JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 32 (2024) 151–156
Received: 14 May 2024
Accepted: 7 July 2024

Performance analysis of selected database management systems

Analiza wydajności wybranych systemów zarządzania bazami danych

Radosław Kowalczyk*, Jakub Smołka

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

Abstract

The article presents the results of performance tests of database management systems. In order to conduct the tests, an application running in Java environment and using an ORM tool such as Hibernate for communication with the database was used. The tests were carried out for test scenarios such as CRUD operations, join queries, aggregate functions, and filter conditions in queries. The DBMSes used for testing were MySQL, PostgreSQL and H2. Performance analysis looked at the execution time of operations, RAM usage and CPU load. Additionally, tests were carried out for different sizes of user groups using the application. PostgreSQL proved to be the best choice for complex operations.

Keywords: MySQL; PostgreSQL; H2

Streszczenie

Artykuł prezentuje wyniki testów wydajności systemów zarzadzania bazami danych. W celu przeprowadzenia badań wykorzystano aplikacje działającą w środowisku Java i wykorzystującą do komunikacji z bazą danych narzędzie ORM jakim jest Hibernate. Badania zostały przeprowadzone dla scenariuszy testowych takich jak operacje CRUD, zapytania złączające, funkcje agregujące oraz warunków filtrujących w zapytaniu. Do testów wykorzystano DBMS: MySQL, PostgreSQL i H2. Analiza wydajności dotyczyła czasu wykonania operacji, wykorzystania pamięci RAM oraz obciążenia procesora. Dodatkowo przeprowadzono testy dla różnej wielkości grup użytkowników korzystających z aplikacji. PostgreSQL okazał się najlepszym wyborem dla złożonych operacji.

Słowa kluczowe: MySQL; PostgreSQL; H2

*Corresponding author

Email address: radoslaw.kowalczyk1@pollub.edu.pl (R. Kowalczyk)

1. Wstęp

Szybki rozwój technologii informacyjnych powoduje duże zapotrzebowanie na jak najlepszy sposób przechowywania i dostępu do danych. Najczęściej stosowane są techniki ORM (Object-Relational Mapping) [1]. Gromadzenie danych przez organizację z różnych sektorów gospodarki sprawia, że system zarządzający danymi powinien być wydajny oraz wychodzić naprzeciw nowym zadaniom stawianym przez klientów. Powoduje to ciągłą ewolucję systemów DBMS (ang. DataBase Managment System).

Artykuł przedstawia wyniki porównania danych otrzymane dla różnego rodzaju zapytań, co pozwala na określenie zalet i wad poszczególnych systemów bazodanowych w połączeniu z ORM [2]. Porównując rezultaty badań, można określić, który system będzie najlepszym wyborem. Duża liczba wykorzystanych scenariuszy pozwala dokładniej odwzorować mocne i słabe strony poszczególnych systemów.

Analiza wydajności systemu bazodanowego jest istotna, ponieważ pomaga zidentyfikować sposób wykorzystania pamięci RAM oraz procesora. Pozwala to na dostosowanie architektury sprzętowej, co wpłynie na zmniejszenie kosztów poniesionych przez użytkowników. Identyfikacja problemów z szybkością działania pozwoli na wprowadzenie optymalizacji systemu, co przyczyni się do wzrostu konkurencyjności w danej branży oraz zwiększenia poziomu zadowolenia klientów. W środowiskach gdzie czas reakcji jest kluczowy, może to znacząco wpływać na wyniki biznesowe.

Jednymi z najpopularniejszych systemów używanych do różnych celów i oferujących różne funkcjonalności są: MySQL, PostgreSQL i H2. Dlatego też te trzy systemy zostały przetestowane pod względem funkcjonalności. Językiem, który został wykorzystany do stworzenia aplikacji upraszczającej korzystanie z bazy danych, jest Java. Wybór ten był podyktowany szerokim zastosowaniem oraz powszechnym użyciem tego języka. Narzędziem ORM, które zostało użyte w tym środowisku, jest najpopularniejsze narzędziem używanym do mapowania obiektowo-relacyjnego w Javie. Narzędziem tym jest Hibernate, który jest wciąż rozwijany i optymalizowany. Wersja narzędzia użyta do budowy aplikacji to 6.4.0 [3].

