JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 22 (2022) 59–65
Received: 4 November 2021
Accepted: 25 November 2021

Comparing the performance of the object-relational mapping programming frameworks available in Java

Porównanie wydajności szkieletów programistycznych mapowania obiektowo-relacyjnego dostępnych w języku Java

Jakub Benedykt Pitera* , Mateusz Połeć*, Grzegorz Kozieł

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

Abstract

The paper concerns a performance evaluation of selected object-relational mapping tools. This work is intended to assist software architects in determining which of the analyzed libraries will be the optimal choice for use in a specific project. The work includes the comparison of Hibernate ORM, EclipseLink, Apache OpenJPA and DataNucleus libraries from the theoretical and practical point of view. Each of the examined tools has been described according to criteria allowing to distinguish the most important features influencing communication with relational databases. These features will then be compared on a practical level by examining the behavior of the test applications. In terms of performance, the Apache OpenJPA library turned out to be the best, but in terms of configuration and availability it significantly differs from other libraries. This was caused by an unintuitive configuration and poor documentation of the technology. In this respect, the Hibernate library definitely dominated thanks to full compatibility with the Spring programming framework, intuitive configuration as well as rich documentation and support by the extensive community of programmers using it.

Keywords: Java ORM; Java Persistence API; performance evaluation

Streszczenie

Artykuł poświęcono wykonaniu analizy porównawczej wybranych narzędzi mapowania obiektowo-relacyjnego. Jego celem jest pomoc architektom oprogramowania w określeniu, która z analizowanych bibliotek będzie optymalnym wyborem do użycia w określonym projekcie. Celem artykułu jest ocena bibliotek Hibernate ORM, EclipseLink, Apache OpenJPA oraz DataNucleus pod kątem teoretycznym oraz praktycznym. Każde z badanych narzędzi opisane zostało według kryteriów pozwalających na wyodrębnienie najważniejszych cech mających wpływ na komunikację z relacyjnymi bazami danych. Cechy te następnie zostały porównane na poziomie praktycznym poprzez zbadanie zachowania aplikacji testowych. Pod względem wydajnościowym, najlepsza okazała się biblioteka Apache OpenJPA, jednak pod względem konfiguracji i dostępności znacznie odstępuje od innych bibliotek. Spowodowane było to nieintuicyjną konfiguracją oraz ubogą dokumentacją technologii. Pod tym względem zdecydowanie górowała biblioteka Hibernate dzięki pełnej kompatybilności ze szkieletem programistycznym Spring, intuicyjnej konfiguracji oraz bogatej dokumentacji i wsparcia przez obszerną społeczność korzystających z niej programistów.

Słowa kluczowe: Java ORM; Java Persistence API; ocena wydajności

*Corresponding author

Email address: jakub.pitera@pollub.edu.pl (J. B. Pitera), mateusz.polec1@pollub.edu.pl (M. Połeć)

1. Wstęp

W dzisiejszych czasach możliwość gromadzenia danych przez systemy informatyczne jest rzeczą powszechnie spotykaną, niezależnie czy mówimy tu o prostych stronach-wizytówkach czy też o skomplikowanych systemach zarządzających całymi korporacjami – wszystkie wykorzystują relacyjne bazy danych. Zdecydowana większość projektowanych aplikacji tworzona jest za pomocą programowania obiektowego [1]. Jednoczesne wykorzystywanie relacyjnych baz danych i podejścia obiektowego w trakcie tworzenia systemów informatycznych spowodowało utworzenie nowej, abstrakcyjnej warstwy, umożliwiającej przedstawianie tabel bazy danych w postaci klas i obiektów w kodzie źródłowym aplikacji. Mechanizm ten został określony mapowaniem obiektowo-relacyjnym (ORM – ang. *object-relational mapping*) [2].

Język programowania Java posiada całą paletę dostępnych bibliotek ORM [3]. Mnogość dostępnych rozwiązań stawia jedno pytanie – której z bibliotek użyć do zrealizowania projektu? Na rynku dostępnych jest wiele rozwiązań: Hibernate ORM, czyli najpopularniejsza biblioteka ORM, która od 2001 jest pionierem stosowanych dzisiaj rozwiązań, przyczyniając się do istotnych zmian w specyfikacji EJB (ang. Enterprise JavaBeans) oraz JPA (ang. Java Persistence API), doprowadzając je do stanu, w którym są obecnie znane; EclipseLink, główna konkurencja technologii Hibernate, wspierająca standardy JPA, JAXB (ang. Java Architecture for XML Binding), JCA (Java Connector Architecture) i SDO (Service Data Objects); biblioteka Apache OpenJPA, która może być używana jako samodzielna warstwa trwałości POJO (ang. Plain Old Java Object) lub zintegrowana z dowolnymi platformami zgodnymi z Java EE (ang. Enterprise Edition); oraz technologia DataNucleus zapewniająca dostęp do rozproszonych systemów baz danych takich jak HBase i Cassandra, graficznych systemów baz danych jak Neo4j, arkuszów kalkulacyjnych w formatach Excel i OpenDocument, danych zapisywa-

nych w formacie JSON dla Amazon i Google Storage czy baz danych w formacie JSON pokroju MongoDB [4]. Biblioteki te zostały wybrane ze względu na popularność, badaną poprzez liczbę powiązanych tematów w wyszukiwarkach Google i Google Scholar oraz serwisów Github i Stack Overflow. Uzyskane wyniki obrazują częstotliwość wykorzystywania badanych technologii, co przekłada się na ich popularność i łatwość w dostępie do informacji oraz wsparcia technicznego.

