DOI: https://doi.org/10.36910/6775-2524-0560-2025-61-16
УДК 004.65
Лавренчук Світлана Василівна, к.т.н., доцент
https://orcid.org/0000-0002-5453-3924
Кайдик Олег Леонтійович, к.т.н., доцент
https://orcid.org/0000-0002-3620-270X
Мельник Катерина Вікторівна, к.т.н., доцент
https://orcid.org/0000-0002-9991-582X
Конкевич Людмила Миколаївна, асистент
https://orcid.org/0000-0002-8279-3133
Лук'янчук Юлія Володимирівна, магістрант
Луцький національний технічний університет, м. Луцьк, Україна

ГІБРИДНА SQL/NOSQL АРХІТЕКТУРА ДЛЯ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ІОТ-МОНІТОРИНГУ ЯКОСТІ ПОВІТРЯ

Лавренчук С. В., Кайдик О. Л., Мельник К. В., Конкевич Л. М., Лук'янчук Ю. В. Гібридна SQL/NoSQL архітектура для оптимізації продуктивності IoT-моніторингу якості повітря. Внаслідок швидкого зростання кількості IoT-пристроїв та їх інтегрованого використання в різних екосистемах виникає потреба у високій масштабованості, ефективному зберіганні та швидкій обробці величезних масивів даних у режимі реального часу. Через велику частоту надходження та різноманітність форматів даних від сенсорів, архітектура зберігання має бути гнучкою і забезпечувати високу продуктивність операцій запису. Це створює виклик щодо вибору оптимальної системи управління базами даних, яка б поєднувала швидкість обробки великих обсягів даних із можливістю ефективного виконання складних аналітичних запитів до історичних даних. У статті представлено результати експериментального дослідження продуктивності реляційної бази даних PostgreSQL та документо-орієнтованої бази даних MongoDB у контексті системи моніторингу якості повітря на базі IoT. Проведено серію контрольованих навантажувальних тестів з варіюванням обсягу даних (100-10000 записів) та кількості паралельних потоків (1-50). Результати показують дев'ятикратну перевагу MongoDB під час операцій запису (34,56 мс проти 338,16 мс у середньому) та майже двократну перевагу PostgreSQL під час операцій читання (14,41 мс проти 25,98 мс). На основі отриманих даних запропоновано гібридний підхід, що поєднує MongoDB для оперативного збору даних та PostgreSQL для довгострокового зберігання та аналітики. Така інтергація реляційних та NoSQL баз даних є технічно оптимальним та економічно вигідним рішенням. Підтверджено зниження вартості на 30-40 %, що робить його привабливим для реального впровадження у промислових масштабах.
Ключові слова: PostgreSQL, MongoDB, NoSQL, IoT, моніторинг якості повітря, гібридна архітектура, індексування.

Lavrenchuk S., Kaidyk O., Melnyk K., Konkevych L., Luk'yanchuk Y. Hybrid SQL/NoSQL Architecture for Optimizing the Performance of IoT Air Quality Monitoring. The rapid growth of IoT devices and their integration in various ecosystems creates a need for high scalability, efficient storage, and fast processing of large amounts of data in real-time. Due to the high frequency of arrival and variety of data formats from sensors, the storage architecture must be flexible and provide high performance for write operations. This creates a challenge in choosing the optimal database management system that would combine the speed of processing large volumes of data with the ability to execute complex analytical queries on historical data effectively. The article presents the results of an experimental study of the performance of the PostgreSQL relational database and the MongoDB document-oriented database in the context of an IoT-based air quality monitoring system. A series of controlled load tests was conducted with varying data volume (100-10,000 records) and the number of parallel threads (1-50). The results show a nine-fold advantage of MongoDB during write operations (34.56 ms vs. 338.16 ms on average) and an almost two-fold advantage of PostgreSQL during read operations (14.41 ms vs. 25.98 ms). Based on the obtained data, a hybrid approach is proposed that combines MongoDB for operational data collection and PostgreSQL for long-term storage and analytics. Such integration of relational and NoSQL databases is a technically optimal and economically beneficial solution. A cost reduction of 30-40% is confirmed, which makes it attractive for real implementation on an industrial scale.
Keywords: PostgreSQL, MongoDB, NoSQL, IoT, air quality monitoring, hybrid architecture, indexing.