2. Przegląd literatury

Tematy związane z wydajnością baz danych są na tyle aktualne i popularne, że pojawiają się w wielu publikacjach naukowych. Porównanie wydajności systemów zarządzania bazami danych PostgreSQL i MySQL w kontekście aplikacji desktopowych zostało przedstawione w artykule „Porównanie wydajności relacyjnych baz danych PostgreSQL oraz MySQL dla aplikacji desktopowych” [4]. Celem badania było porównanie wydajności PostgreSQL i MySQL, aby zidentyfikować, który system osiąga lepsze wyniki. Przeprowadzono analizę podstawowych operacjach bazodanowych: do-

dawanie, pobieranie, aktualizacja i usuwanie rekordów. Wyniki wskazują, że PostgreSQL może oferować lepszą wydajność w aplikacjach desktopowych.

Porównanie systemów DBMS: MySQL, PostgreSQL, MariaDB oraz H2 zostało przedstawione w artykule „Wydajność pracy z bazami danych w aplikacjach JEE" [2]. Badano wydajność systemów DBMS w kontekście aplikacji JEE. Przeprowadzono serię testów wydajnościowych na operacjach CRUD. MySQL wykazał niższą wydajność przy większych zbiorach danych, H2 wykazał najkrótsze czasy wykonania w większości operacji, z wyjątkiem aktualizacji, gdzie najlepiej wypadł PostgreSQL.

Badanie wpływu różnych technologii dostępu do danych zostało przedstawione w artykule „Hybrydowe metody pracy z bazami danych w aplikacjach JEE" [5]. Artykuł prezentuje wyniki badań nad hybrydowymi metodami pracy z bazami danych w celu zoptymalizowania wydajności operacji bazodanowych. Parametrami uwzględnionymi w badaniu były: czas wykonania operacji oraz zużycie pamięci RAM. Wykorzystanie hybrydowych metod pozwoliło na uzyskanie wyższej wydajności niż przy użyciu pojedynczej technologii.

Analiza literatury dowodzi, że wybór systemu zarządzania bazami danych może istotnie wpłynąć na działanie aplikacji. PostgreSQL wypada korzystnie w testach wydajności, szczególnie w aplikacjach wymagających skomplikowanych operacji z niewielką ilością danych.

3. Cel badawczy

Celem badań było ustalenie mocnych i słabych stron systemu zarządzania bazami danych. W tym celu poddano analizie wydajność trzech systemów zarządzających relacyjną bazą danych. Dwa pierwsze są popularnym obecnie wyborem na rynku [6], są to: MySQL i PostgreSql, natomiast trzeci z nich - H2 - jest często używanym rozwiązaniem, które jednocześnie charakteryzuje się mniejszym obciążeniem sprzętowym.

Aby dokonać szczegółowej analizy, przetestowano operacje typu CRUD, wyszukiwania z wykorzystaniem kilku tabel połączonych ze sobą określonymi relacjami, filtrowanie wyników według określonych kryteriów oraz wykorzystywanie funkcji agregujących. Dodatkowo, aby zapewnić warunki podobne do środowiska produkcyjnego, wprowadzono scenariusze testowe dla różnej liczby użytkowników. W analizie wydajności wykorzystano trzy kryteria: czas trwania operacji, wykorzystanie pamięci i procesora [5].

