JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 264–272
Received: 14 June 2023
Accepted: 3 July 2023

Performance analysis of databases created in virtualized and containerized environment

Analiza wydajności baz danych utworzonych w zwirtualizowanym i skonteneryzowanym środowisku

Zygmunt Łata*, Maria Skublewska-Paszkowska

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

Abstract
Database systems are currently implemented on different environments - containerized and virtualized. This paper aimed to compare the performance of relational databases running on these two environments. Therefore, two research hypotheses were formulated. The first hypothesis assumed that databases running on Docker containers are more efficient than those on virtual machines. The second one assumed that the Oracle is the most efficient database regardless of which environment it was run on. MySQL, PostgreSQL, Microsoft SQL Server and Oracle databases were tested. The study measured the execution times of INSERT, UPDATE, DELETE and SELECT queries. Each test was repeated 100 times. It was stated that databases running on Docker containers outperform instances running on virtual machines. Furthermore, it was found that the PostgreSQL database have a definite advantage in performance over the rest, analysed databases, which allowed the second hypothesis to be rejected.

Keywords: performance comparison; relational databases; Docker containers; virtual machines

Streszczenie
Systemy bazodanowe są obecnie implementowane na różnych środowiskach - skonteneryzowanym oraz zwirtualizowanym. Artykuł ma na celu porównać wydajność relacyjnych baz danych uruchomionych w tych dwóch środowiskach. W związku z tym sformułowano dwie hipotezy badawcze. Pierwsza z nich zakładała, że bazy uruchomione na kontenerach Docker są wydajniejsze od tych na maszynach wirtualnych. W drugiej hipotezie przyjęto, że Oracle jest najwydajniejszą bazą niezależnie w jakim środowisku została uruchomiona. Badaniom poddano bazy MySQL, PostgreSQL, Microsoft SQL Server oraz Oracle. W ramach badań zmierzono czasy wykonywania zapytań INSERT, UPDATE, DELETE oraz SELECT. Każdy test został powtórzony 100 razy. Wykazano, że bazy uruchomione na kontenerach Docker przewyższają pod względem wydajności instancje działające na maszynach wirtualnych. Ponadto stwierdzono, że baza PostgreSQL ma zdecydowaną przewagę w wydajności nad pozostałymi, analizowanymi bazami, co pozwoliło odrzucić drugą hipotezę.

Słowa kluczowe: porównanie wydajności; relacyjne bazy danych; kontenery Docker; maszyny wirtualne

*Corresponding author
Email address: zygmunt.lata@pollub.edu.pl (Z. Łata)

1. Wstęp

Zdecydowana większość nowoczesnych aplikacji i systemów wykorzystuje bazy danych do przechowywania niezbędnych informacji w sposób trwały i usystematyzowany oraz do wykonywania na nich podstawowych operacji - zapis, edycja, usuwanie i odczyt. Na przestrzeni lat bazy danych były stale rozwijane i udoskonalane przez programistów, aby sprostać rosnącym potrzebom na rynku, takim jak szybszy dostęp i przetwarzanie danych. Istnieje wiele rodzajów baz danych rozróżnianych ze względu na model danych m.in. relacyjne, obiektowe, NoSQL (nierelacyjne), grafowe oraz dokumentowe (w formacie JSON) [1]. Relacyjne bazy danych wciąż są często stosowane, ponieważ mają swoje zalety i zajmują wysokie pozycje w rankingach popularności [2]. Najpopularniejszymi obecnie wśród nich są Oracle, MySQL, MS SQL Server (Microsoft) oraz PostgreSQL.

Współcześnie wiele firm wybiera wdrażanie baz danych w chmurze jako usługi (ang. Database as a Service, DBaaS), niekiedy wiąże się to z wysokimi kosztami eksploatacji takiego typu bazy i zarządzania nią, gdyż w przypadku takiego modelu przedsiębiorstwo nabywa dostęp do bazy na zasadzie płatnej subskrypcji z dostawcą chmury [3]. Dlatego też znaczna część organizacji decyduje się na utworzenie lokalnej infrastruktury (serwera), na którym wdrażana jest baza danych wykorzystując przy tym technologie wirtualizacji, która pozwala znacznie polepszyć wykorzystanie zasobów sprzętowych fizycznej infrastruktury.

Wirtualizacja pozwala za pomocą oprogramowania utworzyć odizolowaną warstwę sprzętu komputerowego [4]. W takim środowisku zasoby sprzętowe pojedynczego komputera takie jak: procesory, pamięć operacyjna, pamięć masowa mogą zostać rozłożone na wiele maszyn wirtualnych (ang. Virtual Machine, VM), które posiadają własny system operacyjny (ang. Operating System, OS) i działają jako niezależne urządzenia, wykorzystując przy tym jedynie część zasobów sprzętowych bazowego komputera (hosta). Takie podejście

pozwala na utworzenie oddzielnych instancji maszyn wirtualnych z zainstalowanymi i skonfigurowanymi na nich określonymi serwerami baz danych, do których następnie będzie można uzyskać dostęp z zewnątrz za pomocą odpowiednich żądań.

