JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 32 (2024) 231–238
Received: 13 June 2024
Accepted: 5 July 2024

Comparative analysis of database mapping frameworks available in NuGet Manager

Analiza porównawcza szkieletów mapowania bazodanowego dostępnych w managerze NuGet

Paweł Karabowicz, Maciej Klimiuk*, Małgorzata Plechawska-Wójcik

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

Abstract
Every year amount of data significantly grows. This causes a need of tools capable of processing data as quickly as possible. The aim of this article is to perform a comparative analysis of ORM (Object Relational Mapping) tools available in the NuGet manager. The comparison of tools was carried out on two relational database management systems, PostgreSQL and MySQL. Each tool performed eight DML operations, including SELECT, INSERT and UPDATE operations. The operations were repeated 1, 10, 100, 1000, and 10,000 times, respectively. The execution time of each query was measured using the system StopWatch class available in the .NET environment. The results showed that the best ORM tool turned out to be Dapper.

Keywords: performance analysis; ORM; Dapper; OrmLite

Streszczenie
Każdego roku ilość danych znacznie rośnie. Powoduje to konieczność posiadania narzędzi zdolnych do jak najszybszego przetwarzania danych. Celem tego artykułu jest przeprowadzenie analizy porównawczej narzędzi ORM (Object Relational Mapping) dostępnych w menedżerze NuGet. Porównanie narzędzi zostało przeprowadzone na dwóch systemach zarządzania relacyjnymi bazami danych, PostgreSQL i MySQL. Każde narzędzie wykonało osiem operacji DML, w tym operacje SELECT, INSERT i UPDATE. Operacje były powtarzane odpowiednio 1, 10, 100, 1000 i 10 000 razy. Czas wykonania każdego zapytania był mierzony za pomocą klasy systemowej StopWatch dostępnej w środowisku .NET. Wyniki pokazały, że najlepszym narzędziem ORM okazał się Dapper.

Słowa kluczowe: analiza wydajności; ORM; Dapper; OrmLite

*Corresponding authors
Email address: **maciej.klimiuk@pollub.edu.pl** (M. Klimiuk)

1. Wstęp

Współczesny świat potrzebuje ogromnych zasobów danych do prawidłowego funkcjonowania. Większość instytucji, zarówno z sektora prywatnego jak i publicznego, do ich przechowywania wykorzystuje bazy danych. Bazy danych są standardowym sposobem przechowywania danych i korzysta z nich zdecydowana większość aplikacji. Zauważono, że potrzebne są narzędzia, które poprawiają procesy wykorzystania baz danych. Z tego powodu zaczęto wprowadzać rozwiązania mające na celu ułatwienie, a zarazem przyspieszenie procesów wykonywanych podczas połączenia z bazą danych.

Z uwagi na rozwój języków programowania oraz środowisk programistycznych postanowiono usprawnić, a jednocześnie uprościć, sposób zarządzania bazą danych z poziomu pisanego kodu. Stworzono szkielety mapowania bazodanowego (ORM - Object Relational Mapping), które pozwalają zdefiniować dane oraz nimi manipulować bez potrzeby korzystania z zewnętrznych programów do zarządzania bazami danych [1]. Dzięki mapowaniu klas na tabele bazodanowe oraz obiektów na dane, możliwe jest komunikowanie się z bazą danych przy minimalnej znajomości języka SQL. Dla różnych języków czy środowisk narzędzia ORM powstały, aby w jak największym stopniu ułatwić programistom pracę z bazami danych. Z uwagi na duży wybór tego rodzaju szkieletów, warto je sprawdzić i przetestować.

Dodatkowo, w obliczu rosnącej roli danych, istotne staje się wykorzystanie baz danych nie tylko w sferze przechowywania, ale także jako narzędzia ułatwiającego szybkie i efektywne zarządzanie informacjami. Współczesne podejścia do baz danych zdają się kłaść nacisk na prostotę obsługi, dostępność oraz integrację z nowoczesnymi narzędziami programistycznymi. W miarę jak technologia ewoluuje, tak samo ewoluują i strategie zarządzania danymi, dążąc do pełnej optymalizacji procesów związanych z bazami danych.

Najbardziej powszechnym rodzajem baz danych są bazy relacyjne, które korzystają z języka SQL [2]. Do podstawowych procesów, które są wywoływane przez język SQL należy definiowanie struktury danych (DDL – Data Definition Language) oraz manipulowanie nimi (DML – Data Manipulation Language). W zależności od rodzaju bazy danych SQL, stosowane są również inne systemy zarządzania bazą danych (SZBD). Narzędzia ORM tworzone są do pracy właśnie z bazami relacyjnymi. Z uwagi na duży wybór tego rodzaju technologii warto sprawdzić te najpopularniejsze.

W kontekście narzędzi ORM wykorzystywanych w środowisku .NET, manager pakietów NuGet odgrywa

kluczową rolę. Pozwala bezproblemowo pobrać, a następnie dodać do projektu wybrany szkielet lub inne tego typu narzędzie. Kolejnym atutem jest konfigurowanie środowiska z zainstalowanymi bibliotekami. Dzięki managerowi NuGet zarządzanie szkieletami mapowania bazodanowego jest o wiele efektywniejsze i prostsze.

Na czas wykonania zapytań ma wpływ wiele kryteriów, do których z pewnością należy typ przeprowadzanej operacji oraz złożoność zapytań. Drugi parametr wynika z generowania zapytań na podstawie napisanego kodu. Dzieje się tak za pomocą konfiguracji lub adnotacji, które łączą właściwości klas z kolumnami baz danych, pozwalając na ich mapowanie. Z uwagi na bardzo krótki czas wykonywania jednej instrukcji, najlepszym sposobem na zauważenie różnicy jest wywołanie wielu zapytań jednocześnie, dlatego kolejnym ważnym aspektem jest liczba wykonanych zapytań. Na potrzeby badania zostały wybrane jedne z popularniejszych systemów zarządzania bazami danych. Z kolei na wybór szkieletów ORM wpłynęła ich popularność w managerze NuGet.