2. Analiza literatury

Niniejszy rozdział zawiera przegląd renomowanych i wiarygodnych źródeł, w oparciu, o które powstała treść merytoryczna artykułu. Zebrane publikacje zawierają informacje dotyczące dokładnego opisu badanych technologii wraz ze sposobami oraz wynikami ich porównania. Pozwoliły one na zebranie niezbędnej wiedzy wykorzystanej podczas przeprowadzanych badań.

Artykuł „Object-Relational Mapping as a Persistence Mechanism for Object-Oriented Applications" [1] opisuje proces tworzenia aplikacji obiektowych poprzez zwrócenie uwagi na jego kluczowy element – mapowanie obiektowo-relacyjne. Publikacja ta zawiera kompleksowy opis całego mechanizmu i zwraca uwagę na najczęściej pojawiające się problemy, na które każdy programista powinien zwrócić szczególną uwagę. Artykuł ten pozwolił określić jakie elementy powinny znaleźć się w pracy dotyczącej badania bibliotek mapowania obiektowo-relacyjnego, co przełożyło się na sposób organizacji i wykonania części teoretycznej i praktycznej niniejszej pracy.

Publikacja naukowa pod tytułem „Performance Evaluation of Java Based Object Relational Mapping Tool" [2] porusza tematykę badania wydajności wykonywania zapytań wykorzystując szkielety programistyczne mapowania obiektowo-relacyjnego dla języka programowania Java. Autor publikacji poddał badaniom trzy najpopularniejsze narzędzia ORM takie jak Hibernate, Ebean oraz Toplink. Badanie polegało na wykonaniu szeregu podzapytań do bazy danych MySQL oraz zmierzeniu czasu ich wykonania. Badania dla wszystkich szkieletów programistycznych zostały wykonane na tej samej platformie. Informacje dotyczące sposobu przeprowadzenia badań oraz wykorzystania bazy danych MySQL stanowiły wzór testów przeprowadzonych w niniejszej pracy.

Następna pozycja literaturowa to artykuł naukowy pod tytułem „Analiza porównawcza technologii odwzorowania obiektowo-relacyjnego dla aplikacji Java" [3]. Celem wyżej wymienionego artykułu jest porównanie cech, możliwości oraz wydajności technologii mapowania obiektowo-relacyjnego takich jak Hibernate i Oracle TopLink. Wszystkie testy wydajnościowe zostały przeprowadzone na wyszczególnionej platformie, a wykorzystana baza danych to Oracle 10g. Każdy pomiar powtarzany był czterokrotnie, a końcowy wynik został uśredniony. Na potrzeby testów baza danych została wypełniona przykładowymi danymi. Zostało wyszczególnione dziesięć operacji na bazie danych, które modyfikują, zapisują, odczytują oraz usuwają dane. Szczegółowe opisy przeprowadzonych testów zawierają informacje pozwalające na wydajny i efektywny sposób przeprowadzenia badań wydajności bibliotek ORM.

Publikacja o tytule „Object-Relational Mapping Tools and Hibernate" [4] zajmuje się tematyką narzędzi mapowania obiektowo-relacyjnego, głównie skupiając się na technologii Hibernate. Kompleksowo opisuje sposoby zastosowania opisywanych narzędzi, punktuje ich zalety oraz wady. Szkielet programistyczny Hibernate opisuje jako jedno z najważniejszych narzędzi tego typu. Wywiera on również ogromny wpływ na paradygmaty oraz konwencje JPA. Ponieważ istnieje od wielu lat, Hibernate jest według autorów najlepszym wyborem dla narzędzia ORM do użycia z językiem programowania Java dostępnym na rynku. Kompleksowe informacje zawarte w artykule stanowią wzór dla teoretycznego opisu badanych bibliotek i dostarczają bogaty zasób wiedzy dotyczący technologii Hibernate.

„Analysis of ORM Based JPA Implementations" [5] obejmuje przegląd najczęściej używanych dostawców JPA, w szczególności szkieletów programistycznych zapewniających obsługę JPA, takich jak Hibernate, EclipseLink, OpenJPA i DataNucleus. Analiza wydajności powyższych czterech implementacji JPA jest oparta na strukturze ORM, która w największym stopniu przyczyniła się do odkrycia wyzwań i zweryfikowania zagadnień programowania w języku Java. Publikacja doskonale wpasowuje się w tematykę niniejszej pracy, ze względu na informacje dotyczące badanych przez nią bibliotek.

3. Badane biblioteki mapowania obiektowo-relacyjnego

Technologie ORM dostarczane są jako samodzielne biblioteki, możliwe do wykorzystania w każdym projekcie. Narzędzia mapowania obiektowo-relacyjnego stały się niezbędnym elementem szkieletów programistycznych dla języka Java [5].

W tym rozdziale przedstawione zostaną cztery biblioteki ORM, które wykorzystywane są w języku programowania Java.