Konteneryzacja jest równie popularną technologią wykorzystywaną do wdrażania różnych usług, takich jak serwery aplikacji i bazy danych [5]. Podobnie jak wirtualizacja, odseparowuje warstwę sprzętu komputerowego od komputera hosta, w takim podejściu utworzone środowisko współdzieli zasoby sprzętowe i rozdziela je na wiele oddzielnych instancji, zwanych kontenerami. Kontenery nie posiadają własnego systemu operacyjnego, współdzielą one jądro systemu operacyjnego komputera bazowego i zawierają wyłącznie biblioteki systemu operacyjnego wraz z zależnościami wymaganymi do uruchomienia danej usługi. Oferują lepszą wydajność w wykorzystaniu zasobów sprzętowych niż maszyny wirtualne oraz cechują się lepszą przenośnością, ponieważ działają w jednakowy sposób w dowolnej infrastrukturze. Taka koncepcja pozwala na uruchamianie kontenerów jako oddzielnych instancji zawierających kompletne środowisko bazodanowe, najczęściej poprzez wskazanie określonego obrazu bazy danych, które są ogólnie dostępne w publicznym repozytorium DockerHub. Kontener utworzony w taki sposób można następnie bez żadnych przeszkód wdrożyć w chmurze.

Bazując na opisanych powyżej technologiach wdrażania usług, celem artykułu jest porównanie wydajności relacyjnych baz danych utworzonych w zwirtualizowanym i skonteneryzowanym środowisku.

2. Przegląd literatury

Istnieje wiele prac naukowych poświęconych tematyce porównującej wydajność rozwiązań bazodanowych oraz środowisk, w których zostają uruchomione. Autorzy stosują w nich różnorodne metody badawcze pozwalające ocenić i porównać wydajność wybranych przez nich systemów bazodanowych.

Ze względu na charakter pracy, artykułem otwierającym przegląd jest „*An Introduction to Docker and Analysis of its Performance*" [6], który charakteryzuje środowisko skonteneryzowane Docker i jego wydajność. Opisane zostały w nim główne elementy składające się na platformę Docker w porównaniu z technologią wirtualizacji i innymi dostępnymi technologiami konteneryzacji. Oprócz tego zwrócono również uwagę na zalety kontenerów Docker, takie jak: przenośność, szybkość, skalowalność, szybkie dostarczanie oraz gęstość, jak również na ich wady, takie jak: brak pełnej wirtualizacji, brak wsparcia dla starszych komputerów oraz niedostarczone w chwili obecnej rozwiązania związane z bezpieczeństwem kontenerów. Artykuł wykazuje, że kontenery Docker są szybsze w działaniu niż maszyny wirtualne i mniej obciążają zasoby.

W artykule „*Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization*" [7] przeprowadzono porównanie wydajności relacyjnych DBMS w środowisku skonteneryzowanym. Badanymi systemami były MySQL, PostgreSQL, MS SQL Server i Oracle. Każdy z nich został wdrożony jako oddzielna instancja kontenera Docker z domyślnymi ustawieniami, wykorzystując oficjalne dystrybucje obrazów z DockerHub. Przeprowadzono serię eksperymentów z użyciem aplikacji webowej z zaimplementowanymi testami do pomiarów czasowych wykonywania określonych operacji. Eksperymenty obejmowały zapytania SELECT dla trzech różnych rozmiarów zestawów danych. Badania przeprowadzono dla czterech konstrukcji zapytania SELECT: z klauzulą ORDER BY lub WHERE (jednym warunkiem), z podzapytaniem skorelowanym i klauzulą ORDER BY oraz z funkcjami agregującymi i wielokrotnymi złączeniami wewnętrznymi INNER JOIN. Każdy scenariusz został powtórzony 100 razy. Na podstawie analizy otrzymanych wyników autorzy stwierdzili, że najbardziej wydajnym systemem okazał się Oracle. Potwierdza to słuszność hipotezy postawionej przez autorów, ponieważ uzyskał on najlepsze wyniki w prawie każdym scenariuszu badawczym, za wyjątkiem czwartego scenariusza, w którym wydajniejszym systemem okazał się MS SQL Server. Co więcej autorzy potwierdzili swoje przypuszczenia, że czas wykonywania zapytań wybierających dane z bazy jest tym większy, im większy jest zestaw przeszukiwanych danych w bazie.

Artykuł „*Comparison of query performance in relational a non-relation databases*" [8] porównuje wydajność dwóch różnych typów DBMS – relacyjnych i nierelacyjnych. W artykule zostały scharakteryzowane bazy danych SQL oraz NoSQL, zwracając uwagę na różnice pomiędzy nimi, ich cechy, zasadę działania oraz architekturę. Badania przeprowadzono na relacyjnych bazach danych: MySQL, MS SQL Server, Oracle oraz na nierelacyjnych: MongoDB, GraphQL, Apache Cassandra i Redis. Scenariusze skupiały się na wykonywaniu pomiarów czasu operacji wybierania danych oraz manipulacji nimi - wstawiania, aktualizowania oraz usuwania danych. Testy przeprowadzone zostały dla dwóch zestawów danych liczących po 10 000 i 100 000 rekordów. Wyniki jednoznacznie ilustrują przewagę nierelacyjnych baz nad relacyjnymi pod względem uzyskanych czasów poszczególnych operacji. Stosunek wydajności nierelacyjnych do relacyjnych baz wynosił odpowiednio: 1:15 dla wstawiania danych, 1:9 dla aktualizacji danych, 1:6 dla usuwania danych oraz 1:3 dla wybierania danych. MS SQL Server osiągnął najwyższą wydajność wśród relacyjnych baz, natomiast wśród nierelacyjnych najwydajniejsza okazała się MongoDB.

