поєднанні з високоякісним вокодером (як WaveNet) може вважатися кращим вибором, оскільки він часто встановлює стандарт якості. Проте якщо ключовими факторами є швидкість генерації, ефективність і можливість обробки великих обсягів тексту для відео, то системи на кшталт FastSpeech 2 та особливо FastSpeech 2s є значно привабливішими. Вони пропонують відмінний компроміс між швидкістю та якістю, що робить їх більш практичними для багатьох реальних завдань автоматизованого озвучення. Отже, оптимальна TTS-система обирається шляхом зважування переваг у природності звучання проти переваг у швидкості та ефективності генерації відповідно до потреб конкретного проєкту.

Список використаних джерел

1. A Survey on Neural Speech Synthesis / X. Tan, T. Qin, F. Soong, T. -Y. Liu. *arXiv.org*. 2021. URL: https://arxiv.org/abs/2106.15561 (дата звернення: 12.05.2025).
2. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions / J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, Y. Wu. *arXiv.org*. 2017. URL: https://arxiv.org/abs/1712.05884 (дата звернення: 12.05.2025).
3. WaveNet: A Generative Model for Raw Audio / A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu. *arXiv.org*. 2016. URL: https://arxiv.org/abs/1609.03499 (дата звернення: 12.05.2025).
4. Towards achieving robust universal neural vocoding / J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, R. Barra-Chicote, A. Moinet, V. Aggarwal. *arXiv.org*. 2018. URL: https://arxiv.org/abs/1811.06292 (дата звернення: 12.05.2025).
5. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech / Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, T.-Y. Liu. *arXiv.org*. 2020. URL: https://arxiv.org/abs/2006.04558 (дата звернення: 12.05.2025).

УДК 004.415.2:004.657

Лисак В. О., *здобувач вищої освіти 2 курсу спеціальності 122 Комп'ютерні науки*
Зелінська О. В., *канд. техн. наук, доцент, в. о. завідувача кафедри інформаційних технологій*

ОПТИМІЗАЦІЯ ЗАПИТІВ ДО БАЗИ ДАНИХ У ВЕБПРОЄКТАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасних вебпроєктах, де PHP виступає серверною мовою, а MySQL – системою керування базами даних, обсяг інформації постійно зростає. Відповідно ефективність SQL-запитів стає критично важливою для продуктивності системи. Запити без належної оптимізації можуть значно «стиснути» швидкодію програми та спричинити затримки у відповіді [1].

Для забезпечення взаємодії між вебдодатками та базами даних використовуються переважно два підходи: безпосереднє виконання SQL-запитів або застосування ORM (Object-Relational Mapping). Перший підхід полягає у прямому написанні SQL-команд у коді програми для роботи з базою даних. Натомість другий підхід ORM представляє технологію абстракції, що забезпечує відображення структур бази даних на об'єкти цільової мови програмування вебдодатку [7].

Оптимізація запитів є надзвичайно актуальною у контексті сучасних інформаційних систем. Неоптимізовані запити часто стають «вузькими місцями», що уповільнюють виконання операцій з базою даних (БД) і збільшують навантаження на сервер [2]. Своєчасне застосування технік оптимізації дає змогу знизити кількість звернень до БД, прискорити відповіді сервера і покращити користувацький досвід. Актуальність теми зумовлена потребою обробки великих обсягів даних та забезпечення масштабованості вебдодатків.

Сучасні керівництва з оптимізації SQL рекомендують низку перевірених практик. Зокрема, документація MySQL радить використовувати оператор EXPLAIN для отримання плану виконання запиту і визначення, де треба додати індекси [3]. Інші джерела зазначають, що нормалізація структури БД зменшує надлишковість даних і складність запитів, що часто дає змогу спростити потрібні JOIN-операції і підвищити швидкодію [4]. Дослідження оптимізації JOIN-ів підкреслюють важливість індексації стовпців, за якими відбувається з'єднання: без відповідних індексів MySQL змушений робити повне сканування таблиць, що суттєво сповільнює виконання [5]. Також у літературі зазначаються і практики ефективного проєктування запитів (наприклад, уникнення зайвого SELECT*) та використання кешування для зменшення кількості звернень до БД. Зазначені публікації рекомендують мінімізувати перенесення зайвих полів і використовувати обмеження кількості рядків (LIMIT) під час формування результатів [1; 2].