Celem badania jest ocena wybranych typów szkieletów mapowania bazodanowego oraz znalezienie najbardziej optymalnego ORM dostępnego w managerze pakietów NuGet. Do analizy porównawczej wybrano następujące szkielety: Entity Framework Core, NHibernate, Dapper oraz OrmLite. Realizację badania przeprowadzono z wykorzystaniem 16 scenariuszy badawczych zawierających operacje takie jak SELECT, UPDATE oraz INSERT. Najwięcej scenariuszy przewidziano dla operacji wykonywania zapytań, ponieważ uwzględniono zapytania o różnym poziomie złożoności. Analizowano czas realizacji poszczególnych operacji wykonywanych w seriach po 1, 10, 100, 1000 i 10000 powtórzeń.

2. Przegląd literatury

W niniejszym rozdziale została opisana literatura zbliżona tematycznie do zakresu artykułu. Główne pozycje zawierają istotne informacje związane z wydajnością, optymalizacją i sposobem tworzenia zapytań dla najbardziej powszechnych szkieletów mapowania obiektowo-relacyjnego.

Otwarcie przeglądu literaturowego stanowi artykuł pochodzący z czasopisma Journal of Computer Science Institute zatytułowany "Badanie wydajności zapytań SQL w wybranym systemie informatycznym" autorstwa Krzysztofa Barczaka [3]. Artykuł skupia się na optymalizacji zapytań SQL, do wykonania których, posłużył silnik MS SQL. Wymienionymi sposobami w pracy, na zoptymalizowanie zapytań, jest zmniejszenie liczby wybranych kolumn w operacjach typu SELECT, unikanie niepotrzebnego wykorzystania klauzuli ORDER BY, używanie indeksów na kolumnach oraz użycie odpowiedniego rodzaju złączenia tabel w celu przyśpieszenia wykonania zapytania.

Następna praca o tytule "Analiza możliwości optymalizacji zapytań SQL", której autorem jest Piotr Rymarski, zestawia ze sobą najbardziej popularne systemy baz danych oraz wskazuje sposób optymalizacji zapytań [4]. Praca opisuje każdy z systemów baz danych osobno, wymieniając ich zalety oraz wady. Podobnie jak w pierwszym artykule, optymalizacja dotyczy głównie operacji SELECT, JOIN, indeksowania oraz klauzuli GROUP BY, niemniej dodatkowo opisany został sposób optymalizacji klauzuli UNION, pozbycia się redundantnego sortowania i filtrowania przy użyciu operatora SARGable i Non-SARGable. Autor wskazuje, że najbardziej efektywnym sposobem na zmniejszenie czasu wykonania zapytania jest ograniczenie zestawu zwracanych danych.

Kolejna praca naukowa zatytułowana "Analiza wydajności baz danych utworzonych w zwirtualizowanym i skonteneryzowanym środowisku", której autorem jest Zygmunt Łata, ma na celu odpowiedź na hipotezę, który sposób użycia systemu bazodanowego jest najwydajniejszy - czy ten przy pomocy maszyny wirtualnej, czy ten z wykorzystaniem kontenerów Docker [5]. W ramach badań dokonano pomiaru czasu wykonywania zapytań INSERT, UPDATE, DELETE oraz SELECT. Każdy test został powtórzony 100 razy. W porównaniu wykorzystano cztery rodzaje systemów bazodanowych, którymi były MySQL, PostgreSQL, MS SQL Server oraz Oracle XE. Okazało się, że konteneryzacja znacząco przewyższa maszyny wirtualne pod względem efektywności. Czas wykonywania zapytań dla konkretnego systemu bazodanowego zainstalowanego na maszynie wirtualnej był średnio dwukrotnie dłuższy niż dla kontenera zawierającego ów system. Najszybszym systemem bazodanowym, czyli takim, który przetworzył zapytania w jak najkrótszym czasie był PostgreSQL, był on średnio dwukrotnie szybszy od drugiego w kolejności najwydajniejszego systemu bazodanowego.

Drugi artykuł o tytule "A Comparative Study of the Features and Performance of ORM Tools in a .NET Environment" analizuje znaczenie narzędzi ORM w procesie tworzenia systemów informatycznych w środowisku .NET [6]. Autorzy artykułu, Cvetković i Janković, skupiają się na dwóch najpopularniejszych szkieletach ORM w .NET: Entity Framework i NHibernate. Przeprowadzają oni szczegółową analizę porównawczą tych narzędzi, oceniając ich funkcje, sposoby użycia oraz wydajność z perspektywy rozwoju oprogramowania. W celu dokładnego pomiaru wydajności, autorzy przeprowadzają siedem testowych zapytań, z których każde jest wykonywane sto razy. Następnie obliczony jest średni czas wykonania tych zapytań. Wyniki eksperymentów wykazują, że Entity Framework uzyskał lepsze wyniki wydajnościowe niż NHibernate we wszystkich zapytaniach, z wyjątkiem jednego. Na końcu artykułu omawiane są przyczyny, dla których technologia ORM nie jest jeszcze powszechnie stosowana, mimo jej potencjalnych zalet.

Praca naukowa "An evaluation of .NET Object-Relational Mappers in relational databases Entity Framework Core and Dapper" autorstwa Maximiliana Myllyaho Forsberga [7] analizuje dwa narzędzia ORM w kontekście ich wydajności oraz efektywności implementacji w aplikacjach .NET. Autor porównuje Entity Framework Core oraz Dapper, mierząc czas wykonania, alokację pamięci oraz złożoność kodu dla obu narzędzi. Badania pokazują, że Dapper zazwyczaj osiąga lepszą wydajność we

wszystkich kategoriach, co czyni go preferowanym wyborem, jeśli użytkownikowi zależy na najkrótszym czasie wykonania zapytań. Z kolei Entity Framework Core oferuje lepsze wsparcie dla programistów w fazie implementacji, co może korzystnie wpływać na efektywność rozwoju aplikacji.