W artykule „*Microsoft SQL Server and Oracle: Comparative Performance Analysis*" [9] autorzy porównują wydajność dwóch relacyjnych baz danych: MS SQL Server i Oracle. Przeprowadzili testy, dokonując pomiaru czasów wykonania zapytań SELECT dla pojedynczej tabeli oraz dla wielu scalonych tabel za pomocą złączeń wewnętrznych INNER JOIN. Wykonywane zapytania podzielono na: zapytanie proste, z klauzulą WHERE lub ORDER BY. Przy ocenie wydajności uwzględniano również wymagania systemowe i sprzętowe, bezpieczeństwo oraz podstawowe cechy baz da-

nych: obsługiwane systemy operacyjne, języki programowania, składnia, interfejsy oraz języki SQL. Na podstawie analizy wyników autorzy wskazali na przewagę bazy Oracle, ponieważ jest wspierana przez więcej systemów operacyjnych i języków programowania. MS SQL Server przeważa tylko pod kątem używanego języka zapytań T-SQL, który ma prostszą składnią niż PL-SQL używany przez Oracle. W przypadku wymagań systemowych i sprzętowych nie zaobserwowano większych różnic, poza tym, że MS SQL Server wymagał dwukrotnie mniejszego zużycia pamięci RAM. W kwestii bezpieczeństwa Oracle oferuje silniejsze zabezpieczenia, jest bardziej odporny na błędy i uszkodzenia danych. Pod względem czasów wykonywania zapytań SELECT dużą przewagę miał MS SQL Server. Mimo to autorzy nie określają jednoznacznie, który z systemów jest wydajniejszy.

Artykuł „*Huge and Real-Time Database Systems: A Comparative Study and Review for SQL Server 2016, Oracle 12c & MySQL 5.7 for Personal Computer*" [10] porównuje wydajność relacyjnych baz danych: MySQL, MS SQL Server oraz Oracle. Przy analizie wydajności posłużono się pomiarami czasu wykonywanych operacji bazodanowych: aktualizacji kolumny, kopiowania rekordów z jednej tabeli i wstawiania ich do drugiej oraz wybierania danych. Zapytanie SELECT podzielono na: zapytanie proste, z klauzulą WHERE, ORDER BY lub GROUP BY oraz dla wielu scalonych tabel za pomocą wielokrotnych złączeń JOIN. Zestaw danych testowych liczył od 300 000 do 400 000 rekordów. Na podstawie otrzymanych wyników stwierdzono, że najwydajniejszą bazą pod względem szybkości wykonywanych operacji okazał się być MySQL.

Artykuł „*A Comparative Study on the Performance of the Top DBMS Systems*" [11] przedstawia porównanie wydajności relacyjnych systemów bazodanowych MySQL, IBM DB2, Oracle, MS SQL Server oraz MS Access. Testy opierały się na pomiarach czasu wykonywania zapytań UPDATE, DELETE i SELECT. Zapytanie SELECT podzielono na: zapytanie proste, z klauzulą ORDER BY lub WHERE z wieloma warunkami, z wieloma złączeniami wewnętrznymi INNER JOIN, z funkcjami agregującymi bez warunków lub z dodatkowymi warunkami po klauzuli HAVING, z wewnętrznie zagnieżdżonymi zapytaniami SELECT oraz zapytanie łączące wszystkie poprzednie w jedno atomowe. Zestaw danych testowych składał się z 1 000 000 rekordów. Ocenie podlegało również zużycie zasobów: procesora, pamięci RAM, pamięci wirtualnej oraz liczby zajętych wątków. Na podstawie przeprowadzonych eksperymentów stwierdzono, że IBM DB2 jest najwydajniejszym systemem bazodanowym pod względem szybkości wykonywania zapytań. Bezpośrednio za nim znajdują się: MS SQL Server, MySQL oraz Oracle. Natomiast ostatnie miejsce zajął MS Access charakteryzujący się najdłuższym czasem spośród badanych baz danych. Mimo to, zauważono, że baza MS Access wykazywała się najmniejszym zużyciem procesora oraz pamięci.

3. Cel i hipotezy badawcze

Artykuł jest efektem przeprowadzonych badań, których celem była ocena wydajności relacyjnych baz danych w dwóch różnych środowiskach - zwirtualizowanym i skonteneryzowanym. Narzędziem używanym do utworzenia środowisk zwirtualizowanych będzie Oracle VM VirtualBox, natomiast dla środowisk skonteneryzowanych oprogramowanie Docker Desktop. Każdy z systemów zostanie wdrożony jako oddzielna usługa na maszynie wirtualnej i na kontenerze Docker. Badanymi bazami będą MySQL, PostgreSQL, MS SQL Server i Oracle. Do oceny wydajności posłużą metryki oparte na pomiarach czasu wykonywania zapytań SQL typu INSERT, UPDATE, DELETE oraz SELECT.

Na podstawie przeglądu literatury postawione zostały następujące hipotezy badawcze:

H1: Bazy danych uruchomione na kontenerach Docker są wydajniejsze niż bazy uruchomione na maszynach wirtualnych pod względem szybkości wykonywania zapytań SQL.

H2: Baza danych Oracle jest najwydajniejsza pod kątem szybkości wykonywania zapytań SQL wśród badanych baz, niezależnie od środowiska, w którym została uruchomiona.

4. Metodyka badawcza

4.1. Stanowisko badawcze

Stanowiskiem badawczym, na którym zostały przeprowadzone eksperymenty jest fizyczna maszyna Lenovo Legion Y540, pełniąca rolę gospodarza (hosta) z zainstalowanym 64-bitowym systemem operacyjnym Windows 11 Home w wersji 22H2, kompilacji 22621.1265. W Tabeli 1 znajduje się pełna specyfikacja techniczna urządzenia testowego.

