JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 18 (2021) 61–66
Received: 22 September 2020
Accepted: 12 January 2021

Comparison of the performance of relational databases PostgreSQL and MySQL for desktop application

Porównanie wydajności relacyjnych baz danych PostgreSQL oraz MySQL dla aplikacji desktopowej

Bartłomiej Mirosław Klimek*, Maria Skublewska-Paszkowska

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

The aim of this paper was to compare the performance of two relational database management systems PostgreSQL and MySQL. For the purpose of the study a relational database was designed and a piece of software was implemented to connect desktop application with the database system. This software shall also creates entities and relations between them in desired empty schemes on servers for databases. There has also been implemented a desktop application in Java programming language, that allows browsing data stored in database and performing the tests of database performance. Tests addressed basic operations of adding, collecting, updating and deleting data. A hypothesis "PostgreSQL is more efficient for desktop application while loaded with small data, in that case 1000 of queries" was confirmed by obtained results.

Keywords: DBMS; MySQL; PostgreSQL; relational databases; comparison of performance

Streszczenie

Celem niniejszej pracy było porównanie wydajności relacyjnych systemów zarządzania bazami danych PostgreSQL i MySQL. Na potrzeby tego badania została zaprojektowana baza danych oraz opracowano i zaimplementowano oprogramowanie mające łączyć aplikację desktopową z poszczególnymi systemami baz danych. Oprogramowanie to ma również tworzyć encje oraz relacje między nimi w wybranych pustych schematach na bazy na serwerach. Zaimplementowano także aplikację desktopowa w języku programowania Java, pozwalającą na przeglądanie zapisanych danych w bazie oraz przeprowadzenie testów wydajności bazy. Testy dotyczyły podstawowych operacji dodawania, pobrania, aktualizacji i usuwania danych. W pracy postawiono hipotezę "PostgreSQL jest bardziej wydajny dla aplikacji desktopowych podczas małego obciążenia danymi, czyli do 1000 zapytań", która została potwierdzona wynikami uzyskanymi z przeprowadzonych badań.

Słowa kluczowe: SZBD; relacyjne bazy danych; porównanie wydajności

1. Wstęp

Obecnie bazy danych towarzyszą ludziom na co dzień, chociaż nie zawsze są tego świadomi. Praktycznie wszędzie spotykają się z aplikacjami korzystającymi z baz danych, czy to przeglądając Internet, czy korzystając z tych desktopowych, zainstalowanych na swoich komputerach.

Baza danych jest w uproszczeniu zwyczajnym skomputeryzowanym systemem przechowywania rekordów, którego głównym celem jest przechowywanie i dostęp do informacji oraz ich aktualizacja [5].

Pierwsze systemy bazodanowe znacznie różnią się od najczęściej używanych w dzisiejszych czasach rozwiązań i były one zapoczątkowane w latach 60 ubiegłego wieku i były to m.in. model sieciowy CODASYL oraz model hierarchiczny o nazwie IMS [6].

Relacyjny model baz danych przedstawił w roku 1970 pracownik firmy IBM Edgar F. Codd, który opublikował poświęconą im pracę. Do tamtej pory głównie używanymi były modele hierarchiczny i sieciowy, a praca ta zrewolucjonizowała przechowywanie danych w komputerach. Autor argumentował poszukiwania nowego modelu "niezależnością danych" i twierdził, że zaprezentowany przez niego model okazał się lepszy od modeli grafowego czy sieciowego [7].

Wraz z rozwojem komputerów systemy informatyczne coraz bardziej się rozrastają, dlatego potrzeba, aby czas dostępu do danych był jak najkrótszy. Dlatego też praca ta skupia się na zbadaniu wydajności dwóch syste-

*Corresponding author
Email address: bartlomiej.klimek@pollub.edu.pl (B.M. Klimek)

mów relacyjnych baz danych PostgreSQL oraz MySQL na przykładzie aplikacji desktopowej.