3.1. Hibernate ORM

Hibernate to biblioteka utworzona w 2001 roku przez przedsiębiorstwo Red Hat, oparta na licencji GNU Lesser General Public License (LGPL). Hibernate odwzorowuje klasy języka programowania Java na tabele bazy danych oraz typy danych Java na typy danych SQL. Warstwa biblioteki znajduje się pomiędzy kodem źródłowym aplikacji, a serwerem bazy danych, dzięki czemu obsługuje wszystkie prace związane z utrwalaniem danych w oparciu o odpowiednie mechanizmy i wzorce ORM [6].

Hasło „Hibernate ORM" zostało odnalezione w 1 790 000 wynikach wyszukiwarki Google. Google Scholar zawiera 8 980 publikacji związanych z tą technologią. Serwis Github przechowuje 64 987 repozytoria powiązane z biblioteką Hibernate ORM. Na forum

Stack Overflow zadano 88 211 pytania dotyczące tej biblioteki.

3.2. EclipseLink

EclipseLink to oparty o licencję Eclipse Public License projekt o otwartym kodzie źródłowym, opracowany przez firmę Eclipse Foundation. Biblioteka ta zapewnia rozszerzalną strukturę, która pozwala na interakcję z takimi usługami danych, jak bazy danych, usługi sieciowe, mapowanie Object XML (OXM) czy systemy informacji korporacyjnej (EIS). EclipseLink jest oparty na bibliotece TopLink [7].

Hasło „EclipseLink" w wyszukiwarce Google dało 503 000 wyników. W Google Scholar pojawiło się 1 160 publikacji związanych z tą technologią. Serwis Github zawiera 553 repozytoria powiązane z badaną technologią. Na forum Stack Overflow zadano 4914 pytań dotyczących biblioteki EclipseLink.

3.3. Apache OpenJpa

Apache OpenJPA to biblioteka o otwartym kodzie źródłowym, oparta na licencji Apache License 2.0. Narzędzie to opiera się na projekcie Kodo, który pierwotnie opracowany został przez firmę Solar Metric Inc w 2001 roku. Firma ta została następnie przejęta przez BEA Systems, gdzie biblioteka Kodo został rozszerzona, aby móc implementować specyfikację JDO (Java Data Object) oraz JPA (Jakarta Persistence). W roku 2006 BEA Systems przekazał większą część kodu narzędzia Kodo firmie Apache Software Foundation pod nazwą OpenJPA. W maju 2007 r. OpenJPA przeszło z inkubatora do projektu najwyższego poziomu [8].

Hasło „Apache OpenJPA" zostało odnalezione w 103 000 wynikach wyszukiwarki Google. Google Scholar zawiera 6 110 publikacji związanych z tą technologią. Serwis Github przechowuje 124 repozytoria powiązane z biblioteką Apache OpenJPA. Na forum Stack Overflow zadano 1082 pytania dotyczące tej biblioteki.

3.4. DataNucleus

DataNucleus (wcześniej znany jako Java Persistent Objects JPOX) to projekt na licencji Apache 2.0, o otwartym kodzie źródłowym. Projekt JPOX rozpoczął się w 2003 r. i został ponownie uruchomiony jako DataNucleus w 2008 r., oferując przy tym o wiele większy zakres obsługiwanych magazynów danych [9].

Wyszukanie hasła „DataNucleus" dało 91 100 wyników w wyszukiwarce Google i 701 publikacji w usłudze Google Schoolar. Biblioteka DataNucleus jest częścią 146 repozytoriów w serwisie Github. Na forum Stack Overflow zadano 891 pytań z wiązanych z tą technologią.

4. Implementacja aplikacji testowych

Podczas implementacji aplikacji testowych wykorzystano system zarządzania relacyjnymi bazami danych MySQL 8.0 [10]. Baza danych zawiera 14 tabel powiązanych relacjami. W bazie danych występują relacje jeden do jednego, jeden do wielu oraz wiele do wielu z tabelami łączącymi. Tabele zawierają dane w postaci tekstowej, liczbowej oraz pliki w formacie BLOB.

Aplikacja testowa została zaimplementowana w języku programowania Java 11 [11] przy wykorzystaniu narzędzia automatyzującego budowanie projektu Maven 3.6.3 oraz szkielet programistyczny Spring Boot 2.5.4 [12]. Dla każdej z badanych technologii został utworzony moduł zawierający zależności do bibliotek badanych szkieletów programistycznych mapowania obiektowo-relacyjnego. Moduły zostały zintegrowane z projektem Spring Boot Starter Data JPA. Dla każdej aplikacji zdefiniowano obiekty domenowe odzwierciedlające tabele z bazy danych. Utworzono 14 klas zawierających adnotację @Table, @Entity oraz adnotacje umożliwiające tworzenie relacji pomiędzy encjami. Dla relacji wiele do wielu wykorzystano adnotację @ManyToMany oraz @JoinTable, które pozwalają na utworzenie dodatkowej tabeli łączącej zrelacjonowane encje. Zaimplementowane aplikacje wykorzystują pojedynczy wątek oraz zachowują transakcyjność wykonywanych operacji.

