JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 27 (2023) 178–185
Received: 3 May 2023
Accepted: 15 May 2023

The analysis of Java ORM frameworks performance in terms of analytical data processing

Analiza efektywności analitycznego przetwarzania danych w języku Java z wykorzystaniem wybranych narzędzi ORM

Justyna Baran*, Piotr Muryjas

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

Abstract
The aim of this paper is to evaluate Java ORM frameworks in terms of analytical data processing. The analysis includes following technologies: Hibernate, Apache Cayenne, EclipseLink and DataNucleus. Article contains characteristics and importance of ORM technologies, as well as the research of related literature. The main study has been performed with the use of implemented Java applications that enabled to run and measure the execution time of analytical queries with various levels of complexity. The analysis of the obtained results enabled to define Hibernate as the most efficient technology for analytical data processing. Additionally the limitations of examined tools and the significant differences between them have been identified and presented.

Keywords: Java; ORM; analytical data processing; performance

Streszczenie
Celem artykułu jest ocena efektywności analitycznego przetwarzania danych w języku Java z użyciem technologii ORM takich jak Hibernate, Apache Cayenne, EclipseLink oraz DataNucleus. Na wstępie przedstawiono charakterystykę i znaczenie narzędzi ORM oraz dokonano przeglądu literatury przedmiotu. Badania zostały zrealizowane przy użyciu aplikacji zaimplementowanych w języku Java i polegały na zmierzeniu czasu wykonania zapytań analitycznych o różnym stopniu złożoności. Uzyskane wyniki badań pozwalają stwierdzić, iż Hibernate jest najbardziej efektywną technologią stosowaną do analitycznego przetwarzania danych. Dodatkowo, na ich podstawie dokonano identyfikacji ograniczeń zastosowania wybranych technologii ORM oraz wskazano istotne różnice występujące między poszczególnymi narzędziami.

Słowa kluczowe: Java; ORM; analityczne przetwarzanie danych; efektywność

*Corresponding author
Email address: **s99171@pollub.edu.pl** (J. Baran)

1. Wstęp

Dane są nieodłączną częścią większości istniejących aplikacji, które gromadzą je oraz umożliwiają ich przetwarzanie. Oprócz podstawowych operacji takich jak odczyt, zapis, modyfikowanie czy usuwanie rekordów, zebrane dane mogą także być analizowane. Analityczne przetwarzanie danych w dzisiejszych czasach pełni istotną rolę w rozwoju organizacji na wielu płaszczyznach. Dostarcza ono cennych informacji, które umożliwiają wsparcie i optymalizację procesów planowania czy podejmowania decyzji. Aby zapewnić wysoką jakość i skuteczność tych działań, niezbędne jest przeprowadzenie efektywnej analizy danych, która często wiąże się z koniecznością wykonywania złożonych zapytań na dużych zbiorach danych. Z tego względu istotna jest kwestia wydajności tego procesu.

Wśród istniejących paradygmatów programowania pod względem popularności wyróżnia się programowanie obiektowe. W procesie komunikacji między relacyjnymi bazami danych, a obiektowymi językami programowania wsparcie stanowią technologie mapowania obiektowo-relacyjnego, czyli ORM.

Jednym z najpopularniejszych języków obiektowych jest Java, która stanowi zaawansowane, wieloplatformowe rozwiązanie i umożliwia efektywną implementację wydajnego oprogramowania biznesowego. Udostępnia ona liczne dodatkowe biblioteki, a także technologie służące do mapowania obiektowo-relacyjnego. Wybór odpowiedniego narzędzia ORM może zależeć od konkretnych potrzeb i przeznaczenia oraz powinien zostać dostosowany do indywidualnych wymagań rozwiązania oferowanego użytkownikowi końcowemu.

Głównym celem niniejszego artykułu jest ocena wybranych narzędzi ORM z punktu widzenia efektywności ich użycia do analitycznego przetwarzania danych. W dalszej jego części przedstawione zostaną szczegółowe wyniki badań dla czterech technologii przeznaczonych dla języka Java. Wybrane do badań narzędzia to: Hibernate, Apache Cayenne, EclipseLink oraz DataNucleus. Wyniki badań pozwoliły dokonać oceny szybkości wykonywania zapytań analitycznych, uruchamianych w poszczególnych środowiskach, a także zaobserwować różnice występujące pomiędzy nimi.

2. Przegląd literatury

Niniejszy rozdział zawiera analizę literatury obejmującej tematykę technologii ORM dostępnych dla języka Java.

S.N. Bhatii i in. dokonali analizy wydajności narzędzi Hibernate, Ebean oraz Oracle TopLink [1]. Porównanie zostało przeprowadzone na podstawie czasu wykonania wybranych zapytań SQL. Środowisko badawcze przygotowano z wykorzystaniem bazy danych MySQL, zawierającej dwie tabele powiązane relacją jeden do wielu, w których znajdowało się odpowiednio 5200 i 40 rekordów. Podczas badań wykonano instrukcje INSERT, DELETE, UPDATE oraz SELECT z różnymi operatorami. Spośród badanych narzędzi, technologia Ebean okazała się najbardziej wydajna, jedynie w przypadku operacji INSERT lepszy rezultat uzyskała technologia Hibernate. Dla wszystkich przeprowadzonych scenariuszy narzędzie TopLink uzyskało najgorsze wyniki.

Poszerzone badania w tym obszarze przeprowadzili N. Dhingra i in. W ich publikacji [2] dokonano porównania efektywności technologii opartych na standardzie JPA, tj. Hibernate, EclipseLink, OpenJPA oraz DataNucleus. W ramach przeprowadzonych badań zweryfikowano wpływ różnych zapytań na wykorzystanie procesora i dysku oraz zarządzanie wątkami i pamięcią przez maszynę wirtualną Javy (JVM). W badaniach wykorzystano pięć zapytań SQL zawierających: złączenie wewnętrzne, stronicowanie, zliczanie rekordów, funkcje agregujące oraz zapytanie sparametryzowane. Z punktu widzenia w/w kryteriów najlepsze rezultaty uzyskały technologie Hibernate oraz EclipseLink, a narzędzie OpenJPA okazało się najmniej wydajne. Technologia DataNucleus uzyskała akceptowalne wyniki, ale rozwiązanie to nie jest powszechnie stosowane ze względu na duży stopień skomplikowania.