Постановка наукової проблеми. Стрімке зростання кількості IoT-пристроїв створює нові виклики для систем збору та обробки даних. За прогнозами, промислові IoT-системи стикаються з тисячами пристроїв з мільйонами сенсорів, які постійно генерують мільярди точок даних. Системи моніторингу якості повітря є типовим прикладом IoT-застосувань, де важливими є як швидкість запису великих обсягів даних від датчиків, так і можливість ефективного виконання складних аналітичних запитів до накопичених історичних даних.

Вибір оптимальної системи управління базами даних (СУБД) для IoT-застосувань залишається предметом активних досліджень [1-4]. Реляційні бази даних відзначаються своєю здатністю зберігати структуровані дані у табличних структурах з підтримкою ACID властивостей, що забезпечує високу надійність та цілісність даних. NoSQL бази даних, навпаки, оптимізовані для

високої продуктивності операцій читання/запису у розподілених середовищах та пропонують гнучкість схеми даних.

Попередні дослідження показують, що MongoDB демонструє високу продуктивність при роботі з великими обсягами даних завдяки ефективним механізмам індексування [3] та розподіленій архітектурі [4], тоді як PostgreSQL відзначається у транзакційній цілісності та можливостях складних запитів. Однак бракує комплексних досліджень, які б порівнювали ці технології у контексті реальних IoT-систем з різними типами навантаження та рівнями паралелізму, а також розглядали можливості їх спільного використання у гібридній архітектурі [5].

Метою дослідження є експериментальне порівняння продуктивності PostgreSQL та MongoDB для системи моніторингу якості повітря з визначенням оптимальних сценаріїв використання кожної технології та розробкою рекомендацій щодо архітектури баз даних для IoT-застосувань та розробкою гібридної архітектури, що поєднує переваги обох підходів для досягнення максимальної ефективності.

Аналіз досліджень. Порівняння реляційних та NoSQL баз даних є актуальною сферою сучасних досліджень. Реляційні бази даних чітко дотримуються властивостей ACID, що робить їх надійними для критичних систем, де транзакції повинні оброблятися достовірно. NoSQL бази даних, не слідуючи строгим ACID властивостям, можуть досягати вищої продуктивності для операцій запису [6].

Дослідження архітектурних відмінностей показують, що NoSQL бази даних можуть горизонтально масштабуватися через декілька вузлів, оскільки використовують дерева злиття з логарифмічною структурою, які доповнюються та використовують послідовні операції запису чи зчитування [7]. Це є значно швидшим за випадкові операції чи зчитування, що використовуються реляційними базами даних з структурами двійкових дерев.

MongoDB оптимізована для сценаріїв реального часу, де критично важливими є швидкі оновлення та гнучка схема даних, що робить її популярним вибором для IoT застосувань [8]. PostgreSQL, з іншого боку, підходить для застосувань з добре визначеними відношеннями та вимогами до складних запитів [9].

Вибір між SQL та NoSQL базами даних залежить від конкретних потреб проекту: SQL бази підходять для структурованих даних з добре визначеними відношеннями, тоді як NoSQL бази краще справляються з напівструктурованими або неструктурованими даними. Однак бракує емпіричних досліджень, які б кількісно порівнювали продуктивність цих систем у реальних IoT-сценаріях з різними рівнями навантаження та розглядали гібридні підходи до організації баз даних.

Викладення основного матеріалу. Експериментальна система побудована на базі мікроконтролера ESP32-S2 з датчиком якості повітря BME680, який вимірює температуру, вологість, атмосферний тиск та якість повітря. Серверна частина реалізована на платформі .NET 8 з використанням C# та забезпечує одночасний запис даних до PostgreSQL та MongoDB для паралельного порівняння продуктивності.

PostgreSQL налаштовано з трьома B-tree індексами: на поле timestamp для швидкої вибірки за часовим діапазоном, на deviceId для фільтрації даних конкретного датчика, та складений індекс на комбінацію обох полів. MongoDB використовує рушій зберігання WiredTiger з рівнем запису з підтвердженням (acknowledged) та аналогічними індексами на timestamp та deviceId.

Проведено серію контрольованих навантажувальних тестів з варіюванням двох параметрів: кількості записів (100, 500, 1000, 5000, 10000) та кількості паралельних потоків (1, 5, 10, 20, 50). Кожен тест включав операції INSERT та SELECT з вимірюванням часу виконання за допомогою класу Stopwatch з мікросекундною точністю. Тестові дані генерувалися програмно з реалістичними значеннями параметрів якості повітря. Загалом зібрано 301 вимірювання продуктивності для кожної бази даних. Результати тестування операцій INSERT представлено у таблиці 1. При послідовному виконанні (1 потік) MongoDB демонструє значну перевагу, яка зростає зі збільшенням обсягу даних.