Експерти з веброзробки підкреслюють, що ефективність роботи з базами даних є одним із найважливіших чинників, що визначають загальну продуктивність вебдодатків і, як наслідок, задоволеність користувачів від роботи з системою. Фахівці рекомендують веброзробникам дотримуватися комплексу перевірених методик для покращення ефективності роботи з базами даних [8]:

1. Використання EXPLAIN із запитами SELECT.
2. Включення індексів на шуканих стовпцях.
3. Використання ідентифікаційних полів, коли це можливо.
4. Мінімізування NULL-значення за замовчуванням.
5. Використання небуферизованого режиму для запитів.
6. Оптимізація розміру стовпця для ефективності.

1. Використання EXPLAIN із запитами SELECT

Одним із найважливіших інструментів для оптимізації запитів до бази даних у MySQL є оператор EXPLAIN. Він дає змогу розробникам отримати детальну інформацію про те, як MySQL виконує запити SELECT. Аналізуючи вивід EXPLAIN, можна виявити «вузькі місця» запиту, як-от повне сканування таблиць (Full Table Scan), відсутність або неефективне використання індексів, а також некоректне об'єднання таблиць [3].

2. Включення індексів на шуканих стовпцях

Індекси є фундаментальним механізмом для прискорення доступу до даних у базах даних. Вони працюють аналогічно предметному покажчику в книзі: замість повного перегляду всіх сторінок для пошуку потрібної інформації, індекс дає змогу швидко перейти до необхідного місця [6]. Для MySQL індексація стовпців, за якими здійснюється пошук (WHERE), сортування (ORDER BY) або об'єднання таблиць (JOIN) є критично важливими.

3. Використання ідентифікаційних полів, коли це можливо

Ідентифікаційні поля, або первинні ключі (Primary Keys), відіграють центральну роль у структурі реляційних баз даних. Вони забезпечують унікальність кожного запису в таблиці та є основним засобом для швидкого доступу до конкретних даних. Зазвичай первинні ключі реалізуються як автоінкрементні цілі числа, що забезпечує їх компактність і ефективність для індексації [8].

4. Мінімізування NULL-значення за замовчуванням

Під час проєктування структури бази даних важливо враховувати вплив NULL-значень на продуктивність запитів. Хоча NULL може здаватися зручним для представлення відсутності даних, його використання має свої наслідки. По-перше, NULL-значення можуть ускладнити логіку запитів, оскільки для них потрібні спеціальні оператори (IS NULL, IS NOT NULL) замість звичайних порівнянь. По-друге, NULL-значення можуть негативно впливати на ефективність індексів, оскільки індекси не завжди охоплюють рядки з NULL.

5. Використання небуферизованого режиму для запитів

У деяких сценаріях роботи з базою даних, особливо під час обробки великих обсягів даних, використання небуферизованого режиму для запитів може значно покращити продуктивність. За замовчуванням більшість драйверів баз даних (у тому числі для PHP) буферизують усі результати запиту в пам’яті сервера додатків перед тим, як надати їх програмі [4]. Це може призвести до значного споживання пам’яті та затримок, якщо результат запиту великий.

6. Оптимізація розміру стовпця для ефективності

Правильний вибір типів даних та розмірів стовпців є ключовим аспектом оптимізації схеми бази даних. Використання надлишкових типів даних або необґрунтовано великих розмірів стовпців може призвести до зайвого споживання дискового простору та оперативної пам’яті, а також сповільнити виконання запитів. Кожен зайвий байт, що зберігається в таблиці, впливає на швидкість читання даних, оскільки більше даних потрібно завантажувати з диска [5].