Tabela 1: Specyfikacja techniczna fizycznej maszyny gospodarza

Procesor	Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz, 2400 MHz, Rdzenie: 4, Procesory logiczne: 8
Dysk	Model: INTEL SSDPEKNW010T9, Typ: SSD (ang. Solid State Drive), Pojemność: 954 GB
Pamięć RAM	32 GB (DDR4, 2666MHz)

Środowiska zwirtualizowane z określonymi serwerami baz danych uruchomiono na maszynach wirtualnych za pomocą narzędzia Oracle VM VirtualBox w wersji 7.0.6. Bazy MySQL, PostgreSQL oraz MS SQL Server wykorzystywały taki sam obraz systemu operacyjnego Ubuntu w wersji 20.04.5 LTS (Focal Fossa), używany jako serwer. Natomiast baza Oracle wykorzystywała obraz systemu operacyjnego Oracle Linux w wersji 8.7 (R8-U7-x86_64), również w postaci serwerowej. Na każdej maszynie wirtualnej zainstalowano i skonfigurowano konkretną wersję serwera bazodanowego zgodnie z instrukcjami zawartymi na oficjalnych stronach dokumentacji twórców. W Tabeli 2 przedstawiono przydzielone zasoby dla maszyn wirtualnych.

Tabela 2: Zasoby sprzętowe przydzielone maszynom wirtualnym

Liczba rdzeni procesora (wirtualnych procesorów)	2
Dysk	Kontroler: SATA (ang. Serial ATA), Typ: VDI (ang. Virtual Disk Image), Pojemność: 30 GB, Dyna-

	miczny sposób przydzielania pamięci
Pamięć RAM	4 GB

W celu ujednolicenia tworzonych środowisk bazodanowych oraz uwiarygodnienia wyników eksperymentów na maszynie gospodarza korzystającej z systemu operacyjnego Windows został zainstalowany WSL (ang. Windows Subsystem for Linux) pozwalający na uruchamianie dystrybucji systemów operacyjnych bazujących na systemie Linux [12]. Używając narzędzia WSL zainstalowano dystrybucję systemu operacyjnego Ubuntu w wersji 20.04.5 LTS (Focal Fossa) zapewniając tym samym zgodność pod kątem systemów operacyjnych z maszynami wirtualnymi. Domyślną wersję WSL zaktualizowano do wersji WSL 2 wprowadzając tym samym znaczne usprawnienia takie jak wbudowane jądro Linux (ang. Linux kernel) oraz pełną kompatybilność wywołań systemowych zwiększając przy tym wydajność uruchamianych dystrybucji [13].

Środowiska skonteneryzowane z usługami bazodanowymi uruchomiono na kontenerach Docker za pomocą oprogramowania Docker Desktop w wersji 4.17.0 (99724). Narzędzie to używa silnika Docker w wersji 20.10.23 i zostało skonfigurowane tak aby korzystało z silnika WSL 2, zwiększając jednocześnie wydajność niż przy domyślnym silniku Hyper-V. Kontenery Docker zostały uruchomione z domyślnymi ustawieniami zgodnie z przedstawioną dokumentacją dla oficjalnych obrazów baz danych udostępnionych przez twórców na publicznym repozytorium obrazów DockerHub. Kontenery zdefiniowano za pomocą oddzielnych plików Compose, co znacznie uprościło proces uruchamiania i zarządzania nimi. Porty publiczne, na których zostały wystawione kontenery z bazami danych były zgodne z odpowiadającymi im portami na maszynie gospodarza. Do każdego kontenera podmontowano wolumin z ustawionymi opcjami praw dostępu do katalogu roboczego kontenera - odczytu z, zapisu do (ang. read-write, rw). Utworzony w ten sposób wolumin powiązywał katalog roboczy wdrażanego serwera bazodanowego z systemem plików maszyny gospodarza, co pozwoliło na zachowanie stanu bazy danych - jej struktury oraz przechowywanych danych.

Ponadto utworzony został plik konfiguracyjny o rozszerzeniu .wslconfig pozwalający na zmianę domyślnych ustawień przydzielonych zasobów sprzętowych dla wszystkich dystrybucji uruchamianych na WSL 2 [14]. W Tabeli 3 przedstawiono przydzielone zasoby sprzętowe zgodnie z plikiem konfiguracyjnym .wslconfig dla platformy Docker Desktop, wprowadza to ograniczenia dla kontenerów Docker uruchamianych na wszystkich dystrybucjach działających na WSL 2. Widoczny przydział pokrywa się z limitami wskazanymi dla maszyn wirtualnych.

Tabela 3: Zasoby sprzętowe przydzielone platformie Docker Desktop

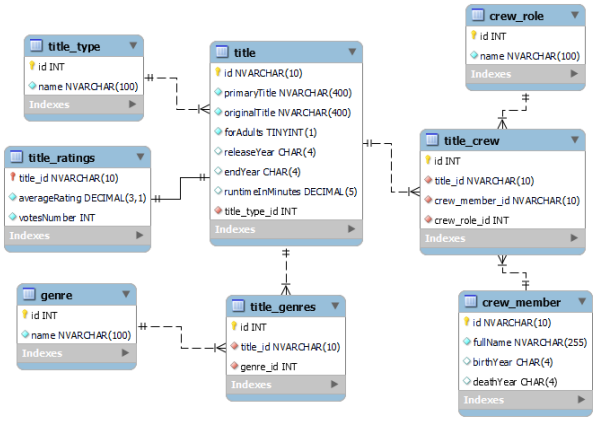
Liczba rdzeni procesora (wirtualnych procesorów)	2
Pamięć RAM	4 GB
Pamięć SWAP	1 GB

4.2. Bazy danych

W eksperymencie porównywano ze sobą następujące systemy bazodanowe:

- MySQL w wersji 8.0.32;
- PostgreSQL w wersji 15.2;
- MS SQL Server w wersji 2022 (16.0.4003.1);
- Oracle XE (Express Edition) w wersji 21c (21.3.0.0) – Production.

Schemat widoczny na rysunku 1 jest modyfikacją koncepcji bazy danych udostępnionej przez serwis IMDb, w której przedstawiony został realny zbiór danych na temat tytułów filmowych, telewizyjnych, seriali oraz powiązanych z nimi celebrytów na całym świecie [15].



