УДК 004.42 DOi 10.30837/ bi.2025.1(102).03

І.В. Кириченко[1], Г.Ю. Терещенко[2], В.В. Ніколенко[3]

[1] Харківський національний університет радіоелектроніки, м. Харків, Україна, iryna.kyrychenko@nure.ua, ORCID iD: 0000-0002-7686-6439

[2] Харківський національний університет радіоелектроніки, м. Харків, Україна, hlib.tereshchenko@nure.ua, ORCID iD: 0000-0001-8731-2135

[3] Харківський національний університет радіоелектроніки, м. Харків, Україна, vitalii.nikolenko@nure.ua

ДОСЛІДЖЕННЯ МЕТОДІВ ЗБЕРІГАННЯ ВІЗУАЛЬНИХ ДАНИХ

Стаття присвячена дослідженню зберігання великих обсягів візуальних даних в програмних системах з використанням різних технологій та підходів до зберігання. В роботі проаналізовано актуальні методи зберігання даних, переваги та недоліки використання реляційних та нереляційних СКБД, наявних вбудованих технологій для роботи з даними великого розміру, MS SQL Server FILESTREAM, порівняно зі зберіганням цих даних в окремому сховищі. Для оцінки методів виконується бенчмаркінг (порівняння продуктивності) запису, читання та фільтрації даних різного розміру та обсягу та порівняння використання системних ресурсів. В теорії, використання FILESTREAM повинно було підвищити продуктивність роботи програмної системи, в той час, як за результатами дослідження при великих навантаженнях воно збільшило час запису на 110%, послідовного читання на 118% та час фільтрації на 32%, порівняно зі зберіганням зображень напряму в базі даних. Зберігання файлів напряму в базі даних підвищило фрагментацію даних та знизило ефективність індексів, що негативно вплинуло на масштабованість програмної системи та результати експериментів з великим обсягом даних. Гібридні підходи показали найкращий результат; серед них Microsoft SQL Server в середньому в 1.95 рази швидший за MongoDB, при цьому MongoDB показав меншу втрату продуктивності для запису та фільтрації з велики обсягом даних.

ASP.NET CORE, BIG DATA, ENTITY FRAMEWORK, FILESTREAM, MONGO DB, MS SQL SERVER, БЕНЧМАРКІНГ, ГІБРИДНЕ СХОВИЩЕ, ЗОБРАЖЕННЯ, НЕРЕЛЯЦІЙНІ БАЗИ ДАНИХ, РЕЛЯЦІЙНІ БАЗИ ДАНИХ, ПРОДУКТИВНІСТЬ

I.V. Kyrychenko, G.Yu. Tereshchenko, V.V. Nikolenko. Research on visual data storage methods. The article is dedicated to the study of the efficiency of various technologies and storage approaches for large volumes of visual data. Through the research, the modern data storage methods were analysed, along with the advantages and disadvantages of using relational and non-relational databases, available built-in technologies for working with larger files, such as MS SQL Server FILESTREAM, compared to storing this data in separate storage. Writing, reading and filtering benchmarks were performed with data of different sizes and volumes to determine the performance, as well as the efficiency of system resources usage. FILESTREAM was supposed to increase the performance of the system, but based on the research results, storing large-sized data increased write time by 110%, sequential read time by 118% and filtering time by 32%, compared to storing images in the database. Direct image storage in the database increased data fragmentation and reduced index efficiency, which ended up lowering system scalability and increasing experiment runtime under high load. Hybrid methods of data storage showed the best results. DBMS Microsoft SQL Server was 1.95 times on average faster than MongoDB, even though MongoDB showed less performance loss for the data writing and filtering.

ASP.NET CORE, BIG DATA, ENTITY FRAMEWORK, FILESTREAM, MONGO DB, MS SQL SERVER, BENCHMARKING, HYBRID STORAGE, IMAGES, NON-RELATIONAL DATABASE, RELATIONAL DATABASE, PERFORMANCE.

