JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 31 (2024) 89–96
Received: 22 February 2024
Accepted: 2 April 2024

Comparative analysis of the performance of selected database management system

Analiza porównawcza wydajności wybranych systemów zarządzania bazami danych

Piotr Marcin Tracz*, Małgorzata Plechawska-Wójcik

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

Abstract
Currently, in the era of large amount of generated data, a big challenge is fast data processing. The aim of the article is to conduct a comparative analysis of the performance of relational and non-relational databases using MS SQL Server, MongoDB, and CouchDB as examples. The comparison was carried out based on developed research scenarios designed to examine the execution time of CRUD operations. A performance ranking of databases was established based on the obtained results, with MS SQL Server taking the first place, MongoDB the second one, and CouchDB the third one. For the operations of deletion, downloading, and updating of data, MS SQL Server proved to be the most efficient database. Only in the operation of inserting data did MongoDB turn out to be the most efficient database.

Keywords: database management system; non-relational databases; relational databases; performance

Streszczenie
Obecnie w erze dużej ilości generowanych danych, dużym wyzwaniem staje się szybkie przetwarzanie danych. Celem artykułu jest dokonanie analizy porównawczej wydajności relacyjnych i nierelacyjnych baz danych na przykładzie MS SQL Server, MongoDB oraz CouchDB. Porównanie zrealizowano na podstawie opracowanych scenariuszy badawczych zaprojektowanych w taki sposób, aby zbadać czas realizacji operacji CRUD. Na podstawie otrzymanych wyników utworzono ranking wydajności baz danych, w którym pierwsze miejsce zajęła baza MS SQL Server, drugie MongoDB, a trzecie CouchDB. Dla operacji usuwania, pobierania i aktualizacji danych najwydajniejszą bazą danych okazała się baza MS SQL Server. Jedynie przy operacji wstawiania danych najwydajniejszą bazą okazało się MongoDB.

Słowa kluczowe: system zarządzania bazą danych; nierelacyjne bazy danych; relacyjne bazy danych; wydajność

*Corresponding author
Email address: piotr.tracz1@pollub.edu.pl (P. Tracz), m.plechawska@pollub.pl (M. Plechawska-Wójcik)

1. Wstęp

W dzisiejszych czasach, w których technologia informatyczna towarzyszy ludziom w codziennym życiu i staje się bardzo ważną częścią życia ludzkiego widoczny jest wzrost ilości generowanych danych. Szacuje się, że ilość nowych danych rośnie w tempie wykładniczym [1]. Codziennie miliardy ludzi, korzystając ze swoich telefonów i laptopów, na których pracują, bawią się i kontaktują z innymi osobami, wytwarzają bardzo duże ilości danych. Aby zaspokoić potrzeby klientów, którymi są szybkie działanie aplikacji i ich niezawodność, opracowywane są coraz nowsze rozwiązania bazodanowe, które mają zapewnić szybsze przetwarzanie danych. Generowane przez klientów dane są również wykorzystywane przez firmy do podejmowania decyzji biznesowych [2], gdzie czas przetworzenia tak dużej ilości danych jest bardzo istotny.

Tradycyjnym podejściem do zapisywania i przechowywania danych jest użycie relacyjnych baz danych [3]. Jednakże, wyzwaniem dla tych baz jest odpowiednia wydajność podczas przetwarzania dużych zbiorów danych tzw. Big Data [4]. Bazy SQL są stosunkowo wydajnymi bazami danych. Jeśli jednak relacyjny model danych zawiera wiele relacji między tabelami, wówczas pobranie danych wymaga użycia wielu złączeń, co znacznie wydłuża czas trwania tej operacji [5].

Obecnie popularne stają się bazy NoSQL, które zastępują tradycyjne relacyjne bazy danych, ponieważ uważa się je za bardziej skalowalne i szybsze [6]. Przykładem ich zastosowania jest zapisywanie danych pochodzących z mediów społecznościowych [7], które są nieustrukturyzowanymi zbiorami danych i często pochodzą z różnych źródeł [8]. Głównym zarzutem stawianym bazom danych NoSQL jest bezpieczeństwo i prywatność, ponieważ nie były one projektowane z myślą o szyfrowaniu danych czy uwierzytelnianiu [9].

W niniejszym artykule dokonano analizy porównawczej wydajności relacyjnych i nierelacyjnych baz danych na podstawie czasu realizacji operacji CRUD (create, read, update, delete). Nasuwa się więc pytanie, czy bardziej wydajne są tradycyjne relacyjne bazy danych, czy jednak coraz popularniejsze bazy NoSQL. Aby odpowiedzieć na to pytanie, zaprojektowano bazę danych, która przechowuje dane o tematyce kolarskiej i zbadano czas wstawiania, usuwania, pobierania i aktualizacji danych. Zdefiniowano również następującą tezę artykułu: Relacyjne bazy danych są bardziej wydajne w kontekście operacji CRUD niż bazy danych NoSQL. W tym artykule poruszono tematykę wydajności baz danych, ponieważ

jest to bardzo ważny aspekt w działaniu aplikacji internetowych, a także w systemach Business Inteligence, gdzie przetwarzane są duże zbiory danych i czas tego przetwarzania jest bardzo istotny.

2. Przegląd literatury

Analiza porównawcza wydajności relacyjnych i nierelacyjnych baz danych na podstawie czasu wykonania operacji została przedstawiona na przykład w artykułach naukowych [10-12]. Najczęściej w analizie wykorzystywano nierelacyjną bazę MongoDB, a także popularne bazy relacyjne takie jak np. MS SQL Server, PostgreSQL czy MySQL.

W artykule [13] porównano wydajność relacyjnych i nierelacyjnych systemów zarządzania bazami danych typu open-source. Wykorzystano do tego badania relacyjną bazę MySQL i nierelacyjną bazę CouchDB. Przeprowadzono porównanie czasu wykonania operacji CRUD (create, read, update, delete) dla różnej ilości danych za pomocą dwóch podobnych aplikacji napisanych w języku Java (jedna aplikacja dla MySQL, druga dla CouchDB), które umożliwiły wyznaczenie czasu tych samych operacji CRUD wykonywanych przez te aplikacje na różnych bazach danych, dla różnych ilości danych i różnej złożoności zapytań. Wyniki, które otrzymano, pokazują, że dla małej liczby rekordów relacyjna baza MySQL jest wydajniejsza niż nierelacyjna baza CouchDB, ale dla dużych zbiorów danych to nierelacyjna baza danych jest bardziej wydajna.

