JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2022) 75–82
Received: 30 September 2022
Accepted: 14 October 2022

Performance analysis of working with databases with Spring and Symfony

Analiza wydajności pracy z bazami danych na przykładzie Spring i Symfony

Ewa Wieleba*, Bartłomiej Wieleba

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

Abstract

The article presents a comparative analysis of the efficiency of work with MySQL and PostgreSQL databases, using the popular Spring (Java) and Symfony programming frameworks. The research was carried out with the use of proprietary test applications that perform CRUD operations on a different number of records. The test results showed that the execution time of writing and deleting data using the Spring application is longer than when performing the same operations in Symfony. On the other hand, in the case of UPDATE and SELECT operations, the operation execution time with the Spring application turned out to be shorter than in the case of Symfony. The test results also confirmed that, regardless of the development framework, MySQL is less efficient than PostgreSQL while operating on 10 000 records except for DELETE, where MySQL combined with Symfony is the fastest.

Keywords: comparative analysis; Spring; Symfony; relational databases

Streszczenie

W artykule przedstawiono analizę porównawczą wydajności pracy z bazami danych MySQL i PostgreSQL, z wykorzystaniem popularnych szkieletów programistycznych Spring (Java) i Symfony. Badania przeprowadzono z wykorzystaniem autorskich aplikacji testowych, realizujących operacje typu CRUD na różnej liczbie rekordów. Wyniki testów wykazały, że czas wykonywania operacji zapisu i usuwania danych przy użyciu aplikacji Spring jest dłuższy niż przy wykonywaniu tych analogicznych operacji w Symfony. Natomiast w przypadku operacji UPDATE i SELECT, czas wykonywania operacji za pomocą aplikacji Spring okazał się krótszy niż w przypadku Symfony. Wyniki testów dowiodły, że niezależnie od szkieletu programistycznego MySQL jest mniej wydajny w stosunku do PostgreSQL przy wykonywaniu poleceń na dużej liczbie (10 000) rekordów dla wszystkich operacji poza operacją DELETE, gdzie MySQL z Symfony jest najszybszy.

Słowa kluczowe: analiza porównawcza; Spring; Symfony; relacyjne bazy danych

*Corresponding author

Email address: ewa.szewczak@pollub.edu.pl (E. Wieleba)

1. Wstęp

Wiele systemów informatycznych składa się z mikroserwisów i od nich zależy ich wydajność. Mikroserwisy odbierają i przetwarzają żądania, łączą się z innymi mikroserwisami lub bazami danych, na których wykonują wiele operacji. Liczba przetwarzanych operacji potrafi być ogromna w związku z czym rośnie obciążenie na mikroserwisach i bazach danych, co wiąże się z ich zawodnością i konsekwentnie z zawodnością systemów, które z nich korzystają.

Często zawodność wynika z opóźnień, kiedy współpracujące ze sobą serwisy zbyt długo czekają na odpowiedź na zadane żądania. Programiści starają się zapobiegać takim przypadkom, ale wszystko wciąż sprowadza się do jednego: szybkości obsługi żądań przez mikroserwisy i konsekwentnie, do szybkości obsługi zapytań przez bazy danych, z którymi współpracują te serwisy. Ważnym elementem decydującym o szybkości jest wybór języka programowania mikroserwisów oraz silnika baz danych. W niniejszym artykule przeprowadzono analizę porównawczą wydajności czasowej aplikacji współpracującej z danymi w relacyjnych bazach danych (MySQL i PostgreSQL), utworzonych w języku PHP (Symfony) i Java (Spring).

2. Przegląd literatury

Autorzy artykułu [1] badali możliwości optymalizacji zapytań SQL na silnikach baz danych Oracle, MySQL, Microsoft SQL Server i PostgreSQL z użyciem własnej aplikacji napisanej w języku Java z wykorzystaniem Hibernate. Z przeprowadzonych badań wynika, że wydajność baz danych zależy przede wszystkim od ilości danych przechowywanych przez bazę, stopnia skomplikowania wykonywanych zapytań oraz liczby zwracanych kolumn i rekordów w zapytaniu. Na szybkość wykonywania zapytań ma także wpływ funkcja grupująca i użycie klauzuli UNION.

Autorka artykułu [2] porównuje wydajność systemów bazodanowych MySQL, PostgreSQL i MS SQL w połączeniu z aplikacją internetową stworzoną w języku Java z Hibernate i JDBC. Autorka zwraca uwagę na powiązania między tabelami. W przypadku zapytań, w których występowało wykorzystanie powiązań między tabelami najlepiej wypadł silnik PostgreSQL, ale przy porównaniu czasów wykonywania zapytań dla pojedynczych tabel okazywało się, że są one do siebie

zbliżone zarówno dla silnika baz danych PostgreSQL, MySQL i MS SQL.

Autorzy artykułu [3] przy pomocy aplikacji Laravel (PHP) sprawdzali wydajność baz danych SQL Server, MySQL oraz PostgreSQL. Badania polegały na pomiarze czasu wykonywania prostych zapytań i operacji z użyciem konkatenacji kolumn oraz tabel. Zebrane wyniki pokazały, że w przypadku wykonywania zapytań dla małej liczby rekordów (do 1000) bardzo dobrze wypada baza MySQL.

Autorzy artykułu [4] porównywali dwa szkielety programistyczne PHP: Laravel i Symfony pod kątem architektury, organizacji kodu, czytelności dokumentacji. Badania wykazały, że oba szkielety doskonale sprawdzają się do tworzenia aplikacji internetowych. Symfony okazał się dobrym wyborem przy budowie rozbudowanych projektów i jest uważany za najbardziej stabilny szkielet programistyczny PHP, natomiast Laravel jest najbardziej popularnym szkieletem programistycznym do tworzenia aplikacji w języku PHP i jest najprostszy do nauki.

