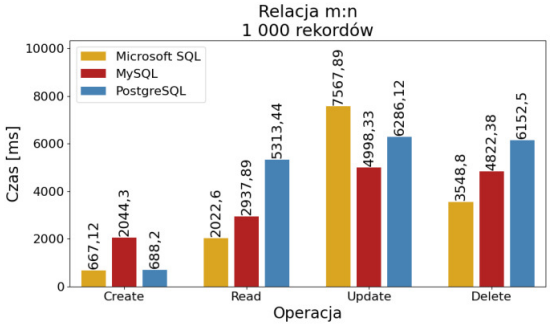
Rysunek 12: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z dwóch tabel połączonych relacją 1:n.



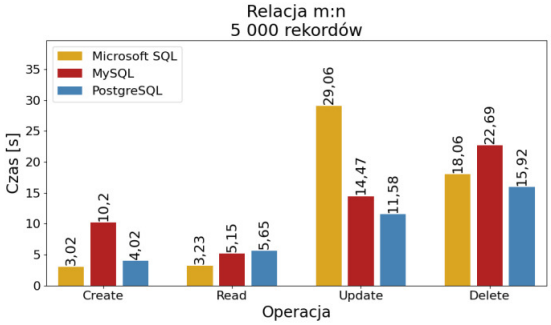
Rysunek 13: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z dwóch tabel połączonych relacją 1:n.

6.4. Operacje dla trzech tabel rozbijających relację m:n

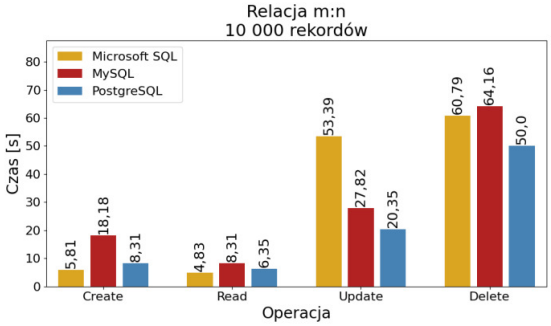
Średnie czasy wykonania operacji CRUD dotyczących trzech tabel połączonych relacjami 1:n (w sposób który rozbija relację m:n), dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 14-17.



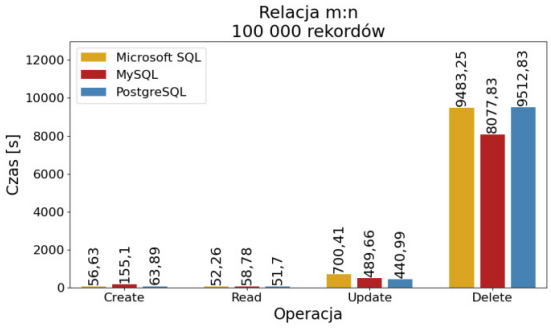
Rysunek 14: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z trzech tabel rozbijających relację m:n.



Rysunek 15: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z trzech tabel rozbijających relację m:n.



Rysunek 16: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z trzech tabel rozbijających relację m:n.



Rysunek 17: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z trzech tabel rozbijających relację m:n.

7. Wnioski

Dla operacji wykonywanych na rekordach jednej tabeli można wskazać, że najgorsze wyniki wydajności czasowej, niezależnie od liczby rekordów i rodzaju operacji, uzyskał silnik MySQL. Silnikiem z najlepszymi wynikami (poza przypadkiem testu operacji usuwania dla 100 000 rekordów) był PostgreSQL.

W przypadku trzech tabel połączonych relacjami 1:1 przewaga PostgreSQL jest nadal widoczna, przy czym wystąpiły 3 przypadki, w których ustąpił on technologii Microsoft SQL Server (operacje tworzenia dla 5 000 i 10 000 rekordów oraz operacja aktualizacji dla 100 000 rekordów). Silnik MySQL w większości przypadków osiągnął najgorsze wyniki – wyjątkami są tutaj operacje odczytu dla 10 000 rekordów oraz operacje odczytu i usuwania dla 100 000 rekordów, gdzie najwolniejszy był Microsoft SQL Server.

