JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 190-196
Received: 22 May 2023
Accepted: 4 July 2023

Comparative analysis of selected databases on the example of a proprietary web application

Analiza porównawcza wybranych baz danych na przykładzie autorskiej aplikacji internetowej

Łukasz Przychodzień*, Dominika Radwan, Grzegorz Kozieł

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

Abstract
Database performance is one of the most important factors affecting the usability of the system. Therefore, the authors of the article decided to examine 3 popular database systems: MySQL, MS SQL and PostgreSQL, analyzing their performance. For this purpose, a test application was prepared and Docker software was used to simulate different hardware parameters. Depending on the selected settings and the number of records, different results were obtained. For small data sets, the differences were almost imperceptible. They have drastically increased for large data sets. In this case, MySQL fared poorly, and MS SQL was the best. This means that the choice of the database is very important, and it is worth considering the available hardware, the amount of data and the queries performed.

Keywords: performance analysis; relational databases; DBMS

Streszczenie
Wydajność bazy danych jest jednym z najistotniejszych czynników wpływających na użyteczność systemu. Dlatego autorzy artykułu postanowili przebadać 3 popularne systemy bazodanowe: MySQL, MS SQL oraz PostgreSQL analizując ich wydajność. W tym celu przygotowano aplikację testową oraz wykorzystano oprogramowanie Docker, aby umożliwić symulację różnych parametrów sprzętowych. W zależności od wybranych ustawień oraz liczebności rekordów uzyskano inne wyniki. Dla małych zestawów danych różnice były niemal niezauważalne. Drastycznie zwiększyły się dla dużych zbiorów danych. W tym przypadku słabo wypadł MySQL, a najlepiej MS SQL. Oznacza to, że wybór bazy danych jest niezwykle istotny, a do jego podjęcia warto rozważyć dostępny sprzęt, liczebność danych oraz wykonywane zapytania.

Słowa kluczowe: analiza porównawcza; relacyjne bazy danych; SZDB

*Corresponding author
Email address: **lukasz.przychodzien@pollub.edu.pl** (Ł. Przychodzień)

1. Wstęp

W obecnych czasach bardzo trudno jest znaleźć aplikację czy stronę internetową, której działanie nie byłoby oparte na wykorzystaniu bazy danych w celu przechowywania informacji koniecznych do poprawnego funkcjonowania systemu. W znacznej większości przypadków wykorzystywane są do tego celu relacyjne bazy danych, przechowujące dane w uporządkowanych tabelach. Każda z tabel zawiera informacje na określony temat, taki jak dane klientów, zamówień czy pracowników. Dodatkowo poszczególne tabele połączone są relacjami pozwalającymi na uporządkowanie związków pomiędzy nimi.

Ważnym aspektem podczas projektowania aplikacji oraz doboru użytych technologii jest kwestia wydajności. Niezwykle istotnym jest, aby zapytania obsługiwane przez system bazodanowy wykonywały się wystarczająco szybko. Dodatkowo należy zwrócić uwagę na zróżnicowanie wielkości przechowywanych zbiorów danych w przypadku typowych aplikacji oraz możliwości wykorzystanej architektury serwerowej, a także kosztów związanych z jej zakupem.

Biorąc pod uwagę powyższe kwestie autorzy artykułu zdecydowali się przeprowadzić porównanie wydajności najpopularniejszych systemów bazodanowych. Dodatkowo, aby zweryfikować wpływ parametrów środowiska do przeprowadzenia testów wykorzystana zostanie technologia wirtualizacji w postaci kontenerów Docker. Ponadto przeprowadzane testy zostaną wykonane dla kilku zbiorów danych o różnej wielkości, dzięki czemu zbadana zostanie również skalowalność systemu bazodanowego.

2. Przegląd literatury

Aby lepiej poznać możliwości wybranych systemów bazodanowych, została przeprowadzona analiza artykułów o podobnym temacie powstałych na przestrzeni ostatnich lat.

W artykule “Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework” [1] autorzy przedstawili porównanie wydajności trzech silników bazodanowych: SQL Server, PostgreSQL oraz MySQL. Do badań wykorzystana została aplikacja napisana w języku PHP z użyciem frameworka Laravel. W badaniach skupiono się na najpopularniejszych operacjach bazodanowych, takich jak np. SELECT, INSERT, DELETE. Czas wykonania instrukcji na bazie

był mierzony z poziomu kodu PHP obliczając różnicę czasu po i przed wykonaniu instrukcji. Działanie każdej instrukcji sprawdzano kolejno na różnych liczbach rekordów badanej tabeli począwszy od 1 aż do 20000 rekordów. Dla małej liczby rekordów tj. do 1000 najszybszy okazał się MySQL, niewiele wolniejszy był PostgreSQL, z kolei znacznie wolniejszy okazał się MSSQL. W przypadku większej liczby rekordów (powyżej 1000) najszybszy okazał się PostgreSQL.