Podobne badania zostały przeprowadzone przez B.Pllana, a ich celem szczegółowym było porównanie wydajności trzech najpopularniejszych technologii ORM implementujących specyfikację JPA, tj. EclipseLink, Hibernate oraz OpenJPA [3]. Autor dokonał oceny efektywności wykonywania operacji DML (INSERT, UPDATE, DELETE) oraz prostych zapytań SELECT. Dla zapytań DML najbardziej efektywną okazała się technologia OpenJPA, na drugim miejscu uplasował się EclipseLink, a na ostatnim Hibernate. Z kolei w przypadku zapytań SELECT narzędzie Hibernate uzyskało najlepsze wyniki, natomiast technologia OpenJPA okazała się najmniej efektywna. Wyniki badań dodatkowo pokazały, iż testowane technologie charakteryzują się różną wydajnością w zależności od rodzaju przeprowadzanych operacji.

Ocenę wydajności technologii Hibernate, EclipseLink, Apache OpenJPA oraz DataNucleus przeprowadzili również M.Połeć i in. [4]. W ich badaniach wykorzystano aplikację webową połączoną z bazą danych MySQL, zawierającą 14 tabel z relacjami jeden do jeden oraz jeden do wielu. Wykorzystane przez nich scenariusze badawcze obejmowały odczytywanie, tworzenie, aktualizowanie oraz usuwanie danych dla różnej liczby rekordów w poszczególnych tabelach. Uzyskane wyniki badań wskazują, iż najwyższą wydajnością charakteryzuje się narzędzie Apache OpenJPA. Jednak pod względem konfiguracji oraz dostępności dokumentacji technologia Hibernate posiada znaczącą przewagę.

W uzupełnieniu warto także przedstawić wnioski z badań przeprowadzonych przez M. Żuchnik i in., dotyczących oceny wydajność komunikacji z bazą danych przy pomocy interfejsu JDBC oraz wybranych technologii ORM [5]. Badaniom zostały poddane narzędzia JDBC, jOOQ, MyBatis oraz Hibernate. Jako kryteria ich oceny przyjęto czas wykonania zapytań dla operacji odczytu, zapisu, aktualizacji i usuwania rekordów, zużycie pamięci RAM, wykorzystanie procesora, popularność narzędzia, podatność na ataki SQL Injection, wsparcie dla różnych dialektów, weryfikację składni oraz konieczność znajomości języka SQL. Dla każdego z rozpatrywanych kryteriów przypisano odpowiednią wagę. Najwyższą z nich otrzymała szybkość wykonywania zapytań oraz odporność na ataki SQL Injection, natomiast za najmniej istotne uznano znajomość SQL oraz weryfikację składni. Uwzględniając zdefiniowane wagi, najbardziej efektywne okazało się rozwiązanie JDBC, na drugim miejscu znalazła się technologia Hibernate, a następnie MyBatis i jOOQ.

Przeprowadzona analiza literatury przedmiotu wskazuje na brak publikacji dotyczących oceny narzędzi ORM z punktu widzenia analitycznego przetwarzania danych. Z tego powodu niniejszy artykuł i zaprezentowane w nim wyniki badań mogą mieć znaczenie i stanowić pomoc w optymalnym wyborze przeznaczonej dla języka Java technologii ORM wykorzystywanej w analitycznym przetwarzaniu danych.

3. Charakterystyka technologii ORM

Obsługę współpracy obiektowych technologii programistycznych z relacyjną bazą danych zapewnia mapowanie obiektowo-relacyjne. Z jego pomocą dane w tabelach relacyjnych oraz obiekty w aplikacji są wzajemnie konwertowane [6]. Ze względu na liczne różnice między modelem relacyjnym i obiektowym, narzędzia ORM stanowią duże usprawnienie, tworząc dodatkową warstwę abstrakcji, która ułatwia synchronizację tabel bazodanowych i obiektów [7].

Spośród technologii ORM dostępnych dla języka Java, w niniejszym artykule zaprezentowano cztery narzędzia o otwartym kodzie źródłowym, które są stabilnymi i rozbudowanymi rozwiązaniami. Oferują one współpracę z licznymi systemami zarządzania bazami danych, a oprócz podstawowych operacji, dostarczają także dodatkowe funkcjonalności wpływające na poprawę wydajności zarządzania danymi. W związku z tym powinny one umożliwić efektywną realizację złożonych zapytań, niezbędnych podczas analitycznego przetwarzania danych.

3.1. Hibernate

Hibernate jest technologią powszechnie stosowaną dla projektów implementowanych w języku Java. Zapewnia ona możliwość korzystania z implementacji JPA (Java Persistence API), a oprócz wbudowanego w tym standardzie języka JPQL (Java Persistence Query Language) dostarcza także jego własną, rozszerzoną wersję –

HQL (Hibernate Query Language). Wśród zaawansowanych funkcjonalności oferowanych przez Hibernate można wymienić kontrolę współbieżnego dostępu do danych, leniwe ładowanie, liczne strategie pobierania danych oraz ich walidację [8].

3.2. EclipseLink

Technologia EclipseLink powstała w oparciu o narzędzie TopLink stworzone przez firmę Oracle. Implementuje ona standard JPA, dostarczając zaawansowaną obsługę komunikacji z relacyjnymi bazami danych. Nie ogranicza się tylko do funkcjonalności dostępnych w standardzie, ale dostarcza także dodatkowych komponentów, które między innymi pozwalają wykorzystać pamięć podręczną i zaawansowane mapowanie oraz zapewniają prostą współpracę z formatami XML i JSON [9].