2. Przegląd literatury

Rozwój technologi powoduje, że pojawiają się prace naukowe dotyczące porównań wybranych zagadnień, w tym systemów bazodanowych, zarówno relacyjnych, jak i nierelacyjnych. Autorzy pracy "Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage" [1] przedstawiają czasowe porównania operacji CRUD (CREATE, READ,UPDATE, DELETE) z podziałem na bazy danych racjonalne i nieracjonalne. W celu przeprowadzenia eksperymentu zostały zaimplementowane dwie analogiczne aplikacje w języku Java, jedna działająca z systemem MySQL, a druga z systemem CouchDB. W artykule przedstawiono dwie struktury danych dla nierelacyjnego systemu CouchDB: pierwsza struktura, w którym każdy dokument musi zawierać odniesienie do innego, a druga struktura, w której każdy dokument zawiera wszystkie dane każdego podmiotu. Druga struktura okazała się bardziej wydajna czasowo niż w podejściu relacyjnym.

Porównanie wydajności czterech systemów relacyjnych z uwzględnieniem wirtualizacji zostało przedstawione w artykule "Porównanie wydajności baz danych MySQL, MSSQL, PostgreSQL oraz Oracle z uwzględnieniem wirtualizacji" [2]. Przeprowadzono serię eksperymentów z użyciem aplikacji, w której zaimplementowane zostały testy do pomiaru czasu wykonania analizowanych instrukcji. Każde zapytanie zostało zmierzone 100-krotnie, a pierwszy pomiar odrzucony. Uzyskane rezultaty potwierdziły hipotezę, że najszybszym systemem bazodanowym jest Oracle.

Wydajność trzech systemów bazodanowych MySQL, PostgreSQL oraz MangoDB wykorzystując aplikację webową napisaną w Django przedstawiono w [3]. W uzyskanych wynikach najbardziej wydajnym systemem okazał się PostgreSQL. W przypadku aplikacji testowej wykonanej na potrzeby tego artykułu, różnica w szybkości między PostgreSQL, a MySQL wynosi około 17%.

Wydajność czterech systemów bazodanowych MySQL, PostgreSQL, MariaDB oraz H2 została przedstawiona w artykule [4]. Porównanie wydajności relacyjnych baz danych polegało na przeprowadzeniu operacji dodania, aktualizacji, usuwania i wybierania danych. Każde badanie zostało wykonane 10 razy dla 1, 100, 2500, 5000, 7500 i 10000 rekordów. MySQL wykazała słabą wydajność podczas pracy z dużą ilością danych. Najlepsze średnie czasy wykonania wszystkich operacji (za wyjątkiem aktualizacji) uzyskała baza danych H2. 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.

Artykuł naukowy "Database systems Performance Evaluation for Iot Applications" [8] skupia się na porównaniu wydajności pomiędzy relacyjnymi systemami bazodanowymi MySQL, PostgreSQL i nierelacyjnym systemem MongoDB dla aplikacji w internecie przedmiotów (ang. Internet of Things, IoT). Autorzy przeprowadzili badania na serwerze lokalnym przy użyciu skryptów napisanych w języku Python, aby zminimalizować opóźnienia. Badane scenariusze przewidywały pobranie danych do 500000 rekordów, operacje dodania danych do 500 zapytań, oraz agregację danych dla maksymalnie 10 zapytań.

Według autorów w przypadku małej liczby pobieranych rekordów najlepiej sprawdził się system PostgreSQL, natomiast w przypadku większej liczby pobieranych rekordów (powyżej 20000) MySQL okazał się lepszy niż PostgreSQL, ale gorszy niż MongoDB.

W drugim scenariuszu badawczym przewidującym dodawanie danych dla niewielkiej liczby zapytań najlepiej sprawdził się system MongoDB, drugie miejsce zajął PostgreSQL, a ostatnie MySQL z podobnymi czasami do PostgreSQL. W przypadku większej liczby zapytań najlepiej sprawdził się PostgreSQL.