Implementacja każdej z bibliotek pozwoliła na analizę poziomu skomplikowania integracji badanych technologii ze szkieletem programistycznym Spring.

Biblioteka Hibernate w bardzo prosty sposób integruje się ze technologią Spring. Bogata dokumentacja pozwoliła znaleźć odpowiedź na każdy pojawiający się problem, a intuicyjność kodu przyspieszyła proces tworzenia modułu przeznaczonego dla tej technologii.

Szkielet programistyczny Spring wspiera integrację z biblioteką EclipseLink, przez co konfiguracja tej technologii jest względnie prosta. Warto natomiast zwrócić uwagę na nieaktualną oraz niekompletną dokumentację. Podczas implementacji, na stronie producenta nie znaleziono przykładów oraz wymagań stawianych przez technologię EclipseLink podczas integracji ze szkieletem programistycznym Spring. Głównymi źródłami wiedzy, wykorzystanymi podczas konfigurowania biblioteki oraz projektowania w niej operacji bazodanowych, były niezależne serwisy dostępne w Internecie oraz dokumentacja szkieletu Spring.

Integracja ze szkieletem programistycznym Spring oraz samo wykorzystanie biblioteki DataNucleus okazało się problematyczne. Głównym problemem była uboga dokumentacja oraz brak przykładów dostępnych w Internecie. Podczas konfiguracji oraz tworzenia zapytań wymagana była analiza bibliotek technologii DataNucleus oraz dostępnych klas implementujących standard JPA.

Również w przypadku technologii OpenJPA występuje trudność ze znalezieniem aktualnej dokumentacji pozwalającej na sprawną integrację środowisk. Dodatkowa utrata wsparcia ze strony Spring znacząco utrudnia implementację.

5. Metodyka badań

Dla każdej z testowanych technologii zaprojektowano aplikacje w języku Java z wykorzystaniem szkieletu programistycznego Spring Boot. Każda z utworzonych

aplikacji miała na celu wykonywanie operacji na tej samej bazie danych z wykorzystaniem ocenianych bibliotek. Aplikacje są do siebie jak najbardziej zbliżone i oddzielne dla każdej z technologii. W ramach porównania przygotowano zestaw operacji odpowiedzialnych za odczyt, tworzenie, aktualizację oraz usuwanie rekordów z bazy danych z wykorzystaniem badanych bibliotek mapowania obiektowo-relacyjnego. Dla każdej operacji wykonano serię dziesięciu prób, a następnie uśredniono otrzymane rezultaty. Następnie rezultaty dotyczące konkretnej metody i wykorzystanej biblioteki zostały porównane z pozostałymi wynikami.

Do zmierzenia czasu wykorzystano podejście programowania zorientowanego aspektowo (AOP), umożliwiającego przechwycenie działań wskazanych metod, poprzez uruchomienie określonego fragmentu kodu przed oraz po logice wykonującej się w przechwytywanych metodach. Implementację tego rozwiązania umożliwił wykorzystywany szkielet programistyczny Spring Boot, dający dostęp do adnotacji @Aspect oraz @Around. Do badań wykorzystano urządzenia o specyfikacji przedstawionej w tabeli 1.

Tabela 1: Dane środowiska testowego

Nazwa	Wartość
System operacyjny	Windows 10
Procesor	Intel Core i5 6300HQ
Pamięć operacyjna	16GB DDR4
Dysk	SSD 500GB

Dla operacji odczytu przygotowano trzy zapytania. Każde z zapytań wykorzystuje inne funkcje pozwalające na wyszukiwanie danych. Pierwsze zapytanie skupia się na wyszukaniu danych bazując na pojedynczej tabeli. W drugim zapytaniu skupiono się na wyszukaniu danych z tabel powiązanych relacją. Trzecie zapytanie wykorzystuje funkcję agregujące. Na listingu 1 – 3 przedstawiono wykorzystane zapytania JPQL podczas badania operacji odczytu.

Listing 1: Operacja odczytu danych z pojedynczej tabeli

```
@Query("select v from Vote v " +
       "where v.post is not null")
Set<Vote> findAllPostVotes();
```

Listing 2: Operacja odczytu danych wykorzysujące zewnętrzne złączenie dwóch tabel

```
@Query("select p from Post p " +
       "left join p.answers a " +
       "where a.id is null")
Set<Post> findAllWithoutAnswers();
```

W celu wykonania badania wykonania operacji zapisu nowych rekordów przygotowano zestaw tysiąca, 5 tysięcy oraz 10 tysięcy danych. Badanie polegało na utworzeniu oraz zapisaniu przygotowanej serii danych do tabeli.

Dla operacji aktualizacji danych przygotowano zestaw tysiąca, 5 tysięcy oraz 10 tysięcy danych. Operacja ma na celu zaktualizowanie danych w dwóch tabelach powiązanych relacją jeden-do-jednego. Przykładowa operacja, umożliwiająca zarówno aktualizację jak i zapis danych przedstawiono na listingu 4.