3.3. DataNucleus

DataNucleus to technologia oferująca wsparcie podczas pracy z różnymi źródłami danych. Oprócz relacyjnych, nierelacyjnych i grafowych baz danych, zapewnia ona współpracę z arkuszami kalkulacyjnymi oraz plikami w formacie OpenDocument. Jedną z dostarczonych metod komunikacji z bazą danych jest implementacja standardu JPA. Inne funkcjonalności tej technologii obejmują także wykorzystanie interfejsów JDO i Jakarta [10].

3.4. Apache Cayenne

W przeciwieństwie do uprzednio przedstawionych rozwiązań, technologia Apache Cayenne nie implementuje standardu JPA, ale wykorzystuje moduł EJBQLQuery obsługujący ciągi znaków zgodnych ze składnią języka JPQL. W celu komunikacji z relacyjną bazą danych udostępnia także interfejs dostarczający gotowy zestaw zapytań wraz z możliwością definiowania własnych wyrażeń. Wbudowane dodatkowe narzędzie z graficznym interfejsem użytkownika – Cayenne Modeler, ułatwia mapowanie obiektów i umożliwia bezpośrednie generowanie kodu Java na podstawie schematu bazy danych [11].

4. Środowisko badawcze

Dla każdej z wybranych technologii zaimplementowana została aplikacja w języku Java, która umożliwia połączenie z bazą danych oraz realizację zaplanowanych scenariuszy badawczych. Do badań wybrane zostały następujące wersje technologii: Hibernate 6.1.7, Apache Cayenne 4.0, EclipseLink 2.7.8 oraz DataNucleus 6.0.0. Aplikacje zostały przygotowane przy pomocy środowiska programistycznego IntelliJ Idea Community Edition 2022.2.2, z wykorzystaniem OpenJDK w wersji 11 i szkieletu programistycznego Spring Boot 3.0.5. Zastosowanie narzędzia Apache Maven usprawniło proces budowania projektu oraz dołączania zależności.

Do przeprowadzenia badań wykorzystany został system zarządzania bazą danych Microsoft SQL Server 2019 Express oraz zintegrowane środowisko do zarządzania jego komponentami, tj. SQL Server Management Studio v18.12.1. Wybranym źródłem danych jest udostępniona przez Microsoft przykładowa relacyjna baza danych AdventureWorks2019, która reprezentuje fikcyjną firmę produkującą rowery i akcesoria rowerowe. W bazie zgromadzone są między innymi dane na temat produktów, pracowników, klientów oraz zamówień [12]. W badaniach uwzględnione zostały tabele o następującej liczbie rekordów: FactInternetSales – 100000, DimDate – 3652, DimProduct – 606, DimProductSubcategory – 37, DimProductCategory – 4, DimCustomer – 18484, DimSalesTerritory – 11.

Badania zostały przeprowadzone z wykorzystaniem tego samego sprzętu komputerowego, dane techniczne przedstawia (Tabela 1).

Tabela 1: Specyfikacja środowiska testowego

Nazwa	Wartość
Laptop	ASUS VivoBook 15 R520UA
Procesor	Intel Core i3-8130U CPU 2.20GHz
System operacyjny	Windows 10 64 bit
Pamięć RAM	8GB (DDR4, 2400MHz)
Dysk	SSD M.2 256 GB

5. Metodyka badawcza

W celu przygotowania aplikacji dla każdej z badanych technologii niezbędne było przeprowadzenie konfiguracji środowiska i połączenia z bazą danych. W kolejnym etapie zdefiniowano klasy reprezentujące poszczególne encje, a następnie zaimplementowane zostały zapytania odpowiadające zdefiniowanym scenariuszom badawczym. Podczas implementacji zapytań wykorzystano udostępnione przez technologie ORM języki: HQL dla technologii Hibernate, JPQL w przypadku EclipseLink i DataNucleus oraz EJBQL dla Apache Cayenne. Aby umożliwić zebranie wyników badań wydajnościowych, dodana została funkcjonalność umożliwiająca pomiar czasu, w jakim wykonywane jest każde zapytanie (w milisekundach). Po uruchomieniu aplikacji wykonanie zapytania było możliwe za pomocą wywołania odpowiedniej komendy z poziomu wiersza poleceń.

W celu zwiększenia dokładności wyników badań, każde zapytanie dla każdej testowanej technologii zostało wykonane 10 razy, a ostateczny poddany porównaniu wynik stanowił średnią wartość z zebranych pomiarów cząstkowych. Przed każdą próbą niezbędne było wyczyszczenie pamięci podręcznej systemu bazy danych przy pomocy polecenia DBCC DROPCLEANBUFFERS. Przeprowadzanie eksperymentu odbyło się bez połączenia internetowego, a oprócz systemu operacyjnego uruchomione zostały tylko niezbędne usługi, tj. stworzona aplikacja oraz serwer baz danych. Pozwoliło to zmniejszyć obciążenie systemu operacyjnego i zminimalizować liczbę działających w tle procesów, które mogłyby korzystać z zasobów procesora w trakcie dokonywania pomiarów.

Oprócz czasu wykonania poszczególnych zapytań porównana została także częstotliwość występowania wzmianek na temat badanych technologii w literaturze na podstawie liczby wyników w wyszukiwarce Google Scholar, która pozwala przeszukiwać bazy danych za-

wierające publikacje naukowe. Uwzględniono wszystkie dostępne dane wyszukiwania na dzień 15.04.2023. Oceniono także wielkość społeczności korzystającej z wybranych rozwiązań na podstawie liczby pytań zadanych w serwisie Stack Overflow dotyczących poszczególnych technologii. Dane zostały zebrane w dniu 15.04.2023 z uwzględnieniem wszystkich dostępnych zapytań w serwisie.