W artykule [14] dokonano porównania wydajności baz danych SQL i NoSQL wykorzystując do tego celu Microsoft SQL Server i MongoDB. Badania przeprowadzono dla dużego zbioru danych w formacie JSON, który nie wymaga normalizacji. Opisano czym są relacyjne i nierelacyjne bazy danych, a także przedstawiono charakterystyczne cechy bazy MongoDB. Aby dokonać porównania wydajności dwóch baz danych wykonano 4 eksperymenty (operacja wstawienia rekordów do baz danych, operacja wyszukania losowego ciągu znaków, operacja wyszukania danego identyfikatora, operacja aktualizacji rekordu). Otrzymane wyniki badań pokazują, że dla dużej ilości nieustrukturyzowanych danych w formacie JSON, wyższą wydajnością charakteryzuje się baza MongoDB. Aby dokonać wyboru, która baza danych będzie bardziej wydajna, należy dokonać analizy jakie dane i w jakim formacie będą przetwarzane.

W artykule [15] przedstawiono porównawcze testy wydajnościowe dwóch dokumentowych baz NoSQL: MongoDB i CouchBase, a także relacyjnej bazy PostgreSQL. Testy wydajnościowe przeprowadzono przy użyciu danych geoprzestrzennych. Zdefiniowano dwa zapytania, które były uruchamiane dla rosnącej liczby użytkowników. Czas trwania zapytań został obliczony przez narzędzie Apache JMeter. Otrzymane wyniki pokazują, że dla zapytań gdzie jedynie pobierano dane, bez wykonania kalkulacji bazy danych NoSQL były szybsze niż relacyjna baza danych. Dla zapytań, gdzie użyto funkcji kalkulujących, bazy NoSQL zapewniały wysoką wydajność, lecz dla danych ze skomplikowaną strukturą relacyjna baza danych okazała się bardziej wydajna.

3. Charakterystyka systemów baz danych

3.1. Relacyjne bazy danych

Twórcą relacyjnego modelu danych jest Edgar Frank Codd, który w 1970 roku, w swoim artykule [16] opisał podstawowe założenia tego modelu. Początki relacyjnych baz danych sięgają końca lat 70. i początku 80. XX wieku, kiedy to powstały pierwsze systemy zarządzania bazami danych, takie jak Oracle i DB2 [17]. Obecnie bazy danych SQL są najpopularniejszym rozwiązaniem bazodanowym.

W relacyjnych bazach danych wykorzystuje się dane, umieszczone w tabelach, które posiadają wiersze i kolumny. Każdy wiersz to pojedyncza jednostka opisywanych obiektów, a kolumny opisują daną cechę obiektu. Aby zidentyfikować wiersze w tabeli stosuje się klucz główny, który jest kolumną lub zbiorem kolumn w tabeli. W relacyjnych bazach danych występują związki między tabelami. Związki są tworzone przez klucz główny jednej tabeli i klucz obcy drugiej tabeli, dzięki czemu można powiązać dane z dwóch tabel. Klucz obcy to kolumna, której wartość wskazuje na klucz główny innej tabeli [18].

Microsoft SQL Server jest jednym z najpopularniejszych relacyjnych systemów zarządzania bazami danych. W badaniu została wykorzystana darmowa wersja oprogramowania Express, która bardzo dobrze sprawdza się do tworzenia nierozbudowanych stron internetowych i aplikacji, np. ze względu na mniejszy maksymalny rozmiar bazy danych i mniejszą maksymalną liczbę rdzeni procesora w stosunku do wersji standardowej. Jest to baza danych wyprodukowana i wspierana przez firmę Microsoft i korzysta z języka Transact-SQL, który pozwala na tworzenie pętli, instrukcji warunkowych, wyzwalaczy, a także procedur i funkcji składowanych.

3.2. Nierelacyjne bazy danych

Historia NoSQL sięga 1998 roku, kiedy to Carlo Strozzi użył tej nazwy dla swojego relacyjnego systemu bazodanowego [19]. Nazwał go w ten sposób, ponieważ jego rozwiązanie nie korzystało z języka SQL. Pierwotnie termin NoSQL został więc użyty do określenia relacyjnej bazy danych, w której nie używano SQL. Termin NoSQL nie był popularny aż do 2009 roku, kiedy to odbyło się spotkanie w San Francisco dotyczące nierelacyjnych baz danych. Spotkanie to nazwano NoSQL, co było propozycją Erica Evansa [20]. Od tego wydarzenia termin NoSQL stał się popularny i w większości przypadków jest odbierany w kontekście nierelacyjnych baz danych.

Dane w bazach typu NoSQL, w odróżnieniu od relacyjnych baz danych, nie są przechowywane w tabelach, a także nie muszą być ustrukturyzowane. Bazy NoSQL nie posiadają stałej struktury, a ponadto akceptują różne typy danych. To w jaki sposób dane są przechowywane w bazie danych definiuje jej rodzaj. Do najważniejszych modeli danych dla nierelacyjnych baz danych zalicza się: klucz-wartość, dokument, rodzina kolumn, graf.

3.2.1. MongoDB

MongoDB to dokumentowa baza NoSQL, która przechowuje dane w dokumentach w binarnej formie JSON, czyli BSON. Zbiory dokumentów tworzą kolekcje, które składają się na bazy danych. Dane nie muszą mieć ściśle określonego schematu, ponieważ struktura w dokumentach jest elastyczna, a język, który wykorzystywany jest do przetwarzania danych to JavaScript [21]. MongoDB oferuje graficzny interfejs użytkownika MongoDBCompass, a także środowisko wiersza poleceń MongoDB Shell, w których można wykonywać polecenia i zarządzać bazami danych

3.2.2. CouchDB

CouchDB jest dokumentową bazą NoSQL, w której dokumenty są zapisywane w formacie JSON. Dokumenty przechowywane są w bazach danych, co odróżnia tę bazę od MongoDB, w której dokumenty zapisywane są do odpowiedniej kolekcji w bazie danych. Do przetwarzania danych używa się języka JavaScript. Baza stworzona została z myślą o zastosowaniach online i mobilnych, nadając się do obsługi systemów CRM (zarządzanie relacjami z klientami) oraz CMS (zarządzanie treścią). Firma Apache dostarcza aplikację webową Fauxton, w której można zarządzać danymi w bazach danych, a także użytkownikami poprzez udzielanie dostępów i uprawnień [22].