Ostatnią pozycją jest praca pod tytułem "Analiza porównawcza wydajności czasowej zapytań do baz danych w języku C#" autorstwa Tomasza Nowickiego oraz Sebastiana Tomczaka [8]. Autorzy porównują ze sobą zapytania napisane w języku SQL jako łańcuch znaków, najpopularniejszy ORM dostępny dla platformy .NET, czyli Entity Framework oraz sparametryzowane zapytanie SQL. Z badań wynika, że wykorzystanie ORM jest najwolniejszym sposobem na wykonywanie zapytań, natomiast najszybszym sposobem jest użycie sparametryzowanego zapytania SQL. Entity Framework jest kilkukrotnie wolniejszy od pozostałych dwóch sposobów co wskazuje na znaczny brak wydajności szkieletów mapowania bazodanowego.

Podsumowując, prace innych autorów skupiają się głównie na porównaniu różnych narzędzi ORM. Szczególną uwagę zwraca się na wydajność, aby na jej podstawie móc określić najlepszy szkielet mapowania bazodanowego w środowisku .NET. Inną poszukiwaną odpowiedzią jest jak najbardziej optymalny sposób pisania zapytań SQL. Oprócz tego zwracana jest także uwaga na wsparcie techniczne danej technologii. W niniejszym artykule, skupiono się na zestawieniu szkieletów mapowania bazodanowego dla konkretnego managera pakietów. Z uwagi na powtarzalność wśród badanych narzędzi wybrana literatura naukowa pozytywnie wpływa na przeprowadzenie badania. Dodatkowo informacje zawarte w pierwszym artykule pozwalają przygotować zoptymalizowane zapytania do testowania.

3. Metodyka badań

Przeprowadzanie analizy porównawczej szkieletów mapowania bazodanowego podzielone zostało na dwie podstawowe części. Pierwszą z nich było przygotowanie środowiska badań, natomiast druga część odnosi się do samej realizacji badań. Sposób, w jaki wyniki zostały opracowane, był zależny od rodzaju operacji. Częścią wspólną było otwarcie oraz zamknięcie kontekstu do bazy danych przy każdej iteracji pętli. Dodatkowo, aby wyeliminować odroczenie inicjalizacji obiektu do czasu, aż będzie potrzebny (lazy loading), wynik z każdego zapytania typu SELECT był konwertowany na listę. Program mierzył czas od pierwszego otwarcia kontekstu aż do wyświetlenia wartości w konsoli. Dla operacji typu UPDATE oraz INSERT czas mierzony był zarówno od otwarcia jak i zamknięcia kontekstu. Dla operacji aktualizacji, pomiar czasu był wstrzymywany na okres wybrania danych z bazy oraz odpowiedniej modyfikacji danych. Gdy dane do aktualizacji były odpowiednio przygotowane, ponownie pomiar został włączony na okres aktualizacji rekordów w bazie danych i ich zapisie. W przypadku operacji dodawania rekordów do bazy danych, w pomiarze znalazła się operacja dodania obiektu do bazy danych oraz zapisanie zmian. Pominięty został czas, gdy w kodzie został utworzony nowy obiekt, który miał zostać dodany do bazy danych.

3.1. Obiekty badań

Wybrane obiekty użyte podczas badania podzielone są na grupę badanych szkieletów programistycznych oraz baz danych, na których te szkielety będą testowane. Do pierwszej grupy wybrano:

- Entity Framework Core,
- Dapper,
- NHibernate,
- OrmLite.

Wszystkie z wymienionych szkieletów są popularnymi narzędziami służącymi do mapowania bazodanowego ORM, z czego każdy ma swoje cechy charakterystyczne.

Szkielet Entity Framework Core jest oficjalnym narzędziem ORM dostarczanym przez Microsoft. Zapewnia szeroki zakres funkcji zarządzających bazą danych. Znacznie usprawnia obsługę wielu operacji za pomocą rozszerzenia LINQ [9].

Narzędzie Dapper zostało stworzone przez Stack Overflow. Dostarcza funkcji pozwalających przejąć kontrolę nad zapytaniami, co pozwala skupić się na wydajności i szybkości działania aplikacji [10].

NHibernate jest narzędziem zainspirowanym Hibernate dla języka Java. Dostarcza zaawansowane funkcje mapowania obiektowo-relacyjnego. Z uwagi na to, że jest starszą biblioteką ORM dla .NET, jego konfiguracja jest bardziej zaawansowana od wymienionych wyżej szkieletów [11].

Ostatnim badanym narzędziem jest OrmLite. Jest to lekki ORM, który cechuje się wysoką elastycznością w tworzeniu zapytań oraz szybkością w ich wykonaniu [12].

Poza szkieletami programistycznymi do mapowania bazodanowego do przeprowadzenia badań zostały użyte następujące systemy zarządzania relacyjnymi bazami danych:

- PostgreSQL,
- MySQL.

Oba wyróżnione systemy oferują szeroki zakres funkcji do zarządzania relacyjnymi bazami danych. PostgreSQL charakteryzuje się otwartym kodem źródłowym oraz dużym wyborem funkcji, która pozwala zachować elastyczność przy korzystaniu z bazy danych. Z kolei MySQL wyróżnia się wydajnością i skalowalnością. Sprawia to, że często jest wykorzystywany przy projektach open source oraz aplikacjach webowych.

3.2. Stanowisko testowe

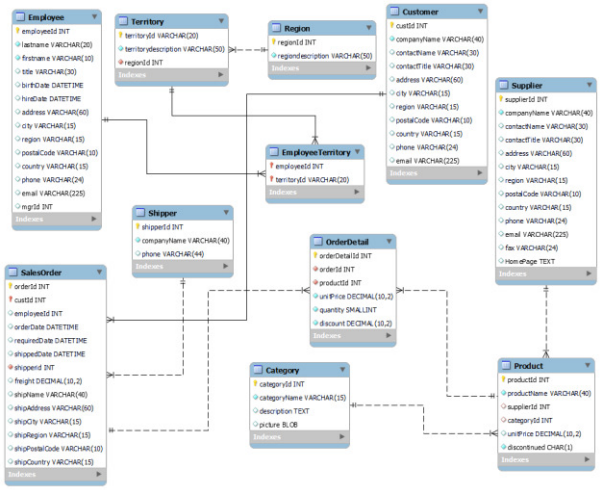
Stanowiskiem testowym przy wykonywaniu analizy porównawczej jest komputer marki Apple, którego główną cechą rozpoznawczą jest system operacyjny macOS dostarczany również przez tę samą firmę. Pozostałe właściwości komputera przedstawione zostały w Tabeli 1.