Autorzy artykułu [5] porównali silniki baz danych MySQL i PostgreSQL. Badania polegały na łączeniu się z bazą danych, tworzeniu tabel, następnie uzupełnieniu tabel danymi, wykonywaniu instrukcji SELECT, a następnie usuwaniu wszystkich danych z bazy. Polecenia były przeprowadzane na 5000, 10000, 15000, 20000, 25000, 30000, 35000 i 40000 rekordach. Wyniki wskazały, że operacje INSERT, SELECT i DELETE są wykonywane szybciej przy użyciu silnika bazy danych MySQL niż przy użyciu silnika baz danych PostgreSQL.

W artykule [6] autorzy przeprowadzili badania wydajności baz danych MySQL, MS SQL, PostgreSQL i Oracle uruchomionych na platformie Docker. Stworzyli do tego celu aplikację testową w języku PHP z użyciem szkieletu programistycznego Symfony z interfejsem graficznym napisanym w Vue.js. Do przeprowadzenia badań stworzyli w aplikacji klasy pozwalające na wykonanie pojedynczych operacji na bazach danych, następnie do tych klas napisali testy jednostkowe. Każdy test, uruchamiający odpowiednią klasę, został wykonany 100-krotnie. Autorzy ~~po przeglądzie literatury~~ założyli, że najszybsze są silniki Oracle i MS SQL. Uzyskane wyniki badań częściowo potwierdziły ich tezę, ponieważ w większości operacji najlepiej wypadał silnik Oracle, a trochę słabszy był MS SQL.

Autorzy artykułu [7] przeprowadzili badania ukierunkowane na pomiar szybkości wykonywania zapytań, obciążenia procesora i pamięci na dysku dla baz danych MySQL, PostgreSQL oraz Firebird na systemie operacyjnym Windows 10 Pro 64-bit. Badania polegały na powtórzeniu 10 razy tego samego scenariusza i obliczeniu średniej wartości pomiarów. Przeprowadzone badania dowiodły, że najmniejsze obciążenie procesora występuje przy użyciu bazy Firebird, natomiast baza MS SQL okazała się najszybsza, zajmują najmniej miejsca na dysku, ale bardzo obciążając procesor.

Autorzy artykułu [8] przeprowadzili badania wydajności dwóch szkieletów programistycznych PHP: Yii i CodeIgniter. Badania wykazały, że Yii jest szybszy w porównaniu z innymi frameworkami i autorzy polecili go do budowania dużych projektów, gdzie istotnym jest czas. Dodatkową jego zaletą jest prostota i automatycznie generowany kod kodu. Natomiast framework CodeIgniter jest łatwiejszy w instalacji, jednak z badań wynika, że może być trudniejszy do nauki, dlatego autorzy zalecają go bardziej doświadczonym programistom.

Autorzy artykułu [9] przeprowadzali badania mające na celu porównanie frameworków: Symfony2 i PhalconPHP pod kątem ich popularności, dostępności dokumentacji, możliwości i narzędzi przyśpieszających proces budowy oprogramowania. Badania polegały na wysyłaniu 2000 żądań przy ustanowionych 10 jednoczesnych połączeniach. Badania wykazały, że Phalcon i Symfony2, mają podobne możliwości routingu, użycia adnotacji i wyrażeń regularnych, ale Symfony2 zapewnia większy wybór formatów plików takich jak XML lub YAML. Silnik szablonów Twig w Symfony2 pozwala na lepszą organizację struktury szablonów. Z kolei silnik szablonów Volt w Phalcon jest bardziej przyjazny dla programistów zaczynających pracę z silnikami szablonów i ładuje się szybciej niż Twig.

W zależności od silnika baz danych, od zastosowanego języka programowania po stronie serwera i od wybranego frameworka może zależeć efektywność działania aplikacji internetowej.

3. Cel badań

Celem badań podjętych w niniejszym artykule jest porównanie wydajności pracy z relacyjnymi bazami danych MySQL i PostgreSQL na przykładzie Spring i Symfony.

Postawiono następujące tezy badawcze:

- Czas wykonywania operacji SELECT, DELETE, UPDATE, INSERT przy użyciu Spring jest krótszy niż przy wykonywaniu tych samych operacji przy użyciu szkieletu programistycznego Symfony w połączeniu z bazami danych PostgreSQL i MySQL.
- Czas wykonywania operacji SELECT dla silnika baz danych MySQL jest krótszy niż dla silnika PostgreSQL dla mniejszej liczby danych.

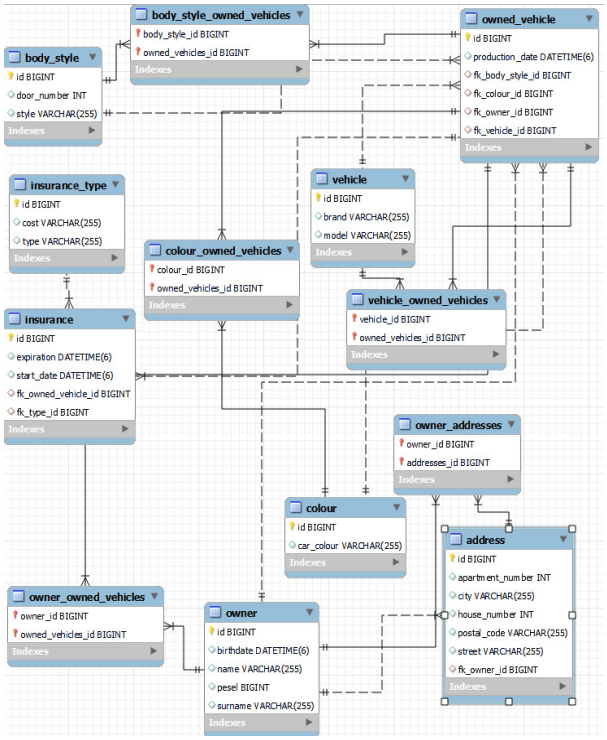
W artykule użyto szkieletu programistycznego Symfony [12] i Spring [11], ponieważ według rankingów [14-15] są one najbardziej popularnymi szkieletami programowania odpowiednio dla języka Java i PHP. Podobnie w przypadku baz danych, wybrano MySQL [19] i PostgreSQL [13] jako jedne z najpopularniejszych silników bazodanowych [16]. W artykule przedstawiono analizę porównawczą dla dwóch popularnych szkieletów programistycznych języków Java i PHP.