4. Środowisko badawcze

4.1. Opis stanowiska badawczego

Badania przeprowadzono na sprzęcie o następujących parametrach:

- Laptop: HP Pavilion 15
- Procesor: Intel Core i5-1135G7
- Zainstalowana pamięć RAM 32 GB
- System operacyjny: Windows 10 Home 22H2
- Dysk: SSD M.2 PCIe 960G

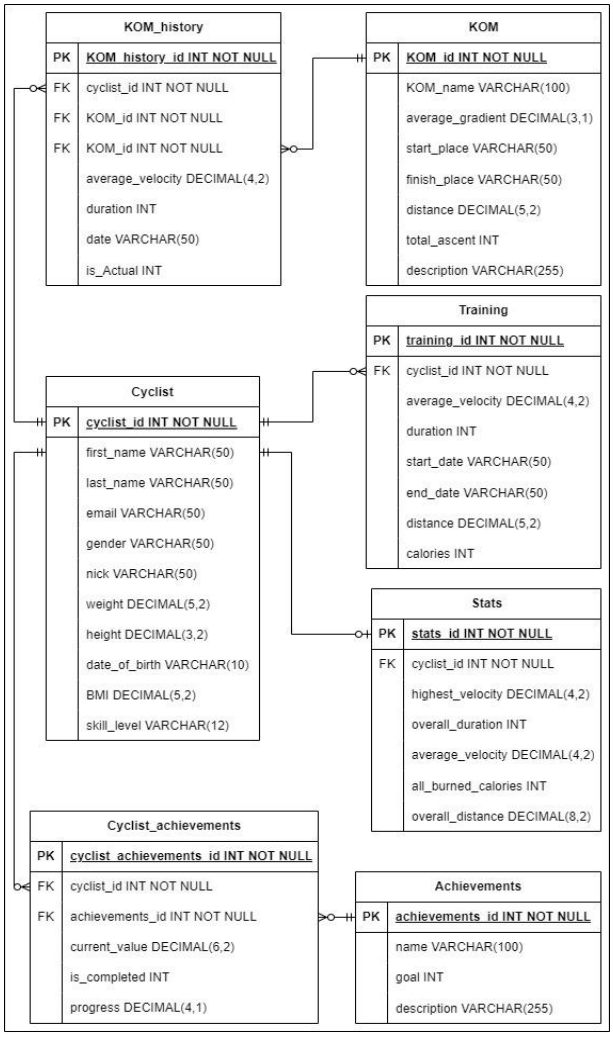
Na opisanym wyżej sprzęcie zainstalowano oprogramowanie umożliwiające wykonanie badań:

- Microsoft SQL Server 2022 Express wraz z graficznym interfejsem użytkownika SSMS w wersji 19.1
- MongoDB 7.0.2 wraz z graficznym interfejsem użytkownika MongoDB Compass 1.40.4 i interfejsem wiersza poleceń MongoDB Shell 2.0.
- CouchDB 3.3.2 wraz z NodeJS w wersji 8.1.0 i biblioteką nano.

4.2. Schematy baz danych

Aby przeprowadzić badania wydajnościowe zaprojektowano bazę danych, która przechowuje dane dotyczące treningów, statystyk i osiągnięć danego kolarza, a także dane o najlepszych wynikach osiąganych na popularnych podjazdach kolarskich. Na Rysunku 1 przedstawiono schemat ERD, który został zaimplementowany w relacyjnej bazie danych. Baza danych zawiera siedem tabel:

- *Cyclist* - zawiera dane o kolarzu.
- *Training* - zawiera parametry wykonanego treningu każdego kolarza.
- *Stats* - zawiera dane opisujące statystykę treningów każdego kolarza.
- *Cyclist_achievements* - zawiera dane o postępach w zdobywaniu osiągnięć danego kolarza.
- *Achievements* - zawiera dane opisujące osiągnięcia, które są do zdobycia przez kolarzy.
- *KOM* - King of the Mountain, zawiera dane opisujące popularne podjazdy kolarskie, na których można się ścigać. *KOM* to termin używany w popularnej aplikacji wspomagającej treningi kolarskie Strava.
- *KOM_history* - zawiera dane o najlepszych czasach przejazdu na danym *KOM*. Kolumna *is_actual* informuje o tym, który przejazd jest najlepszy i kto posiada tytuł króla danego podjazdu.



Rysunek 1: Schemat ERD relacyjnej bazy danych.

Aby stworzyć koncepcyjny schemat nierelacyjnych baz danych, dokonano przekształcenia schematu relacyjnej bazy danych. Bazy NoSQL pozwalają na zapis tych samych danych w zupełnie inny sposób, dlatego tabele zostały przekształcone na kolekcje zawierające zagnieżdżone dokumenty. Każda kolekcja i dokument zagnieżdżony będą odpowiadały tabelom w relacyjnej bazie danych i będą zawierały takie same kolumny. Tabele *KOM* i *KOM_history* zostały przekształcone na kolekcję *KOM*, która zawiera zagnieżdżone dokumenty *KOM_history*, a tabela *Achievements* została przekształcona na kolekcję

bez zmian. Tabele: *Cyclist*, *Training*, *Stats* i *Cyclist_achievements* przekształcono w jedną kolekcję *Cyclist*, która zawiera trzy dokumenty zagnieżdżone *Training*, *Stats* i *Cyclist_achievements*.