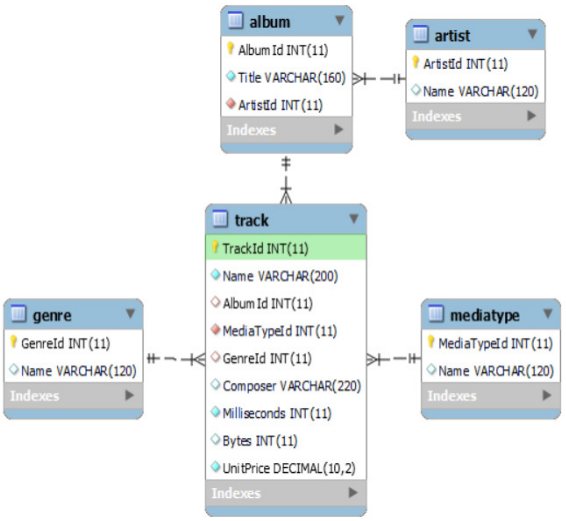
4. Scenariusze i metoda badań

W celu przeprowadzenia badania stworzono relacyjną bazę danych[7] oraz aplikację połączoną z bazą. Schemat bazy danych przedstawiono na rysunku (Rysunek 1). Do zbudowania aplikacji został użyty język obiektowy Java. Połączenie z bazą odbywa się poprzez framework Hibernate, który wykorzystuje API JDBC [8], dlatego też Hibernate może w prosty sposób komunikować się z różnymi systemami baz danych. Do efektywnego zarządzania pulą połączeń w aplikacji zastosowano bibliotekę c3p0. Aktywne połączenia do bazy danych przechowywane są w puli i wypożyczane przez aplikację, zamiast tworzenia nowego połączenia za każdym razem. Dodatkowo c3p0 pozwala na ustawienie właściwości puli, takich jak: minimalna i maksymalna liczba połączeń oraz timeouty. Wykonywanie operacji na stworzonej bazie odbywa się z poziomu aplikacji. W celu analizy różnych przypadków stworzono scenariusze testowe, które zostały przedstawione w Tabeli 1.

Tabela 1: Opis scenariuszy testowych

S1	Wstawienie rekordów do tabeli z wykorzystaniem automatycznego generowania id
S2	Aktualizacja rekordów w tabeli przy użyciu zapytania HQL i metody createQuery()
S3	Wyszukiwanie rekordów o podanych kryteriach
S4	Usuwanie rekordów
S5	Zapytanie łączące tabele: Artist, Album i Track
S6	Zapytanie obliczające średni czas trwania utworów dla każdego albumu
S7	Wykonanie zapytania HQL w celu znalezienia wszystkich utworów danego artysty mające określoną cenę. Zapytanie filtruje utwory należące do określonego wykonawcy i nie przekraczające określonej ceny
S8	Wykonywanie wyszukiwania jednego rekordu poprzez określoną liczbę użytkowników
S9	Wykonywanie wyszukiwania 10000 rekordów poprzez określoną liczbę użytkowników
S10	Wykonywania wyszukiwania złączającego tabele dla jednego rekordu przez wielu użytkowników
S11	Wykonywanie wyszukiwania złączającego tabele dla 10000 rekordów przez wielu użytkowników

Powyższe scenariusze zostały wykonane przy wykorzystaniu trzech systemów zarządzania bazami danych: PostgreSQL, MySQL i H2. Scenariusze od S1 do S6 włącznie były wykonane dla różnej liczby rekordów: 1000, 10000, 50000, 100000, 500000, 1000000. Scenariusze od S7 do S10 włącznie były wykonane dla różnych wielkości grup użytkowników: 10, 50, 100, 500, 1000. W tym celu zaimplementowano wielowątkową architekturę, której zadaniem była symulacja działania użytkowników. Każdy wątek reprezentował oddzielnego użytkownika systemu. Stworzenie grupy wątków pozwoliło zasymulować rzeczywiste obciążenie systemu. W scenariuszach uwzględniono wyszukiwanie zarówno zapytań prostych, jak i złożonych. Za pomocą scenariuszy zamieszczonych w tabeli (Tabela 1) zmierzono czas wykonywania danego scenariusza, użycie procesora oraz wykorzystanie pamięci RAM. Pomiar czasu został zrealizowany przy pomocy funkcji System.currentTimeMillis(), zwracającej czas uruchomieniowy Wirtualnej Maszyny Javy [9], zapisując wartość do zmiennej przed uruchomieniem scenariusza i odejmują tą zmienną od wartości po zakończeniu scenariusza. Pomiar użycia procesora oraz pamięci RAM został wykonany przy pomocy narzędzia JProfiler.



Rysunek 1: Schemat bazy danych wykorzystany podczas badań.