6. Scenariusze badawcze

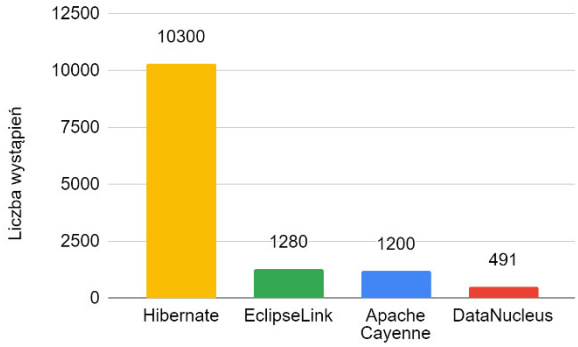
Na potrzeby oceny efektywności analitycznego przetwarzania danych z wykorzystaniem wybranych technologii ORM przygotowane zostały następujące scenariusze badawcze:

1. Wyznaczenie całkowitej wartości sprzedaży dla poszczególnych produktów (S1).
2. Wyznaczenie całkowitej wartości poszczególnych zamówień oraz liczby pozycji dla każdego zamówienia w poszczególnych miesiącach kolejnych lat (S2).
3. Wyznaczenie całkowitej wartości zamówień, liczby zamówień oraz maksymalnej i minimalnej ceny produktów w obrębie danego zamówienia w poszczególnych miesiącach kolejnych lat (S3).
4. Wyznaczenie całkowitej wartości sprzedaży dla poszczególnych podkategorii i kategorii produktów wraz z podsumowaniem całkowitej wartości sprzedaży dla każdej podkategorii oraz łączną wartością sprzedaży dla wszystkich produktów (S4).
5. Wyznaczenie produktów, które nie pojawiły się w żadnych zamówieniach (S5).
6. Wyznaczenie 5 krajów o największej całkowitej wartości sprzedaży (S6).
7. Wyznaczenie liczby klientów, którzy złożyli zamówienia w grudniu poszczególnych lat (S7).
8. Wyznaczenie liczby klientów oraz całkowitej wartości zamówień złożonych między określonymi datami w Australii i Stanach Zjednoczonych (S8).
9. Wyznaczenie liczby klientów, którzy w grudniu poszczególnych lat w Stanach Zjednoczonych złożyli zamówienie zawierające produkt o nazwie HL Mountain Tire (S9).
10. Wyznaczenie produktów sprzedanych w 2018 roku, których całkowita wartość sprzedaży wyniosła więcej niż 10000 (S10).
11. Wyznaczenie danych 3 klientów, których średnia wartość zamówienia jest największa (S11).
12. Wyznaczenie procentowego udziału wartości sprzedaży z poszczególnych krajów w całkowitej wartości sprzedaży (S12).
13. Wyznaczenie procentowego udziału wartości miesięcznej sprzedaży w sprzedaży rocznej dla poszczególnych regionów (S13).
14. Wyznaczenie krajów w poszczególnych regionach, dla których całkowita wartość sprzedaży jest większa od średniej wartości sprzedaży krajowej w danym regionie (S14).
15. Wskazanie dla każdego miesiąca kolejnych lat produktu, którego wartość sprzedaży jest największa (S15).
16. Wyznaczenie liczby klientów i całkowitej wartości sprzedaży w kolejnych latach dla poszczególnych rodzajów rowerów (S16).
17. Wyznaczenie danych klientów, których łączna wartość zamówień jest większa niż całkowita wartość sprzedaży dla kraju Canada w grudniu 2018 roku (S17).
18. Wyznaczenie produktów, na które zostały złożone zamówienia zarówno w regionie Southwest, jak i Australia (S18).
19. Wyznaczenie produktu, który został sprzedany w największej liczbie sztuk w poszczególnych latach dla poszczególnych krajów (S19).

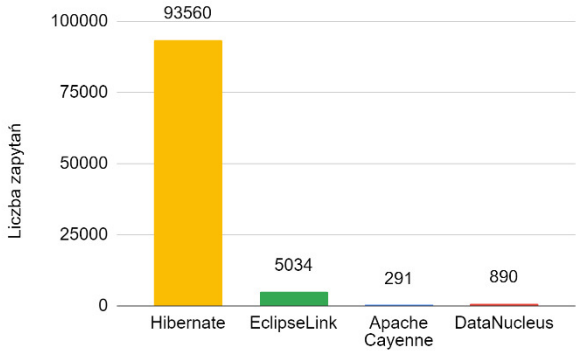
7. Wyniki badań

W niniejszej części artykułu przedstawiono wyniki przeprowadzonych badań dla uprzednio opisanych scenariuszy.

Rysunek 1 ilustruje liczbę wystąpień fraz Hibernate ORM, EclipseLink, Apache Cayenne oraz DataNucleus w wyszukiwarce Google Scholar. Technologia Hibernate pod względem liczby wystąpień w literaturze posiada znaczącą przewagę w porównaniu do pozostałych narzędzi.



Rysunek 1: Częstotliwość występowania wzmianek w literaturze.

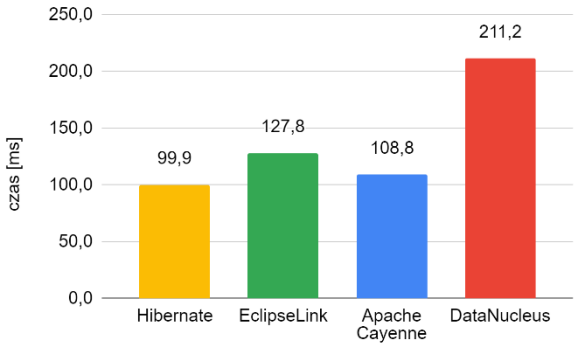


Rysunek 2: Liczba zapytań w serwisie Stack Overflow.

Największa liczba zapytań w serwisie Stack Overflow dotyczy technologii Hibernate, następnie EclipseLink i DataNucleus, a najmniej pytań pojawiło się dla Apache Cayenne, co zostało zobrazowane na Rysunku 2.