5. Scenariusze badawcze

Aby dokonać analizy porównawczej wydajności wybranych systemów zarządzania bazami danych, zdefiniowano scenariusze badawcze, według których zostały wykonane badania. Aby otrzymać jak najbardziej dokładne wyniki i zminimalizować błąd pomiaru, w każdym scenariuszu badawczym wykonano po 30 serii badań dla każdego systemu zarządzania bazą danych dla następującej liczby rekordów w bazie danych: 500, 1000, 2000, 5000, 10000. Dla scenariuszy badawczych, w których badany jest czas odczytywania i aktualizacji danych, przed rozpoczęciem serii badań ładowana jest odpowiednia liczba rekordów do bazy danych. Instrukcje wstawiania i usuwania danych w serii dla bazy MS SQL Server są wykonywane w pętli *while*, ponieważ zapisywanie danych w pamięci podręcznej w tym przypadku nie zakłamuje rezultatów. Pozostałe instrukcje dla bazy relacyjnej i wszystkie operacje dla nierelacyjnych baz danych są uruchamiane bez użycia pętli *while*, gdyż w tym przypadku zapisywanie danych w pamięci podręcznej, powodowałoby zakłamanie rezultatów. Wyniki badań dla relacyjnej bazy danych są zapisywane do pomocniczej tabeli *select_time*, z której podczas analizy wyników zostaną one odczytane, a wyniki badań dla nierelacyjnych baz danych są zapisywane do plików tekstowych. Po wykonaniu badań zweryfikowano poprawność wykonania zapytań, sprawdzając identyczność pobranych danych we wszystkich bazach danych, a także sprawdzono liczbę wstawianych i usuwanych rekordów.

5.1. Scenariusz 1

W tym scenariuszu dokonuje się pomiaru czasu wykonania operacji wstawienia danych do każdej tabeli w relacyjnej bazie danych i kolekcji w bazach NoSQL na podstawie plików wygenerowanych przy pomocy generatora danych fikcyjnych Mockaroo, zarówno dla relacyjnej, jak i nierelacyjnej bazy danych. Wykonano także pomiar czasu usuwania danych ze wszystkich tabel w bazie danych. Usuwanie odbywa się po załadowaniu wszystkich rekordów w danej serii. Badanie zostanie przeprowadzone dla każdego systemu zarządzania bazą danych dla następującej liczby rekordów: 3010, 6010, 12010, 30010, 60010. Są to sumy liczb rekordów we wszystkich tabelach.

5.2. Scenariusz 2

W tym scenariuszu dokonuje się pomiaru czasu wykonania operacji odczytu wszystkich danych z jednej tabeli lub kolekcji. Dane, które przedstawiają wszystkie możliwe do zdobycia osiągnięcia, są odczytywane z tabeli *Achievements*.

5.3. Scenariusz 3

W tym scenariuszu dokonuje się pomiaru czasu wykonania operacji odczytu danych z wykorzystaniem funkcji agregacyjnych i instrukcji grupowania danych. Pobierane dane z tabeli *Cyclist* to lista, która przedstawia poziomy zaawansowania kolarzy wraz z liczbą osób, które deklarują dany poziom zaawansowania.

5.4. Scenariusz 4

W tym scenariuszu dokonuje się pomiaru czasu wykonania operacji odczytu wybranych danych z dwóch tabel *KOM* i *KOM_history* w relacyjnej bazie danych i odczytu tych samych danych zapisanych w jednej kolekcji *KOM*, w której dokumenty zawierają zagnieżdżone dokumenty *KOM_history* w nierelacyjnej bazie danych. Ponadto dane są poddane filtrowaniu, aby spełniały określony w zapytaniu warunek. Pobierane z tabeli dane to lista archiwalnych danych o najszybszym przejeździe każdego odcinka, zdefiniowanego w tabeli *KOM*, który posiada taką historię wraz z najważniejszymi parametrami danego etapu *KOM*.

5.5. Scenariusz 5

W tym scenariuszu dokonuje się pomiaru czasu wykonania operacji odczytu wszystkich danych z czterech tabel *Cyclist*, *Training*, *Stats* i *Cyclist_achievements* w relacyjnej bazie danych i odczytu tych samych danych zapisanych w jednej kolekcji *Cyclist*, w której dokumenty zawierają zagnieżdżone dokumenty *Training*, *Stats* i *Cyclist_achievements* w nierelacyjnej bazie danych. Dane przedstawiają wszystkie treningi i osiągnięcia, a także statystykę każdego z kolarzy.

5.6. Scenariusz 6

W tym scenariuszu dokonuje się pomiaru czasu wykonania operacji odczytu wybranych danych z sortowaniem wyniku otrzymanego z trzech tabel *Cyclist*, *Achievements* i *Cyclist_achievements* w relacyjnej bazie danych i odczytu tych samych danych zapisanych w dwóch kolekcjach *Cyclist*, w której dokumenty zawierają zagnieżdżone dokumenty *Cyclist_achievements* i z kolekcji *Achievements*. Odczytywane dane przedstawiają listę kolarzy wraz z ich osiągnięciami posortowanymi po nazwie osiągnięcia i po informacji czy dane osiągnięcie zostało zdobyte.

5.7. Scenariusz 7

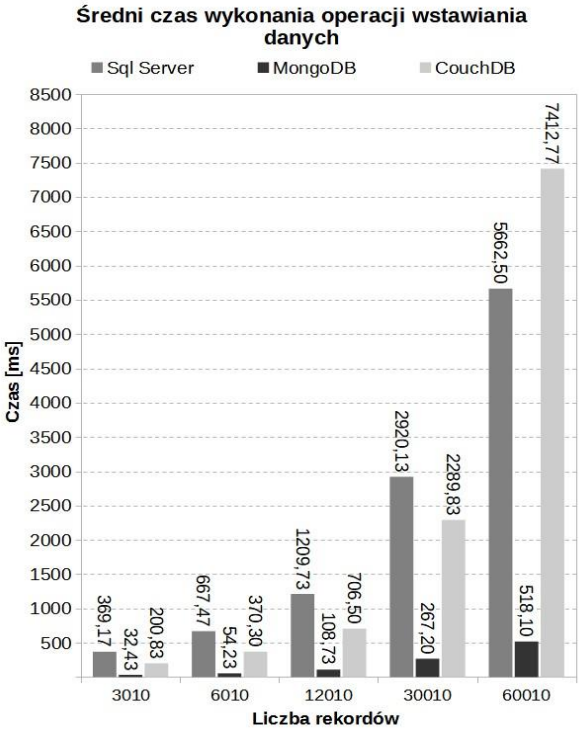
W tym scenariuszu dokonuje się pomiaru czasu wykonania operacji aktualizacji danych w jednej tabeli *Stats* w relacyjnej bazie danych i w zagnieżdżonym dokumencie *Stats* w kolekcji *Cyclist* w nierelacyjnej bazie danych. Aktualizacja danych polega na dodaniu 100 kilometrów do obecnej wartości przebytego dystansu dla statystyk każdego kolarza.