Baza danych zawiera pięć tabel: track, genre, mediatype, album i artist, połączonych odpowiednio relacjami (Rysunek 1). Badania zostały przeprowadzone dla trzech systemów bazodanowych, w których stworzono relacyjną bazę danych. Dane zostały wygenerowane losowo w celu przeprowadzenia poszczególnych operacji. Baza została osadzona na serwerze lokalnym. Do testów użyto poniższego sprzętu (Tabela 2).

Tabela 2: Specyfikacja sprzętowa wykorzystana do badań

Producent	Lenovo
Model	82LN
Procesor	AMD Ryzen 7 5700U
Pamięć RAM	16GB
Karta graficzna	AMD Radeon
System	Windows 11 Home, 64 bitowy

5. Wyniki badań

W dalszej części artykułu przedstawiono wyniki badań w formie tabel. Badania koncentrują się na wyznaczeniu czasów wykonania zapytań przez każdy z systemów zarządzających bazą danych, wykorzystaniu procesora oraz zużyciu pamięci RAM.

Tabela 3 przedstawia czasy wykonania podstawowych operacji CRUD: dodawania, wyszukiwanie, aktualizacja i usuwanie rekordów w bazie danych. W Tabeli 4 pokazano średnie procentowe użycie procesora podczas wykonywania operacji CRUD. Tabela 5 prezentuje dane dotyczące zużycia pamięci RAM podczas wykonywania scenariuszy od S1 do S4.

Dla Tabel 3,4 oraz 5 operacje CRUD mają swoje odpowiedniki w określonych scenariuszach: operacja tworzenia (Create) odpowiada scenariuszowi S1, odczyt (Read) – scenariuszowi S3, aktualizacji (Update) – scenariuszowi S2, natomiast usuwania (Delete) – scenariuszowi S4.

Tabela 3: Scenariusz S1-S4 uzyskane czasy

Operacja	Operacje CRUD		
Mierzone parametry	Średni czas wykonania operacji [ms]		
Liczba rekordów	MySQL	PostgreSQL	H2
1000	1121 358 418 330	786 362 391 323	1019 330 376 343
10000	6614 388 546 352	3905 377 460 434	5443 367 494 427
50000	17249 1209 1013 547	12743 841 788 932	10234 759 875 522
100000	28444 1402 1125 750	21775 1206 975 1494	18202 1042 1096 695
500000	119658 3707 2394 2155	94365 3924 1825 6516	81068 5253 3128 3108
1000000	240438 6966 3115 4079	183610 8896 2707 12540	149978 9448 5669 5706

W tabeli 3 widać, że PostgreSQL uzyskuje lepsze czasy wykonania operacji CRUD, szczególnie przy większej liczbie rekordów.

Tabela 4: Scenariusze S1-S4 - wykorzystanie CPU

Operacja	Operację CRUD		
Mierzone parametry	Średnie użycie CPU [%]		
Liczba rekordów	MySQL	PostgreSQL	H2
1000	5,27 4,98 4,48 4,68	4,52 4,78 4,38 4,09	4,59 5,16 5,26 5,75
10000	9,17 5,37 5,20 5,27	6,14 5,08 5,45 4,10	9,54 5,85 5,65 6,45
50000	9,94 5,92 5,46 5,87	9,08 5,75 5,93 5,84	13,76 6,04 9,16 4,68
100000	10,15 7,03 7,71 7,71	9,65 6,91 7,42 6,72	12,30 6,43 4,77 6,15
500000	11,02 7,32 12,48 6,83	11,02 8,49 8,48 6,93	12,38 6,85 7,59 5,66
1000000	17,18 8,04 16,11 5,95	11,61 6,71 9,66 7,22	11,91 8,67 9,66 6,04

Na podstawie danych zaprezentowanych w Tabeli 4 można zauważyć, że MySQL wykazuje większe zapotrzebowanie na użycie procesora przy większej liczbie rekordów.