Rysunek 1: Struktura bazy danych.

W badaniach zastosowano realne zasoby udostępnione przez serwis IMDb, które ze względu na wprowadzone modyfikacje schematu wymagały ich spreparowania. Opracowane w ten sposób dane wyodrębniono na cztery zestawy z rosnącą liczbą rekordów. Wszystkie bazy wypełniono jednakową treścią, oznaczało to, że początkowe dane pozostawały niezmienne, natomiast do poprzedniej puli danych dodawano liczbę rekordów określoną dla konkretnego zestawu danych do tabeli *title* i do powiązanych z nią tabel. W Tabeli 4 przedstawiono przydział liczby rekordów dla poszczególnych zestawów testowych.

Tabela 4: Przydział liczby rekordów dla poszczególnych zestawów danych

Tabela	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
title	1 000	10 000	100 000	1 000 000
title_genres	1 848	17 171	174 023	1 603 316
crew_member	842	12 971	230 403	1 346 385
title_crew	4 018	72 109	1 013 876	7 750 390
title_ratings	1 000	10 000	100 000	1 000 000

4.3. Metryki pomiaru

Badania przeprowadzone zostały dla wszystkich systemów bazodanowych wymienionych w podrozdziale 4.2, które to zostały zaimplementowane w dwóch różnych środowiskach - zwirtualizowanym oraz skonteneryzowanym. W trakcie trwania każdego eksperymentu uruchomiona była jedynie pojedyncza instancja bazy danych. Testy opierały się na pomiarach czasów wykonywania konkretnych operacji bazodanowych, które zostały powtórzone cyklicznie 100 razy, aby osiągnąć jak najbardziej wiarygodne i prawidłowe wyniki. Eksperymenty zostały podzielone na główne scenariusze ze względu na wykonywany rodzaj zapytania INSERT, UPDATE, DELETE oraz SELECT. Każdy z nich został powtórzony dla kolejnych zestawów danych o zmieniającej się liczbie rekordów przedstawionych w Tabeli 4.

Przy zestawieniu wszystkich pomiarów, pierwszy z nich należało odrzucić, ponieważ znacznie odbiegał od reszty ze względu na dłuższy czas wykonywania zapytania. Sytuacja ta spowodowana była przez brak wcześniejszej analizy składniowej określonego w scenariuszu zapytania, co jednocześnie przekładało się na brak zbudowanego planu wykonania zapytania. W rezultacie pierwsze wykonanie określonego zapytania w scenariuszu poddawane było przez system bazodanowy analizie składniowej. W dalszej kolejności uruchamiany był optymalizator zapytań, który tworzył możliwe plany wykonania zapytania, po czym obliczał ich przybliżone koszty. Następnie wybierał on plan o najniższym możliwym koszcie i zapisywał go, a potem przekazywał go do kolejnego etapu, w którym było wykonywane zapytanie na podstawie tak skonstruowanego planu [17].

Lista kroków, jakie zrealizowano podczas dokonywania pomiaru czasu wykonywania zapytania SQL została zaprezentowana w Tabeli 5.

Tabela 5: Lista czynności realizowanych w trakcie pomiaru czasu wykonywania zapytania SQL

Numer realizowanego kroku	Wykonywana czynność
1	Pomiar czasu przed wykonaniem zapytania SQL
2	Wykonanie określonego zapytania SQL
3	Pomiar czasu po wykonaniu zapytania SQL
4	Obliczenie różnicy pomiędzy pomiarami przed i po wykonaniu zapytania SQL według następującego wzoru: czas_wykonania = (czas_po – czas_przed) * 1000
5	Wyczyszczenie pamięci podręcznej (ang. cache) zapytań, bazy danych

Wyniki pomiarów zapisywano do oddzielnej ramki danych (ang. DataFrame). Po przeprowadzeniu eksperymentu dla konkretnego scenariusza i uzupełnieniu danych w ramce danych została ona wyeksportowana do pliku .csv. Nazwa utworzonego pliku była zgodna z następującym wzorem – scenariusz_X_zestaw_danych_Y.csv, który składał się z dwóch oddzielnych członów. Litera X odnosiła się do realizowanego obecnie scenariusza badań, natomiast litera Y wskazywała na aktualnie wybrany zestaw danych.

4.4. Scenariusze

Pierwszy scenariusz opierał się na wykonaniu operacji INSERT, która polegała na wstawieniu pojedynczego rekordu do tabeli *title*, dane wstawiane były rekord po rekordzie. Budowa zapytania została przedstawiona na listingu 1.

Listing 1: Zapytanie SQL typu INSERT wstawiające pojedynczy rekord do tabeli *title*

```
INSERT INTO title (
    id, primaryTitle, originalTitle, forAdults,
    releaseYear, endYear, runtimeInMinutes,
    title_type_id
)
VALUES (
    :id, :primaryTitle, :originalTitle, :forAdults,
    :releaseYear, :endYear, :runtimeInMinutes,
    :title_type_id
);
```

Drugi scenariusz opierał się na wykonaniu operacji DELETE, która polegała na usunięciu pojedynczego rekordu z tabeli *title_ratings* przy zastosowaniu klauzuli WHERE z jednym warunkiem filtrowania. Składnię zapytania przedstawiono na listingu 2.