W przypadku agregacji dla małej ilości danych najlepiej sprawdził się PostgreSQL, natomiast dla większej liczby rekordów najlepszym okazał się MySQL.

Praca naukowa "Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis" [9] skupia się na porównaniu pięciu najpopularniejszych systemów zarządzania bazami danych. Zbadano między innymi operacje dodawania, pobierania i aktualizacji od 6 do 400000 rekordów. Autorzy doszli do wniosków, że najlepiej sprawdził się system SQLite, natomiast drugie miejsce w badaniach zajął PostgreSQL, który był od niego lepszy w przypadkach dodawania rekordów, ale nie dorównał mu w pozostałych przypadkach. Systemy Oracle i SQL Server miały bardzo podobne wyniki, a najgorzej sprawdził się MySQL.

Przedstawiony przegląd literatury dowodzi, że wydajność systemów MySQL oraz PostgreSQL jest aktualnym tematem. Uzyskane w pracach innych wyniki pozwalają na zdefiniowanie tezy "PostgreSQL jest bardziej wydajny dla aplikacji desktopowych podczas małego obciążenia danymi, czyli do 1000 zapytań".

3. Obiekty badań

3.1. PostgreSQL

PostgreSQL jest szybkim, relacyjnym systemem zarządzania bazami danych. W testach porównawczych wyprzedza lub dorównuje osiągom wielu innych systemów przeznaczonych do zarządzania bazami danych, czy to otwartoźródłowych, czy też zastrzeżonych przez prawo [10].

Wspiera dużą część standardu SQL oraz oferuje wiele nowych rozwiązań takich jak złożone kwerendy, klucze obce, wyzwalacze a także inne oraz może zostać rozszerzony przez użytkownika m.in. o nowe typy danych, funkcje, czy operatory [11].

Początki tego systemu sięgają roku 1986, gdy rozpoczął się projekt POSGRES prowadzony przez profesora Michaela Stonebrakera na Uniwersytecie Kalifornijskim, gdzie rozwinął się z projektu Ingres, skąd wywodzi się jego nazwa oznaczająca w wolnym tłumaczeniu "następujący po Ingresie" (ang. post-Ingres). Z czasem projekt zmienił nazwę na Postgres95, a kolejno w roku 1996 na PostgreSQL, która to utrzymała się do dnia dzisiejszego i podkreśla zgodność ze standardem SQL [12].

PostgreSQL wspiera systemy operacyjne takie jak Linux, Windows oraz macOS, oraz jest zgodny ze standardem ACID (ang. Atomicity, Consistency, Isolation, Durability) dla transakcji zapewniając niepodzielność, spójność, izolację i trwałość danych [13].

3.2. MySQL

MySQL jest najpopularniejszym otwartoźródłowym systemem zarządzania bazami danych. Powstał w roku 1995, natomiast od 2010 jest własnością firmy Oracle i jest przez nią rozwijany oraz dystrybuowany w dwóch wersjach: wersji darmowej "The Community Edition", oraz wersji płatnej "Enterprise Edition", która to jest rozszerzona względem wersji darmowej. Wersja bezpłatna w zupełności wystarcza do zapewnienia niezawodności i bezpieczeństwa dla aplikacji [14].

MySQL w wersji 8.0 ma być nową generacją baz danych, dzięki obsłudze SQL jak i NoSQL z notacją JSON, a także wspólne wyrażenia tablicowe, które upraszczają i poprawiają przejrzystość kodu SQL [14].

4. Metoda badań

Do badań została użyta specjalnie na tę potrzebę zaprojektowana aplikacja w języku programowania Java. Do połączenia aplikacji z bazą danych opracowano oprogramowanie REST API które umożliwia na komunikację z serwerem przy pomocy protokołu HTTP oraz wysyłanie zapytań w notacji JSON. API zostało zaprojektowane tak, by można było zbadać operacje dodawania danych oraz ich pobierania z relacjami z inną encją, bądź bez relacji. Można również zbadać operacje aktualizacji danych i ich usuwania, w obu tych przypadkach bez żadnych relacji łączących.