Listing 3: Operacja odczytu danych wykorzystująca funkcję agregującą

```
@Query("select u from User u " +
       "inner join u.userDetails ud " +
       "inner join u.votes v " +
       "where ud.gender = :gender " +
       "group by v.user " +
       "having count(v.user) >= 10")
Set<User> findAllByGenderWithMin10Vote(String gender);
```

Listing 4: Operacja umożliwiająca zapis oraz aktualizację danych

```
@Transactional
@Override
public void createOrUpdateAll(Set<User> users) {
    userRepository.saveAll(users);
}
```

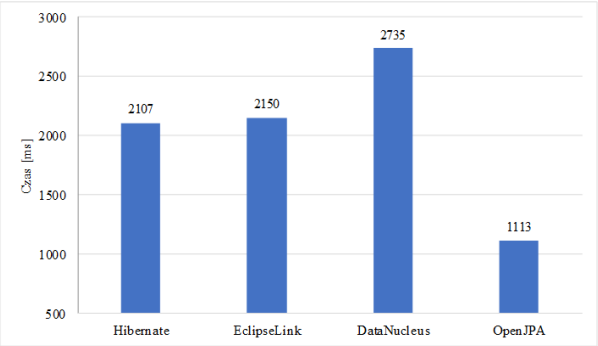
Dla operacji usuwania zaplanowano zestaw instrukcji umożliwiający usunięcie rekordu oraz relacyjnie powiązanych z nią rekordów z innych tabel. Przygotowano trzy instrukcje mające na celu usunięcie 100, 500 oraz tysiąca rekordów. Na listingu 5 przedstawiono instrukcję wykorzystaną podczas usuwania danych.

Listing 5: Operacja usunięcia rekordów wraz z relacyjnie powiązanymi danymi z innych tabel

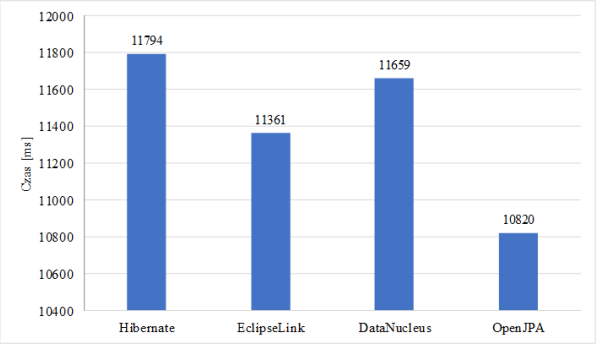
```
@Transactional
@Override
public void deleteAll(Set<User> users) {
    final var userIds = users.stream()
            .map(User::getId)
            .collect(Collectors.toSet());
    postService.deleteAllPostsByUsersIds(userIds);
    voteService.deleteVotesByUsersIds(userIds);
    answerService.deleteAnswersByUsersIds(userIds);
    commentService.deleteCommentsByUsersIds(userIds);
    userDetailsService.deleteAllByIds(users);
    userRepository.deleteAll(users);
}
```

6. Wyniki badań

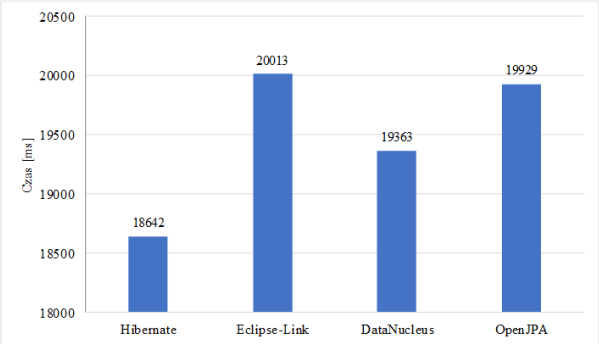
Tabela 2 przedstawia zbiorcze zestawienie otrzymanych rezultatów dla zapytań wyszukiwania rekordów. Czasy odczytu danych przedstawiono na rysunkach 1 – 3. Analizując otrzymane rezultaty można stwierdzić, że najbardziej optymalną technologią jest OpenJPA. Technologia ta wykazuje ponad dwukrotnie szybsze działanie wykonania operacji na pojedynczej tabeli względem innych technologii. Również dla operacji, które wymagają odczytu danych z tabel połączonych relacją wykazuje nieznaczną przewagę. Dla operacji wykorzystującej funkcję agregującą uzyskała nieznacznie wolniejszy czas, jednak otrzymany rezultat nie odbiega od uzyskanego przez pozostałe szkielety programistyczne. Warto wyróżnić również technologię Hibernate, która względem wyszukiwania na pojedynczej tabeli nie ustępuje technologii EclipseLink oraz DataNucleus, natomiast wykazuje szybsze działanie dla operacji wykorzysującej funkcje agregujące.



Rysunek 1: Średni czas odczytu danych z pojedynczej tabeli ze zbioru 200 tyś. rekordów.



Rysunek 2: Średni czas odczytu danych z wykorzystaniem relacji ze zbioru 500 tyś. rekordów.

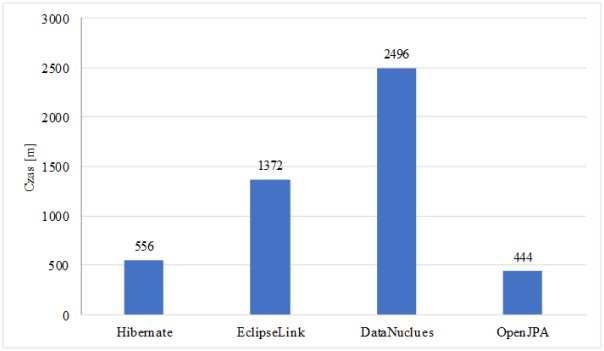


Rysunek 3: Średni czas odczytu danych z wykorzystaniem funkcji agregującej ze zbioru 500 tyś. rekordów.