Listing 2: Zapytanie SQL typu DELETE usuwające pojedynczy rekord z tabeli *title_ratings* z jednym warunkiem filtrowania

```
DELETE FROM title_ratings
WHERE title_id = 'tt0000514';
```

Trzeci scenariusz opierał się na wykonaniu operacji UPDATE, która polegała na modyfikacji pojedynczego rekordu w tabeli *title_ratings* przy zastosowaniu klauzuli WHERE z jednym warunkiem filtrowania. Wartość w kolumnie *averageRating* jest wypełniana wygenerowaną liczbą pseudolosową (zmiennoprzecinkową) z przedziału (5, 10), zaokrąglaną do 1 miejsca po przecinku. Zapytanie zostało zilustrowane na listingu 3.

Listing 3: Zapytanie SQL typu UPDATE modyfikujące pojedynczy rekord w tabeli *title_ratings* z jednym warunkiem filtrowania

```
UPDATE title_ratings
SET averageRating = ROUND((RAND() * 5) + 5, 1)
WHERE title_id = 'tt0000264';
```

W przypadku scenariuszy nr 1-3 konieczne było sekwencyjne wykonanie następujących czynności w celu przywrócenia stanu bazy danych do stanu sprzed wykonania zapytania:

- dezaktywacja/usunięcie kluczy obcych, które są powiązane z tabelą (opcjonalnie);
- usunięcie wszystkich danych z tabeli;
- wstawienie danych do tabeli, odpowiednio dla badanego zestawu danych;
- aktywacja/utworzenie kluczy obcych, które są powiązane z tabelą (opcjonalnie).

W scenariuszu nr 1, czynności były realizowane po zakończeniu wszystkich iteracji, tj. po wykonaniu

wszystkich zapytań i zebraniu wszystkich pomiarów. Natomiast dla scenariuszy nr 2-3 czynności były wykonywane po każdej iteracji, tzn. po wykonaniu zapytania i dokonania zapisu pomiaru.

Czwarty scenariusz opierał się na wykonaniu operacji SELECT i został podzielony na dwa przypadki.

Pierwszy przypadek A polegał na selekcji rekordów z tabeli *title* przy zastosowaniu klauzuli WHERE z pojedynczym warunkiem filtrowania. Wybierane są wszystkie rekordy, w których wartość kolumny *title_type_id* wynosi 1. W rezultacie zwracane są wszystkie tytuły, w których typ tytułu został przypisany jako film. Nałożony został również limit zwracanych rekordów przy pomocy klauzuli LIMIT, maksymalna liczba zwracanych rekordów została zawężona do 100. Oprócz tego dane zostały posortowane rosnąco względem wartości w kolumnie *id* poprzez wykorzystanie klauzuli ORDER BY z opcją ASC. Budowa zapytania została przedstawiona na listingu 4.

Listing 4: Zapytanie SQL typu SELECT wybierające dane z tabeli *title* z pojedynczym warunkiem filtrowania, limitem zwracanych rekordów i sortowaniem po jednej kolumnie

```
SELECT * FROM title
WHERE title_type_id = 1
ORDER BY id ASC
LIMIT 100;
```

Drugi przypadek B polegał na selekcji rekordów ze scalonych ze sobą dwóch tabel poprzez wykorzystanie klauzuli INNER JOIN. Tabela *title_genres* została połączona złączeniem wewnętrznym z tabelą *genre*. W rezultacie zwracane są rekordy zawierające identyfikator, nazwę gatunku oraz liczbę tytułów o określonym gatunku. W tym celu została użyta pojedyncza funkcja agregująca COUNT, która zwróciła liczbę rekordów na podstawie kolumny *title*_id. Funkcja agregująca wykorzystywała pogrupowane dane według kolumn *genre_id* oraz name przy pomocy klauzuli GROUP BY. Ponadto dane zostały posortowane rosnąco względem wartości w kolumnie *genre_id* przy pomocy klauzuli ORDER BY z opcją ASC. Składnię zapytania zilustrowano na listingu 5.

Listing 5: Zapytanie SQL typu SELECT wybierające dane z tabeli *title_genres* z pojedynczą funkcją agregacji, pojedynczym złączeniem wewnętrznym, grupowaniem danych i sortowaniem po jednej kolumnie

```
SELECT
    tg.genre_id as genre_id,
    g.name as genre_name,
    COUNT(tg.title_id) as titles_counter
FROM title_genres tg
INNER JOIN genre g on tg.genre_id = g.id
GROUP BY tg.genre_id, g.name
ORDER BY genre_id ASC;
```

5. Wyniki

Wyniki dotyczące scenariusza badawczego nr 1, w którym mierzony był czas wykonywania instrukcji INSERT polegającej na wstawieniu pojedynczego rekordu do tabeli *title* zostały przedstawione w Tabeli 6.

Wyniki dotyczące scenariusza badawczego nr 2, w którym badany był czas wykonywania instrukcji DELETE usuwającej pojedynczy rekord z tabeli *title_ratings* spełniający określony warunek zostały przedstawione w Tabeli 7.

Tabela 6: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 1

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	4,059	4,080	4,578	4,410
MySQL maszyna wirtualna	9,047	9,986	9,040	9,382
PostgreSQL kontener	1,665	1,664	1,660	1,655
PostgreSQL maszyna wirtualna	2,549	2,747	2,562	2,344
Microsoft SQL Server kontener	7,404	7,839	11,835	13,978
Microsoft SQL Server maszyna wirtualna	9,000	13,447	18,651	20,595
Oracle kontener	9,505	10,072	15,023	14,218
Oracle maszyna wirtualna	10,628	10,658	10,223	11,314

Tabela 7: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 2

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	4,107	4,804	4,502	4,871
MySQL maszyna wirtualna	9,332	9,046	9,219	11,429
PostgreSQL kontener	1,893	2,142	3,687	3,200
PostgreSQL maszyna wirtualna	2,668	2,743	4,653	3,998
Microsoft SQL Server kontener	11,883	18,183	75,155	308,270
Microsoft SQL Server maszyna wirtualna	16,745	22,428	80,450	364,955
Oracle kontener	9,478	8,797	16,369	13,422
Oracle maszyna wirtualna	10,539	9,575	17,543	23,677