Artykuł „Performance analysis of relational databases Oracle and MS SQL based on desktop application" [2] opisuje porównanie wydajności relacyjnych baz danych Oracle oraz MS SQL. Do testów autorzy wykorzystali autorską aplikację napisaną w języku C# z użyciem technologii MVVM (Model-View-View-Model). Dziewit et al. badali czas odpowiedzi systemów bazodanowych w zależności od wykonywanych zapytań wywoływanych przez zewnętrzną (aplikacja pozasystemowa względem bazy danych) aplikację desktopową wspomagającą zarządzanie warsztatem samochodowym. Czasy wykonania poszczególnych zapytań zostały wyznaczone na podstawie statystyk wykonanych poleceń zapisywanych automatycznie przez systemy bazodanowe. Autorzy analizowali czasy wykonania operacji języka DML zarówno dla danych tekstowych, jak i binarnych. Badania wykazały, że baza MS SQL lepiej realizuje operacje insert i update niż baza Oracle, z kolei Oracle jest wydajniejszy w przypadku operacji typu select na danych binarnych.

Autorzy pracy pod tytułem "Analiza porównawcza baz danych" [3] przeprowadzili badania najpopularniejszych systemów zarządzania bazami danych: MySQL, PostgreSQL oraz Firebird. Analizowali oni szybkość wykonania zapytań, obciążenie procesora oraz pamięci na dysku z użyciem autorskiej aplikacji przygotowanej w oparciu o standardy TPC (Transaction Processing Performance Council). Aplikacja została napisana w języku Java. Testy wydajności przeprowadzono na dwóch komputerach o różnych parametrach technicznych.

Pod względem szybkości wykonania zapytań najlepsza okazała się baza danych MySQL. Identycznie wygląda sytuacja pod względem ilości zajmowanego miejsca w pamięci – najlepsze wyniki uzyskała baza MySQL. Wyniki badań obciążenia procesora wykazały zależność od rodzaju wykonywanej operacji. Podczas dodawania dużej liczby rekordów najmniejsze obciążenie procesora zaobserwowano dla silnika bazy Firebird.

W pracy [4] porównana została wydajność baz danych MySQL, PostgreSQL, MariaDB oraz H2. Autorzy wyznaczyli średni czas wykonania operacji dodania, aktualizacji, usuwania oraz wybierania danych. Każde badanie zostało wykonane 10 razy dla 1, 100, 2500, 5000, 7500 i 10000 rekordów, a następnie obliczony został czas średni. MySQL wykazała słabą wydajność podczas pracy z dużą ilością danych. Najlepsze średnie czasy wykonania wszystkich operacji (za wyjątkiem aktualizacji) uzyskała baza H2. Dla operacji sortowania, złączenia, wybierania różnica w czasie dość istotna ok. 100 ms. Wyniki otrzymane dla MySQL, MariaDB oraz PostgreSQL są zbliżone dla operacji wybierania, złączenia i usuwania danych, jednakże PostgreSQL okazał się trochę szybszy od pozostałych dwóch. PostgreSQL wykazał najlepsze wyniki dla operacji aktualizacji.

Innym rozwiązaniem bazodanowym cieszącym się coraz większą popularnością są bazy nierelacyjne. Artykuł „A comparison of database performance of MariaDB and MySQL with OLTP workload" opisuje porównanie relacyjnej bazy MySQL z nierelacyjnym systemem MariaDB [5]. Autorzy analizowali wydajność obu baz na przykładzie systemu klasy OLTP wspomagającego pracę lotniska. Ich pomiary wykazały znaczną przewagę bazy relacyjnej MySQL.

Podobne badania opisuje artykuł „Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database" [6]. Jego twórcy porównali wydajność 3 systemów NoSQL z 3 bazami relacyjnymi badając czasy wykonania operacji CRUD. W przypadku tej pracy lepsze wyniki osiągnęły rozwiązania nierelacyjne.

Zainteresowanie tematyką wydajności relacyjnych baz danych w ostatnich latach jest bardzo wysokie. Powstały liczne prace naukowe weryfikujące efektywność zarówno baz relacyjnych jak i nierelacyjnych, sprawdzające ich działanie w połączeniu z aplikacjami internetowymi oraz desktopowymi. Jednak nie znaleziono badań wykorzystujących framework Spring, jak również wirtualizację z użyciem środowiska Docker. Wykorzystanie konteneryzacji umożliwia przeprowadzenie testów wydajnościowych dla różnorodnych parametrów technicznych środowiska testowego oraz ich analizę w sposób niespotykany w żadnej z opisanych badań.

3. Relacyjne systemy baz danych

Za twórcę relacyjnych baz danych uznaje się Edgara Coda, który zaproponował to rozwiązanie w roku 1969, gdy pracował dla IBM-u. Cechą charakterystyczną baz relacyjnych jest przedstawienie przechowywanych danych w formie tabeli z wierszami i kolumnami [7]. Każda tabela zawiera w sobie rekordy danych (pojedynczy rekord zwany jest krotką) gdzie każda krotka jest definiowana przez typ atrybutu oraz unikalną dla tej krotki wartością.