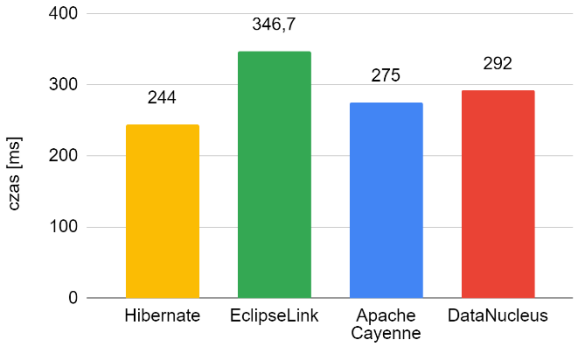
Wyniki badań dla scenariusza S1 przedstawiono na Rysunku 3. Najkrótszy średni czas wykonania zapytania uzyskano w technologii Hibernate. Narzędzie Apache

Cayenne umożliwiło realizację scenariusza w nieznacznie dłuższym czasie (9%). Trzecia w kolejności znalazła się technologia EclipseLink (28% dłuższy czas), natomiast DataNucleus jest narzędziem, dla którego średni czas wykonania scenariusza S1 okazał się ponad dwukrotnie dłuższy w porównaniu do Hibernate.



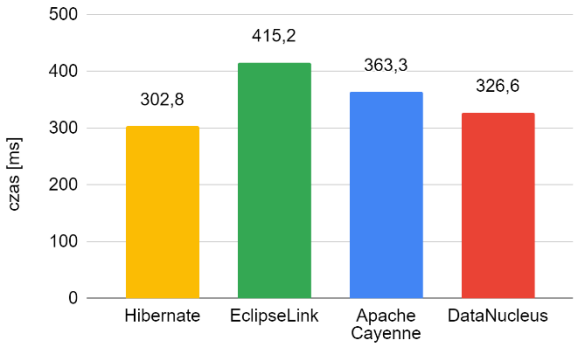
Rysunek 3: Średni czas wykonania scenariusza S1.

Rysunek 4 przedstawia wyniki badań dla scenariusza S2. Wykonanie zapytania w najkrótszym czasie było możliwe z wykorzystaniem technologii Hibernate. Kolejne miejsca w rankingu efektywności zajęły technologie Apache Cayenne oraz DataNucleus. Scenariusz S2 był realizowany najdłużej w technologii EclipseLink.



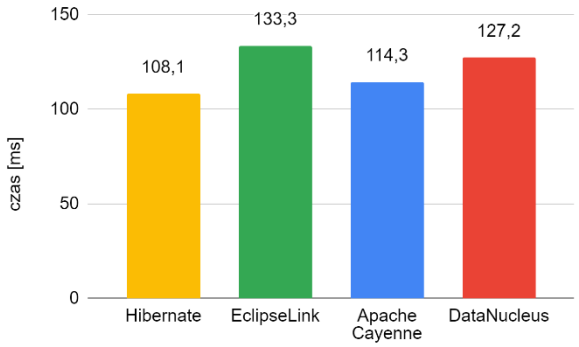
Rysunek 4: Średni czas wykonania scenariusza S2.

Rysunek 5 obrazuje średni czas realizacji scenariusza S3 dla wszystkich środowisk. Najlepsza z punktu widzenia efektywności okazała się technologia Hibernate, następnie DataNucleus, Apache Cayenne i ostatecznie EclipseLink.



Rysunek 5: Średni czas wykonania scenariusza S3.

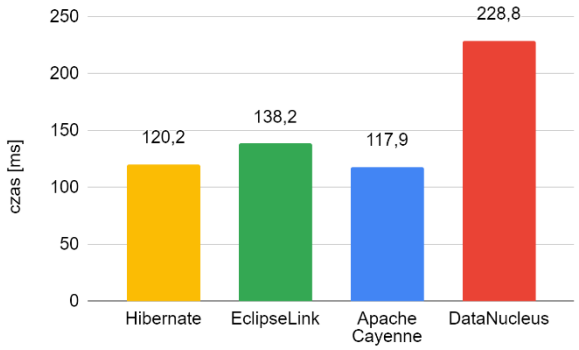
Scenariusz S4 został zrealizowany z wykorzystaniem natywnego zapytania SQL, ponieważ we wszystkich omawianych technologiach ORM udostępnione obiektowe sposoby definiowania zapytań nie pozwalają na zastosowanie klauzuli ROLLUP. Wyniki tego eksperymentu przedstawiono na Rysunku 6.



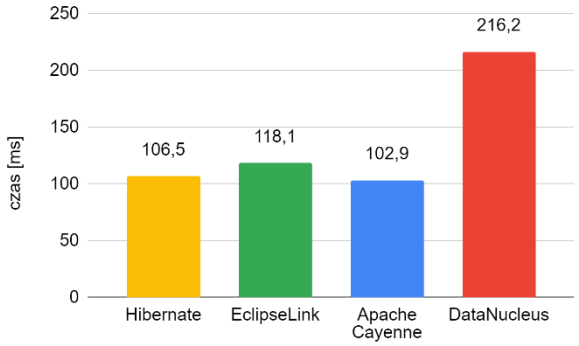
Rysunek 6: Średni czas wykonania scenariusza S4.

W przypadku natywnego zapytania SQL ponownie technologia Hibernate zapewniła najkrótszy czas jego wykonania, jakkolwiek jej przewaga nad pozostałymi nie jest już tak znacząca jak w dotychczas zaprezentowanych scenariuszach.

Rysunki 7 i 8 prezentują wyniki realizacji scenariuszy S5 i S6. W tych przypadkach najkrótszy średni czas wykonania zapytania zapewniła technologia Apache Cayenne. Nieznacznie ustępują jej narzędzia Hibernate oraz EclipseLink. Najgorszy wynik uzyskała technologia DataNucleus.

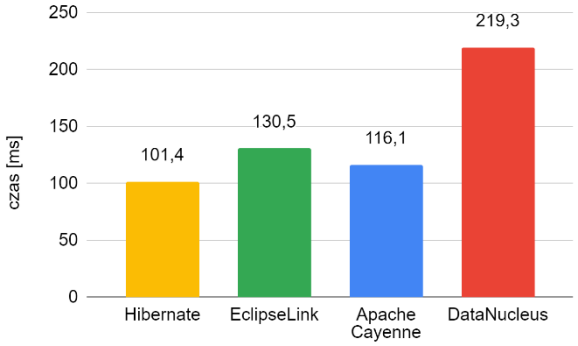


Rysunek 7: Średni czas wykonania scenariusza S5.