Każdy test zostanie wykonany 10 razy dla 1, 100 i 1000 powtórzeń pętli obsługującej daną operację, dla każdego badanego systemu baz danych. Każde powtórzenie pętli wysyła do bazy jedno zapytanie w notacji JSON. Aby wyniki testów były miarodajne badania zostały przeprowadzone na pustych schematach baz danych, osobnych dla każdego zestawu operacji dla encji bez relacji, kolejno: dodawania, pobrania, aktualizacji i usuwania, a osobno dla encji z relacjami w tym przypadku scenariusze dodawania, a następnie pobierania danych. Po każdym takim zestawie operacji schemat bazy danych został usunięty, a jego miejsce zajmował nowy, pusty schemat. Dla wyników została obliczona średnia oraz odchylenie standardowe w zaokrągleniu do części dziesiętnej.

Wszystkie badania zostały przeprowadzone na bazie danych opracowanej specjalnie na potrzeby aplikacji desktopowej służącej do zarządzania bazą filmów i seriali. Schemat bazy widnieje na rysunku 1. Do badań przeprowadzanych na encji bez relacji została użyta encja 'movie', natomiast do badań dla encji z relacjami zostały użyte tabele 'tvseries' oraz 'tvseassons'. Typ LONGTEXT przy wybranych kolumnach został użyty na wypadek dłuższego opisu, natomiast w badaniach nie został wykorzystany odpowiedni dla tego typu rozmiar danych i mógłby zostać on zastąpiony przez typ VARCHAR.

Do przeprowadzenia badań służył moduł testowy aplikacji przedstawiony na rysunku nr 2.

Aby wykonać wybrany scenariusz konieczne było zaznaczenie pola znajdującego się obok opisu, oraz wybranie liczby powtórzeń w górnej części widoku przedstawionego na rysunku 2. Możliwe jest także wybranie początkowego klucza głównego dla operacji pobierania, aktualizacji i usuwania, jednak wszystkie przeprowadzone badania zostały wykonane od klucza głównego o numerze 1.

Czasy wykonywania wybranego scenariusza zostały zmierzone w aplikacji desktopowej przy pomocy funkcji System.nanoTime() zwracającej czas uruchomionej Wirtualnej Maszyny Javy [15], której wartość została zapisana do zmiennej przed wykonaniem wybranego scenariusza, a po jego wykonaniu do nowej zmiennej została zapisana różnica aktualnej wartości zwróconej przez funkcję System.nanoTime() i zmiennej zawierającej zapisany czas sprzed wykonania scenariusza. Czasy zostały zwrócone w konsoli zintegrowanego środowiska programistycznego.

Badania zostały przeprowadzone na komputerze Asus ROG GL553 z 64 bitowym procesorem Intel Core i5-7300HQ 2.50 GHz, pamięcią RAM 16GB oraz 64 bitowym systemem operacyjnym Windows 10 Home w wersji 1909.