4. Metoda i środowisko badań

Do realizacji badań opracowano 3 aplikacje testowe:

- Java ze Spring – aplikacja typu REST,
- PHP z Symfony – aplikacja typu REST,
- aplikacja do pomiaru czasu wykonywania operacji, przeznaczona do zbierania wyników badań w języku Java.

Operacje na bazie danych realizowano za pomocą narzędzi ORM – Hibernate w Spring oraz Doctrine w Symfony. Schemat bazy danych samochodów, na której wykonano testy, przedstawiony jest na Rys. 1. Nazwy tabel i pól przygotowano w j. angielskim, aby zachować zgodność językową z nazwami klas, wykorzystywanych w kodzie aplikacji. .



Rysunek 1: Schemat ERD bazy danych.

Testy polegały na wykonywaniu zapytań zapisywania rekordów do tabeli *owned_vehicle*, aktualizowania danych takich jak marka, model, początek i koniec okresu na który ubezpieczono pojazd, ilość drzwi jaką posiada pojazd, styl karoserii pojazdu, adres właściciela pojazdu i kolor pojazdu, usuwania kaskadowego rekordów z tabeli *owned_vehicle*, dodawania rekordów do tabeli *owned_vehicle* . Testy realizowano dla różnej liczby rekordów (100, 1000 i 10000). Każda z operacji była powtarzana 50 razy, Czas wykonania operacji mierzony był poprzez obliczenie różnicy czasu przed i po wykonaniu danej operacji.

Do testów został użyty komputer o specyfikacji:

- procesor Ryzen 4600h,
- 24GB RAM DDR 4 3200MHz,
- dysk SSD,
- system operacyjny Windows 10 Home Edition.

Aplikacje testowe zaimplementowano w językach:

- Java (JDK 18) i Spring Boot 2.7.0,
- PHP 8.1.6 i Symfony 6.1.0.

5. Elementy implementacji aplikacji testowych

Przykładowe instrukcje UPDATE i SELECT dla Spring zostały pokazane na przykładach 1 i 2. Analogiczne operacje zostały zaimplementowane za pomocą Doctrine w Symfony.

Przykład 1: Instrukcja update w Spring Boot

```
@Query(value = "update owned_vehicle set
fk_colour_id =?1, fk_body_style_id =?2 where
  id=?3", nativeQuery = true)
void updateOwnedVehicleById(Long colour,
   Long bodyStyle, Long id);

@Transactional
public void updateOwnedVehicle(OwnedVehicleDto dto)
{
  ownedVehicleRepository
  .updateOwnedVehicleById(dto.getFkColourId(),
    dto.getFkBodyStyleId(), dto.getId());
}
```

Przykład 2: Instrukcja SELECT z JOIN w Spring Boot

```
@Query(value = "select " + "v.brand, " +
       "v.model, " + "i.start_date, " +
        "i.expiration, " + "bs.door_number, " +
        "bs.style, " + "a.city, " +
        "a.street, " + "a.apartment_number, " +
        "a.house_number, " + "c.car_colour " +
        "from owned_vehicle " +
        "union all vehicle v on v.id =
         owned_vehicle.fk_vehicle_id " +
        "union all insurance i on
       owned_vehicle.id = i.fk_owned_vehicle_id " +
       "union all body_style bs on bs.id =
        owned_vehicle.fk_body_style_id " +
       "union all address a on
      owned_vehicle.fk_owner_id = a.fk_owner_id " +
           "union all colour c on c.id =
   owned_vehicle.fk_colour_id", nativeQuery = true)

List<OwnedVehicleResponse> selectWithJoin();
//-----------------------------------------------
public List<OwnedVehicleResponse> selectWithJoin()
{
  return
     ownedVehicleJoinRepository.selectWithJoin();
}
```

6. Wyniki badań

W tabeli 1 przedstawiono wyniki porównania średnich czasów wykonywania operacji do pobierania i aktualizacji danych dla systemów bazodanowych MySQL i PostgreSQL w aplikacji Spring Boot.

W tabeli 2 przedstawiono analogiczne wyniki dla aplikacji Symfony.

6.1. Operacje dla 100 rekordów

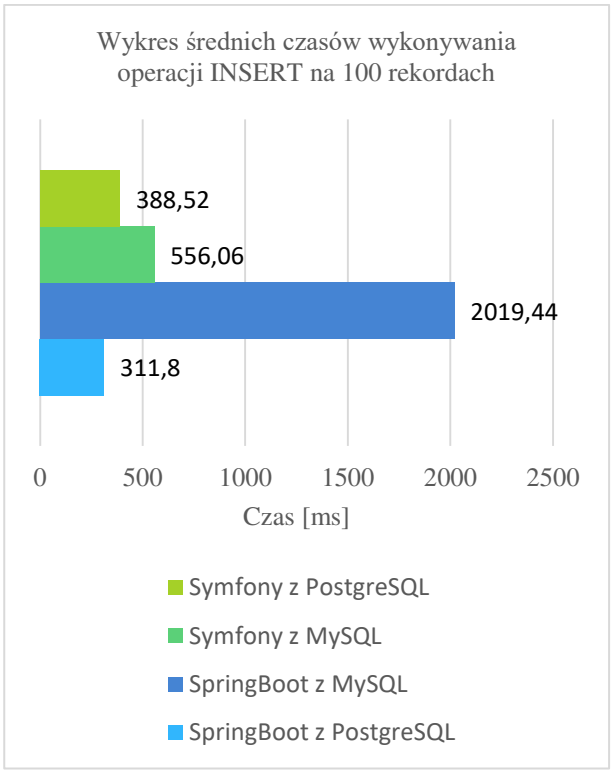
Średnie czasy wykonywania operacji na bazie danych dla 100 rekordów przedstawiono na rysunkach 2-5.

