УДК 004.04

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ПОБУДОВИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ВИКОРИСТАННЯ РІЗНИХ ОБ'ЄКТНО-РЕЛЯЦІЙНИХ ФРЕЙМВОРКІВ JAVA З СУБД ПРИ РОЗРОБЦІ ВЕБ-ДОДАТКІВ

Горбік Максим Миколайович
к.т.н., доцент каф. ІС Глава Марія Геннадіївна
Національний університет «Одеська політехніка», УКРАЇНА

АНОТАЦІЯ. Робота присвячена створенню рекомендаційної системи використання різних об'єктно-реляційних фреймворків Java з системами управління базами даних при розробці веб-додатків. В основі ідеї лежить аналіз особливостей фреймворків та їх роботи з різними типами баз даних.

Вступ. З останніми технологічними розробками та зростанням обсягу даних у сфері веб-додатків виникає потреба в ефективному доступі до інформації та її обробці. А все більша цифровізація задає попит на розробку та породжує велику кількість користувачів мережі, а відповідно і даних. В залежності від структури даних, для їх зберігання можна використовувати реляційні або NoSQL бази даних. Java є третьою в світі за популярністю мовою програмування[1], а тому була обрана для дослідження. Сучасна розробка бекенд частини на Java передбачає можливість використання об'єктно-реляційних фреймворків для спрощення взаємодії між такими базами та програмним продуктом. Однак, для досягнення оптимальної продуктивності та швидкості відгуку застосунків, необхідно обрати саме той набір технологій, який водночас забезпечить задоволення потрібних показників ефективності та дозволить вести розробку швидко та без залучення високорівневих фахівців.

Мета роботи. Мета роботи полягає в розробці методики використання різних об'єктно-реляційних фреймворків Java, зокрема Hibernate, Spring Data, та JDBC з різними варіантами СУБД при розробці веб-додатків з урахуванням вимог до швидкодії та термінів розробки додатку.

Основна частина роботи. Методика роботи передбачає оцінку низки критеріїв, за якими можливо обрати краще програмне рішення для конкретного веб-додатка. При розробці веб-додатків важливо враховувати різні аспекти, включаючи вимоги до швидкодії та термінів розробки. Один із ключових аспектів такої розробки полягає в ефективній роботі з реляційними базами даних, що вимагає перетворення реляційних даних в об'єктну модель, а також забезпечення зручного та продуктивного програмування. Першим кроком для вибору технології буде оцінка знань команди розробників та аналіз архітектури. Об'єктно-реляційне відображення (ORM) є набором технік та методологій програмування, які надають зручність та ефективність при роботі з реляційними базами даних, дозволяючи розробникам зосередитися на бізнес-логіці додатків, а не на деталях роботи з базами даних, але можлива ситуація, коли у використанні бази повноцінної бази даних немає необхідності (наприклад, для веб агрегатору, що використовує сторонній API), або додаток потребуватиме лише базових запитів до бази. ORM фреймворки же використовуються для автоматизації процесу маппінгу об'єктів на таблиці бази даних, що дозволяє здійснювати операції створення, зчитування, оновлення та видалення даних за допомогою об'єктно-орієнтованого підходу, що є більш релевантним для систем з великою кількістю складних сутностей. ORM фреймворки надають високорівневий інтерфейс для роботи з базами даних, що дозволяє розробникам уникнути прямого взаємодії з SQL запитами та забезпечує абстракцію від деталей низькорівневої роботи з базами даних. Це спрощує процес розробки та підтримки веб-додатків, оскільки розробнику не потрібно вручну створювати та оптимізувати SQL запити, і він може зосередитися на більш високорівневих завданнях. Однак, вибір найкращого ORM фреймворка для конкретного проекту може бути складним завданням, оскільки існує багато факторів, які впливають на продуктивність, швидкодію та простоту використання таких технологій.

Перший з основних факторів, на котрий потрібно звернути увагу - це ефективність генерації SQL запитів. Деякі ORM фреймворки використовують складні алгоритми для автоматичного

створення оптимальних SQL запитів, що може позитивно вплинути на швидкодію додатка. Крім того, підтримка кешування результатів запитів і передбачення попереднього завантаження (eager loading) даних можуть значно зменшити кількість виконуваних запитів до бази даних і покращити швидкодію додатка.

Інший фактор - підтримка різних типів відношень між об'єктами. Деякі ORM фреймворки надають зручний спосіб визначення зв'язків типу один до одного, один до багатьох та багато до багатьох. Це дозволяє розробникам з легкістю моделювати складні відношення між об'єктами в базі даних, що спрощує розробку та підтримку системи[2].

Також варто звернути увагу на механізми кешування і оптимізації доступу до даних. Деякі ORM фреймворки підтримують кешування результатів запитів або навіть повного об'єктного кешування, що дозволяє зберігати часто використовувані дані в оперативній пам'яті та зменшує навантаження на базу даних. Крім того, оптимізовані механізми доступу до даних можуть покращити продуктивність додатка, дозволяючи ще ефективніше виконувати запити до бази [3].