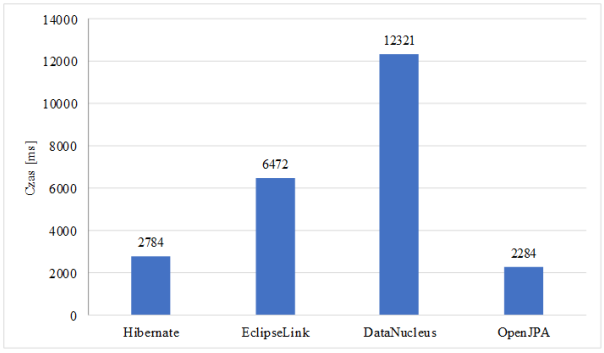
Tabela 2: Średni czas odczytu danych w milisekundach

Seria / Tech.	Zapytanie nr 1	Zapytanie nr 2	Zapytanie nr 3
Hibernate	2107 ms	11794 ms	**18642 ms**
EclipseLink	2150 ms	11361 ms	20013 ms
DataNucleus	2735 ms	11659 ms	19363 ms
OpenJPA	**1113 ms**	**10820 ms**	19929 ms

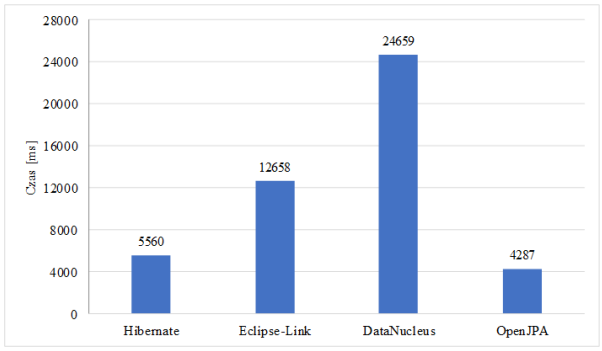
Analogiczne badanie przeprowadzono dla operacji zapisu danych. Rysunki 4-6 prezentują czasy wykonania badanych operacji, porównanie uzyskanych wyników przedstawiono w tabeli 3. Analizując otrzymane rezultaty można stwierdzić, że najwydajniejszą technologią w zakresie zapisu jest OpenJPA. Warto również zwrócić uwagę na technologię Hibernate, która dla małego zbioru otrzymuje czas porównywalny względem OpenJPA, natomiast różnice zwiększają się wraz ze wzrostem liczby operacji. Technologie EclipseLink oraz DataNucleus znacząco odbiegają od wcześniej wspomnianych technologii wykazując kilkukrotnie większy czas realizacji zadania zapisu.



Rysunek 4: Średni czas zapisu 1 tyś. rekordów.



Rysunek 5: Średni czas zapisu 5 tyś. rekordów.



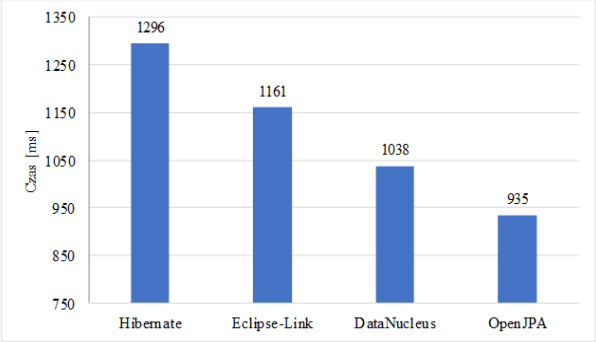
Rysunek 6: Średni czas zapisu 10 tyś. rekordów.

Tabela 3: Średni czas zapisu danych w milisekundach

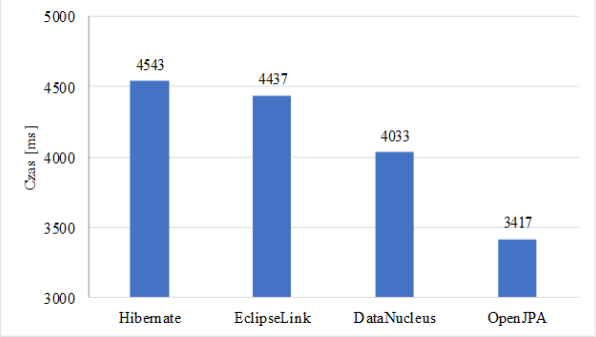
Seria / Tech.	1 tyś. rekordów	5 tyś. rekordów	10 tyś. rekordów
Hibernate	556 ms	2784 ms	5560 ms
EclipseLink	1372 ms	6472 ms	12658 ms
DataNucleus	2496 ms	12321 ms	24659 ms
OpenJPA	**444 ms**	**2284 ms**	**4287 ms**