Tabela 1: Średnie czasy wykonywania zapytań w SpringBoot

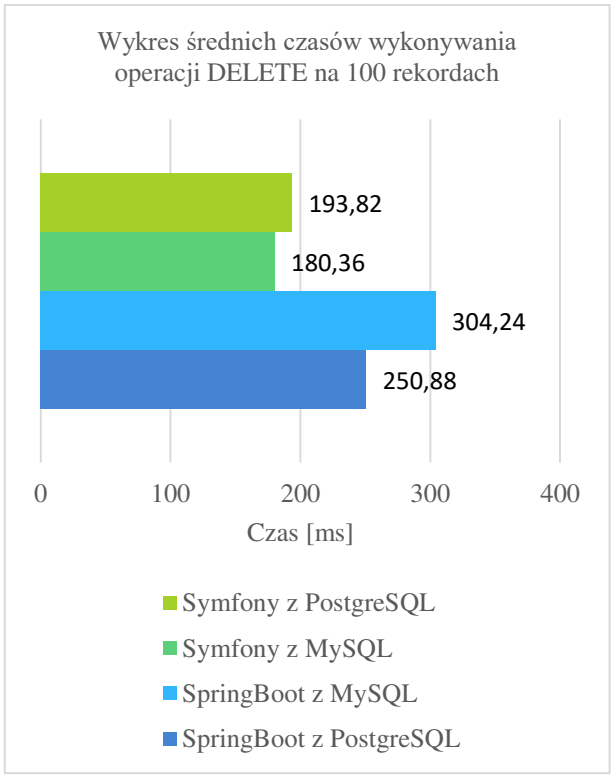
Tabela średnich czasów wykonywania zapytań w SpringBoot w PostgreSQL i MySQL						
	SpringBoot z PostgreSQL			SpringBoot z MySQL		
	100 rekordów	1000 rekordów	10000 rekordów	100 rekordów	1000 rekordów	10000 rekordów
INSERT	311,8 ms	2000,22 ms	18661,44 ms	2019,44 ms	19842,42 ms	204407,14 ms
DELETE	250,88 ms	2000,24 ms	18511 ms	304,24 ms	2711,88 ms	27825,04 ms
UPDATE	8,16 ms	20,2 ms	143,52 ms	11,4 ms	30,82 ms	232,92 ms
SELECT	31,58 ms	41,94 ms	180,42 ms	39,62 ms	55,78 ms	219,62 ms

Tabela 2: Średnie czasy wykonywania zapytań w Symfony

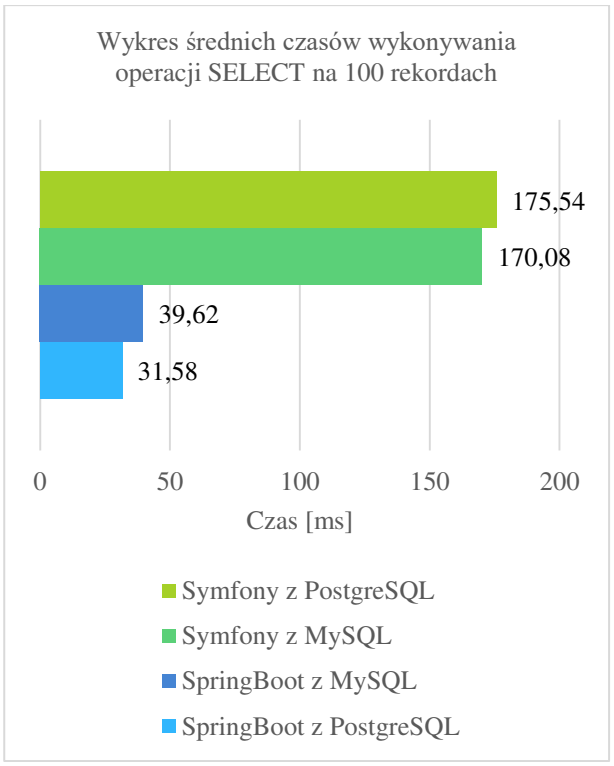
Tabela średnich czasów wykonywania zapytań w Symfony w PostgreSQL i MySQL						
	Symfony z PostgreSQL			Symfony z MySQL		
	100 rekordów	1000 rekordów	10000 rekordów	100 rekordów	1000 rekordów	10000 rekordów
INSERT	388,52 ms	1835,84 ms	17403,34 ms	556,06 ms	3373,82 ms	32157,26 ms
DELETE	193,82 ms	225,22 ms	627,76 ms	180,36 ms	193,96 ms	331,02 ms
UPDATE	175,98 ms	191,34 ms	376,4 ms	174,42 ms	196,9 ms	389,18 ms
SELECT	175,54 ms	187,24 ms	355,72 ms	170,08 ms	185,36 ms	378,48 ms



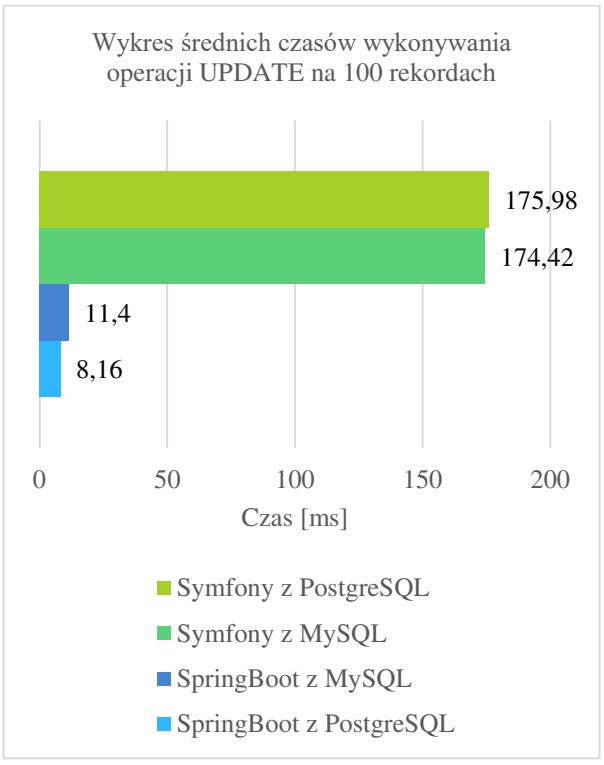
Rysunek 2: Wykres średnich czasów wykonania operacji INSERT na 100 rekordach.