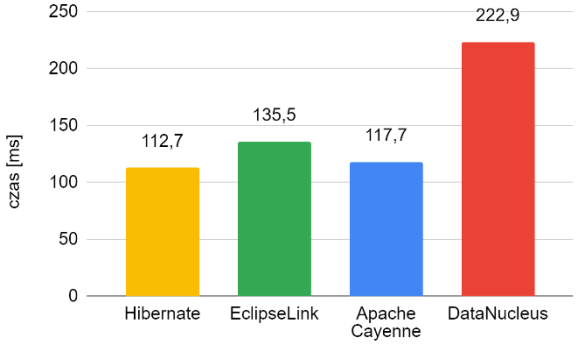


Rysunek 8: Średni czas wykonania scenariusza S6.

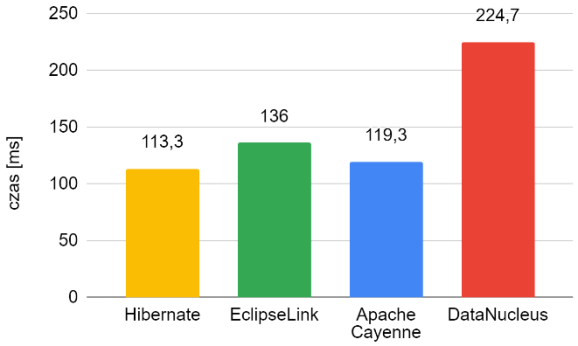
Wizualizacje Rysunek 9 do Rysunek 15 prezentują wyniki badań dla scenariuszy S7, S8, S9, S10, S16, S17 i S18. We wszystkich wymienionych przypadkach najbardziej wydajna okazała się technologia Hibernate. Bardzo dobre wyniki analitycznego przetwarzania danych uzyskano również przy użyciu technologii Apache Cayenne. Na trzeciej pozycji w tym rankingu znalazła się technologia EclipseLink. Natomiast przetwarzanie danych w narzędziu DataNucleus wymagało najdłuższego czasu, niemal dwukrotnie dłuższego niż w technologii Hibernate.



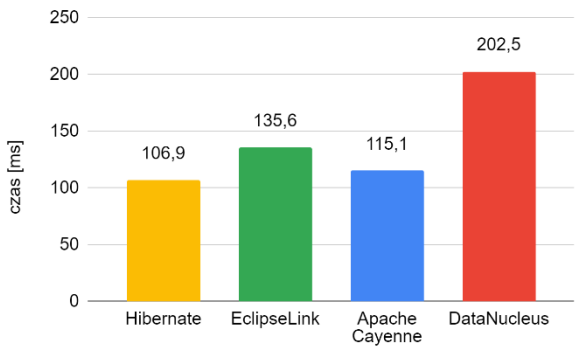
Rysunek 9: Średni czas wykonania scenariusza S7.



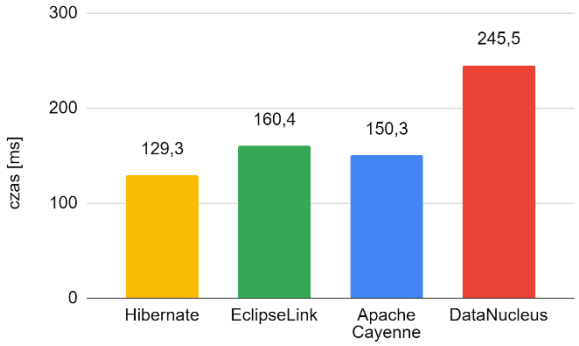
Rysunek 10: Średni czas wykonania scenariusza S8.



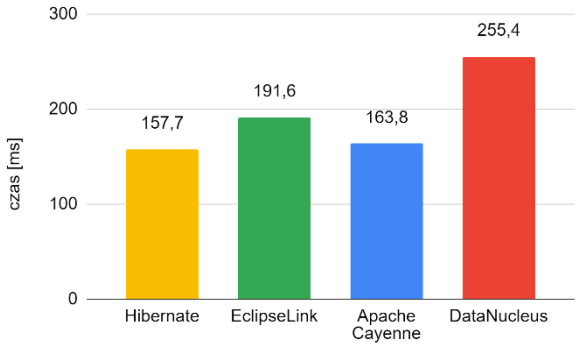
Rysunek 11: Średni czas wykonania scenariusza S9.



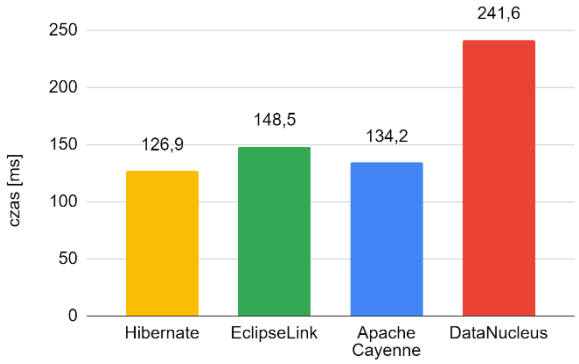
Rysunek 12: Średni czas wykonania scenariusza S10.



Rysunek 13: Średni czas wykonania scenariusza S16.



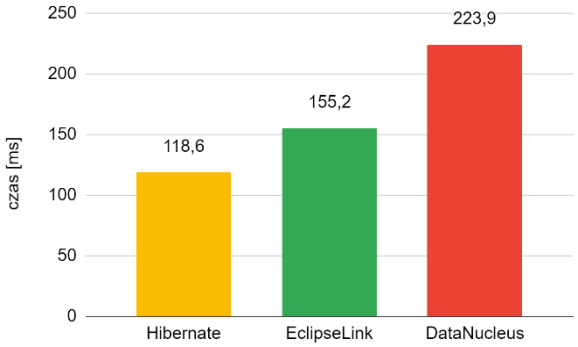
Rysunek 14: Średni czas wykonania scenariusza S17.