Tabela 1: Konfiguracja komputera użytego do testów

Właściwość	Opis
System operacyjny	macOS Sonoma 14.4
Model laptopa	MacBook Pro 14-inch, 2021
Procesor	Apple M1 Pro
Pamięć RAM	16GB
Dysk twardy	Macintosh HD

3.3. Przebieg testów

Do przygotowania środowiska została wykorzystana publicznie dostępna relacyjna baza danych Northwind [13]. Odpowiednio przygotowany plik z bazą danych został zaimportowany do systemów zarządzania relacyjnymi bazami danych MySQL oraz PostgreSQL. Skrypt SQL zawierał polecenia DDL oraz DML, które stworzyły bazę danych (Rysunek 1) oraz dodały wstępne wartości do odpowiednich tabel.



Rysunek 1: Schemat bazy danych Northwind.

Drugą część środowiska stanowi aplikacja do przeprowadzenia testów, która została wykonana przez autorów w środowisku .NET w języku C#. Aplikacja zawierała szkielety programistyczne omawiane podczas badania. Każdy ze szkieletów programistycznych wymagał zainstalowania odpowiednich bibliotek oraz odpowiedniego połączenia do bazy. Narzędzia Entity Framework oraz NHibernate wymagały przygotowania odpowiednich klas mapujących, które odwzorowały strukturę bazy danych na odpowiednie obiekty. Dodatkowo narzędzie NHibernate wymagało przygotowania odpowiednich klas mapujących, które definiowały reguły mapowania klas na odpowiednie tabele bazodanowe, znajdujące się w bazie danych. Z perspektywy ilości kodu to właśnie przygotowanie programu ze szkieletem NHibernate było najbardziej czasochłonne. Entity Framework Core umożliwia automatyczne wygenerowanie klas, co przyspieszyło implementowanie tego narzędzia. Z kolei narzędzia Dapper oraz OrmLite opierają się na języku SQL, co pozwala uniknąć mapowania na obiekty, ale wymaga znajomości tego języka. Dzięki temu ich implementacja przebiegła bezproblemowo. Następnie zostało przygotowanych osiem zapytań w języku SQL, które zostały sprawdzone pod względem poprawności działania. Ostatnią czynnością było odpowiednie przygotowanie zapytań dla testowanych narzędzi.

4. Scenariusze badawcze

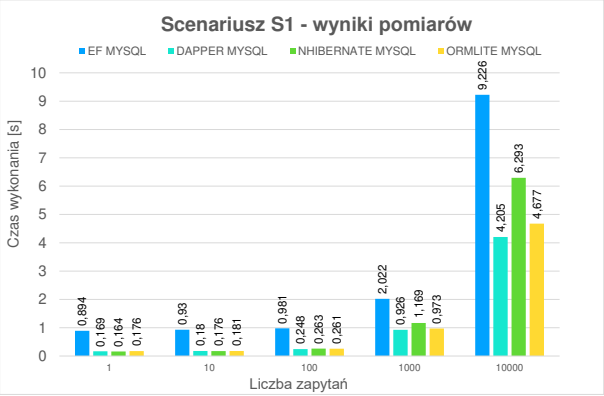
Na potrzeby wykonania testów wyznaczonych zostało osiem zapytań SQL z czego każdy został przetestowany na dwóch bazach danych. Dla lepszego zobrazowania różnic wynikających z liczby wykonanych zapytań każdy przypadek został wykonany w kilku seriach. Dla badanych szkieletów programistycznych przygotowano następujące scenariusze badawcze:

1. Wyznaczenie nazwy produktu, zdjęcia, ceny jednostkowej oraz id produktu, gdzie cena jednostkowa jest większa niż średnia cena jednostkowa wszystkich produktów.
   a) MySQL – S1.
   b) PostgreSQL – S2.
2. Wyznaczenie imienia, imienia nazwiska pracownika oraz opisu regionu, w którym pracownik pracuje.
   a) MySQL – S3.
   b) PostgreSQL – S4.
3. Wyznaczenie nazwy firmy klienta, nazwiska pracownika oraz nazwy produktu.
   a) MySQL – S5.
   b) PostgreSQL – S6.
4. Wyznaczenie szczegółów zamówienia dla klienta o identyfikatorze równym 10, w tym identyfikatora zamówienia, daty zamówienia, daty wysyłki, kosztów frachtu, nazwy firmy klienta, imienia i nazwiska pracownika, identyfikatora produktu, nazwy produktu, ceny jednostkowej, ilości i rabatu.
   a) MySQL – S7.
   b) PostgreSQL – S8.
5. Wyznaczenie nazwy produktu i sumy zamówionych ilości każdego produktu, posortowane malejąco według sumy zamówień.
   a) MySQL – S9.
   b) PostgreSQL – S10.
6. Wyznaczenie nazwy firmy klienta, liczby unikalnych zamówień oraz całkowitego kosztu zamówień, po uwzględnieniu rabatów, posortowane malejąco według całkowitego kosztu zamówień.
   a) MySQL – S11.
   b) PostgreSQL – S12.
7. Aktualizacja ceny jednostkowej wszystkich produktów w kategorii o identyfikatorze równym 3, zwiększając ceny o 10%.
   a) MySQL – S13.
   b) PostgreSQL – S14.
8. Wstawienie nowego produktu do tabeli ‘Product’ z nazwą “Inserted product”, ‘supplierid’ równym 1, ‘categoryid’ równym 1, ceną jednostkową 1.00 oraz oznaczeniem, że produkt nie jest wycofany (‘discontinued’ = 0).
   a) MySQL – S15
   b) PostgreSQL – S16.