Rysunek 3: Wykres średnich czasów wykonywania operacji DELETE na 100 rekordach.



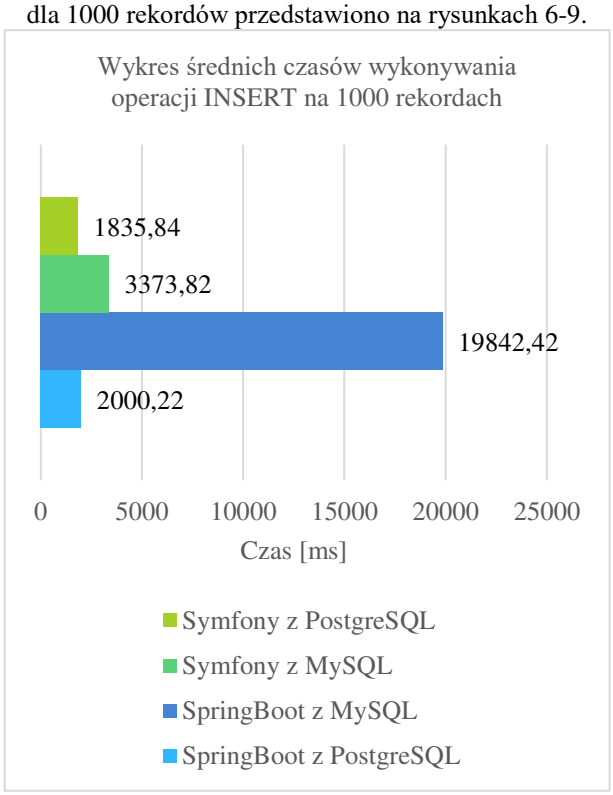
Rysunek 4: Wykres średnich czasów wykonywania operacji SELECT na 100 rekordach.



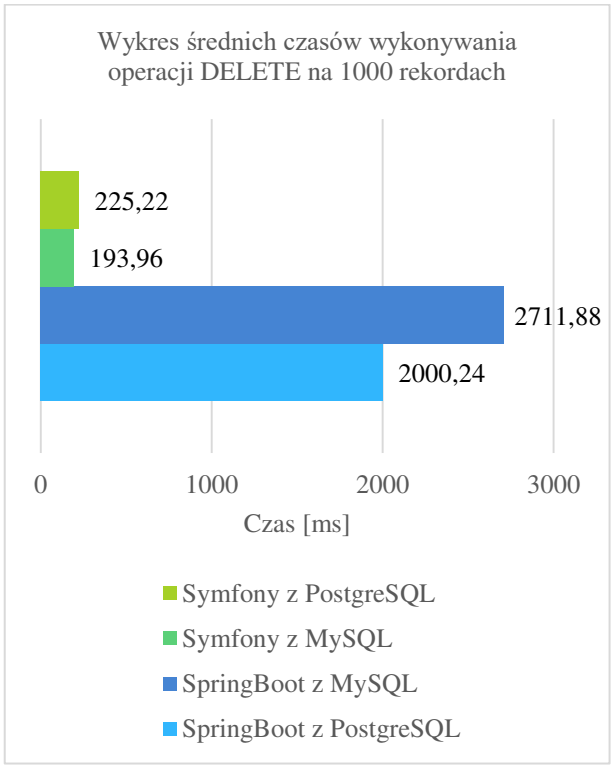
Rysunek 5: Wykres średnich czasów wykonywania operacji UPDATE na 100 rekordach.

6.2. Operacje dla 1000 rekordów

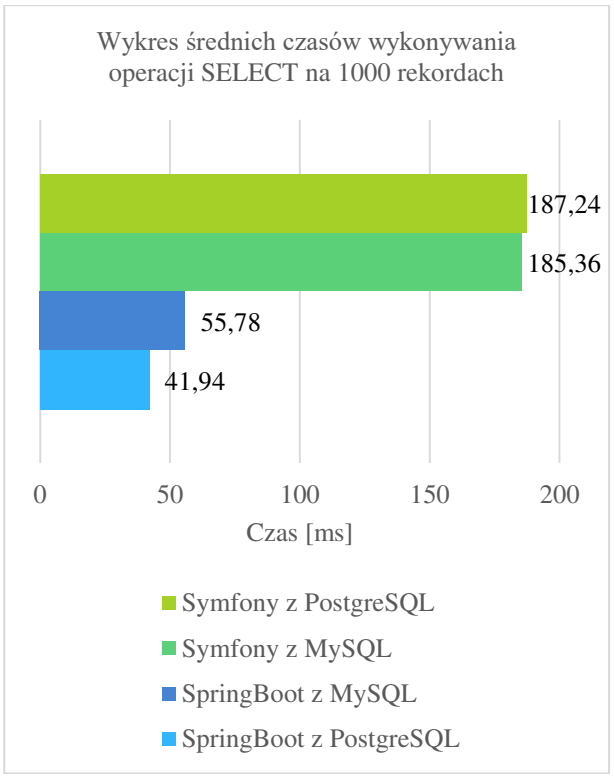
Średnie czasy wykonywania operacji na bazie danych dla 1000 rekordów przedstawiono na rysunkach 6-9.