Rozpatrując wyniki czasowe dla dwóch tabel połączonych relacją 1:n można stwierdzić, że silnik MySQL uzyskał najgorsze wyniki poza przypadkami testowania operacji odczytu i usuwania dla 100 000 rekordów (najwolniejszy był wtedy Microsoft SQL Server). Trudno o wskazanie innych tendencji, ponieważ dla danego rodzaju operacji w zależności od liczby rekordów najszybszy okazywał się PostgreSQL albo Microsoft SQL Server.

Przypadek trzech tabel rozbijających relację m:n jest najbardziej zróżnicowany pod względem wyników osiąganych przez poszczególne silniki. Dla operacji tworzenia danych najszybszym okazał się Microsoft SQL Server, a najwolniejszym silnik MySQL. Dla operacji odczytu (poza testem dla 100 000 rekordów) najszybszy okazał się Microsoft SQL Server. Biorąc pod uwagę operację aktualizacji najwolniejszy okazał się Microsoft SQL Server, a najszybszy (poza testem dla 1 000 rekordów) PostgreSQL. Na temat operacji usuwania danych trudno jest wskazać konkretną tendencję.

Na podstawie przeprowadzonych badań można stwierdzić, że tworząc aplikację, w której do wykonywania zapytań używany jest Enitity Framework, a w strukturze bazy danych dominują pojedyncze tabele lub tabele połączone relacjami 1:1, jako silnik baz danych warto zastosować PostgreSQL, a unikać MySQL. Jeśli natomiast w strukturze bazy danych dominują tabele połączone relacjami 1:n, które nie rozbijają relacji m:n, kierując się wydajnością czasową należy unikać silnika MySQL, natomiast wybrać spośród technologii Microsoft SQL Server i PostgreSQL (zależnie od dominującej operacji na danych jaka będzie wykonywana przez system informatyczny). Jeżeli zaś w strukturze bazy danych dominują relacje m:n rozbite na dwie relacje 1:n wybór silnika bazy danych będzie zależał od rodzaju najczęściej wykonywanej operacji i liczby rekordów, dla których ta operacja będzie dotyczyła.

Literatura

[1] S. Cvetković, D. Janković, A Comparative Study of the Features and Performance of ORM Tools in a .NET Environment, Objects and Databases, Lecture Notes in Computer Science 6348 (2010) 147-158 https://doi.org/10.1007/978-3-642-16092-9_14.

[2] D. Zmaranda, L. Pop-Fele, C. Gyorödi, R. Gyorödi, G. Pecherle, Performance comparison of CRUD methods using NET object relational mappers: A case study, International Journal of Advanced Computer Science and Applications 11(1) (2020) 55-65, https://dx.doi.org/10.14569/ijacsa.2020.0110107.

[3] C.A. Győrödi, D.V. Dumşe-Burescu, D.R. Zmaranda, R.Ş. Győrödi, G.A. Gabor, G.D. Pecherle, Performance analysis of nosql and relational databases with couchdb and mysql for application's data storage, Applied Sciences (Switzerland) 10(23) (2020) 1-21 https://dx.doi.org/10.3390/app10238524.

[4] T. Seser, V. Plestina, F. Marjanica, Performance analysis of SQL prepared statements in CRUD operations. 7th International Conference on Smart and Sustainable Technologies, SpliTech (2022), https://dx.doi.org/10.23919/SpliTech55088.2022.9854303.

[5] A. Gruca, P. Podsiadło, Performance Analysis of .NET Based Object–Relational Mapping Frameworks, Beyond Databases, Architectures and Structures, Communications in Computer and Information Science 424 (2014) 40-49, https://dx.doi.org/10.1007/978-3-319-06932-6_5.

[6] A tour of the C# language, https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/, [17.06.2023]

[7] What is .NET? Introduction and overview, https://learn.microsoft.com/en-us/dotnet/core/introduction, [17.06.2023]

[8] Entity Framework Core, https://learn.microsoft.com/en-us/ef/core/, [17.06.2023]

[9] Microsoft SQL Server 2022 Licensing guide, https://download.microsoft.com/download/9/3/d/93d32de6-f268-45ed-ba25-2f9a6756b6af/SQL_Server_2022_Licensing_guide.pdf, [17.06.2023]

[10] What is MySQL?, https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, [17.06.2023]

[11] PostgreSQL - about https://www.postgresql.org/about/, [17.06.2023]