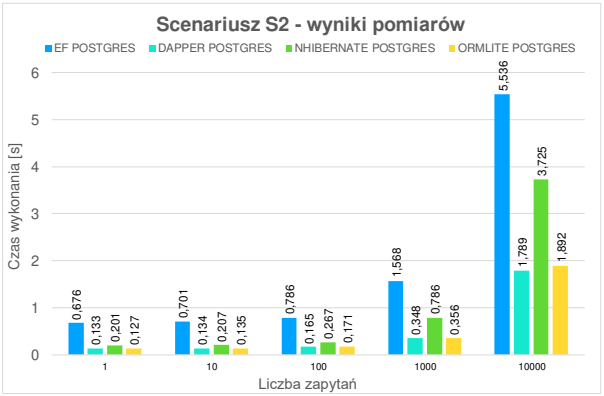
5. Wyniki badań

W poniższym rozdziale przedstawione zostały uzyskane wyniki badań poszczególnych scenariuszy dla badanych

szkieletów mapowania bazodanowego. Rysunki 2-17 prezentują wykresy słupkowe z wynikami badań dla scenariuszy S1-S16, gdzie danymi są czasy wykonania poszczególnych zapytań w określonej ilości razy.

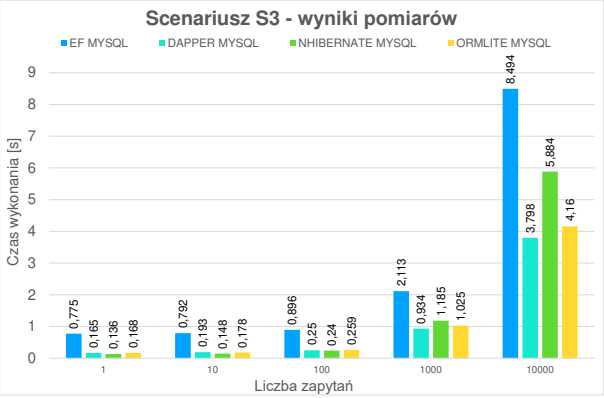


Rysunek 2: Wyniki pomiarów dla Scenariusza 1.

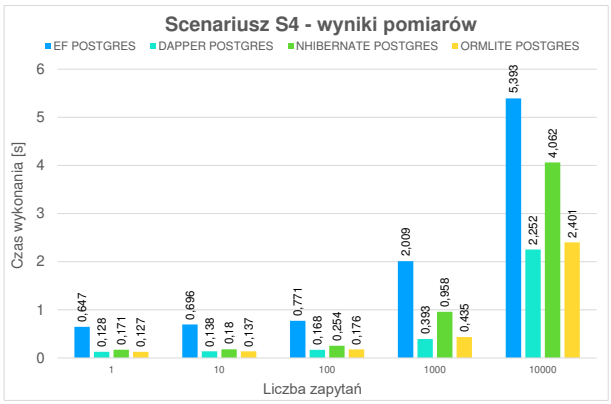


Rysunek 3: Wyniki pomiarów dla Scenariusza 2.

Rysunek 2 i 3 przedstawiają wyniki pomiarów kolejno systemów zarządzania bazami danych MySQL i PostgreSQL dla prostego zapytania opartego o jedną tabelę. Na każdym wykresie przestawiono wyniki dla czterech analizowanych szkieletów.

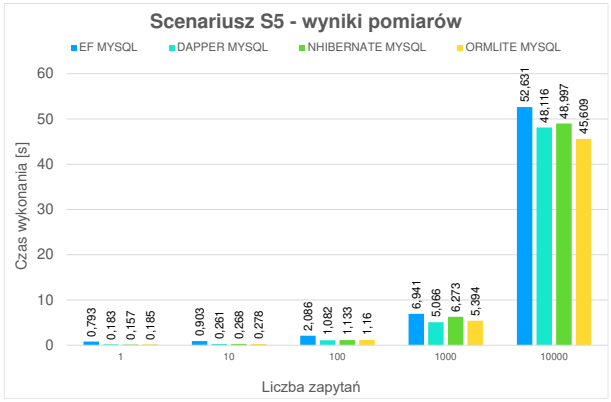


Rysunek 4: Wyniki pomiarów dla Scenariusza 3.

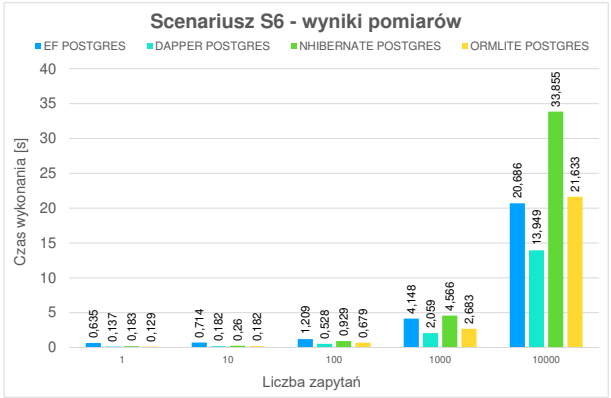


Rysunek 5: Wyniki pomiarów dla Scenariusza 4.

Na Rysunku 4 i 5 znajdują się wykresy z wynikami badań dla dwóch systemów zarządzania bazami danych MySQL oraz PostgreSQL. Podane rezultaty zostały otrzymane dla niezłożonego zapytania opartego o kilka tabel. Poszczególne kolory odnoszą się do badanych szkieletów ORM.

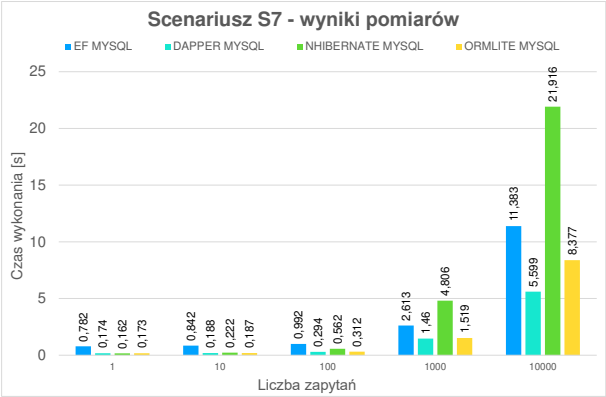


Rysunek 6: Wyniki pomiarów dla Scenariusza 5.