Kolejną cechą wyróżniającą bazy relacyjne jest wykorzystanie relacji, czyli połączeń pomiędzy tabelami, co pozwala na łączenie tabel na wiele sposobów umożliwiając wyszukiwanie zaawansowanych połączeń pomiędzy danymi. Dodatkowo umożliwia to na ograniczenie redundancji przechowywanych danych. Każda tabela dzieli z inną tabelą co najmniej jeden atrybut, jeśli jest z nią w relacji 'jeden do jednego', 'wiele do jednego' albo 'wiele do wielu'.

3.1. MySQL

Stworzony przez szwedzką firmę MySQL AB w roku 1995, która została wykupiona przez Sun Microsys-

tems, obecnie funkcjonujący pod nazwą Oracle Corporation. Dziś jest jednym z najbardziej popularnych open sourcowych systemów zarządzania relacyjną bazą danych [8].

Podobnie jak inne relacyjne bazy danych, MySQL przechowuje dane w tabelach składających się z wierszy i kolumn. Użytkownicy mogą definiować dane, manipulować nimi, kontrolować je i wyszukiwać zapytania przy użyciu języka Structured Query Language, bardziej znanego jako SQL [9].

Wśród jego licznych zastosowań można wymienić pakiet oprogramowania służącego do rozwoju aplikacji internetowych znany pod akronimem LAMP (Linux, Apache, MySQL, PHP) oraz jego alternatywę, czyli pakiet LEMP. W przypadku tego drugiego serwer Apache został zastąpiony serwerem Nginx (czyt. engine-x, stąd litera E w skrócie).

3.2. PostgreSQL

Początki systemu PostgreSQL datuje się na lata 80. ubiegłego wieku. Powstał on na Uniwersytecie Kalifornijskim w Berkeley. Obecnie nad rozwojem języka czuwa grupa entuzjastów pod nazwą "PostgreSQL Global Development Group" [10]. PostgreSQL jest bazą obiektowo-relacyjną dystrybuowaną na darmowej licencji open source.

Dane przechowywane są w postaci tabel połączonych relacjami. Dodatkowo, PostgreSQL integruje obiektowe podejście do baz danych z relacyjnym pozwalając na wykorzystanie narzędzi znanych z relacyjnych baz danych, ale poszerzonych o możliwości związane z obiektowością. Podobnie jak w programowaniu obiektowym możemy stworzyć klasę oraz jej obiekty i klasę dziedziczącą po tej klasie.

Relacyjne właściwości bazy pozwalają z kolei na używanie strukturalnego języka zapytań znanego z relacyjnych baz danych do efektywnego i szybkiego przeszukiwania danych niedostępnego w obiektowych bazach danych. Dodatkowo PostgreSQL wspiera transakcje zgodne z zasadami ACID (Atomicity, Consistency, Isolation, Durability - ang. atomowość, spójność, izolacja, trwałość). Dzięki czemu gwarantuje poprawne przetwarzanie wykonywanych operacji [11].

3.3. MS SQL

Microsoft SQL Server to system zarządzania bazą danych, rozpowszechniany przez korporację Microsoft. Jest to główny produkt bazodanowy tej firmy oraz trzeci co do popularności najczęściej wybierany system bazodanowy na świecie. Głównym językiem zapytań wykorzystywanym w MS SQL jest Transact-SQL. Poza edycjami czysto komercyjnymi Microsoft udostępnia również edycje darmowe do dowolnego zastosowania takie jak Express i Developer [12].

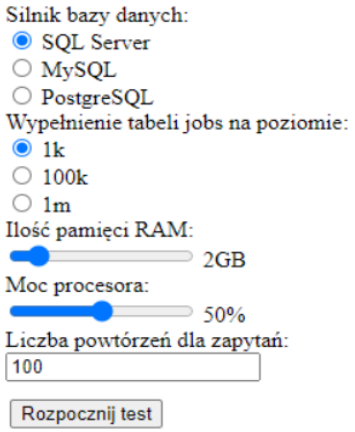
MS SQL, podobnie jak inne relacyjne bazy danych, przechowuje dane w tabelach, kolumnach i wierszach. Każdy wpis jest zdefiniowany unikalnym identyfikatorem. Od samego początku twórcy systemu za nadrzędny cel przyjmowali wydajność oraz niezawodność opracowywanej bazy danych. Równie istotna była skalowalność opracowywanego systemu, czyli jego dostosowanie do obsługi nawet bardzo dużych zbiorów danych.

Microsoft SQL Server jest szeroko stosowany we wdrożeniach korporacyjnych. Ponadto udostępnia on kilka narzędzi ETL (Extract, Transform and Load) oraz usługi raportowania, w których można dodawać, modyfikować i wyszukiwać dane przy użyciu ustandaryzowanego języka zapytań strukturalnych (SQL). MS SQL to rozwijająca się platforma danych używana w rozwiązaniach biznesowych i danych o znaczeniu krytycznym w środowisku lokalnym, w chmurze, czy na platformach hybrydowych [13].