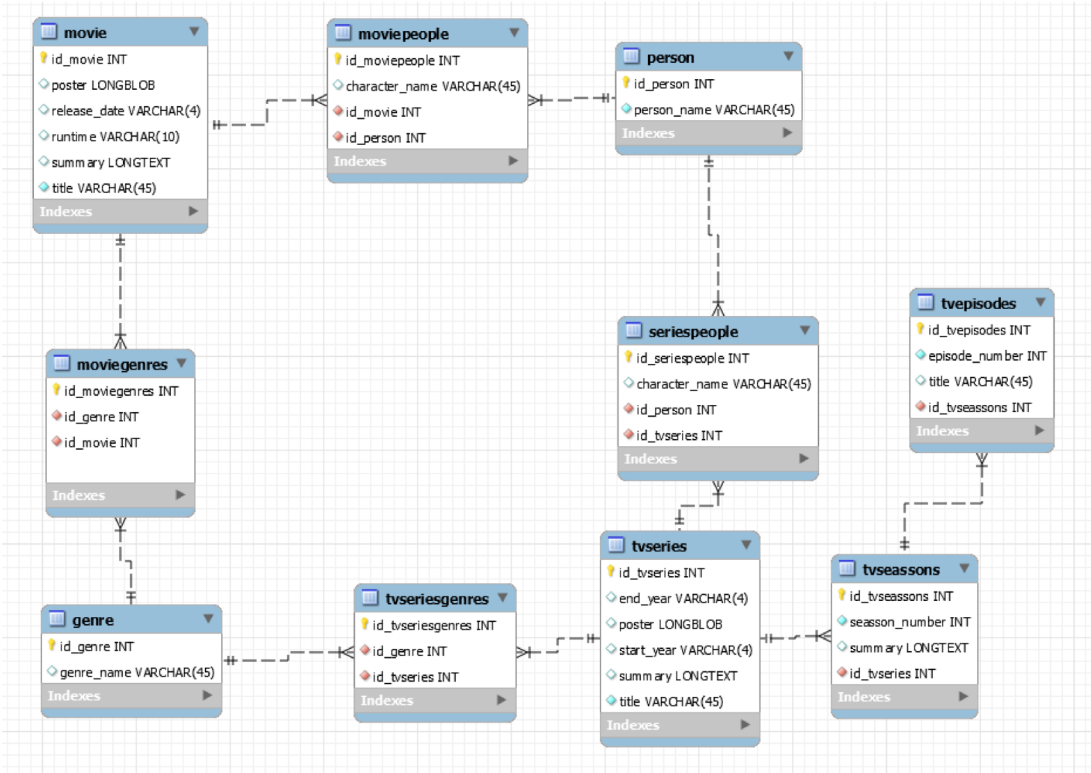
5. Wyniki badań

5.1. Operacja dodania danych

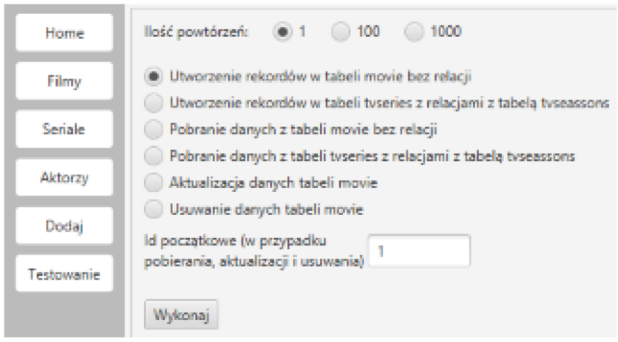
Dla operacji dodawania danych zostały przygotowane dwa scenariusze. Pierwszy przewidujący dodanie danych do encji bez żadnych relacji, a drugi dodanie danych do dwóch encji, które zostaną połączone relacją.

W tabeli 1 przedstawiono średnie czasy oraz odchylenie standardowe dla operacji dodania danych bez relacji dla poszczególnych baz danych. Można tu zaobserwować, że w przypadku PostgreSQL czasy odchylenia standardowego są niższe. Tendencja ta utrzymuje się przy kolejnych scenariuszach.

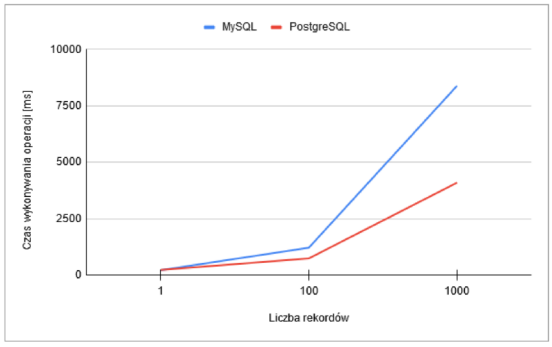
Na rysunku 3 przedstawiono wykres obrazujący średnie czasy wykonania operacji dodawania bez relacji dla poszczególnych baz danych. Widać na nim rosnącą wraz z ilością danych przewagę systemu PostgreSQL, dla której czas przy 100 i 1000 powtórzeń jest o połowę krótszy.