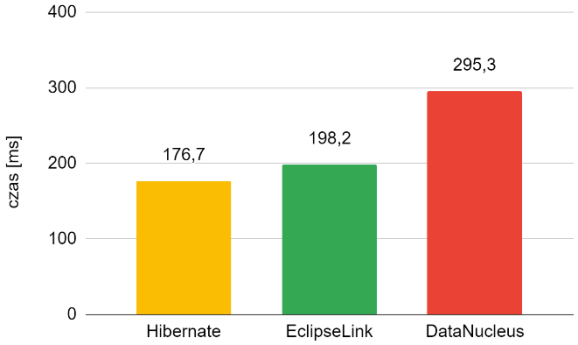
Rysunek 15: Średni czas wykonania scenariusza S18.

Na Rysunkach 16 i 17 zostały zaprezentowane wyniki realizacji scenariuszy S12 i S13 dla wszystkich omawianych technologii z wyjątkiem Apache Cayenne. W przypadku wykorzystanej wersji tego narzędzia

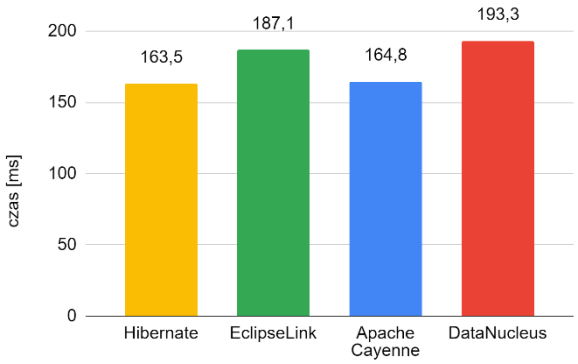
i języka EJBQL nie jest możliwe zastosowanie operacji dzielenia w klauzuli SELECT. Obydwa scenariusze zostały wykonane najszybciej z użyciem technologii Hibernate, a następnie EclipseLink i DataNucleus.



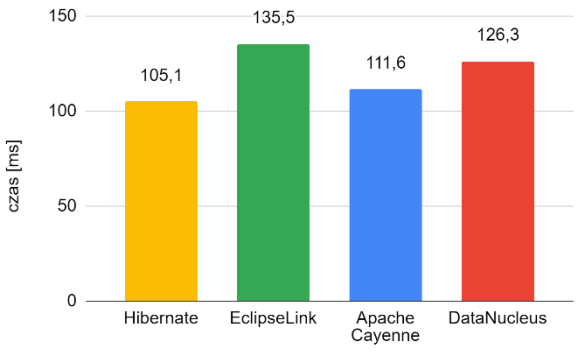
Rysunek 16: Średni czas wykonania scenariusza S12.



Rysunek 17: Średni czas wykonania scenariusza S13.



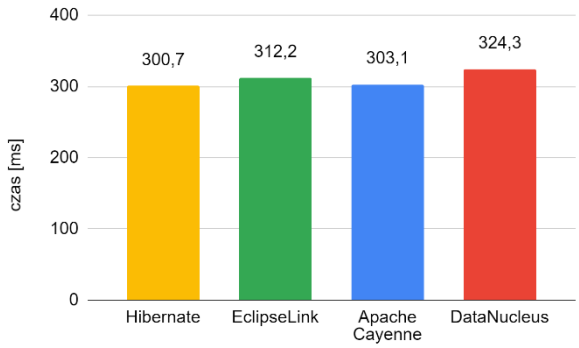
Rysunek 18: Średni czas wykonania scenariusza S11.



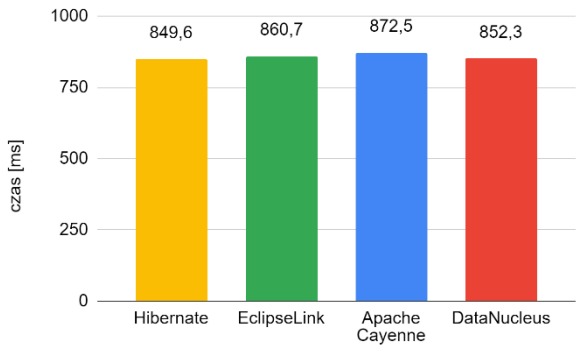
Rysunek 19: Średni czas wykonania scenariusza S14.

Scenariusze S11 i S14 wymagają zastosowania podzapytania w klauzuli FROM. Wykorzystane do implementacji zapytań języki JPQL i EJBQL nie obsługują tego typu operacji. Z tego powodu scenariusze te zostały zrealizowane z wykorzystaniem zapytania SQL, a wyniki tych badań przedstawiono na Rysunkach 18 i 19. Technologia Hibernate umożliwiła realizację wspomnianych scenariuszy w języku HQL z wykorzystaniem mapowania obiektowo-relacyjnego. W tym przypadku średni czas wyniósł 170,2 ms dla scenariusza S11 i 107,9 ms dla scenariusza S14.

W celu wykonania scenariuszy S15 i S19 konieczne jest zastosowanie klauzuli TOP w podzapytaniu, co okazało się niemożliwe w językach JPQL i EJBQL. Zapytania zostały zdefiniowane z zastosowaniem języka SQL, a średni czas ich wykonania zaprezentowano na Rysunkach 20 i 21.



Rysunek 20: Średni czas wykonania scenariusza S15.



Rysunek 21: Średni czas wykonania scenariusza S19.

Technologia Hibernate pozwala również na zastosowanie mapowania obiektowo-relacyjnego do realizacji tych scenariuszy. Średni czas wykonania scenariusza S15 wyniósł 357,4 ms, a scenariusza S19 - 858,5 ms.

8. Wnioski

Przeprowadzone badania oraz analiza ich wyników pozwoliły ocenić efektywność analitycznego przetwarzania danych z wykorzystaniem wybranych technologii ORM przeznaczonych dla języka Java.