6. Wyniki badań

6.1. Wyniki dla Scenariusza 1

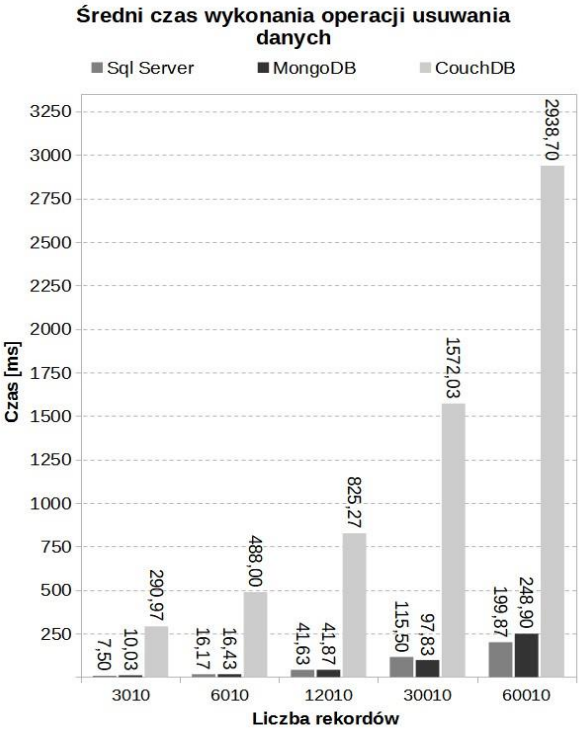
Analizując dane przedstawione na Rysunku 2, można zaobserwować, że wzrost liczby rekordów wyraźnie wpływa na wydłużenie czasu wykonania operacji wstawiania danych we wszystkich trzech bazach. Przy czym, baza CouchDB wykazuje szczególnie znaczące wydłużenie czasu dla liczby rekordów równej 30010 i 60010. Dla

pozostałych baz oraz dla mniejszej liczby rekordów w przypadku bazy CouchDB, około dwukrotne zwiększenie liczby rekordów, dwukrotnie zwiększa średni czas wstawienia danych. Porównując relacyjną bazę i NoSQL, stwierdzono, że bazy NoSQL zapewniają znacznie krótsze czasy wstawiania danych. Dla MongoDB są to czasy około 10 razy krótsze, niż dla pozostałych baz danych. Najdłuższe czasy wstawiania danych obserwuje się w przypadku MS SQL Server, z wyjątkiem serii badań dla największej liczby rekordów, czyli 60010. Dla tej serii najdłuższy czas uzyskano dla bazy CouchDB.



Rysunek 2: Średni czas wykonania instrukcji wstawienia danych.

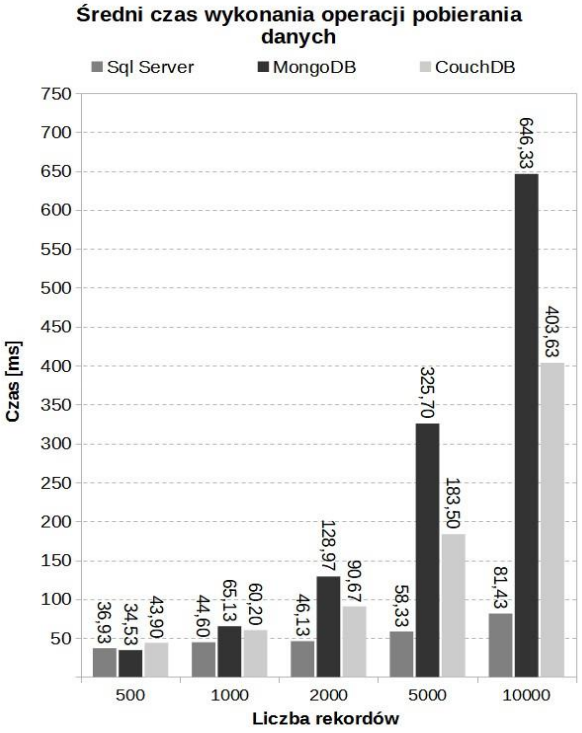
Analizując dane przedstawione na Rysunku 3, można zaobserwować, że wzrost liczby rekordów wpływa na wydłużenie czasu usuwania danych we wszystkich trzech bazach. Porównując relacyjną bazę i NoSQL, stwierdzono, że baza MS SQL Server zapewnia krótsze czasy usuwania danych od baz NoSQL. MongoDB zapewnia dłuższe czasy wykonania tej operacji od MS SQL Server o 34% dla 3010 rekordów, o 2% dla 6010 rekordów, o 0,6% dla 12010 rekordów, o 25% dla 60010 rekordów, a dla 30010 rekordów to baza MongoDB zapewnia krótszy czas usuwania danych o 15%. Najkrótsze czasy usuwania danych zapewnia CouchDB. Dla liczby rekordów 3010 i 6010 CouchDB zapewnia około 30 razy dłuższe czasy usuwania danych, dla 12010 około 20 razy dłuższe, a dla 30010 i 60010 około 15 razy dłuższe czasy od pozostałych dwóch baz.



Rysunek 3: Średni czas wykonania instrukcji usuwania danych.

6.2. Wyniki dla Scenariusza 2

Analizując dane przedstawione na Rysunku 4, można zaobserwować, że wzrost liczby rekordów wpływa na wydłużenie czasu pobrania danych z jednej tabeli lub kolekcji we wszystkich trzech bazach.