Вступ

В теперішній час спостерігається зріст обсягу візуальних даних через поширення комп'ютерного зору, медичної діагностики, автономних систем, соціальних мереж тощо. Наприклад, за 2022 в соціальній мережі Snapchat щохвилини надсилалось 2.43 мільйони «снепів» – повідомлень, що містять фото або відео [1]. За 2024 рік це число збільшилось до 3.3 мільйонів [2] Управління цими даними є критично важливим для забезпечення їхньої доступності, безпеки, масштабованості та ефективного використання в подальшому.

Загальноприйнятий метод зберігання візуальних даних – так зване, гібридне сховище, коли файли зберігаються на окремому сховищі, а в базі даних містяться лише метадані та посилання на файл на цьому сховищі [3].

Але існують окремі випадки, коли використання хмарних засобів неможливо:

– вимоги безпеки програмної системи – міститься конфіденційні дані чи інформація, яку необхідно зберігати лише на власній інфраструктурі програмної системи;

– вимоги ACID – необхідність атомарності та транзакційної цілісності при роботі з даними;

– робота в ізольованих середовищах без доступу до хмарних засобів.

Зростання обсягів візуальної інформації вимагає вдосконалення підходів до її зберігання та обробки, тому доцільним є порівняння продуктивності різних

методів зберігання великих обсягів візуальних даних для реляційних та нереляційних баз даних.

1. Опис предметної області

Об'єктом роботи є дослідження методів зберігання великого обсягу візуальних даних в реляційних та нереляційних базах даних та порівняння з загальноприйнятим методом гібридного сховища – коли дані роділено між базою даних та окремим сховищем, на якому можна зберігати саме неструктуровані дані.

Предметом дослідження є продуктивність порівняних методів та ефективність використання ними системних ресурсів, вплив кількості та розміру даних на отримані показники, що в свою чергу є показником масштабованості програмної системи – однієї з найважливіших характеристик під час роботи з великими обсягами даних.

Сервер програмного застосунку, сервера бази даних та файлове сховище було розміщено локально для порівняння різних підходів в рівних умовах. Для отримання параметрів продуктивності програмної реалізації методів було виконано бенчмарки з використанням бібліотеки BenchmarkDotNet. Отримані результати дозволили зробити висновки щодо доцільності використання тих чи інших методів зберігання великих обсягів даних.

2. Аналіз задач та методів програмної оптимізації зберігання великих обсягів візуальних даних

Зазвичай, програмні системи складаються з клієнтської частини – користувацького інтерфейсу застосунку, серверної частини, яка містить бізнес-логіку застосунку, та бази даних програмної системи.

Швидкодія застосунку залежить як від швидкодії серверної частини програмного застосунку, так і від продуктивності сервера бази даних.

Для підвищення продуктивності серверної частини застосунку використовуються наступні прийоми:

Кешування актуальних даних – зберігання даних, які були нещодавно записані, прочитані чи відредаговані, в оперативній пам'яті серверу застосунку. Таким чином забезпечується швидкий доступ до даних, які найбільш ймовірно будуть необхідні на клієнтській частині, завдяки відсутності необхідності повторного запиту до бази даних.

Вивільнення програмних ресурсів – хоч кешування й пришвидшує доступ до даних, це призводить до великих рівнів використання оперативної пам'яті. Якщо в системі не вистачатиме пам'яті, будуть виконані механізми «збирання сміття» - вивільнення найбільш неактуальних даних. Зазвичай в цьому випадку звільнюється невеликий обсяг пам'яті, внаслідок чого пам'ять знову швидко заповнюється та знову викликається збирач сміття. Щоб запобігти цьому, можна власноруч очищати існуючий кеш, що значно зменшить обсяги використання оперативної пам'яті, зменшить кількість викликів збирача сміття. Знаходження балансу між кешуванням даних та частотою вивільнення даних позитивно вплинуть на ефективність застосунку.

Пагінація (від англ. page – сторінка). Зазвичай на клієнтській частині застосунку в залежності від предметної області необхідно від 5 до 100 записів, тому доцільно отримувати та передавати інформацію з баз даних невеликими порціями – сторінками. Використання цього прийому, окрім зменшення потреб в оперативній пам'яті, також зменшить обсяги передачі даних від бази даних до серверної частини застосунку.