4. Aplikacja testowa

Do przeprowadzenia testów przygotowana została autorska aplikacja napisana w oparciu o szkielet programistyczny Spring. Pozwoliła ona na zautomatyzowanie oraz przyspieszenie wykonywania pomiarów. Na rysunku 1 zaprezentowano ekran główny aplikacji. Do przeprowadzenia serii testów należy wybrać analizowaną bazę danych, ustawić odpowiednie parametry testowe oraz liczbę powtórzeń w serii oraz nacisnąć przycisk "Rozpocznij test".

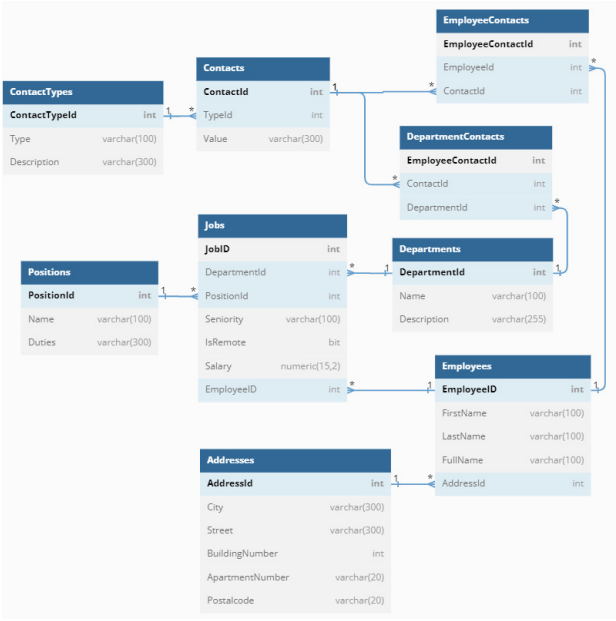
Aplikacja automatycznie uruchamia wybraną bazę danych na kontenerze, a następnie wykonuję serię zapytań według przygotowanych scenariuszy testowych, uwzględniając wybraną liczbę powtórzeń. Czas rozpoczęcia oraz zakończenia obsługi każdego z zapytań jest rejestrowany, a na ich podstawie wyliczany jest dokładny czas wykonania danej instrukcji. Zebrane w ten sposób pomiary następnie zapisywane są do pliku csv. Tak przygotowane dane zostały następnie opracowane z użyciem języka Python.



Rysunek 1: Ekran główny aplikacji.

4.1. Schemat bazy danych

Na potrzeby przeprowadzenia badań przygotowana została baza testowa o schemacie przedstawionym na rysunku 2. Zawiera ona informacje o pracownikach zatrudnionych w firmie, ich danych, umowach, czy wynagrodzeniu.



Rysunek 2: Schemat ERD bazy testowej.

5. Metodyka

Podstawową operacją realizowaną przez większość systemów bazodanowych jest pobieranie danych spełniających zadane warunki. W tabeli 1 przedstawiono zestawienie scenariuszy testowych wykorzystanych podczas testów. Każdy z opisywanych scenariuszy uwzględnia inny typ zapytania DQL (ang. Data Query Language - język do definiowania zapytań).

Tabela 1: Scenariusze testowe

Scenariusz	Kod SQL
I	SELECT c.Value FROM Contacts c WHERE c.Value LIKE '%@gmail.com';
II	SELECT TOP 5 Salary FROM Jobs j ORDER BY Salary DESC LIMIT 5;
III	SELECT c.Value, e.FullName FROM Contacts c JOIN EmployeeContacts ec ON c.ContactId = ec.ContactId JOIN Employees e ON ec.EmployeeId = e.EmployeeId;
IV	SELECT COUNT(DISTINCT j.EmployeeId), d.Name FROM Jobs j JOIN Departments d ON j.DepartmentId = d.DepartmentId GROUP BY d.DepartmentId, d.Name;
V	SELECT SUM(CASE WHEN j.IsRemote = 0 THEN 1 ELSE 0 END) AS office, SUM(CASE WHEN j.IsRemote = 1 THEN 1 ELSE 0 END) AS remote, d.Name FROM Jobs j JOIN Departments d ON j.DepartmentId = d.DepartmentId GROUP BY d.DepartmentId, d.Name;
VI	WITH CTE(remoteCNT, DepartmentId) AS (SELECT SUM(j.IsRemote), j.DepartmentId FROM Jobs j GROUP BY DepartmentId) SELECT d.Name, c.remoteCNT FROM cte c JOIN Departments d ON c.DepartmentId = d.DepartmentId;
VII	SELECT e.FullName FROM Employees e WHERE NOT EXISTS (SELECT 1 FROM EmployeeContacts ec WHERE ec.EmployeeId = e.EmployeeID)
VIII	SELECT e.FullName FROM Employees e WHERE EXISTS (SELECT 1 FROM EmployeeContacts ec WHERE ec.EmployeeId = e.EmployeeID)
IX	SELECT DISTINCT a.City FROM Addresses a;
X	SELECT j.EmployeeID, 'low income' AS ds FROM Jobs j WHERE j.Salary < 1000 UNION ALL SELECT j.EmployeeID, 'middle class' FROM Jobs j WHERE j.Salary BETWEEN 1000 AND 10000 UNION ALL SELECT j.EmployeeID, 'upper class' FROM Jobs j WHERE j.Salary > 10000
XI	SELECT City, Street, BuildingNumber, ApartmentNumber FROM Addresses WHERE City IN ('Warszawa', 'Kraków', 'Wrocław')
XII	SELECT City, Street, BuildingNumber, ApartmentNumber FROM Addresses WHERE City NOT IN ('Warszawa', 'Kraków', 'Wrocław')