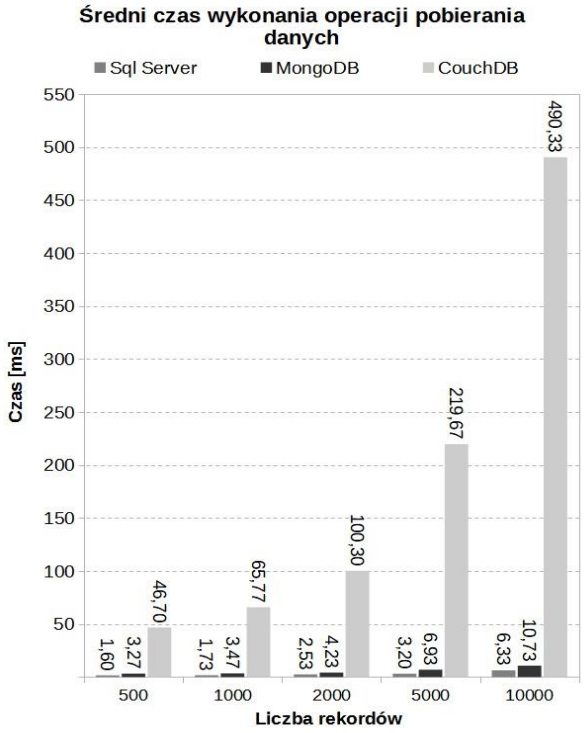
Rysunek 4: Średni czas pobrania danych dla Scenariusza 2.

Największe wydłużenie czasu trwania operacji pobierania danych występuje w bazach NoSQL, zwłaszcza dla coraz większej liczby rekordów. Porównując relacyjną

bazę i NoSQL, stwierdzono, że w tym scenariuszu baza MS SQL Server zapewnia krótsze czasy pobierania danych od baz NoSQL. Jedynie dla najmniejszej liczby rekordów, gdzie wyniki wszystkich baz są zbliżone do siebie, MongoDB zapewnia o 21% krótszy czas pobrania danych od CouchDB i o 6% krótszy od MS SQL Server. Na szczególną uwagę w przypadku MS SQL Server zasługuje fakt, że czas otrzymany dla największej liczby rekordów jest tylko 2,2 razy dłuższy od czasu uzyskanego dla najmniejszej liczby rekordów. W przypadku baz NoSQL, jest to około 18 razy dłuższy czas w przypadku MongoDB i około 9 razy dłuższy dla CouchDB. Dla większej liczby rekordów niż 500 czasy te są zdecydowanie najdłuższe w środowisku MongoDB.

6.3. Wyniki dla Scenariusza 3

Analizując dane przedstawione na Rysunku 5, można zaobserwować, że wzrost liczby rekordów nieznacznie wpływa na wydłużenie czasu pobrania danych używając funkcji agregacyjnych dla MS SQL Server i MongoDB.

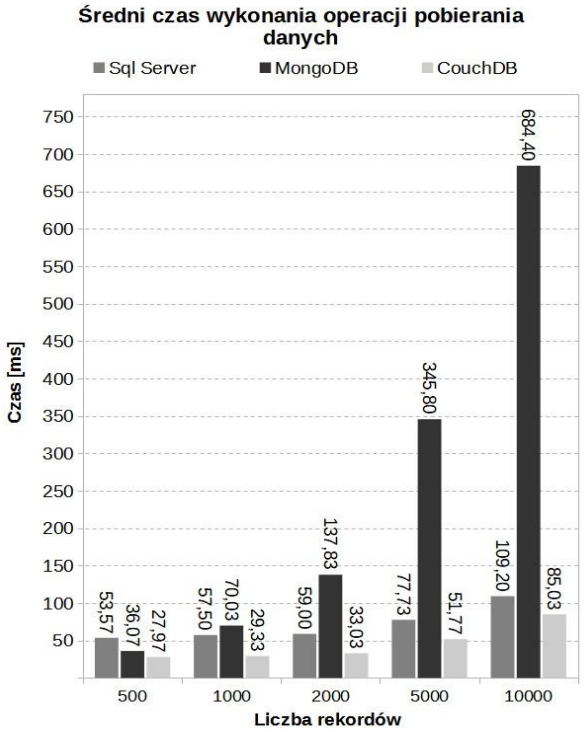


Rysunek 5: Średni czas pobrania danych dla Scenariusza 3.

Dla bazy CouchDB wzrost liczby rekordów znacząco wpływa na wydłużenie czasu tej operacji. Porównując relacyjną bazę i NoSQL, stwierdzono, że w tym scenariuszu baza MS SQL Server zapewnia krótsze czasy pobierania danych od baz NoSQL. MS SQL Server zapewnia około dwukrotnie krótsze czasy od MongoDB, a także dla tych baz danych czas wykonania pobrania danych dla największej liczby rekordów jest tylko 3 razy dłuższy niż czas otrzymany dla najmniejszej liczby rekordów. Zdecydowanie najdłuższe czasy obserwuje się w przypadku bazy CouchDB, dla której były one nawet około 70 razy dłuższe niż dla bazy relacyjnej zawierającej 10000 rekordów.

6.4. Wyniki dla Scenariusza 4

Analizując dane przedstawione na Rysunku 6, można zaobserwować, że wzrost liczby rekordów wpływa na wydłużenie czasu pobrania danych, zwłaszcza dla MongoDB. Dla bazy CouchDB otrzymane czasy dla największej liczby rekordów są tylko 3 razy dłuższe od czasów otrzymanych dla najmniejszej liczby rekordów, a dla MS SQL Server są tylko 2 razy dłuższe. Porównując relacyjną bazę i NoSQL, stwierdzono, że w tym scenariuszu baza CouchDB zapewnia krótsze czasy pobierania danych od bazy MS SQL Server i MongoDB. Jedynie dla 500 rekordów, najdłuższy czas obserwuje się w bazie MS SQL Server, dla większej liczby rekordów to MongoDB charakteryzuje się najdłuższym czasem wykonania tego rodzaju operacji. Niemożliwym jest jednoznacznie stwierdzić, który typ baz danych jest bardziej wydajny w tym scenariuszu, gdyż CouchDB zapewnia krótsze czasy od MS SQL Server, ale baza MS SQL Server zapewnia krótsze czasy w porównaniu do drugiej bazy NoSQL, czyli MongoDB.