Tabela 5: Scenariusz S1-S4 - zużycie pamięci

Operacja	Operacje CRUD		
Mierzone parametry	Średnie zużycie RAM [MB]		
Liczba rekordów	MySQL	PostgreSQL	H2
1000	25,64	23,11	17,68
	22,92	24,44	22,50
	24,42	23,20	23,57
	20,70	20,67	18,95
10000	26,56	29,06	27,32
	23,84	27,02	24,44
	26,42	23,45	28,89
	21,13	30,54	23,21
50000	98,39	99,81	98,41
	25,79	24,90	29,53
	65,28	70,08	43,80
	21,42	22,95	22,64
100000	106,00	105,10	107,70
	30,93	26,18	35,14
	104,10	89,12	78,88
	23,29	24,52	26,05
500000	286,40	285,90	322,50
	23,44	24,36	50,49
	353,60	352,00	280,20
	52,47	25,43	26,95
1000000	556,20	545,70	594,40
	23,01	24,98	51,47
	632,10	654,80	474,80
	52,91	28,03	50,49

Analiza danych z Tabeli 5 wskazuje, że H2 charakteryzuje się niższym zużyciem pamięci RAM niż PostgreSQL i MySQL dla małej liczby rekordów.

Tabela 6 przedstawia uzyskane czasy wykonania złożonych zapytań dla scenariuszy S5 – S7. Tabela 7 zawiera średnie użycie procesora podczas wykonywania złożonych zapytań. Natomiast wykorzystanie pamięci RAM przedstawiono w Tabeli 8.

Tabela 6: Scenariusze S5 - S7

Operacje	Wykonywanie zapytań złączających wiele tabel		
Mierzone parametry	Średni czas wykonania operacji [ms]		
Liczba rekordów	MySQL	PostgreSQL	H2
1000	571	583	572
	538	579	512
	537	549	550
10000	859	811	821
	549	589	530
	646	653	648
50000	1446	1672	1270
	777	616	790
	1202	1051	1098
100000	2247	1929	1878
	878	674	876
	1566	1254	1603
500000	6792	3832	6540
	2810	801	1501
	6642	3269	5873
1000000	11637	7813	12319
	5800	1061	2404
	12090	5284	18777

W tabeli 6 widać, że w przypadku małych i średnich wielkości zbiorów rekordów czasy wykonania zapytań są podobne. Natomiast dla 1000000 rekordów PostgreSQL uzyskuje krótszy czas wykonania zapytań.

Tabela 7: Scenariusze S5-S7 użycie CPU

Operacje	wykonywanie zapytań złączających wiele tabel		
Mierzone parametry	Średnie użycie CPU [%]		
Liczba rekordów	MySQL	PostgreSQL	H2
1000	5,46	5,78	5,85
	5,06	4,98	4,78
	4,97	5,07	5,27
10000	6,05	6,15	6,17
	5,36	5,17	5,07
	5,26	5,06	5,56
50000	6,25	6,24	6,62
	7,41	6,74	5,50
	8,09	6,63	7,02
100000	7,02	7,20	7,22
	9,37	9,63	10,55
	8,77	8,78	7,32
500000	8,47	8,29	8,01
	7,42	7,50	8,85
	10,73	10,26	8,38
1000000	12,30	9,47	12,88
	6,83	6,73	7,22
	16,30	14,36	10,15

Z danych zawartych w Tabeli 7 widać, że użycie procesora we wszystkich systemach bazodanowych jest podobne. Dopiero dla 1000000 rekordów widać mniejsze użycie CPU przez PostgreSQL.

Tabela 8:Scenariusze S5-S7 użycie RAM

Operacje	Wykonywanie zapytań złączających wiele tabel		
Mierzone parametry	Średnie zużycie RAM [MB]		
Liczba rekordów	MySQL	PostgreSQL	H2
1000	20,84	20,89	22,48
	21,74	21,56	23,53
	27,47	22,18	20,64
10000	27,67	25,86	23,95
	22,96	23,09	25,53
	29,48	27,33	23,14
50000	59,57	77,94	61,66
	26,89	70,77	26,95
	63,81	50,97	24,95
100000	101,1	112,00	88,60
	72,64	68,31	29,77
	65,81	80,90	42,31
500000	456,90	445,30	306,40
	23,50	26,40	69,32
	252,80	238,10	108,60
1000000	835,50	865,30	534,10
	21,77	26,08	83,78
	590,60	616,40	123,30

Dane przedstawione w tabeli 8 pokazują, że dla scenariusza S5 i S7 przy mniejszej liczbie rekordów zużycie RAM jest podobne. Natomiast dla dużej liczby rekordów zużycie pamięci RAM w systemach MySQL i PostgreSQL znacznie wzrasta. W scenariuszu S5 odpowiednio: MySQL o 3910,56%, PostgreSQL o 3877,98% , H2 o 2607,74%. Natomiast dla scenariusza S7 wzrost zużycia RAM wyniósł: MySQL o 2049,98%, PostgreSQL o 2679,08% , H2 o 497,38%.