Проведене дослідження підтверджує, що оптимізація запитів до бази даних є критично важливою для забезпечення високої продуктивності та масштабованості сучасних вебпроєктів на PHP з MySQL. Аналіз останніх досліджень та рекомендацій фахівців виявив низку ключових практик, впровадження яких дає змогу ефективно вирішувати проблему «вузьких місць», спричинених неоптимізованими SQL-запитами. Зокрема, систематичне використання EXPLAIN для аналізу запитів SELECT, грамотне індексування шуканих стовпців, застосування ідентифікаційних полів, мінімізація NULL-значень та використання NOT NULL, а також задіяння небуферизованого режиму для великих обсягів даних і ретельна оптимізація розмірів стовпців – усі ці методики є фундаментальними. Їх комплексне застосування не лише покращує швидкість взаємодії вебдодатків з базою даних, але й підвищує загальну стабільність системи і задоволеність користувачів, що є ключовим у динамічному середовищі веброзробки.

Список використаних джерел

1. Оптимізація запитів SQL в PHP. *Foxminded.* 09.04.2025. URL: https://surl.li/vkxnrj (дата звернення: 13.05.2025).

2. PHP Performance Tuning Tips and Tricks. *MoldStud Research Team.* URL: https://surl.li/hcjqxy (дата звернення: 13.05.2025).

3. MySQL 8.4 Reference Manual. Optimization. Oracle. 2024 (section «Optimizing Queries with EXPLAIN»). URL: https://surl.lu/hrcnww (дата звернення: 13.05.2025).

4. Prakash A. MySQL Query Optimization: Faster Performance & Data Retrieval. Airbyte Data Engineering. 05.09.2025. URL: https://surl.li/kwjwxg (дата звернення: 13.05.2025).

5. Crudu A. Optimizing MySQL Join Queries for Efficiency. *Moldstud Tech Blog.* 2024. URL: https://surl.li/wqupiw (дата звернення: 13.05.2025).

6. Shykunov K. Індекси в MySQL – чому вони потрібні та як з ними працювати. *Developers.org.ua.* 01.11.2023. URL: https://dou.ua/forums/topic/45982/ (дата звернення: 18.05.2025).

7. Керування базами даних та інтеграція з веб-додатками: MySQL, MongoDB і SQL Server. *Redstone.* 14.10.2025. URL: https://redstone.agency/blog/keruvannia-bazamy-danych-ta-intehracija-z-veb-dodatkamy-mysql-mongodb-i-sql-server/ (дата звернення: 18.05.2025).

8. Rose-Collins F. 10 найкращих практик оптимізації баз даних для веб-розробників. *Ranktracker.* 03.11.2023. URL: https://www.ranktracker.com/uk/blog/top-10-best-practices-for-optimizing-databases-for-web-developers/ (дата звернення: 18.05.2025).

УДК 004.8:613.2

Ліваковський В. К., *здобувач вищої освіти 4 курсу спеціальності 122 Комп'ютерні науки,*
Римар П. В., *старший викладач кафедри інформаційних технологій*

РОЗРОБКА ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ХАРЧУВАННЯМ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Раціональне харчування є ключовим фактором підтримки здоров'я людини. У сучасному інформаційному просторі користувачі стикаються з браком персоналізованих інструментів та складністю формування здорових звичок. Цифрові платформи мають потенціал для вирішення цих проблем [1]. У дослідженні [2] детально розглянуто класифікацію рекомендаційних систем, серед яких контентно-орієнтовані та колаборативні фільтри. Саме їх поєднання дає змогу створити гібридну модель, здатну адаптуватися до поведінкових змін користувача. У роботі [3] автор досліджує використання нейромереж у побудові рекомендацій, акцентуючи на їх здатності навчатися на великих обсягах користувацьких даних та формувати точні прогнозні оцінки.

Окремий напрям розвитку систем – це застосування комп'ютерного зору для аналізу візуальних характеристик страв, їх складу та відповідності дієтичним обмеженням, що представлено у праці [4]. Водночас впровадження інтелектуальних платформ для харчування супроводжується низкою викликів. Серед основних – необхідність побудови якісної бази харчових продуктів із точною калорійністю та мікроелементами, адаптованої до локального ринку; забезпечення етичного збору персональних даних; труднощі з підтриманням мотивації користувача в довгостроковій перспективі. Для подолання цих бар'єрів важливо передбачити гейміфікацію процесу, гнучке навчання системи на індивідуальних даних та прозору політику конфіденційності.