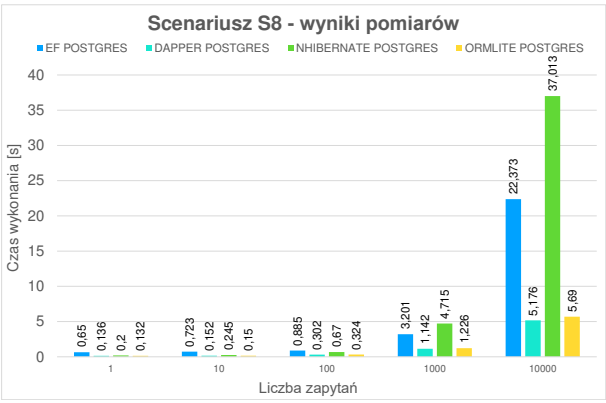


Rysunek 7: Wyniki pomiarów dla Scenariusza 6.

Rysunek 6 i 7 przedstawiają wyniki badań dla scenariusza S5 oraz S6. Ich motywem było przetestowanie zapytania wykorzystującego wiele kolumn z różnych tabel. Również jak poprzednie wykresy, wyniki zostały zobrazowane dla omawianych narzędzi ORM.



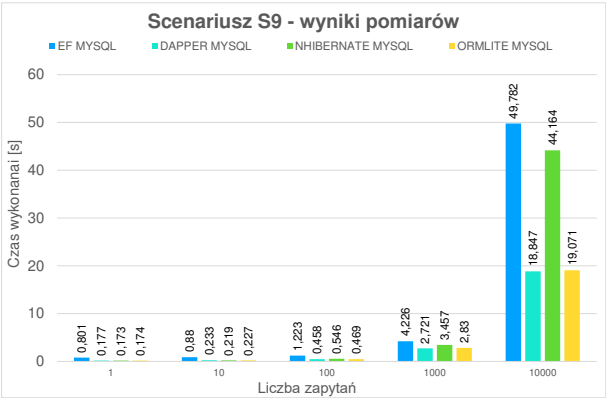
Rysunek 8: Wyniki pomiarów dla Scenariusza 7.



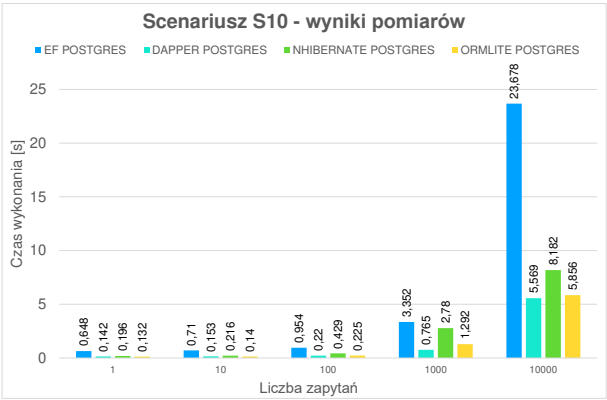
Rysunek 9: Wyniki pomiarów dla Scenariusza 8.

Na Rysunku 8 i 9 zobrazowane zostały dwa wykresy dla następujących systemów zarządzania bazami danych: MySQL i PostgreSQL. Zawarte w nich wyniki przedstawiają rezultat badań dla zapytania złożonego z wielu kolumn z wielu tabel. Dla każdego badanego szkieletu zostały również pogrupowane według liczby wykonanych zapytań.

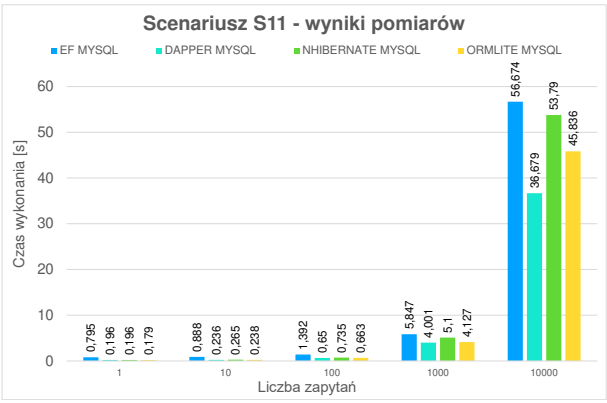
Rysunek 10 oraz 11 prezentują wyniki pomiarów dla systemów zarządzania bazami danych MySQL oraz PostgreSQL w kontekście zapytania zawierającego funkcję agregującą oraz grupowanie. Każdy z wykresów ukazuje rezultaty dla czterech analizowanych narzędzi ORM.



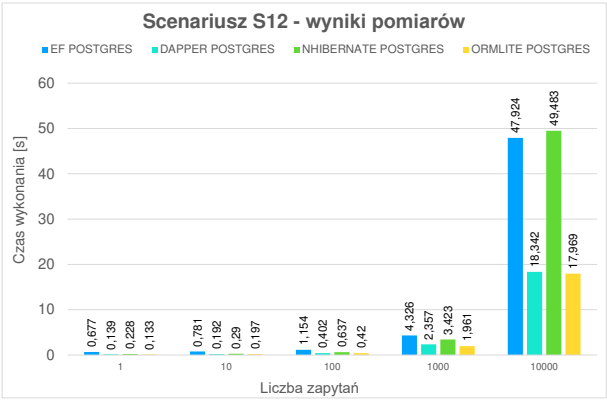
Rysunek 10: Wyniki pomiarów dla Scenariusza 9.



Rysunek 11: Wyniki pomiarów dla Scenariusza 10.



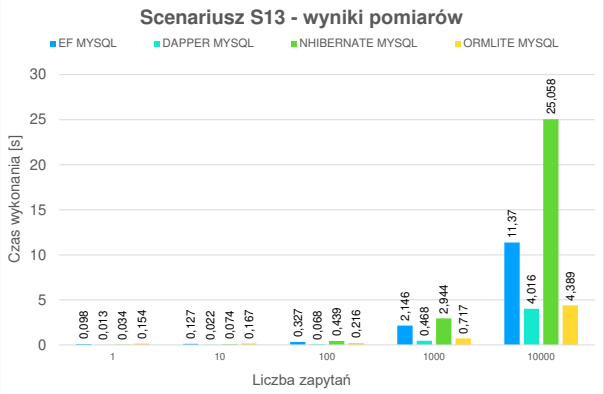
Rysunek 12: Wyniki pomiarów dla Scenariusza 11.