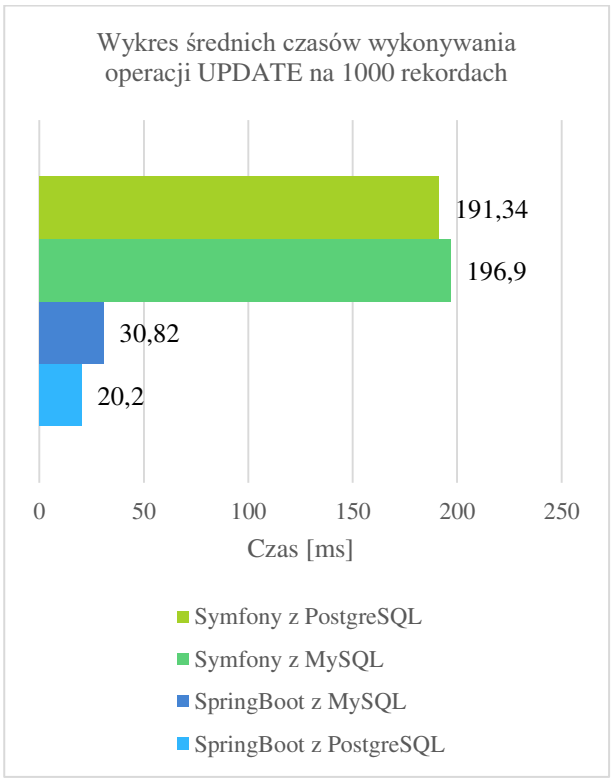
Rysunek 6: Wykres średnich czasów wykonywania operacji INSERT na 1000 rekordach.



Rysunek 7: Wykres średnich czasów wykonywania operacji DELETE na 1000 rekordach.



Rysunek 8: Wykres średnich czasów wykonywania operacji SELECT na 1000 rekordach.

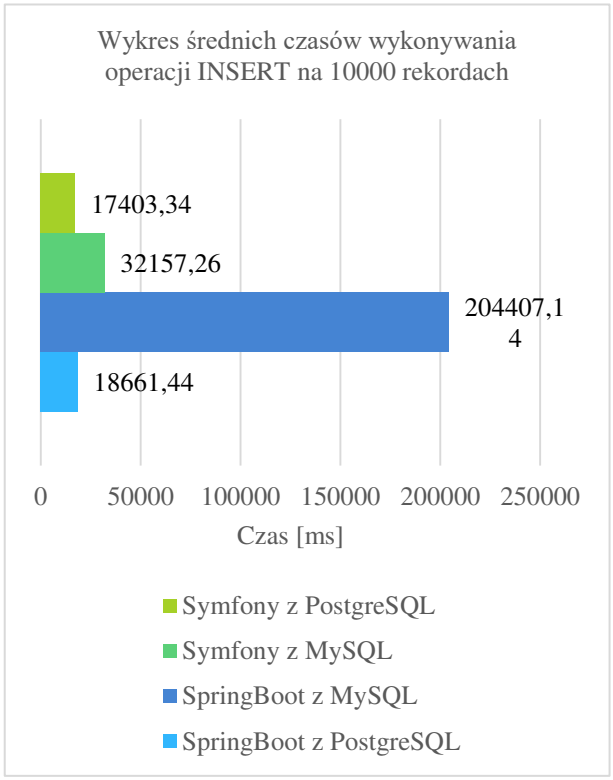


Rysunek 9: Wykres średnich czasów wykonywania operacji UPDATE na 1000 rekordach.

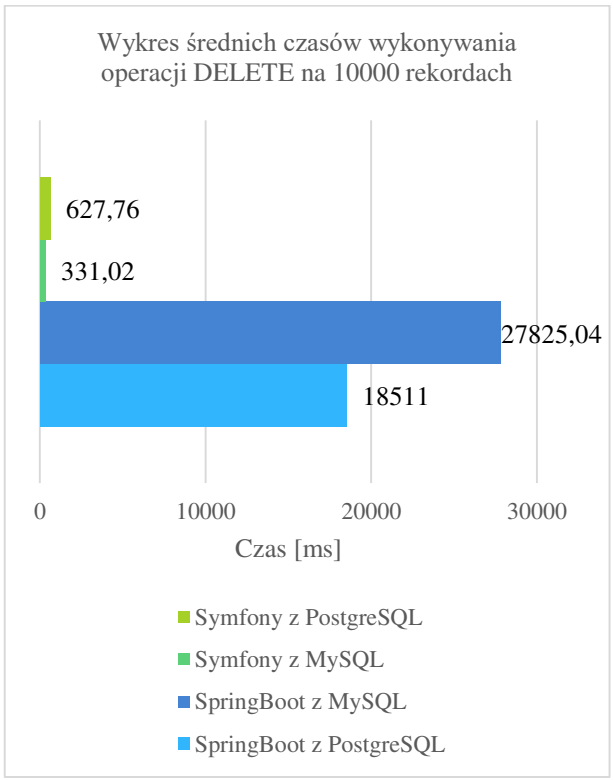
6.3. Operacje dla 10000 rekordów

Średnie czasy wykonywania operacji na bazie danych dla 10000 rekordów przedstawiono na rysunkach 10-13.