Rysunek 1: Diagram EER (ang. Enhanced entity-relationship).



Rysunek 2: Moduł testowania bazy danych w aplikacji desktopowej.



Rysunek 3: Średni czas operacji dodawania danych bez relacji.

Tabela 1: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji dodania danych bez relacji w milisekundach.

L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	214,2 ±27, 4	226,2 ±9, 2
100	1211 ±197, 1	736,8 ±29, 8
1000	8390,5 ±158, 9	4095.7 ±83, 2

W przypadku operacji dodawania danych z relacją sytuacja jest podobna, jak w przypadku dodawania danych bez relacji (tabela 2).

Wykres przedstawiony na rysunku 4 obrazuje, że czasy dodania danych z relacjami są większe niż w poprzednim przypadku. Różnica pomiędzy badanymi systemami baz danych w pewnym momencie jest prawie dwukrotna, na korzyść systemu PostgreSQL.

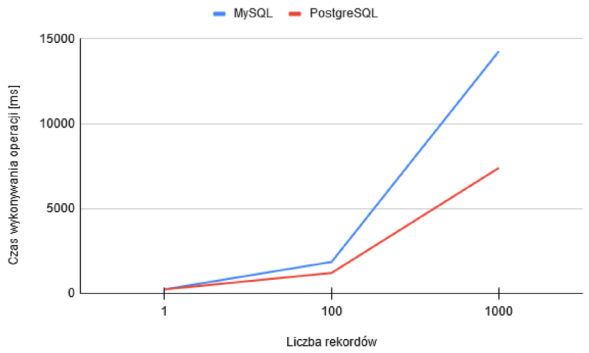
Tabela 2: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji dodania danych z relacją w milisekundach.

L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	240,4 ±3, 9	243,5 ±7, 2
100	1856,3 ±83, 2	1211 ±44
1000	14272,7 ±1014, 4	7399 ±54, 2

5.2. Operacja pobierania danych

Podobnie do operacji dodawania danych w tym przypadku również zbadano dwa scenariusze. Pierwszy przewidujący pobieranie danych bez relacji, a drugi z relacjami.

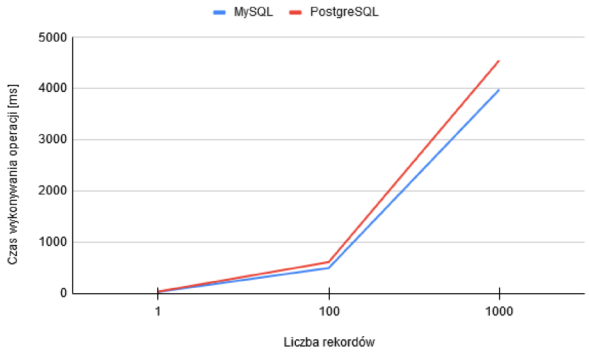
W przypadku pobierania danych bez relacji niewielką przewagę zyskał system zarządzania bazami danych MySQL, a czasy wykonywania operacji są do siebie zbliżone dla obu systemów co można zaobserwować w tabeli 3 i na rysunku 5.



Rysunek 4: Średni czas operacji dodawania danych z relacją.