Таблиця 1. Продуктивність операцій INSERT при послідовному виконанні

Кількість записів	PostgreSQL (мс)	MongoDBc (мс)	Співвідношення
100	163,88	105,61	1,55:1
500	526,55	154,16	3,42:1
1000	746,04	238,32	3,13:1
5000	10145,18	1074,24	9,44:1
10000	35837,64	2125,37	16,86:1

Найбільш суттєва різниця спостерігається при великих обсягах: для 10000 записів PostgreSQL витратив 35,8 секунди проти 2,1 секунди у MongoDB, що демонструє шістнадцятикратну перевагу NoSQL рішення.

Паралелізація операцій запису показала нелінійну залежність продуктивності від кількості потоків (таблиця 2).

Таблиця 2. Вплив паралелізації на продуктивність INSERT (для 100 записів)

Кількість потоків	PostgreSQL (мс)	MongoDB (мс)	Прискорення PostgreSQL, разів	Прискорення MongoDB, разів
1	163,88	105,61	1,00	1,00
5	22,19	16,70	7,39	6,32
10	21,84	9,04	7,50	11,68
20	10,14	5,59	16,16	18,89
50	20,14	5,96	8,14	17,72

MongoDB демонструє стабільнішу поведінку при високому рівні паралелізму, зберігаючи продуктивність 5-17 мілісекунд при 5-50 потоках, тоді як PostgreSQL показує варіативність від 10 до 22 мілісекунд.

Операції SELECT демонструють протилежну картину з переконливою перевагою PostgreSQL (таблиця 3).

Таблиця 3. Продуктивність операцій SELECT

Кількість записів	PostgreSQL (мс)	MongoDBc (мс)	Співвідношення
100	0,79	41,04	1:52
500	2,22	12,56	1:5,7
1000	2,64	10,97	1:4,2
5000	2,97	8,50	1:2,9
10000	5,55	9,52	1:1,7

Для вибірки 100 записів PostgreSQL у 52 рази швидший за MongoDB (0,79 мс проти 41,04 мс). Перевага зменшується при збільшенні обсягу, але залишається стабільною навіть для 10000 записів (в 1,7 разів швидше).

Агрегована статистика на основі 301 операції для кожної бази даних представлена у таблиці 4.

Таблиця 4. Загальна статистика продуктивності

Метрика	PostgreSQL	MongoDB	Співвідношення
Середній час INSERT (мс)	338,16	34,56	9,78:1
Мінімальний час INSERT (мс)	0,83	0,49	1,69:1
Максимальний час INSERT (мс)	37104,19	2637,81	14,07:1
Середній час SELECT (мс)	14,41	25,98	1:1,80
Мінімальний час SELECT (мс)	0,79	7,34	1:9,29
Максимальний час SELECT (мс)	182,20	53,47	3,41:1
Використання пам'яті (МБ)	40,63	44,64	0,91:1
Загальна кількість операцій	301	301	1:1

MongoDB показує майже десятикратну перевагу у середньому часі для операцій INSERT та значно меншу варіативність результатів (діапазон 5376:1 проти 44700:1 у PostgreSQL), що вказує на більш передбачувану поведінку під різними навантаженнями.

Переважна продуктивність MongoDB у операціях запису пояснюється архітектурними відмінностями [10]. PostgreSQL виконує при кожній операції вставки: перевірку цілісності даних, оновлення всіх індексів, запис у WAL журнал, керування блокуваннями для ACID властивостей [11]. MongoDB використовує LSM-tree архітектуру WiredTiger з послідовними операціями запису та блокування на рівні документа, що мінімізує конкуренцію при паралельному доступі [7].

Перевага PostgreSQL у операціях читання обумовлена ефективністю B-tree індексів для діапазонних запитів [3]. PostgreSQL виконує операцію знаходження у індексі з часовою складністю

O(log n) та послідовне читання k записів з O(k). MongoDB має додаткові накладні витрати на десеріалізацію BSON-документів [12] та перевірку умов на рівні документу.