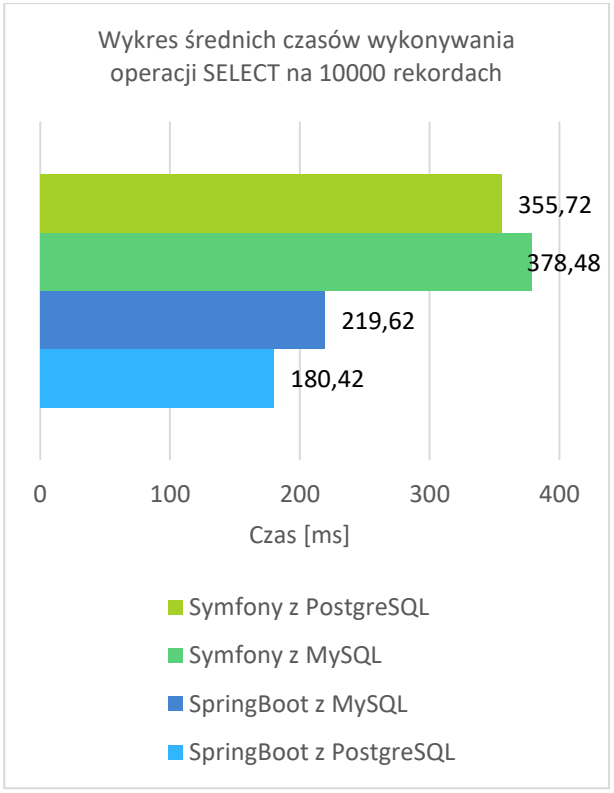
.



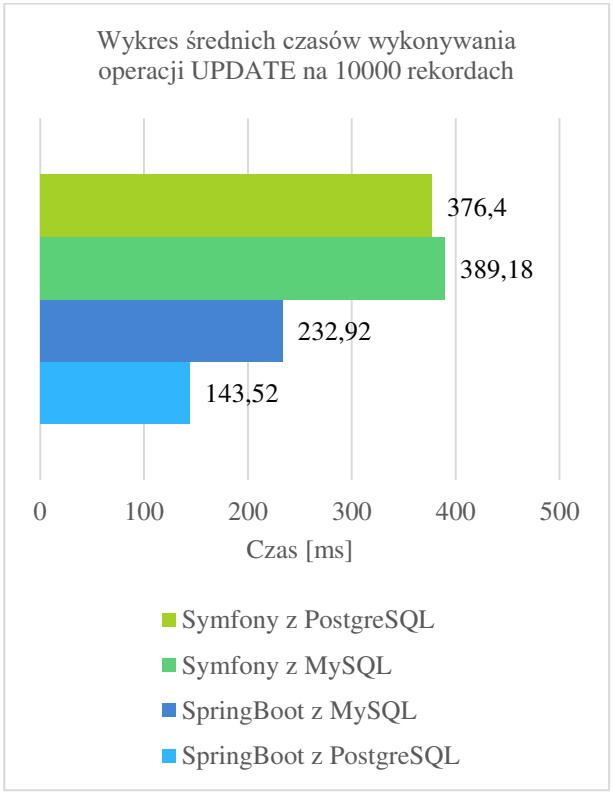
Rysunek 10: Wykres średnich czasów wykonywania operacji INSERT na 10000 rekordach.



Rysunek 11: Wykres średnich czasów wykonywania operacji DELETE na 10000 rekordach.



Rysunek 12: Wykres średnich czasów wykonywania operacji SELECT na 10000 rekordach.



Rysunek 13: Wykres średnich czasów wykonywania operacji UPDATE na 10000 rekordach.

7. Wnioski

Zebrane w tabeli 1 dane wskazują, że w połączeniu z frameworkiem Spring Boot, silnik PostgreSQL jest zdecydowanie szybszy od silnika MySQL. Najbardziej widać to przy wykonywaniu operacji INSERT, gdzie czas wykonywania operacji dla 10000 rekordów w PostgreSQL jest zdecydowanie krótszy niż w przypadku silnika MySQL. Znacząca różnica występuje również dla operacji DELETE, zwłaszcza dla 10 000 rekordów, gdzie czas wykonywania operacji dla PostgreSQL jest znacznie krótszy niż w przypadku MySQL.

Nieco mniejsza różnica występuje przy porównywaniu czasów operacji aktualizacji i pobierania danych (tabela 1). Tutaj również szybszy okazał się PostgreSQL. Dla 10 000 rekordów przy użyciu aplikacji napisanej w języku Java w SpringBoot operacja UPDATE wykonuje się o około 38,4% krócej dla PostgreSQL niż w przypadku MySQL i operacja SELECT o około 17,8% krócej.

Wyniki badań z tabeli 2 dla Symfony wskazują, że czas trwania operacji INSERT jest zdecydowanie krótszy dla bazy PostgreSQL niż MySQL. Dla pozostałych operacji można zaobserwować, że czas zależy od liczby rekordów. W Symfony z MySQL czasy trwania operacji DELETE, UPDATE i SELECT na 100 rekordach są krótsze niż dla analogicznych zapytań w bazie PostgreSQL. Natomiast czas trwania operacji DELETE, UPDATE i SELECT na 1000, 10000 rekordach jest krótszy dla przypadku Symfony z PostgreSQL.

Podsumowując silniki baz danych MySQL i PostgreSQL w połączeniu z aplikacją Spring można dojść do wniosku, że silnik PostgreSQL jest wydajniejszy od silnika MySQL w przypadku wykonywania operacji na dużej liczbie rekordów.

Biorąc pod uwagę wszystkie operacje przeprowadzone na 100, 1000 i 10000 rekordach, czasy wykonania operacji dla MySQL i Spring są dłuższe w porównaniu do innych wariantów aplikacji. Szczególnie można to zaobserwować na 10000 rekordach przy operacji INSERT (Rys 10) i DELETE (Rys 11) gdzie średni czas wykonania operacji dodawania i usuwania nowego rekordu do bazy PostgreSQL jest zdecydowanie krótszy (Tabela 1, Tabela 2, Rys. 10, Rys. 11). Mniejsza różnica występuje przy porównywaniu średnich czasów wykonywania operacji UPDATE i SELECT, jednak wciąż szybszy okazał się silnik PostgreSQL (Rys 13, Rys 12).