Aby lepiej zrozumieć wpływ wielkości zbioru danych na wydajność analizowanego systemu testy zostały przeprowadzone z wykorzystaniem 3 zestawów danych o różnej liczbie rekordów opisanej w tabeli 2 z uwzględnieniem poszczególnych tabel bazy.

Za najbardziej znaczącą uznano tabelę Jobs, odzwierciedlającą umowy o pracę zawarte z pracownikami przedsiębiorstwa. Przechowuje ona informacje, o tym który pracownik pracuje na jakim stanowisku oraz do którego działu należy. Rozszerza ona dodatkowo te dane o informacje takie jak: tytuł pracownika w firmie, informację czy pracownik wykonuje swoje obowiązki zdalnie, a także o wysokość jego miesięcznej wypłaty. Dlatego też w dalszej części artykułu tabela Jobs nazywana jest tabelą główną, a liczba rekordów w niej przechowywanych jest równoznaczna do poziomu wypełnienia bazy.

Tabela 2. Liczebność rekordów w tabelach

Nazwa Tabeli	Liczba rekordów		
	Zestaw 1	Zestaw 2	Zestaw 3
Addresses	1000	100000	1000000
Employees	1000	100000	1000000
Jobs	1000	100000	1000000
Positions	1000	100000	1000000
Contacts	1100	100000	1000000
ContactTypes	3	3	3
Departments	100	100	100
DepartmentContacts	100	100	100
EmployeeContacts	1000	100000	1000000

Kolejnym istotnym czynnikiem analizowanym w badaniach jest wpływ parametrów środowiska uruchomieniowego na wydajność obsługi zapytań. W celu uwzględnienia go wykorzystane zostało oprogramowanie do konteneryzacji Docker. Tabela 3 zawiera opis 3 konfiguracji testowych użytych w badaniach oraz wartości 2 parametrów testowych dla każdej z tych konfiguracji.

Tabela 3. Konfiguracje parametrów testowych

Konfiguracja	Ilość pamięci RAM	Maksymalne dopuszczalne wykorzystanie CPU
Konfiguracja 1	2 GB	20%
Konfiguracja 2	4 GB	40%
Konfiguracja 3	6 GB	60 %

Parametry ustawiane były podczas uruchamiania kontenera. Do ustawienia wybranej wartości pamięci wykorzystano flagę --memory. Domyślnie kontener Dockerowy ma nieograniczony dostęp do zasobów

procesora maszyny, na której jest uruchomiony. Dlatego na potrzeby testów wykorzystano parametr --cpus, określający maksymalne wykorzystanie procesora przez uruchamiany kontener [14].

Do przeprowadzenia badań wykorzystany został komputer o następujących parametrach technicznych:

- Procesor: Intel i5 7300hq,
- GPU: NVIDIA GeForce GTX 1050 Mobile,
- RAM: 16 GB 3200 MHz,
- Dysk: SSD NVMe 3200 MB/s / 1900 MB/s.

6. Analiza wyników

W celu uniknięcia wpływu losowych zakłóceń na uzyskane wyniki, każde z badań przeprowadzono 100 razy. Uzyskane wyniki poddano obróbce, usuwając z nich po 3 wartości skrajne. Pozostałe wyniki zostały przeanalizowane i zaprezentowane w dalszej części artykułu.

6.1. Wyniki dla zbioru 1 000 rekordów

W przypadku najmniejszego zestawu danych testowych oraz najsłabszych parametrów sprzętowych wszystkie bazy uzyskały bardzo dobre wyniki, co pokazuje tabela 4. Różnica w wynikach pomiędzy systemami jest nieznaczna, jednak przy 9 z 12 badanych scenariuszy najlepiej wypadł PostgreSQL. Z kolei dla zapytań wykorzystujących instrukcje UNION ALL i NOT IN lepszy wynik uzyskał MySQL. MS SQL wygrał w scenariuszu z użyciem NOT EXISTS.