Екстраполяція результатів на реальні сценарії показує критичність вибору бази даних при масштабуванні. Для системи з 100 датчиками, що надсилають дані кожні 5 секунд (72000 записів/годину), MongoDB забезпечує латентність близько 3 мс/запис, тоді як PostgreSQL показує 20 мс/запис. При 1000 датчиків (720000 записів/годину) перевага MongoDB стає критичною для підтримки роботи системи в реальному часі.

Результати експериментального дослідження виявили чітку спеціалізацію СУБД: MongoDB продемонструвала дев'ятикратну перевагу в пропускній здатності операцій запису, тоді як PostgreSQL показує майже двократну перевагу в швидкості виконання операцій читання (вибірки даних). Виявлена функціональна асиметрія продуктивності створює оптимальні передумови для впровадження гібридної архітектури. Вона базується на принципі функціонального розподілу ролей між різними СУБД відповідно до їх сильних сторін: MongoDB виконує функцію операційного сховища для високошвидкісного збору та обслуговування актуальних даних (період 7-14 днів), а PostgreSQL виконує роль аналітичного сховища для довгострокового зберігання та виконання складних запитів над історичними даними.

Оперативний шар на MongoDB обслуговує актуальні дані за останній тиждень або два з мінімальною латентністю завдяки LSM-tree архітектурі. Цей шар характеризується високою інтенсивністю операцій запису (до 90%), порівняно невеликим обсягом даних, високими вимогами до швидкості відгуку, та простими запитами на читання.

Аналітичний шар на PostgreSQL зберігає повну історію вимірювань та забезпечує ефективне виконання складних запитів завдяки оптимізованим B-tree індексам. Цей шар характеризується низькою інтенсивністю запису, великим обсягом накопичених даних, складними аналітичними запитами, та менш критичними вимогами до латентності.

Процес міграції даних з оперативного до аналітичного шару є дуже важливим для забезпечення цілісності системи. Базовий алгоритм міграції включає чотири основні етапи: ідентифікацію даних для міграції (вибірка записів старших за встановлений поріг з MongoDB), валідацію та трансформацію (перевірка цілісності та перетворення формату), запис до PostgreSQL (атомарна вставка даних), та видалення з MongoDB (очищення оперативного шару від перенесених даних).

Для забезпечення надійності міграції використовується механізм транзакційного журналу, який фіксує кожен етап процесу та дозволяє відновити стан системи після збою. Журнал міграції містить записи про початок процесу, список ідентифікаторів даних що мігрують, результат запису до PostgreSQL, та підтвердження видалення з MongoDB.

Така архітектура забезпечує кінцеву узгодженість даних. Для системи моніторингу якості повітря оптимальною є змішана стратегія: дані записуються лише у MongoDB при першому прийомі (мінімальна латентність), а міграція до PostgreSQL відбувається пакетно через встановлені інтервали часу (наприклад, щогодини) для даних старших за певний поріг (наприклад, 12 годин).

Оперативний шар на MongoDB налаштовується з акцентом на швидкість запису: використовується механізм зберігання даних WiredTiger з механізмом стиснення snappy, встановлюється підтверджений рівень гарантії запису (acknowledged), налаштовується TTL індекс на поле timestamp з автоматичним видаленням записів старших за 14 днів, та оптимізується розмір кешу WiredTiger (який типово встановлюється як 50 % доступної оперативної пам'яті).

Аналітичний шар на PostgreSQL конфігурується для ефективного виконання складних запитів: збільшуються параметри обсягу пам'яті для кешування сторінок даних, що найчастіше використовуються, shared_buffers (25 % доступної оперативної пам'яті), work_mem (256-512 МБ), та effective_cache_size (50 % доступної оперативної пам'яті), налаштовується autovacuum для регулярного очищення застарілих версій рядків, створюються спеціалізовані індекси (BRIN індекси для часових полів), та використовуються поділ таблиць за часовими діапазонами.

Важливою для продуктивності є оптимізація міграційного процесу. Замість індивідуального копіювання використовується пакетна обробка з розміром пакету 1000-5000 записів. Вибірка даних з бази даних MongoDB реалізується із застосуванням механізму пакетної обробки курсорів (cursor batching). У свою чергу, вставлення даних до PostgreSQL здійснюється за допомогою команди COPY або підготовлених пакетних операторів INSERT (prepared batch INSERT statements), що забезпечує максимальну швидкість інтеграції.