Wyniki uzyskane w Tabeli 6 odnoszące się do scenariusza badawczego nr 1, pokazują, że najlepszym wydajnościowo systemem bazodanowym pod względem szybkości wykonywania zapytań SQL typu INSERT okazał się system PostgreSQL. Dla wszystkich zestawów danych średni czas wykonywania zapytań w przypadku bazy PostgreSQL na kontenerze Docker wynosił 1,7 ms, dlatego też zajął on pierwsze miejsce wśród badanych systemów bazodanowych. Odnosząc się do wyników uzyskanych w pierwszym scenariuszu dla zestawu 4, ostatnie miejsce z najdłuższym czasem wynoszącym 20,6 ms zajął MS SQL Server na maszynie wirtualnej.

Otrzymane wyniki pomiarów w Tabeli 7 dotyczące scenariusza badawczego nr 2, obrazują, że najwydajniejszą bazą danych dla wszystkich zestawów danych jest PostgreSQL osiągając wyniki bliskie 2-5 ms. Równie wydajną bazą danych okazuje się być baza MySQL, osiągając podobne wyniki. Najgorszą natomiast przy tego rodzaju operacjach okazała się być baza MS SQL Server. W początkowej fazie eksperymentu system nieznacznie odbiegał od pozostałych, jednakże wraz z przekroczeniem określonego progu 10 000 rekordów czas wykonywania zapytania drastycznie wzrastał. Dla zestawu 4 uzyskano wyniki w przedziale 308-365 ms.

Wyniki dotyczące scenariusza badawczego nr 3, w którym sprawdzany był czas wykonywania instrukcji UPDATE modyfikującej pojedynczy rekord z tabeli *title_ratings* spełniający określony warunek zostały przedstawione w Tabeli 8.

Tabela 8: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 3

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	4,135	4,626	4,486	5,086
MySQL maszyna wirtualna	9,506	8,699	9,357	12,791
PostgreSQL kontener	1,922	2,179	3,817	3,381
PostgreSQL maszyna wirtualna	3,161	3,079	4,790	4,092
Microsoft SQL Server kontener	13,312	19,109	78,860	309,793
Microsoft SQL Server maszyna wirtualna	18,797	25,219	84,461	371,181
Oracle kontener	48,591	56,526	63,944	59,614
Oracle maszyna wirtualna	50,727	54,686	60,560	62,214

Na podstawie uzyskanych wyników w Tabeli 8 odnoszących się do scenariusza badawczego nr 3, stwierdzono, że najkrótszy czas wykonywania zapytań dla wszystkich zestawów danych osiągnął PostgreSQL z czasem rzędu 2-5 ms. Inną równie wydajną bazą okazuje się być MySQL, osiągając wynik na poziomie 4-13 ms. Różnica czasu wykonywania zapytań pomiędzy PostgreSQL, a MySQL była niewielka i wynosiła kilka milisekund. Najgorzej dla zestawu 1 oraz 2 wypadła baza Oracle z czasem wynoszącym około 48,6- 56,5 ms. Podczas gdy dla zestawu 3 oraz 4 najmniej wydajną okazuje się być baza MS SQL Server z wynikiem nawet 372 ms.

Wyniki dotyczące scenariusza badawczego nr 4 zostały przedstawione w Tabelach 9-10

Tabela 9: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 4 przypadku A

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	1,399	1,667	1,745	3,200
MySQL maszyna wirtualna	1,646	1,903	1,779	2,022
PostgreSQL kontener	1,083	1,297	1,292	2,341
PostgreSQL maszyna wirtualna	1,054	2,010	1,813	2,059
Microsoft SQL Server kontener	6,199	7,011	15,627	30,355
Microsoft SQL Server maszyna wirtualna	10,923	15,138	126,430	432,398
Oracle kontener	11,363	12,037	16,668	26,359
Oracle maszyna wirtualna	12,162	18,409	70,055	174,274

Tabela 10: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 4 przypadku B

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	3,317	18,700	223,249	6995,514
MySQL maszyna wirtualna	4,067	19,881	266,028	8401,838
PostgreSQL kontener	0,912	0,829	0,809	1,639
PostgreSQL maszyna wirtualna	1,117	1,056	0,875	257,710
Microsoft SQL Server kontener	7,744	9,065	30,143	211,049
Microsoft SQL Server maszyna wirtualna	12,832	13,570	47,429	341,674
Oracle kontener	16,509	14,566	44,437	280,221

Oracle maszyna wirtualna	15,730	21,188	44,750	166,475

W czwartym scenariuszu dla przypadku A mierzony był czas wykonywania instrukcji SELECT wybierającej dane z tabeli *title* z pojedynczym warunkiem filtrowania, limitem i sortowaniem. Analizując otrzymane wyniki w Tabeli 9, zauważono, że dwie bazy danych wyróżniały się na tle pozostałych, były to bazy PostgreSQL i MySQL. Średnie czasy dla bazy PostgreSQL i MySQL nie różniły się znacznie od siebie, oscylowały w granicach 1-3,2 ms. Porównując ze sobą uzyskane czasy dla zestawu 4, zdecydowaną przewagę w wydajności nad pozostałymi, analizowanymi bazami danych miała baza MySQL na maszynie wirtualnej. Kolejno po niej znaj-dowały się odpowiednio bazy PostgreSQL na maszynie wirtualnej, PostgreSQL na kontenerze Docker i MySQL na kontenerze Docker. Dla wcześniejszych zestawów od 1 do 3 przewagę nad maszynami wirtualnymi miały kontenery Docker. Najgorszym wyborem dla zestawu 4 okazała się być baza MS SQL Server na maszynie wirtualnej zajmująca ostatnie miejsce, dla której czas wyniósł około 432 ms.