Ці прийоми варто застосувати для оптимізації програмного застосунку, проте їх ефективність суттєво обмежується при неоптимальному виборі СУБД та архітектури системи.

Наприклад, для підвищення продуктивності зберігання бінарних даних в базі даних MS SQL Server, Microsoft пропонують використовувати FILESTREAM для зберігання файлів розміром від 1МБ [4].

Дослідження використання FILESTREAM [5-7] для навчальної системи, в якій зберігаються різні неструктуровані файли, такі як зображення, word документи, відео тощо підтвердило ефективність використання цієї технології на невеликій кількості даних. Доцільно дослідити використання FILESTREAM з великими наборами даних та використанням актуальних архітектурних підходів до розробки програмного забезпечення та сучасних популярних бібліотек та технологій.

3. Методи зберігання великих обсягів візуальних даних

Порівняння методів зберігання даних доцільно почати з аналізу переваг та недоліків використання реляційних та нереляційних баз даних.

Реляційні бази даних не пристосовані до зберігання зображень в них. Ці бази даних були спроектовані для роботи зі структурованими даними невеликого розміру в таблицях з чіткими зв'язками [8]. Ці бази даних розраховані на вертикальне масштабування, тобто збільшення потужностей та обсягів сховища на одній машині. Неможливо нескінченно збільшувати потужність одного сервера, саме тому для зберігання великих обсягів візуальної інформації необхідно горизонтальне масштабування, яке передбачає додавання нових машин в існуючу систему. В більшості з випадків нативна підтримка горизонтального масштабування для реляційних баз даних відсутня або дуже обмежена, тому необхідно використання зовнішніх інструментів. Також не всі реляційні бази даних можуть зберігати великі дані. В базах даних,

що не підтримують (binary) large object (B)LOB, зображення необхідно зберігати в форматі base64, що ще додатково збільшує витрати часу на конвертацію даних та власне розмір самих даних.

З іншого боку є нереляційні бази даних (NoSQL), які відрізняються від реляційних підходами до зберігання, організації, обробки та масштабування даних. Вони мають більшу гнучкість зберігання даних, більш пристосовані саме для роботи з великими обсягами даних. Також вони підтримують горизонтальне масштабування [9].

Проблемою роботи з файлами є неструктурованість даних та довільний розмір цих даних. Стиснуте Full HD зображення (1920x1080) може займати до 800КБ, коли як UHD 8K зображення (7680x4320) може бути розміром до 20МБ. В реляційних базах даних таблиці даних розділено на сторінки невеликого розміру (8КБ для MS SQL Server за замовченням). Постійні запис та видалення файлів призводять до фрагментації баз даних чи дискового простору, що в свою чергу збільшує розмір файлів баз даних, зменшує продуктивність запитів та використання індексів, збільшує розсіяність даних дисковим простором.

Для вирішення цієї проблеми в Microsoft SQL Server розроблено інструмент FILESTREAM, який дозволяє зберігати неструктуровані дані у файловій системі сервера бази даних окремо від решти даних [4]. Для того, щоб зберігати великі файли в файловій системі бази даних, необхідно створити колонку бінарних даних varbinary(max) з атрибутом FILESTREAM.

```
CREATE TABLE Images (
[Id] [uniqueidentifier] ROWGUIDCOL NOT NULL UNIQUE,
[FileName] VARCHAR(50) NOT NULL,
[Content] VARBINARY(MAX) FILESTREAM NOT NULL );
```

Зміст атрибуту [Content] зберігатиметься окремим файлом на дисковому просторі серверу бази даних, а в таблиці бази даних буде міститись посилання на цей файл. При цьому, робота з таблицею з полем FILESTREAM для розробника нічим не відрізняється від звичайних таблиць

FILESTREAM використовує системний кеш NT для кешування файлів. Кешування файлів у системі дозволяє знизити вплив зберігання файлів на продуктивність сервера бази даних. Використання окремого кешу дозволяє не використовувати буферний пул SQL Server, що значить, що на запити доступно більше пам'яті бази даних.