Результати дослідження також підкреслюють фінансову доцільність переходу до гібридного архітектурного підходу. Деталізований порівняльний аналіз сукупної вартості використання різних конфігурацій зберігання даних чітко ілюструє значні відмінності у структурі експлуатаційних витрат, що є вагомим чинником при прийнятті рішення на користь використання гібридної архітектури зберігання та опрацювання даних, отриманих від IoT-сенсорів (таблиця 5).

Таблиця 5. Економічний аналіз архітектурних варіантів (100 датчиків, 3 роки)

Архітектура	Капітальні витрати (на придбання обладнання, ліцензій, тощо), USD	Операційні витрати (на експлуатацію) за 1 рік, USD	Загальна вартість системи (3 роки), USD	Економія
PostgreSQL (моно)	23,500	14,400	66,700	Базова
MongoDB (моно)	17,500	11,400	51,700	-22%
Гібридна	15,000	10,500	46,500	-30%

Використання гібридної архітектури оптимізує розподіл ресурсів між спеціалізованими шарами: оперативний шар на MongoDB використовує компактний кластер з 2-3 серверів середньої потужності (сумарно 6000-9000 USD), аналітичний шар на PostgreSQL використовує єдиний потужний сервер (близько 5000-7000 USD), що дає сумарні капітальні витрати 11000-16000 USD. Операційні витрати суттєво зменшуються завдяки структурним перевагам гібридної архітектури, що підтверджує її економічну доцільність: автоматизована міграція даних мінімізує рутинні адміністративні задачі, скорочуючи тижневе навантаження на обслуговування до 0,5-1 години; функціональна спеціалізація систем MongoDB та PostgreSQL зменшує необхідність у складному і дороговартісному тонкому налаштуванні; незалежне масштабування дозволяє здійснювати точкові інвестиції лише у компоненти, що виступають вузькими місцями, запобігаючи надмірному використанні ресурсів; підвищена відмовостійкість, що забезпечена надмірністю двох незалежних сховищ, знижує ризик та вартість простоїв.

Отже, переваги гібридного підходу: оптимальна продуктивність для кожного типу операцій, економія близько 30 % вартості, підвищена відмовостійкість, гнучка масштабованість та ефективне використання ресурсів.

Успішне впровадження гібридної архітектури вимагає дотримання ряду кращих практик та попереднього аналізу моделі поведінки даних. Наприклад, вибір точки розподілу даних (часового порогу між оперативним та аналітичним шарами) повинен базуватися на аналізі шаблонів доступу: якщо 90% запитів звертаються до даних за останні 7 днів, оптимальний поріг становить 10-14 днів з запасом.

Стратегія індексування повинна відрізнятися між шарами: в MongoDB рекомендується мінімалістичний підхід з 2-3 індексами оскільки кожен індекс уповільнює запис, в PostgreSQL створюються спеціалізовані індекси для типових аналітичних запитів включаючи часткові індекси, BRIN-індекси для великих часових діапазонів, та GIN/GiST-індекси при необхідності.

Для забезпечення надійності та безперебійної роботи гібридної архітектури важливо неперервно моніторити дані та здійснювати вчасно оповіщення, що дасть можливість оперативно виявляти проблеми, тому необхідно:

– відстежувати технологічне відставання (lag) між системами та швидкість міграції даних (кількість записів на годину, яка має залишатися стабільною);

– контролювати розмір операційного шару (обсяг даних у MongoDB не повинен перевищувати встановлений ліміт для N днів);

– оцінювати продуктивність ключових запитів за метриками P95/P99 латентності;

– перевіряти стан реплікації (зокрема, відставання між первинним та вторинними вузлами).

Стратегія резервного копіювання для гібридної системи включає різні підходи для кожного шару: оперативний шар MongoDB частіше створює backup (кожні 6-12 годин) оскільки містить критичні свіжі дані, використовується інкрементальне резервне копіювання (incremental backup) для мінімізації впливу на продуктивність. Аналітичний шар PostgreSQL створює backup рідше (щодня або щотижня), використовується повне резервне копіювання (full backup) у поєднанні зі стисненням

для значної економії дискового простору, оскільки дані в цьому шарі змінюються набагато повільніше, ніж в оперативному.