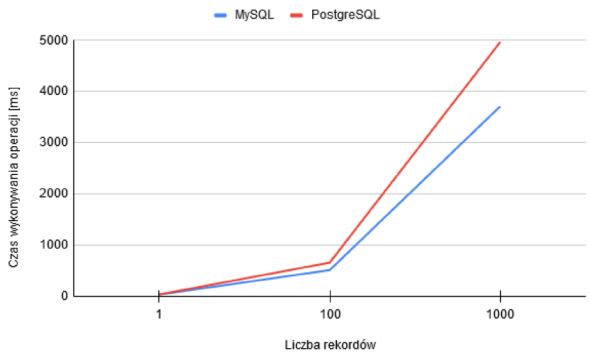
Tabela 3: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji pobierania danych bez relacji w milisekundach.

L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	32,7 ±1, 8	37,5 ±2, 15
100	498,1 ±31, 3	612,5 ±28, 3
1000	3982,3 ±174, 2	4553,9 ±51, 3



Rysunek 5: Średni czas operacji pobierania danych bez relacji.

W przypadku pobierania danych z relacją sytuacja jest podobna jak w przypadku pobierania danych bez relacji, gdzie przewagę miał system MySQL, natomiast tym razem różnica w czasach jest nieco większa. Wyniki przedstawiono w tabeli 4 i na rysunku 6.



Rysunek 6: Średni czas operacji pobierania danych z relacją.

Tabela 4: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji pobierania danych z relacją w milisekundach.

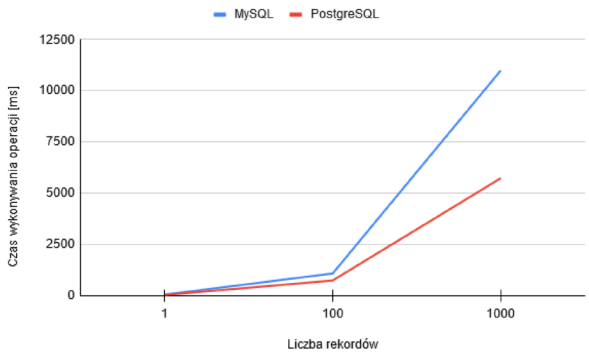
L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	35,2 ±2, 9	37 ±2, 2
100	513,1 ±38, 1	658,2 ±26, 4
1000	3713,4 ±131, 6	4969,7 ±60

5.3. Operacja aktualizacji danych

Operacja aktualizacji danych, której wyniki zostały przedstawione w tabeli 5 i na rysunku 7. Tak jak w przypadku operacji dodawania danych, operacja ta wykonuje się szybciej w przypadku systemu PostgreSQL.

Tabela 5: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji aktualizacji danych bez relacji w milisekundach.

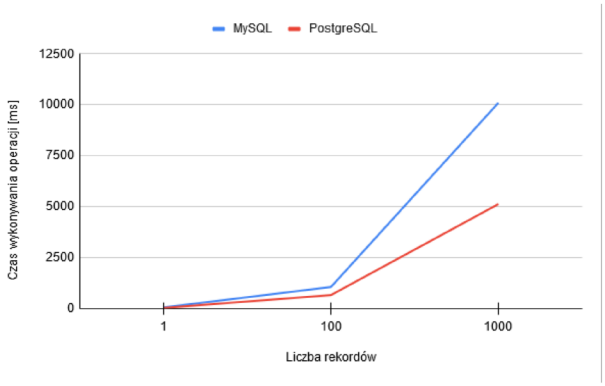
L. rekordów	MySQL [ms]	PostgreSQL [ms]
1	49,4 ±14, 5	31,8 ±2, 9
100	1080,7 ±52, 4	739,7 ±23
1000	11002 ±178, 5	5738,3 ±168, 8



Rysunek 7: Średni czas operacji aktualizacji danych bez relacji.

5.4. Operacja usuwania danych

Operacja usuwania danych, dla której wyniki zostały przedstawione w tabeli 6 oraz na wykresie na rysunku 8 prezentuje wyniki podobne do operacji aktualizacji danych. Ponownie lepsze czasy osiągnął tu system PostgreSQL.