Na podstawie wyników badań wykonanych z użyciem scenariuszy S4, S11, S12, S13, S14, S15 i S19 można zaobserwować pewne ograniczenia technologii ORM, które w udostępnianych zorientowanych obiektowo metodach nie zapewniają wszystkich elementów

składni języka SQL występujących w jego standardzie. Rozwiązaniem tego problemu jest wykonanie operacji przetwarzania danych z użyciem natywnych zapytań SQL. Jednak takie podejście eliminuje możliwość korzystania z zalet technologii ORM, takich jak bezpośrednie operowanie na obiektach, wygodna konwersja między obiektami i tabelami w bazie danych oraz przenośność rozwiązania na inne systemy zarządzania bazą danych.

Dla większości scenariuszy badawczych (17 na 19) najkrótszy średni czas wykonania zapytań uzyskano w technologii Hibernate. Tylko w dwóch przypadkach jej wydajność była nieznacznie niższa w porównaniu do najlepszego wyniku. Narzędzie Hibernate umożliwiło także realizację największej liczby scenariuszy badawczych (18 na 19) przy pomocy zorientowanego obiektowo języka HQL. Świadczy to o wysokim stopniu zaawansowania technologii i możliwości realizacji nawet bardzo złożonych zapytań. Technologia ta sprawdziła się również najlepiej w przypadku scenariuszy wykonanych przy pomocy języka SQL. Warto również zauważyć, że w przypadku porównania czasu wykonania zapytań przy użyciu języka HQL, średni czas był niewiele dłuższy (średnio o ok. 7%) od uzyskanego podczas korzystania z języka SQL. Fakt ten stanowi dowód na wysoką wydajność mapowania obiektowo-relacyjnego zaimplementowanego w tej technologii. Jego użycie nie powoduje dużego spadku efektywności przetwarzania danych zdefiniowanego w natywnym języku SQL. W porównaniu do pozostałych technologii, Hibernate charakteryzuje się największą społecznością korzystającą z technologii i największą liczbą wystąpień wzmianek w literaturze przedmiotu. Posiada również rozbudowaną i aktualną dokumentację, a proces konfiguracji czy budowania zapytań jest najszybszy i najmniej skomplikowany. Wymienione powyżej zalety w istotny sposób wpływają na wybór tej technologii ORM przez większość twórców aplikacji webowych, zapewniając w ten sposób skrócenie czasu ich produkcji oraz efektywne ich wykorzystanie w docelowym środowisku klienta końcowego.

Technologia Apache Cayenne w 13 na 19 przypadków zapewniła uzyskanie zbliżonych średnich czasów realizacji scenariuszy do tych, jakie zmierzono w środowisku Hibernate. Z tego powodu może ona stanowić alternatywę dla rozwiązań bazujących na implementacji standardu JPA. Jednak znacznymi wadami tego rozwiązania są ograniczenia związane z definiowaniem złożonych zapytań, niewielka liczba materiałów dotyczących danej technologii oraz bardzo mała społeczność. Również korzystanie z tego narzędzia, w porównaniu do technologii Hibernate, nie było intuicyjne, wymagało umiejętności konfiguracji mapowania obiektów i relacji w języku xml, a czas definiowania zapytań był znacznie dłuższy.

Dla większości scenariuszy badawczych (14 na 19) technologia DataNucleus okazała się zdecydowanie najmniej wydajna. Niewielka społeczność oraz niska dostępność dodatkowych materiałów dotyczących tej technologii negatywnie wpływa na niską skłonność wyboru tego narzędzia do analitycznego przetwarzania danych.

Z kolei technologia EclipseLink nie wyróżniła się znacząco pod względem wydajnej realizacji zdefiniowanych scenariuszy. Zaletą tego rozwiązania jest rozbudowana, poparta wieloma przykładami dokumentacja oraz większa społeczność w porównaniu do Apache Cayenne i DataNucleus, dzięki czemu podczas korzystania z EclipseLink zapewnione jest większe wsparcie.

Na podstawie otrzymanych wyników badań własnych i powyższych wniosków można stwierdzić, że technologia Hibernate charakteryzuje się największą wydajnością i jest najbardziej efektywnym rozwiązaniem ORM w przypadku analitycznego przetwarzania danych.

Literatura

[1] S. N. Bhatti, Z. H. Abro, F. Rufabro, Performance evaluation of java based object relational mapping tools, Mehran University Research Journal of Engineering and Technology 32(2) (2013) 159-166.

[2] N. Dhingra, E. Abdelmoghith, H. T. Mouftah, Performance Evaluation of JPA Based ORM Techniques, 2nd International Conference on Computer Science Networks and Information Technology (2016) 15-23.

[3] B. Pllana, Performance Analysis of Java Persistence API Providers, UBT International Conference (2018) 100-107.

[4] M. Połeć, J. Pitera, G. Kozieł, Comparing the Performance of the Object-Relational Mapping Programming Frameworks Available in Java, Journal of Computer Sciences Institute 22 (2022) 59-65.

[5] M. Żuchnik, P. Kopniak, Comparative analysis of connection performance with databases via JDBC interface and ORM programming frameworks, Journal of Computer Sciences Institute 21 (2021) 309-315.

[6] Z. Rosiek, Mapowanie obiektowo-relacyjne ORM-czy tylko dobra idea, Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki 4(4) (2010) 99-112.

[7] R. G. Sfirlogea, A Decision Support Model for using an Object-Relational Mapping Tool in the Data Management Component of a Software Platform, University of Utrecht Master's thesis, 2015.

[8] Dokumentacja techniczna technologii Hibernate, https://hibernate.org/orm, [10.04.2023].

[9] Dokumentacja techniczna technologii EclipseLink, https://wiki.eclipse.org/EclipseLink, [10.04.2023].

[10] Dokumentacja techniczna technologii DataNucleus, https://www.datanucleus.org, [10.04.2023].

[11] Dokumentacja techniczna technologii Apache Cayenne, https://cayenne.apache.org, [10.04.2023].

[12] M. Mitri, Teaching Tip: Active Learning via a Sample Database: The Case of Microsoft's Adventure Works, Journal of Information Systems Education 26(3) (2015) 177-186.