Висновки та перспективи подальшого дослідження. Експериментальне дослідження продуктивності PostgreSQL та MongoDB у системі моніторингу якості повітря виявило значні відмінності у характеристиках обох систем. MongoDB демонструє значну перевагу в операціях запису, зафіксовану експериментально як майже десятикратна (з середньою латентністю 34,56 мс проти 338,16 мс у порівнянні з PostgreSQL). Ця висока продуктивність зумовлена її внутрішньою архітектурою – використання механізму зберігання WiredTiger на основі структури LSM-дерева забезпечує переважно послідовні операції запису даних на диск. Крім того, застосування блокування на рівні документа мінімізує конкуренцію та взаємне блокування між паралельними операціями запису, що є важливим моментом для систем високошвидкісного збору даних. PostgreSQL показує двократну перевагу при операціях читання (середній час 14,41 мс проти 25,98 мс у MongoDB) завдяки ефективності B-tree індексів для діапазонних запитів.

MongoDB є стабільнішою під різними навантаженнями з варіативністю результатів у 8,3 разів меншою порівняно з PostgreSQL, що робить її поведінку більш передбачуваною. Гібридний підхід є оптимальним рішенням для IoT-систем моніторингу, дозволяючи використати переваги обох технологій: MongoDB для операційного збору актуальних даних з високою швидкістю запису та PostgreSQL для довгострокового зберігання історичних даних з ефективними аналітичними можливостями.

Економічний аналіз показує зниження сукупної вартості володіння на 30-40% протягом трирічного періоду завдяки оптимізації використання ресурсів та зменшенню операційних витрат. Разом з тим, успіх впровадження гібридної архітектури значно залежить від кількох взаємопов'язаних речей: коректного вибору точки розподілу даних між оперативним (MongoDB) та аналітичним (PostgreSQL) шарами; ефективної реалізації автоматизованої міграції даних з гарантуванням кінцевої узгодженості; впровадження комплексного моніторингу продуктивності та стану обох шарів; спеціалізованої оптимізації конфігурації кожної бази даних під її цільове робоче навантаження; а також розробки всеосяжної технічної документації архітектурного рішення. Лише синхронна увага до цих моментів забезпечить максимальну ефективність та надійність системи.

Подальші дослідження в цій предметній області можна сфокусувати на розширенні порівняльного аналізу за рахунок тестування спеціалізованих NoSQL-баз даних для часових рядів, впровадженні фреймворків для потокової обробки даних з метою реалізації обробки в режимі реального часу. Перспективними напрямами дослідження є також використання машинного навчання для предикції навантаження на систему та масштабування архітектури до мультирегіонального рівня для обслуговування глобально розподілених IoT-систем.

Список використаних джерел:

1. Нежуренко О. Оптимізація зберігання даних IoT-пристроїв у розподілених базах даних: виклики масштабування та ефективності. *Наука і техніка сьогодні*. 2025. № 7(48). URL: https://doi.org/10.52058/2786-6025-2025-7(48)-1739-1753 (дата звернення: 15.10.2025).
2. Ma X. The analysis of the internet of things database query and optimization using deep learning network model. PLOS ONE. 2024. Vol. 19, no. 6. P. e0306291. URL: https://doi.org/10.1371/journal.pone.0306291 (дата звернення: 16.10.2025).
3. Axell, C., Schøien, E., Thon, I. L., Vågene, L. O., & Tveiten, L. Insertion speed of indexed spatial data: comparing MySQL, PostgreSQL and MongoDB. 2022. URL: https://folk.idi.ntnu.no/baf/eremcis/2022/Group02.pdf (дата звернення: 06.11.2025).
4. Al Maamari S. R. S., Nasar M. A Comparative Analysis of NoSQL and SQL Databases: Performance, Consistency, and Suitability for Modern Applications with a Focus on IoT. *East Journal of Computer Science*. 2025. Vol. 1, no. 2. P. 10–15. URL: https://doi.org/10.63496/ejcs.vol1.iss2.76 (дата звернення: 16.11.2025).
5. OUKHOUYA, Lamya, HADDADI, Anass El, ER-RAHA, Brahim, et al. Designing Hybrid Storage Architectures with RDBMS and NoSQL Systems: A Survey. *International Conference on Advanced Intelligent Systems for Sustainable Development*. Cham: Springer Nature Switzerland, 2022. p. 332-343.
6. SQL vs NoSQL: Which One is Better to Use. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/sql/sql-vs-nosql-which-one-is-better-to-use/ (дата звернення: 11.10.2025).
7. What Is a Log-Structured Merge Tree (LSM Tree)? *Aerospike*. URL: https://aerospike.com/blog/log-structured-merge-tree-explained/ (дата звернення: 08.10.2025).
8. Trung, H. Database Performance Evaluation and Applications of Data Science for IoT Platform Analysis. *International Journal Of Computer Information Systems And Industrial Management Applications*. 2021, 13 pp. 124-135.
9. Salunke S. V., Ouda A. A Performance Benchmark for the PostgreSQL and MySQL Databases. *Future Internet*. 2024. Vol. 16, no. 10. P. 382. URL: https://doi.org/10.3390/fi16100382 (дата звернення: 12.11.2025).