Wyniki badań przeprowadzonych przy użyciu aplikacji Symfony, czas wykonywania operacji INSERT niezależnie od liczby rekordów, krótszy jest przy użyciu bazy danych PostgreSQL. W przypadku pozostałych operacji: UPDATE, SELECT, DELETE w przypadku wykonania operacji na 100 rekordach, czas zapytania jest krótszy dla bazy MySQL. Natomiast przy wykonywaniu operacji na większej ilości danych zdecydowanie lepiej sprawdza się baza danych PostgreSQL (Tabela 2, Rys 3, Rys 4, Rys 5).

Na podstawie przeprowadzonej analizy porównawczej obalono słuszność postawionych tez, że „Czas wykonywania operacji SELECT, DELETE, UPDATE, INSERT przy użyciu szkieletu programistycznego Spring jest krótszy niż przy wykonywaniu tych samych operacji przy użyciu szkieletu programistycznego Symfony w połączeniu z bazami danych PostgreSQL

i MySQL". Nie potwierdziła się też teza, że „Czas wykonywania operacji SELECT dla silnika MySQL jest krótszy niż dla silnika PostgreSQL przy operowaniu na mniejszej ilości danych.".

Szkielet programistyczny Spring okazał się wydajniejszy od Symfony w połączeniu zarówno z bazą PostgreSQL jak i MySQL dla operacji SELECT i UPDATE (Tabela 1, Rys 4, Rys 5, Rys 8, Rys 9, Rys 12, Rys 13). Natomiast porównując czas wykonywania operacji INSERT i DELETE zdecydowanie wydajniejszy okazał się Symfony (Tabela 2, Rys 2, Rys 3, Rys 6, Rys 7, Rys 10, Rys 11).

Zebrane wyniki udowadniają, że czas wykonywania operacji SELECT dla silnika baz danych MySQL jest krótszy niż dla silnika PostgreSQL, w połączeniu z aplikacją Symfony niezależnie od liczby rekordów. Natomiast czas wykonywania polecenia SELECT przy użyciu aplikacji Spring jest dłuższy w połączeniu z bazą danych MySQL niż PostgreSQL (Tabela 1, Tabela 2).

Na podstawie przeprowadzonych badań można stwierdzić, że budując aplikacje, w których główną rolę grać będzie pobieranie dużych ilości danych i ich modyfikowanie, lepszym rozwiązaniem będzie wybór języka Java i Spring w połączeniu z bazą danych PostgreSQL. Rezultaty częściowo zgadzają się z wynikami badań przeprowadzonymi w artykule [2], gdzie PostgreSQL był wydajniejszy dla większych zestawów danych jednak badania przeprowadzone w tym artykule wskazują, że jest też wydajniejszy dla mniejszych zestawów danych (100 rekordów). Wyniki przeprowadzonych badań jednak zaprzeczają wynikom uzyskanym w artykule [5], PostgreSQL jest wydajniejszy niż MySQL.

Trudno też jest jednoznacznie wskazać najlepszą konfigurację.

Literatura

[1] P. Rymarski, G. Kozieł, Analiza możliwości optymalizacji zapytań SQL, Journal of Computer Sciences Institute 19 (2021) 151–158.

[2] K. Lachewicz, Analiza wydajności systemów bazodanowych: MySQL, MS SQL, PostgreSQL w kontekście aplikacji internetowych, Journal of Computer Sciences Institute 14 (2020) 94–100.

[3] R. Wodyk, M. Skublewska-Paszkowska, Porównanie wydajności relacyjnych baz danych SQL Server, MySQL oraz PostgreSQL z zastosowaniem aplikacji webowej i frameworku Laravel, Journal of Computer Sciences Institute 17 (2020) 358–364.

[4] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi, A. E. Yamami, A comparative study of laravel and symfony PHP frameworks, International Journal of Electrical and Computer Engineering 9 (2019) 704-712.

[5] S. Andjelic, S. Obradovic, B. Gacesa, A performance analysis of the dbms – mysql vs postgresql, Komunikacie 10 (2008) 53-57.

[6] R. Kleweka, W. Truskowski, M. Skublewska-Paszkowska, Porównanie wydajności baz danych MySQL, MSSQL, PostgreSQL oraz Oracle z uwzględnieniem wirtualizacji, Journal of Computer Sciences Institute 16 (2020) 279–284.

[7] S. Stets, G. Kozieł, Comparative analysis of databases working under the control of Windows system, Journal of Computer Sciences Institute 13 (2019) 298–301.

[8] S.T Ali, J. Long, Quality Evaluation of PHP Frameworks, International Journal of Scientific & Engineering Research 10 (2019) 1454-1458.

[9] N. Prokofyeva, V. Boltunova, Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems, Procedia Computer Science 104 (2017) 51 – 56.

[10] Baza danych PostgreSQL, https://vavatech.pl/technologie/bazy-danych/postgresql, [06.06.2022].

[11] Dokumentacja SpringBoot, https://spring.io/projects/spring-boot , [07.06.2022].

[12] Dokumentacja Symfony, https://symfony.com/doc/current/index.html, [15.09.2022].

[13] Dokumentacja PostgreSQL, https://www.postgresql.org/docs/, [15.09.2022].

[14] Najpopularniejsze szkielety programistyczne do tworzenia aplikacji przy użyciu języka PHP w 2022 roku, https://www.linkedin.com/pulse/2022-most-popular-php-frameworks-infogenlabsinc/, [15.09.2022].

[15] Najpopularniejsze szkielety programistyczne do tworzenia aplikacji przy użyciu języka Java, https://www.geeksforgeeks.org/top-10-most-popular-java-frameworks-for-web-development/, [15.09.2022].

[16] Ranking popularności silników baz danych, https://db-engines.com/en/ranking, [12.10.2022].

[17] Dokumentacja szkieletu programistycznego Hibernate, https://hibernate.org/orm/documentation/6.1/, [12.10.2022].

[18] Dokumentacja biblioteki PHP Doctrine, https://symfony.com/doc/current/doctrine.html, [12.10.2022].

[19] Dokumentacja silnika baz danych MySQL, https://dev.mysql.com/doc/, [12.10.2022].