Wyniki uzyskane podczas badania wydajności aktualizacji danych w bazie przedstawiono na rysunkach 7- 10. Tabela 4 przedstawia średni czas wykonania operacji aktualizacji danych. Analizując otrzymane

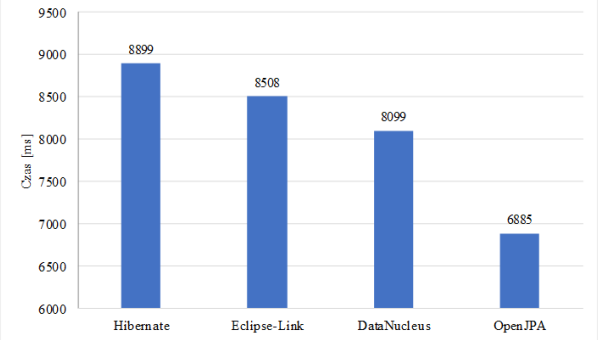
rezultaty można stwierdzić, że najwydajniejszą technologią w zakresie aktualizacji danych jest OpenJPA. Dla mniejszego zbioru danych różnice pomiędzy technologiami są nieznaczne, natomiast rosną one wraz ze wzrostem liczby wykonywanych operacji.



Rysunek 7: Średni czas aktualizacji 1 tyś. rekordów.



Rysunek 8: Średni czas aktualizacji 5 tyś. rekordów.

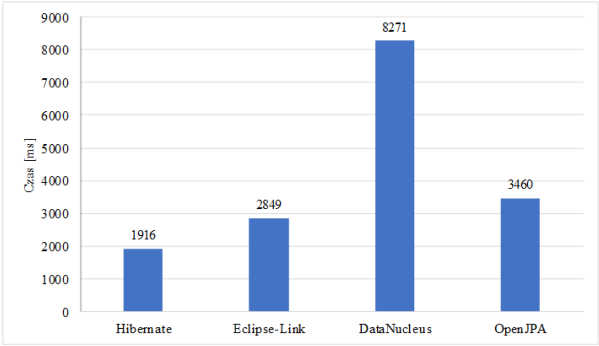


Rysunek 9: Średni czas aktualizacji 10 tyś. rekordów.

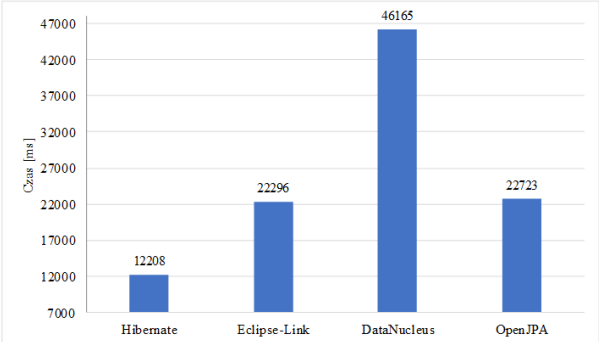
Tabela 4: Średni czas aktualizacji danych w milisekundach

Seria / Tech.	1 tyś. Rekordów	5 tyś. rekordów	10 tyś. rekordów
Hibernate	1296 ms	4543 ms	8899 ms
EclipseLink	1161 ms	4437 ms	8508 ms
DataNucleus	1038 ms	4033 ms	8099 ms
OpenJPA	**935 ms**	**3417 ms**	**6885 ms**

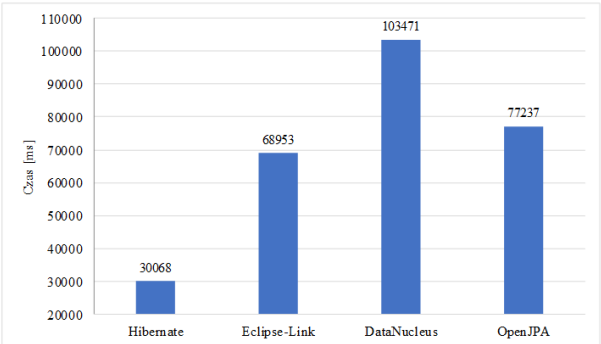
Badanie czasu wykonania zapytania usuwania rekordów wykazało, że najwydajniejszą technologią w zakresie usuwania danych jest Hibernate uzyskując minimum dwukrotnie lepszy czas względem innych technologii. Wyniki uzyskane przez poszczególne technologie zostały zaprezentowane na rysunkach 10 – 12. Dodatkowo Hibernate uzyskuje najmniejszy wzrost czasu przy wzroście liczby operacji do wykonania podczas usuwania. Tabel 5 zawiera zbiorcze zestawienie otrzymanych rezultatów.



Rysunek 10: Średni czas usunięcia 100 rekordów.



Rysunek 11: Średni czas usunięcia 500 rekordów.



Rysunek 12: Średni czas usunięcia 1 tyś. rekordów.

Tabela 5: Średni czas usunięcia danych w milisekundach

Seria / Tech.	1 tyś. Rekordów	5 tyś. rekordów	10 tyś. rekordów
Hibernate	**1916 ms**	**12208 ms**	**30068 ms**
EclipseLink	2849 ms	22296 ms	68953 ms
DataNucleus	8271 ms	46165 ms	103471 ms
OpenJPA	3460 ms	22723 ms	77237 ms

7. Wnioski

Wykorzystując wybrane materiały oraz publikacje, dokonano przeglądu czterech wykorzystywanych w szkielecie programistycznym Spring technologii ORM:

Hibernate ORM, EclipseLink, Apache OpenJPA oraz DataNucleus. Następnie wykonano praktyczne porównanie wydajności wybranych technologii.