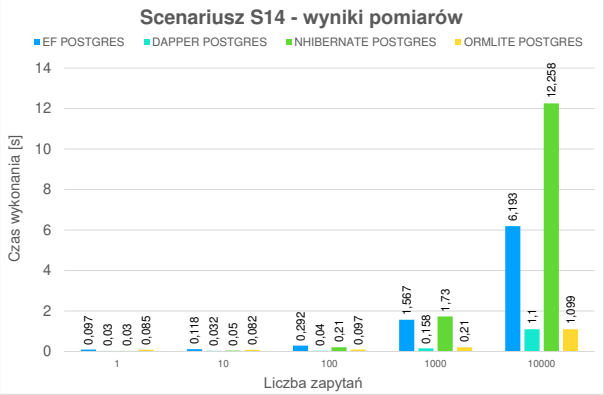
Rysunek 13: Wyniki pomiarów dla Scenariusza 12.

Rysunek 12 i 13 prezentują wyniki badań dla scenariuszy S11 i S12, które miały na celu przetestowanie złożonego zapytania wykorzystującego wiele funkcji agregujących oraz grupowanie. Tak jak poprzednie wykresy, przedstawione rezultaty odnoszą się do omawianych szkieletów mapowania bazodanowego.

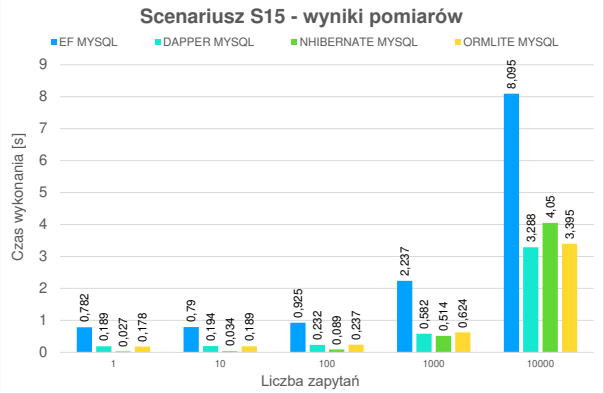
Rysunki 14 i 15 ilustrują dwa wykresy dotyczące systemów zarządzania bazami danych MySQL i PostgreSQL. Przedstawione na nich wyniki badań odnoszą się do zapytania aktualizującego dane w tabeli. Otrzymane rezultaty zostały opracowane dla analizowanych narzędzi ORM.



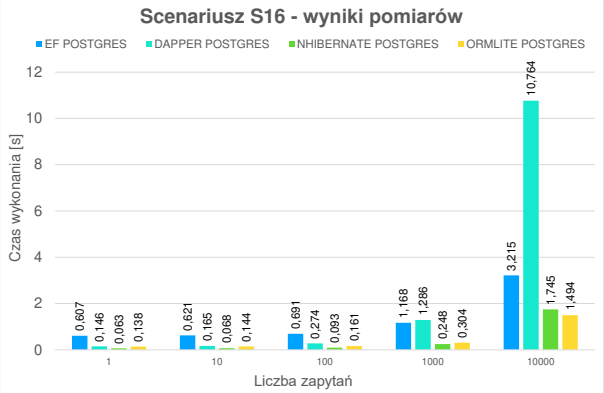
Rysunek 14: Wyniki pomiarów dla Scenariusza 13.



Rysunek 15: Wyniki pomiarów dla Scenariusza 14.



Rysunek 16: Wyniki pomiarów dla Scenariusza 15.



Rysunek 17: Wyniki pomiarów dla Scenariusza 16.

Na Rysunku 16 i 17 przedstawione zostały wyniki badań dla omawianych szkieletów mapowania bazodanowego. Zmierzone czasy zostały otrzymane dla zapytania aktualizującego dane w tabeli. Każdy z wykresów odpowiada osobnemu systemowi zarządzania bazami danych: MySQL oraz PostgreSQL.

6. Dyskusja wyników

Analiza wyników rozpoczyna się od sześciu zapytań typu SELECT, różniących się poziomem złożoności. Pierwsze zapytanie zostało napisane z użyciem klauzuli WHERE, jednej klauzuli typu INNER JOIN oraz podzapytania, które zawierało agregację danych AVG. Dla pierwszej serii powtórzeń dla systemu MySQL, najkrótszy czas wykonania zapytania uzyskało narzędzie Dapper, natomiast dla systemu PostgreSQL zapytanie najkrócej wykonywało się dla szkieletu OrmLite. Narzędziem, które uzyskało najdłuższy czas wykonania w przypadku obu systemów było Entity Framework. Dla pozostałych narzędzi mapujących czas wykonania był bardzo zbliżony do szkieletu Dapper. Wraz ze wzrostem liczby powtórzeń zapytania zwiększały się różnice w czasie wykonania. Przy największej badanej liczbie powtórzeń, zapytania zostały wykonane najszybciej przez narzędzie Dapper. Otrzymany wynik był zbliżony dla obu systemów zarządzania bazami danych.

Drugie zapytanie składa się z wielu klauzuli typu JOIN, użytych w celu wybrania wartości z różnych tabel. W przypadku jednego powtórzenia zapytania dla systemu MySQL, najszybszym narzędziem okazał się NHibernate. Z kolei dla systemu PostgreSQL najkrótszy czas wykonania uzyskał szkielet OrmLite. Przy największej testowanej liczbie powtórzeń zapytania, najkrótszy czas wykonania został zmierzony dla szkieletu Dapper.

Trzecie zapytanie jest podobne do drugiego, ponieważ również składa się z wielu operacji typu JOIN, jednak zwraca ono znacznie większą liczbę rekordów. Dla systemu MySQL zapytanie jest najszybciej wykonywane przez narzędzie NHibernate. Jednak dla systemu PostgreSQL, to szkielet OrmLite wykonał je najkrócej. W przypadku dziesięciu tysięcy powtórzeń dla systemu MySQL najkrótszy czas wykonania uzyskało narzędzie OrmLite, natomiast dla systemu PostgreSQL najszybszym narzędziem okazał się Dapper.