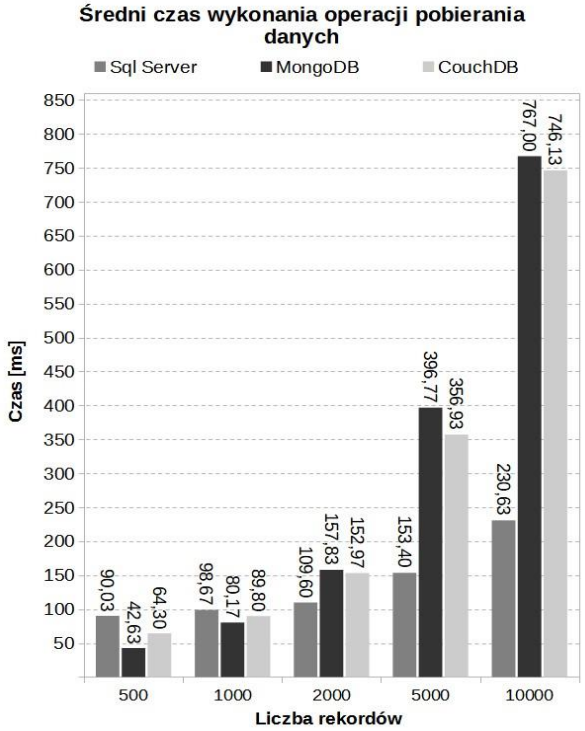


Rysunek 6: Średni czas pobrania danych dla Scenariusza 4.

6.5. Wyniki dla Scenariusza 5

Analizując dane przedstawione na Rysunku 7, można zaobserwować, że wzrost liczby rekordów wpływa na wydłużenie czasu pobrania danych we wszystkich bazach danych. Największe wydłużenie czasu wykonania zapytania widać dla baz NoSQL, dla bazy SQL wydłużenie czasu między najmniejszą a największą liczbą rekordów wynosi 230%. Porównując relacyjną bazę i NoSQL, stwierdzono, że w tym scenariuszu baza SQL zapewnia krótsze czasy pobierania danych od bazy MongoDB i CouchDB. Jedynie dla 500 i 1000 rekordów, najdłuższy czas obserwuje się dla bazy MS SQL Server, a najkrótszy dla MongoDB. Dla większej liczby rekordów to

MongoDB zapewnia najdłuższe czasy, a najkrótsze MS SQL Server.



Rysunek 7: Średni czas pobrania danych dla Scenariusza 5.

6.6. Wyniki dla Scenariusza 6

Otrzymane czasy dla bazy CouchDB były dłuższe nawet 10 tysięcy razy dla 5000 rekordów w porównaniu z bazą MS SQL Server, dlatego nie przeprowadzono 30 serii badań, a jedynie dwie, aby przekonać się czy tak duże różnice się powtórzą. Ponadto nie uzyskano wyniku dla 10000 rekordów ze względu na zbyt długi czas trwania operacji. Nie zestawiono tych wyników z wynikami dla innych baz przedstawionych w Tabeli 1, ponieważ wykonano tylko dwie serie badań. Wartości, które otrzymano dla 500, 1000, 2000, 5000 rekordów to kolejno około: 12000 ms, 41600 ms, 154500 ms i 915350 ms.

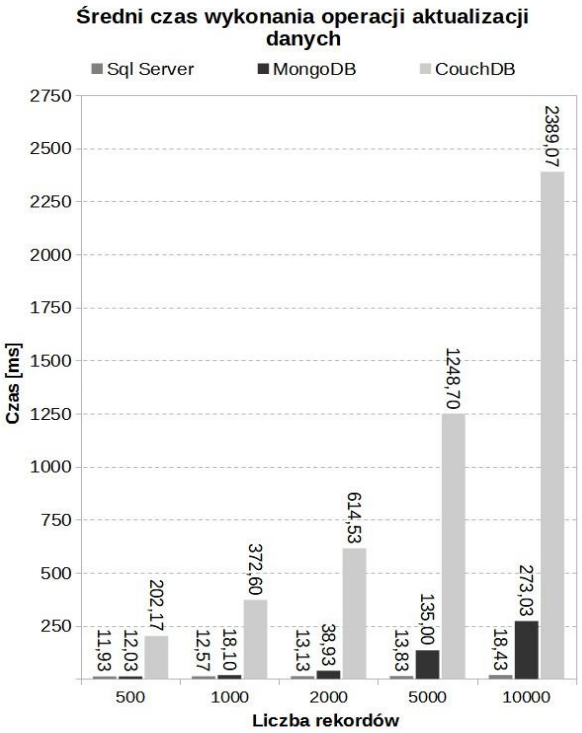
Tabela 1: Średni czas pobrania danych dla Scenariusza 6

Baza danych	MS SQL Server	MongoDB
Liczba rekordów	Średnia [ms]	Średnia [ms]
500	46,7	224,93
1000	49,67	917,27
2000	55,33	3285,1
5000	84,2	18376,97
10000	120,03	70890,4

Przy prezentacji wyników nie przedstawiono wykresu ze średnimi wartościami czasu trwania operacji pobierania danych, ponieważ porównanie tak otrzymanych wyników badań nie prowadzi do otrzymania wiarygodnych wniosków. Jedyne, co można stwierdzić w tej sytuacji to fakt, że nie należy dokonywać operacji łączenia kolekcji w bazach NoSQL, ponieważ jest to wysoce niewydajne działanie.

6.7. Wyniki dla Scenariusza 7

Analizując dane przedstawione na Rysunku 8, można zaobserwować, że wzrost liczby rekordów wpływa na wydłużenie czasu aktualizacji danych. Największe wydłużenie czasu wykonania zapytania widać dla baz NoSQL. Dla bazy SQL otrzymany czas aktualizacji 500 rekordów jest o 35% krótszy od czasu aktualizacji 10000 rekordów, gdzie w przypadku MongoDB jest on krótszy o 96%, a dla CouchDB o 92%. Porównując bazę relacyjną i NoSQL, stwierdzono, że w tym scenariuszu baza SQL zapewnia krótsze czasy aktualizacji danych w porównaniu do baz MongoDB i CouchDB. Najdłuższe czasy aktualizacji danych występują w przypadku bazy CouchDB. Baza MongoDB dla małej liczby rekordów zapewnia krótkie czasy wykonania operacji aktualizacji, lecz wraz ze wzrostem liczby rekordów, MongoDB, zapewnia czasy nawet 14 razy dłuższe od MS SQL Server.



Rysunek 8: Średni czas aktualizacji danych.