Також ORM фреймворки підтримують механізми міграції бази даних, що дозволяє автоматично змінювати схему бази даних відповідно до змін у моделі додатку. Це полегшує процес розгортання та підтримки додатку, забезпечуючи синхронізацію структури бази даних зі змінами в коді додатку. І крім того, при використанні специфічних для фреймворків мов складання запитів, можливо підтримувати структуру системи незалежною від реалізації СУБД, з якими працює додаток.

Для більш детального дослідження були обрані найбільш популярні ORM фреймворки Java[4], а також базова модель підключення Java додатку до бази[5], а саме:

1. Hibernate – найпопулярніший ORM фреймворк Java. Підтримує JPA специфікації.
2. Spring Data JPA – проект Spring, що спрощує роботу з JPA.
3. JDBC - низькорівневий інтерфейс для роботи з базами даних.
4. EclipseLink – інший відомий фреймворк з відкритим вихідним кодом, більш прив’язаний до стандарту JPA.

Для порівняння функціоналу ORM-фреймворків було створено простий web додаток – CRM систему обробки замовлень кафе. Було використано найпопулярніші на даний момент[6] СУБД та Spring Boot як основний фреймворк для розробки. Було проведено моделювання сутностей Order (замовлення), Customer (клієнт), Product (продукт) та OrderItem (позиція замовлення), і були створені відповідні таблиці у базі даних. Було налаштовано зв'язки між сутностями: один до багатьох між Order і Customer, багато до багатьох між Order і Product за допомогою OrderItem як проміжної таблиці. Було розглянуто кілька ключових аспектів для проведення порівняльного аналізу, зокрема швидкість виконання операцій з базою даних, простота використання фреймворків та терміни розробки додатку. Також були враховані вимоги до швидкодії додатку, а також можливість використання додаткових функціональностей, таких як entity graphs, та projections.

Для оцінки швидкості виконання операцій з базою даних було проведено низку тестів, включаючи вставку, оновлення, видалення та вибірку даних з бази. Кожен тест виконувався на різних ORM-фреймворках з використанням різних варіантів СУБД, таких як MySQL(v 8.0.31), PostgreSQL (v15) та Oracle(v21.3).

Для проведення дослідження було створено експериментальне середовище, що складалося з сервера бази даних, веб-додатку та набору тестових сценаріїв. Використовувалася машина з процесором Intel Core i7, 8 ГБ оперативної пам'яті та HDD-накопичувачем.

Для кожного фреймворку було реалізовано набір тестових сценаріїв, що включали операції вставки, оновлення, видалення та вибірки даних з бази. Кожен сценарій виконувався на всіх розглянутих фреймворках з використанням різних варіантів СУБД. Для кожної операції було виміряно час виконання і збережено результати для подальшого аналізу. Було розраховано середні значення часу виконання операцій вставки, оновлення, видалення та вибірки даних з бази для кожного варіанта СУБД. Результати представлені в таблиці нижче у таблиці 1.

Таблиця 1 – Час обробки запитів з фреймворками та СУБД

Фреймворк	СУБД	Вставка (мкс)	Оновлення (мкс)	Видалення (мкс)	Вибірка (мкс)
Hibernate	MySQL	667	671	412	249
Hibernate	PostgreSQL	311	383	198	82
Hibernate	Oracle	277	248	167	39
Spring Data	MySQL	998	391	527	362
Spring Data	PostgreSQL	586	257	258	148
Spring Data	Oracle	408	158	175	80
JDBC	MySQL	585	641	494	204
JDBC	PostgreSQL	376	260	292	104
JDBC	Oracle	247	243	179	32
EclipseLink	MySQL	831	687	361	503
EclipseLink	PostgreSQL	485	390	173	152
EclipseLink	Oracle	396	270	133	64

При аналізі не враховувався час першого запиту, оскільки він включає час витрачений на парсинг самого запиту. Усього було зроблено по 1000 транзакцій на кожну комбінацію бази та фреймворку. Батчі не використовувалися. Додаткових індексів у базах створено не було.

Слід ще раз зауважити, що окрім створених самими базами індексів первинного ключа, додаткових індексів використано не було. А згідно з дослідженням Pedro Martins [7], належне використання індексів може збільшити продуктивність бази PostgreSQL на 91% і вона зможе перевершити продуктивність бази Oracle до 75% в окремих випадках.

Далі для оцінки простоти використання кожного фреймворку було пораховано кількість рядків коду, необхідних для реалізації типових операцій з базою даних. Результати представлені в таблиці 2.