Na czwarte zapytanie składa się użycie wielu klauzuli typu JOIN, wybór wielu danych oraz zastosowanie klauzuli WHERE. Jednokrotne wykonanie tego rodzaju zapytania dla systemu MySQL w najkrótszym czasie cechuje narzędzie NHibernate, z kolei dla systemu PostgreSQL zapytanie najszybciej wykonuje szkielet OrmLite. Przy wzroście liczby powtórzeń najkrótszy czas wykonania zapytania dla obu systemów zarządzania bazą danych osiągnęło narzędzie Dapper.

Piąte zapytanie zawiera klauzule JOIN, GROUP BY oraz ORDER BY. Ten rodzaj zapytania dla systemu MySQL został najszybciej wykonany przez narzędzie NHibernate, jednak dla systemu PostgreSQL najkrótszy czas wykonania uzyskał szkielet OrmLite. Przy dziesięciu tysiącach powtórzeń zapytania, najlepszy rezultat otrzymało narzędzie Dapper.

Szóste zapytanie składa się z dwóch klauzuli JOIN, jednej klauzuli GROUP BY, polecenia ORDER BY oraz funkcji agregacji SUM. Najkrótszy czas wykonania zapytania przypadający na jedno powtórzenie dla obu systemów uzyskało narzędzie OrmLite. Również przy dziesięciu tysiącach powtórzeń zapytania dla systemu PostgreSQL to szkielet OrmLite uzyskał najkrótszy czas wykonania. Z kolei dla MySQL to narzędzie Dapper wykonało je najszybciej. Dalsza analiza wyników zrealizowana jest na podstawie jednej operacji typu UPDATE oraz jednej operacji typu INSERT.

Celem operacji UPDATE było podniesienie wszystkich cen produktów w konkretnej kategorii o 10%. Operacja typu UPDATE, zarówno dla systemu MySQL jak i PostgreSQL, została najszybciej wykonana przez narzędzie Dapper. Wynik taki osiągnięto zarówno dla jednego i dziesięciu tysięcy powtórzeń. Zapytanie typu INSERT, którego celem było wstawienie nowego produktu, zostało wykonane najszybciej przez narzędzie NHibernate. Z kolei dla dziesięciu tysięcy powtórzeń najkrótszy czas wykonania zapytania dla systemu PostgreSQL uzyskał szkielet OrmLite, a w przypadku systemu MySQL, to Dapper okazał się najszybszym narzędziem.

7. Wnioski

Podsumowując otrzymane wyniki, dla większości operacji typu SELECT najszybszym narzędziem wśród testowanych okazał się Dapper. Analogicznie został on również najszybszym szkieletem dla operacji typu UPDATE. Z kolei dla zapytań typu INSERT, to NHibernate okazał się najszybszym narzędziem ORM. Jednakże dla większej liczby powtórzeń, to Dapper szybciej wykonywał zapytania. Zarówno Dapper i NHibernate jak i OrmLite cechują się podobną szybkością wykonania zapytań, niezależnie od użytego systemu zarządzania relacyjną bazą danych jak i wybranego poziomu złożoności zapytań. W przypadku wszystkich testowanych typów zapytań, Entity Framework cechował się najdłuższym czasem wykonania zapytań, niezależnie od liczby powtórzeń. Wskazuje to jednoznacznie, że jest to najwolniejsze narzędzie mapowania obiektowo-relacyjnego wśród badanych. Pozostałe szkielety cechują się porównywalną szybkością wykonania zapytań. W przypadku najwolniejszego narzędzia, realizacja operacji DML opierała się na właściwościach klas oraz rozszerzeniu LINQ, co znacząco spowolniło ich wykonanie. Jednak dzięki tym funkcjonalnościom środowiska .NET, obsługiwanie zapytań SQL jest prostsze dla programistów. Uwzględniając wyniki badań zarówno dla różnych rodzajów SZBD jak i liczby wykonanych zapytań, narzędzie Dapper okazało się najszybsze, co sprawia, że wypada najkorzystniej spośród analizowanych szkieletów mapowania bazodanowego. Dlatego ten szkielet jest najlepszym wyborem w sytuacji, gdy operacje są realizowane na skomplikowanej strukturze danych. W pojedynczych przypadkach można stosować wolniejsze, lecz przystępniejsze narzędzie dla programistów – Entity Framework Core.

Literatura

[1] What is an ORM – The Meaning of Object Relational Mapping Database Tools, https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/, [10.11.2023].

[2] What is SQL (Structured Query Language)?, https://aws.amazon.com/what-is/sql/, [10.06.2024].

[3] K. Barczak, The Examination of SQL Queries Efficiency in Chosen IT System, Journal of Computer Sciences Institute 28 (2023) 186–189, https://doi.org/10.35784/jcsi.3606.

[4] P. Rymarski, Analysis of the possibilities of optimizing SQL queries, Master thesis, Lublin University of Technology, Lublin, 2021.

[5] Z. Łata, Performance analysis of databases created in virtualized and containerized environment, Master thesis, Lublin University of Technology, Lublin, 2023.

[6] S. Cvetković, D. Janković, A Comparative Study of the Features and Performance of ORM Tools in a .NET Environment, In International Conference on Object and Databases (2010) 147–158, https://doi.org/10.1007/978-3-642-16092-9_14.

[7] M. Myllyaho Forsberg, An evaluation of .NET Object-Relational Mappers in relational databases Entity Framework Core and Dapper, Bachelor thesis, Umeå University, Umeå, 2022.

[8] T. Nowicki, S. Tomczak, Comparative analysis of the time performance of database queries in C# language, Master thesis, Lublin University of Technology, Lublin, 2022.

[9] What Is C# Entity Framework? A Comprehensive Guide, https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/entity-framework-in-c-sharp, [10.11.2023].

[10] Welcome To Learn Dapper, https://www.learndapper.com/, [10.11.2023].

[11] NHibernate, https://www.tutorialspoint.com/nhibernate/nhibernate_overview.htm, [10.11.2023].

[12] OrmLite, https://ormlite.com/javadoc/ormlite-core/doc-files/ormlite.html, [10.11.2023].

[13] Baza danych Northwind, https://github.com/harryho/db-samples, [05.06.2024].