Tabela 9 przedstawia czasy uzyskane dla wykonania scenariuszy S8 – S11, które przeprowadzono dla różnej liczby użytkowników. Wykorzystanie procesora poka-

zano w Tabeli 10. Tabela 11 zawiera wyniki badań dotyczące wykorzystania pamięci RAM podczas wykonywania zapytań do bazy danych.

Tabela 9: Scenariusze S8-S11

Operacje	Wyszukiwanie		
Mierzone parametry	Średni czas operacji dla użytkownika [ms]		
Liczba użytkowników	MySQL	PostgreSQL	H2
10	719,2 1096,3 909,5 1872	757,8 947,5 902,2 1728,7	942,2 1480,6 808,2 1668,3
50	1258,8 2540,5 1440,9 4225,2	928,5 1665,5 1434,5 4229,2	1825,9 3324,8 1425,4 4566
100	1592,9 4118,1 2017 7336,9	1685,0 2809,6 1911,1 6281,8	2220,0 3741,5 1770,7 8124,5
500	3034,4 12261,9 2916,9 34476,03	9470,9 11694,7 10047,3 32711,3	3267,0 8579,5 2661,4 29155,1
1000	3590,7 14243,7 3547,5 -	22953,4 27288,9 23318,4 84705,1	4661,4 9737,8 4561,6 -

W Tabeli 9 widać, że dla 10 użytkowników różnice w czasie odpowiedzi są minimalne, natomiast przy większej liczbie użytkowników czasy odpowiedzi dla PostgreSQL i H2 znacząco się zwiększają. Przy 1000 użytkowników bazy danych H2 i MySQL uległy awarii.

Tabela 10: Scenariusze S8- S11 użycie CPU

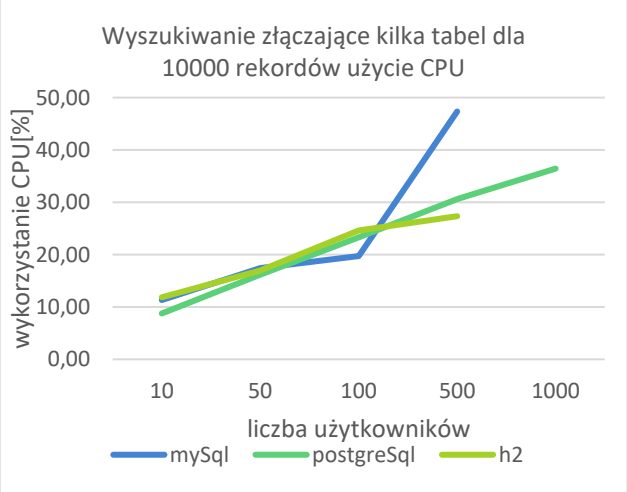
Operacje	Wyszukiwanie		
Mierzone parametry	Średnie użycie procesora [%]		
Liczba użytkowników	MySQL	PostgreSQL	H2
10	7,60 7,31 7,11 11,33	7,40 7,41 7,30 8,76	6,72 8,86 9,76 11,85
50	9,06 11,87 10,80 17,42	7,47 8,59 10,04 16,18	8,19 12,71 10,04 16,97
100	14,57 18,81 14,19 19,74	8,45 13,98 11,67 23,30	13,97 15,29 18,43 24,62
500	14,70 41,52 15,13 47,34	9,87 30,92 14,91 30,62	31,54 21,14 20,36 27,34
1000	14,83 50,32 15,43 -	10,96 31,74 15,20 36,43	42,03 32,94 21,47 -

Dane w Tabeli 10 wyraźnie pokazują, że zarządzanie dużą liczbą równoczesnych połączeń stanowi wyzwanie dla wszystkich testowanych systemów. Wykorzystanie procesora jest mniejsze w PostgreSQL niż w pozostałych dwóch systemach bazodanowych. Dodatkowo dla 1000 użytkowników MySQL i H2 przestają działać.