W czwartym scenariuszu dla przypadku B mierzony był czas wykonywania instrukcji SELECT wybierającej dane z tabeli *title_genres* z pojedynczą funkcją agregacji, pojedynczym złączeniem wewnętrznym, grupowaniem danych i sortowaniem. Według uzyskanych wyników w Tabeli 10, najlepszą okazuje się baza PostgreSQL na kontenerze Docker osiągając czas rzędu 1,6 ms dla największego zestawu danych. W zestawieniu z pozostałymi systemami można zauważyć, że różnica pomiędzy czasami dla zestawu 4 była bardzo wysoka i wynosiła ponad 100 ms. Najgorzej w tym zestawieniu wypadła baza danych MySQL na kontenerze Docker i na maszynie wirtualnej z uwagi na jej bardzo oddalony od tej granicy wynik, który wyniósł niemalże 7-8,4 s.

6. Podsumowanie

Reasumując, celem artykułu było porównanie ze sobą wydajności systemów bazodanowych uruchomionych w środowisku skonteneryzowanym oraz zwirtualizowanym. Badanymi systemami bazodanowymi były MySQL, PostgreSQL, Microsoft SQL Server i Oracle. Przy ocenie wydajności posłużono się średnimi czasami wykonywanych zapytań SQL typu INSERT, UPDATE, DELETE oraz SELECT. Zdefiniowano dwie hipotezy badawcze. Pierwsza hipoteza H1 zakładała, że bazy danych uruchomione na kontenerach są wydajniejsze od baz uruchomionych na maszynach wirtualnych. W drugiej hipotezie H2 ustalono, że Oracle jest najwydajniejszy spośród testowanych baz danych niezależnie w jakim środowisku został uruchomiony. Na podstawie wyników przedstawionych w Tabelach 6-10 hipoteza H1 została potwierdzona. Podczas gdy hipoteza H2 została odrzucona, gdyż wykazano, że PostgreSQL charakteryzował się zdecydowanie większą wydajnością pod względem szybkości wykonywania zapytań spośród analizowanych systemów bazodanowych. Porównywalne wyniki do PostgreSQL uzyskała baza MySQL, osiągająca nieznacznie dłuższe czasy wykonywania zapytań. Z kolei system Oracle znajdował się na trzecim, bądź też ostatnim miejscu w zależności od badanego scenariusza. Podsumowując, uzyskane wyniki różnią się nieco od dotychczasowych badań dotyczących omawianych problemów. W obecnej chwili wykazano, że najlepszą bazą jest PostgreSQL, a po niej osiągając zbliżone wyniki znajduje się baza MySQL. Różnice te mogą wynikać z nieustannego postępu technologicznego oraz zmian wprowadzanych w systemach bazodanowych, konfiguracji baz danych, wyboru środowisk, na których są uruchomione, ustawień połączenia oraz języka, w jakim została napisana aplikacja.

Literatura

[1] What is a database? – Simple definition by Oracle, https://www.oracle.com/pl/database/what-is-database/, [02.2023].

[2] Complete ranking of relational DBMS, https://db-engines.com/en/ranking/relational+dbms, [02.2023].

[3] Database as a Service (DBaaS) by TechTarget, https://www.techtarget.com/searchdatamanagement/definition/database-as-a-service-DBaaS, [02.2023].

[4] IBM topics series – What is virtualization?, https://www.ibm.com/topics/virtualization, [02.2023].

[5] IMB topics series – What is containerization?, https://www.ibm.com/topics/containerization, [02.2023].

[6] B.B. Rad, H.J. Bhatti, M. Ahmadi, An Introduction to Docker and Analysis of its Performance, International Journal of Computer Science and Network Security (IJCSNS) 17(3) (2017) 228-235.

[7] R. Klewek, W. Truskowski, M. Skublewska-Paszkowska, Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization, Journal of Computer Sciences Institute (JCSI) 16 (2020) 279-284.

[8] R. Čerešňák, M. Kvet, Comparison of query performance in relational a non-relation databases, Transportation Research Procedia 40 (2019) 170–177.

[9] M. Ilić, L. Kopanja, D. Zlatković, M. Trajković, D. Ćurguz, Microsoft SQL Server and Oracle: Comparative Performance Analysis, The 7th International Conference on Knowledge Management and Informatics, Vrnjačka Banja, 7(poz. 5) (2021) 33-40.

[10] K. Islam, K. Ahsan, S. A. K. Bari, M. Saeed, S. Asim, Huge and Real-Time Database Systems: A Comparative Study and Review for SQL Server 2016, Oracle 12c & MySQL 5.7 for Personal Computer, Journal of Basic & Applied Sciences 13 (2017) 481-490.

[11] Y. Bassil, A Comparative Study on the Performance of the Top DBMS Systems, Journal of Computer Science & Research 1(1) (2012) 20-31.

[12] What is a WSL 2?, https://learn.microsoft.com/en-us/windows/wsl/about#what-is-wsl-2, [03.2023].

[13] Comparing WSL versions, https://learn.microsoft.com/en-us/windows/wsl/compare-versions, [03.2023].

[14] Settings configuration in WSL – Microsoft manual, https://learn.microsoft.com/en-us/windows/wsl/wsl-config, [03.2023].

[15] Specification of IMDb non-commercial datasets, https://developer.imdb.com/non-commercial-datasets/, [03.2023].

[16] Lecture about query optimization – Section 10.2, https://edu.pjwstk.edu.pl/wyklady/szb/scb/rW10.htm, [03.2023].