10. Makris, A., Tserpes, K., Spiliopoulos, G., Zissis, D., & Anagnostopoulos, D. MongoDB Vs PostgreSQL: A comparative study on performance aspects. *GeoInformatica*, 2021, *25*(2), 243-268.
11. INSERT. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/sql-insert.html (дата звернення: 03.11.2025).
12. Performance research of C# programming language data serializers using the developed software product for testing / O. Balalaieva et al. *Reporter of the Priazovskyi State Technical University. Section: Technical sciences*. 2023. No. 47. P. 8–24. URL: https://doi.org/10.31498/2225-6733.47.2023.299923 (дата звернення: 09.11.2025).

References:

1. Nezhurenko O. Optimization of Data Storage for IoT Devices in Distributed Databases: Scaling and Efficiency Challenges. Science and Technology Today. 2025, no. 7(48). URL: https://doi.org/10.52058/2786-6025-2025-7(48)-1739-1753 (date of access: 15.10.2025).
2. Ma X. The analysis of the internet of things database query and optimization using deep learning network model. PLOS ONE. 2024. Vol. 19, no. 6. P. e0306291. URL: https://doi.org/10.1371/journal.pone.0306291 (дата звернення: 16.10.2025).
3. Axell, C., Schøien, E., Thon, I. L., Vågene, L. O., & Tveiten, L. Insertion speed of indexed spatial data: comparing MySQL, PostgreSQL and MongoDB. 2022. URL: https://folk.idi.ntnu.no/baf/eremcis/2022/Group02.pdf (date of access: 06.11.2025).
4. Al Maamari S. R. S., Nasar M. A Comparative Analysis of NoSQL and SQL Databases: Performance, Consistency, and Suitability for Modern Applications with a Focus on IoT. *East Journal of Computer Science*. 2025. Vol. 1, no. 2. P. 10–15. URL: https://doi.org/10.63496/ejcs.vol1.iss2.76 (date of access: 16.11.2025).
5. OUKHOUYA, Lamya, HADDADI, Anass El, ER-RAHA, Brahim, et al. Designing Hybrid Storage Architectures with RDBMS and NoSQL Systems: A Survey. *International Conference on Advanced Intelligent Systems for Sustainable Development*. Cham: Springer Nature Switzerland, 2022. p. 332-343.
6. SQL vs NoSQL: Which One is Better to Use. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/sql/sql-vs-nosql-which-one-is-better-to-use/ (date of access: 11.10.2025).
7. What Is a Log-Structured Merge Tree (LSM Tree)? *Aerospike*. URL: https://aerospike.com/blog/log-structured-merge-tree-explained/ (date of access: 08.10.2025).
8. Trung, H. Database Performance Evaluation and Applications of Data Science for IoT Platform Analysis. *International Journal Of Computer Information Systems And Industrial Management Applications*. 2021, 13 pp. 124-135.
9. Salunke S. V., Ouda A. A Performance Benchmark for the PostgreSQL and MySQL Databases. *Future Internet*. 2024. Vol. 16, no. 10. P. 382. URL: https://doi.org/10.3390/fi16100382 (date of access: 12.11.2025).
10. Makris, A., Tserpes, K., Spiliopoulos, G., Zissis, D., & Anagnostopoulos, D. MongoDB Vs PostgreSQL: A comparative study on performance aspects. *GeoInformatica*, 2021, *25*(2), 243-268.
11. INSERT. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/sql-insert.html (date of access: 03.11.2025).
12. Performance research of C# programming language data serializers using the developed software product for testing / O. Balalaieva et al. *Reporter of the Priazovskyi State Technical University. Section: Technical sciences*. 2023. No. 47. P. 8–24. URL: https://doi.org/10.31498/2225-6733.47.2023.299923 (date of access: 09.11.2025).