Tabela 11: Scenariusze S8-S11 użycie RAM

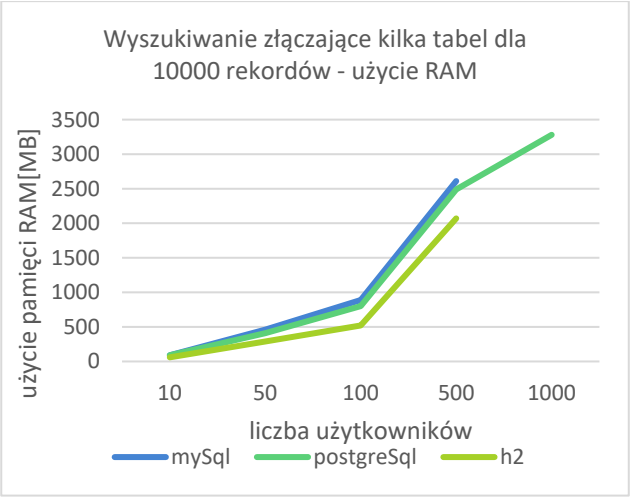
Operacje	Wyszukiwanie		
Mierzone parametry	Średnie użycie RAM [MB]		
Liczba użytkowników	MySQL	PostgreSQL	H2
10	33,3 80 25,7 90	29,5 80 61,3 90	23,4 60 75,8 60
50	55 340 68,7 460	64,5 210 70,6 410	55,1 220 77,3 290
100	94,1 610 124,4 890	65,7 350 88,3 800	91,9 490 140,3 520
500	94,6 1620 159,9 2610	83 960 93,3 2490	126,8 1060 177,5 2070
1000	124,4 1971 202,1 -	83,4 1200 122,3 3280	159,5 1240 178,1 -

Na podstawie danych z Tabeli 11 można zauważyć, że bazy danych MySQL i H2 uległy awarii przy 1000 użytkownikach. Poniżej przedstawiono wykresy zależności wykorzystania procesora od liczby użytkowników dla tej operacji.



Rysunek 2: Wykres użycia CPU dla scenariusza S11.

Na rysunku (Rysunek 2) widać znaczny wzrost użycia procesora w MySQL dla 500 użytkowników, co mogło spowodować wyczerpanie zasobów systemowych przy 1000 użytkowników w konsekwencji doprowadzając do zawieszenia bazy. Natomiast dla bazy H2 nie zaobserwowano problemów z wykorzystaniem zasobów, jednak mimo to doszło do zablokowania systemu. Analizując wykorzystanie pamięci RAM z rysunku (Rysunek 3), również nie można znaleźć przyczyny takiego zachowania aplikacji. W dziennikach logów otrzymano następujący komunikat (Rysunek 4).



Rysunek 3: Wykres użycie pamięci RAM dla scenariusza S11.

```
WARNING: Thread[#53,C3P0PooledConnectionPoolManager[identityToken->1hgekqgb21gjmq6qgysc9|l4e9658b5]-HelperThread-#1,5,main] -- caught unexpected Exception while executing posted task.
java.lang.IllegalStateException: Timer already cancelled.
```

Rysunek 4: Dziennik z otrzymanym wyjątkiem.

Analizując dzienniki z aplikacji można wysnuć wniosek, że problem wystąpił w zarządzaniu pulą połączeń. Czas oczekiwania na odpowiedź z bazy danych przewyższył przyjęty czas na udzielenie odpowiedzi wynoszący 30 sekund.