7. Wnioski

W sześciu scenariuszach badawczych bardziej wydajna jest baza MS SQL Server w porównaniu z bazami NoSQL. Okazała się ona najwydajniejsza dla operacji usuwania, pobierania danych i ich aktualizacji. Przy operacji usuwania danych, wyniki otrzymane dla środowiska MS SQL Server i MongoDB były zbliżone do siebie i różnica wynosiła tylko 2% dla 6010 rekordów i 0,6% dla 12010 rekordów. Dla operacji pobierania danych MS SQL Server wykazał się wysoką wydajnością zwłaszcza dla dużej liczby rekordów. Dla jednej operacji pobierania danych wraz z filtrowaniem wyniku najwydajniejszą bazą była CouchDB. Warto zauważyć wysoką

wydajność MongoDB dla operacji pobierania danych, gdzie liczba rekordów wynosiła 500, w takim przypadku była ona najwydajniejszą z badanych baz. Niestety, przy coraz większej liczbie rekordów, ta baza okazywała się niewydajna dla tych operacji. Wykonywanie operacji łączenia kolekcji przy pobieraniu danych w bazach NoSQL jest bardzo niewydajne, co widać po otrzymanych wynikach badań w Scenariuszu 6. Można stwierdzić, że nie należy łączyć danych między kolekcjami w bazach NoSQL, ponieważ te bazy nie są przystosowane do tego typu operacji. Jedynie przy operacji wstawiania danych zdecydowanie najwydajniejszą bazą było MongoDB, następnie CouchDB, a najmniej wydajną bazą był MS SQL Server, co oznacza, że przy operacji wstawiania danych bazy NoSQL są wydajniejsze. Operacja aktualizacji danych, najszybciej przebiegła w środowisku MS SQL Server. Baza zapewniła zdecydowanie najkrótsze czasy, a także wydłużenie czasu operacji aktualizacji między najmniejszą a największą liczbą rekordów było najmniejsze spośród badanych baz i wyniosło 54%. Biorąc pod uwagę wszystkie otrzymane wyniki za najwydajniejszą bazę danych można uznać MS SQL Server, następnie MongoDB, a najmniej wydajną bazą jest CouchDB. Generalizując, teza artykułu, która brzmi: ”Relacyjne bazy danych są bardziej wydajne niż bazy danych NoSQL w kontekście operacji CRUD” została potwierdzona.

Literatura

[1] D. Chauhan, K. L. Bansal, Using the advantages of NOSQL: a case study on MongoDB, International Journal on Recent and Innovation Trends in Computing and Communication 5(2) (2017) 90–93.

[2] K. Goel, A. H. Ter Hofstede, Privacy-breaching patterns in nosql databases, IEEE Access 9 (2021) 35229–35239.

[3] A. Abdullah, Q. Zhuge, From relational databases to NoSQL databases: Performance evaluation, Research Journal of Applied Sciences, Engineering and Technology 11(4) (2015) 434–439.

[4] T. Patel, T. Eltaieb, Relational database vs NoSQL, Journal of Multidisciplinary Engineering Science and Technology (JMEST) 2(4) (2015) 691-695.

[5] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, D. Wilkins, A comparison of a graph database and a relational database: a data provenance perspective, Proceedings of the 48th annual Southeast regional conference (2010) 1–6, https://doi.org/10.1145/1900008.1900067.

[6] Y. Li, S. Manoharan, A performance comparison of SQL and NoSQL databases, Proceedings of 2013 IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM) (2013) 15–19, https://doi.org/10.1109/PACRIM.2013.6625441.

[7] J. Bhogal, I. Choksi, Handling big data using NoSQL, Proceedings of 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (2015) 393–398, https://doi.org/10.1109/WAINA.2015.19.

[8] P. P. Srivastava, S. Goyal, A. Kumar, Analysis of various NoSql database, Proceedings of 2015 International Conference on Green Computing and Internet of Things (ICGCIoT) (2015) 539–544, https://doi.org/10.1109/ICGCIoT.2015.7380523.

[9] S. Sicari, A. Rizzardi, A.Coen-Porisini, Security&privacy issues and challenges in NoSQL databases, Computer Networks 206 (2022) 108828–108830.

[10] B. Jose, S. Abraham, Performance analysis of NoSQL and relational databases with MongoDB and MySQL, Materials today: Proceedings 24 (2020) 2036–2043.

[11] C. Gyorödi, R. Gyorödi, R. Sotoc, A comparative study of relational and non-relational database models in a Web-based application, International Journal of Advanced Computer Science and Applications 6(11) (2015) 78–83.

[12] K. Fraczek, M. Plechawska-Wojcik, Comparative analysis of relational and non-relational databases in the context of performance in web applications, Beyond Databases, Architectures and Structures, Proceedings of 13th International Conference Towards Efficient Solutions for Data Analysis and Knowledge Representation (2017) 153–164, https://doi.org/10.1007/978-3-319-58274-0_13.

[13] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Ş. Győrödi, G. A. Gabor, G. D. Pecherle, Performance analysis of NoSQL and relational databases with CouchDB and MySQL for application’s data storage, Applied Sciences 10(23) (2020) 8524–8545.

[14] A. Malik, A. Burney, F. Ahmed, A comparative study of unstructured data with SQL and NO-SQL database management systems, Journal of Computer and Communications 8(4) (2020) 59–71.

[15] S. Schmid, E. Galicz, W. Reinhardt, Performance investigation of selected SQL and NoSQL databases, Proceedings of the AGILE (2015) 1–5.

[16] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM 13 (1970) 377–387.

[17] D. D. Chamberlin, Early history of SQL, IEEE Annals of the History of Computing 34 (2012) 78–82.

[18] J. R. Groff, P. N. Weinberg, A. J. Oppel, SQL: The Complete Reference Vol. 2, McGrawHill/Osborne, 2002.

[19] H. Khazaei, M. Fokaefs, S. Zareian, How do I choose the right NoSQL solution? A comprehensive theoretical and experimental survey, Big Data and Information Analytics 1(2) (2016) 185–216.

[20] V. Abramova, J. Bernardino, P. Furtado, Experimental evaluation of NoSQL databases, International Journal of Database Management Systems 6(3) (2014) 1–16.

[21] R. Arora, R. R. Aggarwal, Modeling and querying data in mongodb, International Journal of Scientific and Engineering Research 4(7) (2013) 141–144.

[22] I. Carvalho, F. Sá, J. Bernardino, Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB, Algorithms 16(2) (2023) 78–95.