Tabela 4. Wyniki dla konfiguracji testowej 1

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	10.9	7.1	7.9
Scenariusz II	9.2	5.6	12.7
Scenariusz III	17.1	8.8	15.4
Scenariusz IV	19.3	7.8	15.3
Scenariusz V	14.2	7.5	10.9
Scenariusz VI	15.7	8.1	12.4
Scenariusz VII	8.5	7.7	7.1
Scenariusz VIII	16.7	9.5	23.6
Scenariusz IX	14.9	8.3	36.1
Scenariusz X	13.9	20.0	17.7
Scenariusz XI	17.9	7.0	11
Scenariusz XII	10.7	12	14

Tabela 5. Wyniki dla konfiguracji testowej 2

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	2.9	1.8	3.3
Scenariusz II	2.5	1.5	2.5
Scenariusz III	6.1	3.8	6.8
Scenariusz IV	4.2	2.4	3.7
Scenariusz V	3.7	2.0	3.1
Scenariusz VI	3.7	2.0	3.1
Scenariusz VII	3.1	2.0	5.9
Scenariusz VIII	4.3	3.0	6.4
Scenariusz IX	3.3	2.4	5.4
Scenariusz X	4.4	4.9	4.6
Scenariusz XI	3.2	2.7	3.2
Scenariusz XII	4.6	5.0	4.1

Ponownie dla najmniejszego zbioru danych, w przypadku konfiguracji 2 uzyskano bardzo zbliżone wyniki dla wszystkich analizowanych systemów, co obrazują wyniki zaprezentowane w tabeli 5. Przeciętny czas operacji we wszystkich analizowanych scenariuszach był rzędu kilku milisekund. Z niewielką przewagą w 10 analizowanych zapytaniach najwydajniejszy okazał się PostgreSQL.

Tabela 6. Wyniki dla konfiguracji testowej 3

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	2.5	1.6	2.4
Scenariusz II	2.0	1.4	2.0
Scenariusz III	4.6	3.5	5.0
Scenariusz IV	3.8	2.3	2.5
Scenariusz V	4.2	2.1	2.8
Scenariusz VI	3.2	1.9	2.4
Scenariusz VII	2.5	1.7	3.2
Scenariusz VIII	3.3	2.1	3.3
Scenariusz IX	2.8	1.9	2.9
Scenariusz X	3.6	3.8	2.2
Scenariusz XI	3.1	1.9	3.8
Scenariusz XII	4.7	3.5	4.8

Dla konfiguracji testowej 3, tak jak w przypadku poprzedniej, testy przy wypełnieniu bazy na poziomie 1000 rekordów zakończyły się wynikami rzędu kilku milisekund oraz niemal niezauważalnymi różnicami pomiędzy badanymi systemami. Ponownie najwydajniejszy okazał się w tym przypadku PostgreSQL, co można zaobserwować w tabeli 6.

6.2. Wyniki dla zbioru 100 000 rekordów

W przypadku drugiego zestawu danych testowych oraz konfiguracji 1 uzyskano bardziej zróżnicowane czasy pomiarów, co jest widoczne w tabeli 7. W znacznej większości scenariuszy najgorzej wypadł MySQL uzyskując znacząco większe czasy od pozostałych systemów. Bardzo dobrze radził sobie PostgreSQL, notując najlepsze wyniki w 8 z 12 przypadków testowych. W pozostałych 4 najlepiej wypadł MS SQL.

Tabela 7. Wyniki dla konfiguracji testowej 1

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	547.2	249.8	28.6
Scenariusz II	320.8	65.7	308.6
Scenariusz III	4621.0	1603.3	2031.8
Scenariusz IV	1363.8	559.1	600.4
Scenariusz V	1861.5	350.9	270.6
Scenariusz VI	1188.4	235.0	310.7
Scenariusz VII	912.5	380.7	736.6
Scenariusz VIII	1512.0	900.7	432.7
Scenariusz IX	589.2	250.7	450.6
Scenariusz X	1231.1	2485.9	99.6
Scenariusz XI	467.9	181.4	111.6
Scenariusz XII	1350.4	1042.6	39.9

Tabela 8. Wyniki dla konfiguracji testowej 2

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	216.3	98.3	3.1
Scenariusz II	123.7	32.7	86.9
Scenariusz III	2641.1	622.0	1102.8
Scenariusz IV	524.7	272.6	321.4
Scenariusz V	665.6	109.4	129.5
Scenariusz VI	434.8	98.0	147.4
Scenariusz VII	447.4	127.9	239.9
Scenariusz VIII	625.5	300.4	146.9
Scenariusz IX	272.5	69.2	131.0
Scenariusz X	635.7	683.9	30.8
Scenariusz XI	211.9	64.3	54.2
Scenariusz XII	626.3	480.7	4.3

Również dla tej konfiguracji testowej zwiększenie liczby rekordów testowych spowodowało wydłużenie czasów badanych operacji oraz zwiększenie różnic pomiędzy analizowanymi systemami, co widać w tabeli 8. Dla zbioru 100 000 oraz konfiguracji numer 2 w 11 z 12 scenariuszy testowych najgorszy wynik odnotował MySQL. W 6 najwydajniejszy okazał się PostgreSQL. Nietypowe wyniki uzyskano w scenariuszu X, gdzie PostgreSQL okazał się najwolniejszy, a znaczną przewagą wygrał MS SQL.