6. Wnioski

Celem badania było porównanie wybranych systemów zarządzania bazami danych, poprzez przetestowanie ich w różnych warunkach. Otrzymane dane pozwoliły stwierdzić, który z systemów jest odpowiedni do wykorzystanie podczas budowania aplikacji bazodanowej. Na podstawie wyników otrzymanych dla scenariuszy S1-S4 można wyciągnąć następujące wnioski:

1. PostgreSQL jest wydajny i ekonomiczny pod względem zużycia zasobów, ale może napotykać wyzwania przy dużych zestawach danych, szczególnie w operacjach usuwania rekordów.
2. MySQL prezentuje wysoką wydajność zwłaszcza przy usuwaniu dużych zbiorów danych, Jednak może wymagać większych zasobów sprzętowych.
3. H2 zapewnia wysoką wydajność przy mniejszych zbiorach danych, w przypadku większych zbiorów może mieć problem z zarządzaniem zasobami.

Analizując wyniki dla kolejnych scenariuszy S5-S7 można wyciągnąć następujące wnioski:

1. PostgreSQL wykazuje najwyższą wydajność szczególnie dla dużych zestawów danych, kosztem wyższego zużycia CPU I RAM.
2. MySQL wykazał wysoką wydajność przy mniejszych zestawach danych, ale napotyka problem podczas przetwarzania większych zestawów.
3. H2 ma najwyższą wydajność przy mniejszych zbiorach danych.

Scenariusze S8-S11 są to testy obciążające bazę poprzez zwiększanie liczby użytkowników, Na podstawie wyników wykonania scenariuszy wyciągnięto wnioski:

1. PostgreSQL osiąga wysoką wydajność pod względem czasu operacji dla mniejszej liczby użytkowników i mniejszej ilości danych, ale przy wyższym obciążeniu wydajność spada. Zachowuje jednak zdolność do przetwarzania zapytań tam gdzie inne systemy przestają działać.
2. MySQL utrzymuje stabilną wydajność przez cały czas, napotyka problemy dopiero dla 1000 użytkowników i wyszukiwania złożonych zapytań, czego skutkiem jest zawieszenie wykonywanej operacji.
3. H2 wykazuje problemy z wydajnością czasową i zużyciem zasobów przy złożonych operacjach z duża liczbą rekordów. Podobnie jak w przypadku MySQL przestaje działać przy dużym obciążeniu.

Podsumowując wyniki badań można dojść do wniosku, że PostgreSQL dostarcza efektywne rozwiązania do obsługi złożonych zapytań i może być najlepszym wyborem dla aplikacji o wysokim obciążeniu i złożoności operacji, pod warunkiem zapewnienia odpowiednich zasobów, MySQL i H2 mogą być bardziej odpowiednie dla aplikacji z mniejszymi wymaganiami względem złożoności operacji lub środowisk, gdzie balans pomiędzy wydajnością a zużyciem zasobów jest bardziej krytyczny.

Literatura

[1] N.Dhingra, E.Abdelmoghith, H.T.Mouftah, Performance Evaluation of JPA Based ORM Techniques, In Proceedings of the 2nd International Conference on Computer Science Networks and Information Technology (2016) 263–269.

[2] M.Grzesińska, M.Waszczyńska, B.Pańczyk, JEE Database Applications Performance, Informatyka, Automatyka, Pomiar W Gospodarce I Ochronie Środowiska 6(4) (2016) 73–76, https://doi.org/10.5604/01.3001.0009.5194.

[3] Hibernate technical documentation, https://hibernate.org, [06.04.2024].

[4] B.M.Klimek, M.Skublewska-Paszkowska, Comparison of the performance of relational databases PostgreSQL and MySQL for desktop application, Journal of Computer Sciences Institute 18 (2021) 61–66, https://doi.org/10.35784/jcsi.2314.

[5] K.Jóźwicka, M.Mitrus, Hybrid methods of working with databases in JEE applications, Journal of Computer Sciences Institute 12 (2019) 167–171, https://doi.org/10.35784/jcsi.433.

[6] Relational DBMS popularity ranking, https://db-engines.com/en/ranking/relational+dbms, [06.04.2024].

[7] R.Elmasri, S.Navathe, Fundaamentals of Database Systems, Pearson, Boston Munich, 2016.

[8] C.Bauer, G.King, G.Gregory, Java Persistence with Hibernate, Helion, Gliwice, 2017.

[9] Java Documentation, https://docs.oracle.com/javase/8/docs/api, [06.04.2024].