Таблиця 2 – Кількість рядків коду для створення запитів засобами фреймворків

Фреймворк	Вставка (рядків коду)	Оновлення (рядків коду)	Видалення (рядків коду)	Вибірка (рядків коду)
Hibernate	15	14	12	14
Spring Data	5	8	6	7
JDBC	18	18	16	16
EclipseLink	16	14	18	19

Враховувалася кількість рядків між початком об’яви типового методу та його закриваючою дужкою. Для Spring Data не враховувалися об’яви полів класу та додаткові конструкції, що складали об’яву та ін’єкцію репозиторію та анотацію над класом. Для EclipseLink схожі об’яви враховувалися, тому що вони могли бути об’явлені в межах методу. Тому для кожного виду операцій можливо вважати, що рядків для EclipseLink може бути на 2 менше. Слід зауважити, що складність коду для JDBC є найвищою, крім того, кількість рядків не враховує налаштування самого підключення до бази. Hibernate та EclipseLink є майже еквівалентними за складністю налаштувань, але потребує Hibernate трохи менше коду. Spring Data є безумовним лідером за простотою виконання запитів, зрозумілістю та ненадлишковістю коду і легкістю налаштування завдяки середовищу Spring.

Зважаючи на доступність додаткових технологій оптимізації, було вирішено додатково перевірити швидкодію для складної транзакції, котра б виконувала усі чотири CRUD операції. Після цього було проведено ще один тест і отримано наступні результати: завдяки кешуванню першого рівня, наявного в JPA-compliant фреймворках, якими є Hibernate та EclipseLink, а відповідно і Spring Data, вдалося скоротити час повторного читання даних у 6,5 разів для Spring Data (23 мкс в середньому), та більш ніж у 90 разів для Hibernate (3 мкс в середьому) та EclipseLink (5 мкс в середньому). Водночас, JDBC кешування не підтримує в початковому вигляді.

Висновки. При виборі фреймворку для розробки веб-додатку необхідно враховувати вимоги до швидкодії та термінів розробки. Для проектів з дуже високими вимогами до швидкодії

можна рекомендувати використання Hibernate, оскільки він забезпечує найвищу продуктивність (до 20 разів швидше за Spring Data при читанні) при адекватній складності налаштувань та вимог до знань розробника. Для проектів з меншими вимогами до швидкодії можна рекомендувати використання Spring Data, оскільки цей фреймворк дозволяє дуже сильно спростити розробку та підтримку коду (сумарна кількість рядків коду більше ніж вдвічі менша в порівнянні з Hibernate, не враховуючи налаштування), а також пришвидшити процес розробки.

EclipseLink хоч і відповідає в більшій мірі швидкодії та функціоналу Hibernate, та все ж поступається йому (сумарно на 20% більше коду та на 20% повільніше за Hibernate) та не має настільки вичерпної документації та поширеності, а отже не має очевидних переваг.

JDBC можна рекомендувати для невеликих застосунків, які виконують нескладні запити. Написання великого додатку з використанням JDBC потребуватиме значних зусиль розробника через високу вірогідність помилок в ручному написанні запитів, а також більшу вірогідність вразливостей через відсутність вбудованих в ORM фреймворки засобів захисту.

З боку СУБД найкраще себе проявила система Oracle в загальному тесті та добре показала PostgreSQL. В залежності від потреб замовника та бюджету, можна рекомендувати обидві системи. MySQL показала гірший відгук та не мала видимих переваг над іншими засобами в контексті даної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. TIOBE Index for May 2023. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 10.05.23).
2. Java & Databases: An Overview of Libraries & APIs. URL: https://www.marcobehler.com/guides/java-databases (дата звернення: 02.04.23).
3. Hibernate ORM. URL: https://hibernate.org/orm/ (дата звернення: 02.04.23).
4. 10 Best Java Frameworks to Use in 2023. URL: https://hackr.io/blog/java-frameworks (дата звернення: 02.04.23).
5. Java JDBC API. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/ (дата звернення: 02.04.23).
6. DB-Engines Ranking - Trend Popularity. URL: https://db-engines.com/en/ranking_trend (дата звернення: 10.03.23).
7. Comparing Oracle and PostgreSQL, Performance and Optimization. URL: https://link.springer.com/chapter/10.1007/978-3-030-72651-5_46 (дата звернення: 15.04.23).
8. How Much Projections Can Help? URL: https://arnoldgalovics.com/jpa-projections-comparison/ (дата звернення: 01.05.23)
9. Using Projections In Your Data-Access Layer. URL: https://arnoldgalovics.com/jpa-projections/ (дата звернення: 01.05.23)
10. Müllenbach, Sabine, Kern-Bausch, Lore & Kolonko, Matthias. “Conceptual Modeling LanguageAgila Mod”. Herald of Advanced Information Technology. Publ. Science i Technical. 2019; Vol.2No.4:p.246–258. Odesa. Ukraine. DOI: https://doi.org/10.15276/hait.04.2019.1

DEVELOPMENT AND RESEARCH OF BUILDING A RECOMMENDATION SYSTEM USING DIFFERENT JAVA OBJECT-RELATIONAL FRAMEWORKS WITH DBMS IN THE DEVELOPMENT OF WEB APPLICATIONS

Horbik Maksym

PhD, Associate professor of the Department of IS Maria Glava

Odesa Polytechnic National University, UKRAINE

ANNOTATION. The work is devoted to the creation of a recommendation system for the use of various Java object-relational frameworks with database management systems in the development of web applications. The basis of the idea is the analysis of the features of frameworks and their work with effective types of databases.