Przy wypełnieniu tabel na poziomie 100 000 oraz konfiguracji 3 bardzo słabe wyniki uzyskał MySQL, który zajął ostatnie miejsce w przypadku 10 analizowanych scenariuszy. Znacznie lepsze wyniki uzyskał PostgreSQL, który okazał się najwydajniejszy w 7 z 12 przeprowadzonych testów. Średnie czasy uzyskane w tym zestawieniu zaprezentowano w tabeli 9.

Tabela 9. Wyniki dla konfiguracji testowej 3

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	142	89	8
Scenariusz II	108	20	91
Scenariusz III	1682	518	994
Scenariusz IV	325	199	232
Scenariusz V	376	95	93
Scenariusz VI	237	63	106
Scenariusz VII	179	65	188
Scenariusz VIII	288	189	117
Scenariusz IX	119	57	116
Scenariusz X	275	540	27
Scenariusz XI	121	52	9
Scenariusz XII	137	33	35

6.3. Wyniki dla zbioru 1 000 000 rekordów

Zwiększenie rozmiaru bazy do poziomu 1 000 000 rekordów oraz wybranie parametrów technicznych z zestawu 1 podkreśliło różnice między analizowanymi systemami, co pokazują wyniki zanotowane w tabeli 10. Wyraźną przewagę uzyskał w tym przypadku MS SQL. Czasy operacji dla wszystkich analizowanych systemów uległy znacznemu wydłużeniu. Najgorsze wyniki uzyskała baza MySQL.

Tabela 10. Wyniki dla konfiguracji testowej 1

Scenariusz	Średni czas [s]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	8.6	3.9	0.003
Scenariusz II	3.4	1.6	2.5
Scenariusz III	98.6	22.9	6.2
Scenariusz IV	29.7	7.0	4.8
Scenariusz V	57.0	4.4	2.4
Scenariusz VI	22.8	3.1	3.4
Scenariusz VII	32.2	14.6	3.6
Scenariusz VIII	39.5	14.1	1.9
Scenariusz IX	6.8	2.5	1.8
Scenariusz X	24.0	27.2	1.0
Scenariusz XI	3.8	2.1	0.4
Scenariusz XII	14.1	11.7	0.03

W przypadku konfiguracji testowej 2 testy dla największego z rozpatrywanych zestawów danych ponownie podkreśliły różnice pomiędzy systemami, co jest widoczne w tabeli 11. Nieunikniony okazał się również wzrost czasu wykonania operacji względem mniejszych rozmiarów bazy, przy zachowaniu identycznych parametrów testowych. W przypadku 10 scenariuszy najszybszy okazał się MS SQL. Dla 9 rozpatrywanych zapytań najwolniejszy okazał się MySQL, dla 3 PostgreSQL.

Tabela 11. Wyniki dla konfiguracji testowej 2

Scenariusz	Średni czas [s]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	4.7	1.6	0.004
Scenariusz II	1.2	1.0	1.2
Scenariusz III	39.3	9.6	2.3
Scenariusz IV	9.5	2.7	1.8
Scenariusz V	19.0	2.3	0.9
Scenariusz VI	7.6	1.1	1.2
Scenariusz VII	14.7	9.4	1.2
Scenariusz VIII	21.5	7.2	0.6
Scenariusz IX	3.0	1.6	0.5
Scenariusz X	10.8	16.1	0.3
Scenariusz XI	1.1	1.1	0.2
Scenariusz XII	3.0	5.5	0.006

Wyniki w przypadku 3 konfiguracji testowej dla zbioru 1 000 000 rekordów ponownie pokazały, że baza MySQL bardzo słabo radzi sobie z dużą liczbą rekordów. Zajęła ona ostatnie miejsce w 10 z 12 rozpatrywanych scenariuszy. Z kolei najlepszy okazał się MS SQL, który był najszybszy dla 9 zapytań testowych. Wyniki uzyskane w tej serii testów zaprezentowano w tabeli 12.

Tabela 12. Wyniki dla konfiguracji testowej 3

Scenariusz	Średni czas [s]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	4.1	1.1	0.006
Scenariusz II	0.83	0.76	1.02
Scenariusz III	22.1	7.0	2.3
Scenariusz IV	7.7	1.6	1.9
Scenariusz V	10.7	1.6	1.0
Scenariusz VI	4.6	0.7	1.3
Scenariusz VII	7.5	4.1	1.4
Scenariusz VIII	10.9	3.7	0.8
Scenariusz IX	1.5	1.3	0.8
Scenariusz X	3.9	9.3	0.4
Scenariusz XI	0.87	0.61	0.14
Scenariusz XII	3.510	2.634	0.005

7. Podsumowanie

W niniejszym artykule dokonano porównania trzech relacyjnych baz danych: MySQL, MS SQL oraz PostgreSQL pod względem ich wydajności. W porównaniu uwzględniono wpływ parametrów technicznych środowiska testowego, znaczenie wielkości zbioru danych testowych oraz szereg różnorodnych scenariuszy testowych. Na tej podstawie można wyciągnąć następujące wnioski.