Rysunek 8: Średni czas operacji usuwania danych bez relacji.

Tabela 6: Średni czas i odchylenie standardowe dla 10 powtórzeń operacji usuwania danych bez relacji w milisekundach.

L. rekordów	MySQL	PostgreSQL
1	$56{,}9 \pm 20{,}8$	$33{,}3 \pm 2{,}2$
100	$1060{,}4 \pm 61{,}5$	$663{,}6 \pm 23{,}4$
1000	$10095{,}2 \pm 203{,}8$	$5127{,}4 \pm 88{,}3$

6. Wnioski

W niniejszym artykule przedstawiono badania porównawcze wydajności PostgreSQL i MySQL na podstawie aplikacji desktopowej, napisanej w języku Java. Na potrzeby artykułu stworzono także dwie aplikacje RESTAPI służące do połączenia aplikacji z wybranymi systemami zarządzania bazą danych. Na podstawie przedstawionych wyników badań można wyciągnąć wnioski:

1. W przypadku operacji dodania, aktualizacji i usuwania danych z bazy danych system bazodanowy PostgreSQL okazał się blisko dwukrotnie wydajniejszy.
2. W przypadku operacji pobierania danych z bazy różnice były znacząco mniejsze w porównaniu do innych operacji, lecz lepsze wyniki uzyskał system MySQL.
3. Duże różnice w czasach wykonywania poszczególnych operacji na korzyść systemu PostgreSQL mogą wynikać z jego lepszego przystosowania do notacji JSON [16].
4. Postawiona teza "PostgreSQL jest bardziej wydajny dla aplikacji desktopowej podczas małego obciążenia danymi, czyli do 1000 zapytań" została udowodniona.

Literatura

[1] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Ş. Győrödi, G. A. Gabor, D. Pecherle. Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage, Applied Sciences 10(23) (2020) 8524–8545.

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

[3] B. Nejman, B. Pańczyk, Wydajność pracy z bazami danych aplikacji tworzonych w Django, Journal of Computer Science Institute 11 (2019) 82–85.

[4] K. Krocz, O. Kizun, M. Skublewska-Paszkowska, Analiza porównawcza wydajności relacyjnych baz danych MySQL, PostgreSQL, MariaDB oraz H2, Journal of Computer Science Institute 14 (2020) 1–7.

[5] C. J. Date, An Introduction to Database Systems, Pearson Education, 2004.

[6] Historia baz danych, https://www.quickbase.com/articles/timeline-of-database-history, [11.09.2020].

[7] Historia relacyjnych baz danych, https://twobithistory.org/2017/12/29/codd-relational-model.html, [12.09.2020].

[8] C. Asiminidis, G. Kokkonis, S. Kontogiannis, Database systems Performance Evaluation for IoT Applications, International Journal of Database Management Systems (IJDMS) 10(6) (2018) 1–14.

[9] Md. I. Hossain, S. Mahmud, T. D. Santa, Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis, Daffodil International University Dhaka, Bangladesh, 2019.

[10] R. Obe, L. Hsu, PostgreSQL: Up and Running. A practical guide to the advanced open source database, Third edition, O'Reilly Media, 2018.

[11] Dokumentacja PostgreSQL, https://www.postgresql.org/docs/11/intro-whatis.html, [11.09.2020].

[12] Dokumentacja PostgreSQL: Historia, https://www.postgresql.org/docs/8.4/history.html, [11.09.2020].

[13] Artykuł porównujący PostgreSQL i MySQL, https://www.2ndquadrant.com/en/postgresql/postgresql-vs-mysql/, [11.09.2020].

[14] E. Vanier, B. Shah, T. Malepati, Advanced MySQL 8, Packt Publishing Ltd., 2019.

[15] Dokumentacja java: Class System, https://docs.oracle.com/javase/7/docs/api/java/lang/System.html, [18.09.2020].

[16] Blog enterprisedb.com, https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features, [20.09.2020].