Każda z badanych bibliotek wykorzystuje adnotacje JPA, przez co sam kod reprezentujący encje, serwisy i repozytoria jest do siebie bardzo podobny w każdym z omawianych przypadków. Z tego też powodu na poziom skomplikowania korzystania z danej biblioteki najbardziej wpływał proces konfiguracji narzędzia. Zdecydowanie najlepiej pod tym względem wypadła biblioteka Hibernate, która ze wszystkich narzędzi najlepiej integruje się ze szkieletem programistycznym Spring, oraz posiada najbogatszą dokumentację techniczną wraz z najliczniejszą, wspierającą się społecznością programistów wykorzystujących to narzędzie. Najgorzej natomiast wypadła biblioteka DataNucleus oraz Apache OpenJPA. Obie biblioteki nie posiadają wsparcia ze strony szkieletu programistycznego Spring, nie posiadają aktualnej dokumentacji, która jest również zbyt uboga, by umożliwić sprawną integrację środowisk.

W przypadku operacji wyszukiwania przeprowadzanej na jednej tabeli oraz na tabeli połączonej relacjami, najlepsza okazała się biblioteka Apache OpenJPA. Podczas wykorzystania funkcji agregujących prym wiodła technologia Hibernate ORM. Ogólnie podczas operacji wyszukiwania najlepiej wypadła biblioteka Apache OpenJPA, natomiast najgorzej biblioteka DataNucleus.

Badania operacji zapisu zaowocowały następującym wnioskiem – narzędzie Hibernate ORM jest najefektywniejsze w przypadku małych zbiorów danych, natomiast im większy jest ten zbiór, tym wydajniejsza okazuje się biblioteka Apache OpenJPA. Najgorzej podczas przeprowadzania zapisu danych wypadała technologia DataNucleus.

Test aktualizacji wykonany na małym i dużym zbiorze danych wykazał, że najwydajniejsza w tym przypadku jest biblioteka Apache OpenJPA, jednak biblioteka DataNucleus niewiele jej ustępuje. Najgorzej podczas testów wypadło narzędzie Hibernate ORM.

Podczas usuwania danych prym wiódł Hibernate ORM, a zdecydowanie najgorzej wypadła biblioteka DataNucleus.

Najlepiej w części praktycznej wypadła biblioteka Apache OpenJPA, na drugim miejscu natomiast znajduje się technologia Hibernate ORM.

Na podstawie przeprowadzonych badań oraz zdobytych informacji można wysunąć następujący wniosek – w przypadku, gdy w tworzonym systemie najważniejszą rolę gra wydajność, najlepszą opcją jest wykorzystanie technologii Apache OpenJPA, ponieważ uzyskała ona najlepsze wyniki w przypadku operacji wyszukiwania, zapisu i aktualizacji danych, a w przypadku operacji usuwania danych niewiele ustępowała ona technologii Hibernate. Jeżeli jednak wydajność nie jest kluczowa dla działania tworzonej aplikacji, lepszym wyborem okazuje się być technologia Hibernate ORM, która przewyższa OpenJPA pod względem prostej konfiguracji, popularności oraz dostępu do bogatych zasobów wiedzy i doświadczonej społeczności.

Literatura

[1] J. M. Barnes, Object-relational mapping as a persistence mechanism for object-oriented applications, Macalester College, 2007.

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

[3] P. Błoch, M. Wojciechowski, Analiza porównawcza technologii odwzorowania obiektowo-relacyjnego dla aplikacji Java. XIII Konferencja PLOUG: Systemy informatyczne. Projektowanie, implementowanie, eksploatowanie, Zakopane, 2007.

[4] B. B. Correa, Y. Wang, E. Zimanyi, Object-relational mapping tools and Hibernate, Universite libre de Bruxelles, 2017.

[5] N. Dhingra, Analysis of ORM based JPA Implementations, University of Ottawa, 2017.

[6] C. Bauer, K. Gavin, G. Gary, Java Persistence. Programowanie aplikacji bazodanowych w Hibernate. Wydanie II, Helion, 2016.

[7] Dokumentacja techniczna biblioteki ORM Eclipselink https://www.eclipse.org/eclipselink/documentation, [22.10.2021].

[8] Dokumentacja techniczna biblioteki ORM Apache OpenJPA, http://openjpa.apache.org/documentation.html, [22.10.2021].

[9] Dokumentacja techniczna biblioteki ORM DataNucleus, https://www.datanucleus.org/products/accessplatform_6_0/, [22.10.2021]

[10] K. Appigatla, MySQL 8 Cookbook, Packt Publishing, 2018.

[11] J. Bloch, Java. Efektywne programowanie. Wydanie III, Helion, 2018.

[12] S. Raemaekers, A. Van Deursen, J. Visser, The maven repository dataset of metrics, changes, and dependencies, 10th Working Conference on Mining Software Repositories (2013) 221-224.