Nie istnieje jeden uniwersalny i najlepszy system bazodanowy. W zależności od testowanego zapytania, parametrów technicznych, czy rozmiaru przechowywanych danych testowych poszczególne bazy danych mogą osiągać lepsze lub gorsze wyniki.

Dla małych zbiorów danych różnice w uzyskiwanych czasach wynoszą około 1 ms lub nawet mniej. Jest to różnica, która z punktu widzenia człowieka nie jest zauważalna. W takiej sytuacji czas wykonania zapytań nie jest czynnikiem decydującym o wyborze systemu bazodanowego podczas tworzenia aplikacji.

Wówczas należy rozważyć dodatkowe czynniki takie jak dostępność programistów specjalizujących się w danym języku, ceny licencji, czy dostępne funkcjonalności rozważanych systemów.

W przypadku średnich baz danych widoczne jest duże zróżnicowanie wyników w zależności od analizowanego scenariusza oraz znaczna poprawa wydajności w przypadku poprawy parametrów technicznych. W takiej sytuacji warto rozważyć dostępne zasoby sprzętowe oraz konkretne zapytania, które najczęściej będą wykonywane przez tworzoną aplikację.

Podczas testów przeprowadzanych dla dużych zbiorów danych tak jak w przypadku zestawu 3 widać było wyraźną różnicę w wydajności badanych systemów. W większości analizowanych scenariuszy najlepsze czasy uzyskał MS SQL. Uzyskał on niewielką przewagę nad PostgreSQL. Najsłabiej wypadł w tym przypadku MySQL.

Uzyskane wyniki są dość oczekiwane oraz odpowiadają one wynikom przedstawionych w pracach innych autorów [1-4]. Dodatkowo zgodnie z przewidywaniami można zaobserwować, że w przypadku zwiększania wielkości zbioru danych testowych czasy wykonywania zapytań operacji zwiększają się. Analogicznie poprawa parametrów technicznych środowiska uruchomieniowego prowadzi do zwiększenia wydajności, w przypadku każdego z analizowanych systemów bazodanowych.

Literatura

[1] R. Wodyk, M. Skublwska-Paszkowska, Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework, Journal of Computer Sciences Institute 17 (2020) 358-364, https://doi.org/10.35784/jcsi.2279.

[2] G. Dziewit, J. Korczyński, M. Skublewska-Paszkowska, Performance analysis of relational databases Oracle and MS SQL based on desktop application, Journal of Computer Sciences Institute 8 (2018) 263-269, https://doi.org/10.35784/jcsi.693.

[3] S. Stets, G. Kozieł, Analiza porównawcza baz danych – praca dyplomowa magisterska, Politechnika Lubelska, Lublin 2019.

[4] K. Krocz, O. Kizun, M. Skublewska-Paszkowska, Analiza wydajności relacyjnych baz danych MySQL, PostgreSQL, MariaDB oraz H2 – praca dyplomowa magisterska, Politechnika Lubelska, Lublin 2020.

[5] S. Tongkaw, A. Tongkaw, A comparison of database performance of MariaDB and MySQL with OLTP workload, IEEE Conference on Open Systems, Langkawi 2016, 117-119.

[6] C. -O. Truica, F. Radulescu, A. Boicea and I. Bucur, Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database, 20th International Conference on Control Systems and Computer Science, Bucharest 2015, 191-196.

[7] P. Beynon-Davies, Systemy baz danych, Wydawnictwa Naukowo Techniczne, 2003.

[8] Strefa wiedzy Vavatech, https://vavatech.pl/technologie/bazy-danych/mysql, [10.02.2023].

[9] MySQL - Wikipedia, https://en.wikipedia.org/wiki/MySQL, [10.02.2023].

[10] Opis oprogramowania PostgreSQL, https://www.postgresql.org/about/, [12.02.2023].

[11] PostgreSQL - Amazon AWS, https://aws.amazon.com/rds/postgresql/what-is-postgresql/, [12.02.2023].

[12] Microsoft SQL Server - Wikipedia, https://pl.wikipedia.org/wiki/Microsoft_SQL_Server, [20.04.2023].

[13] Opis systemu MS SQL na portalu Atlantic, https://www.atlantic.net/vps-hosting/what-is-mssql/, [21.04.2023].

[14] Konfiguracja kontenera Docker, https://www.baeldung.com/ops/docker-memory-limit, [21.03.2023].