Зберігання файлів на окремому від бази даних сховищі є універсальним вирішенням наведеної вище проблеми і для реляційних, і для нереляційних баз даних. Воно повинно знизити навантаження на сервери баз даних, зменшити фрагментацію даних та використання системних ресурсів, тим самим підвищивши продуктивність продуктивність програмної системи.

Серед нереляційних СУБД (MongoDB, Cassandra, Neo4j, Azure Cosmos DB тощо), було обрано MongoDB через її популярність, простоту роботи з нею, підтримку файлів будь-яких розмірів завдяки GridFS – технології, яка розділяє документи на невеликі частини, та власне зручність зберігання та обробки даних в ній [10].

Отже, досліджено ефективність зберігання великих обсягів візуальної інформації в MS SQL Server без та з використанням FILESTREAM, MongoDB зі зберіганням файлів в базі даних та обидві СУБД з використанням окремого сховища для файлів.

4. Аналіз методів зберігання великих обсягів візуальних даних в реляційних та нереляційних базах даних

Як предметну область для дослідження було обрано соціальні мережі. З кожним роком зростає кількість користувачів інтернету. За 2022 по 2024 рік кількість користувачів збільшилась на 500 мільйонів людей, з 5 мільярдів до 5.5, або з 63% до 68% популяції землі. І більш ніж 90% з онлайн користувачів – користувачі соціальних мереж [1, 2]. З кожним днем кількість візуальної інформації в соціальних мережах зростає, тому знаходження оптимального методу її зберігання покращить досвід кожного з користувачів соціальних мереж

Запропоновано мінімальний набор сутностей «Користувач» — «Публікація» — «Зображення», якого повинно бути достатньо щоб експериментально дослідити ефективність різних методів зберігання даних.

У випадку з MongoDB, «Зображення» буде вкладено в документ «Публікація», також метадані зображення було відокремлено в документ «Метадані», вкладений в зображення.

Для визначення ефективності методів зберігання графічної інформації в базах даних, необхідно перевірити запис, читання та фільтрацію даних для кожного з підходів з різними обсягами тестових даних.

Отже, порівняно роботу MS SQL Server без використання FILESTREAM для зберігання зображень та з використанням FILESTREAM для зображень з публікацій. MongoDB перевірено без вкладення зображень в документи публікації, з вкладенням зображень. Також для обох СУБД виконані тести з гібридним сховищем – зображення містяться на окремих сховищах, а в базах даних тільки метадані та власне посилання на ці файли.

Порівняно обсяг зайнятого дискового простору

у базах даних (та хмарних сховищах), використання оперативної пам'яті та ресурсів процесора та час виконання кожної з дій. Для проведення аналізу використались набори даних, що імітують типову активність користувачів у соціальних мережах. Візуальні дані діляться на дві категорії: невеликі файли (400–800 КБ) — стиснені зображення FullHD (1920×1080) — та великі файли (5-10 МБ) — стиснені зображення високої якості 8K UHD (7680×4320).

Кожен із цих наборів протестовано на всіх запропонованих методах, щоб визначити, як розмір файлів впливає на продуктивність баз даних. Залежність часу виконання кожної з операцій для різних обсягів даних показує, наскільки ці методи оптимізовані для зберігання великого обсягу даних.

Сервер програмної системи, сервер бази даних та сховище файлів розміщено на локальному пристрої для мінімізації різних факторів під час тестування, таких як затримка передачі даних між серверами, швидкодія серверів в залежності від апаратного забезпечення тощо. Для серверної частини застосунку використано Microsoft.EntityFrameworkCore та MongoDB.EntityFrameworkCore для експериментів з Microsoft SQL Server та MongoDB відповідно. Це дозволяє мати однакову кодову базу для експериментів з різними базами даних, що дозволить отримати ще більш об'єктивні результати щодо різних конфігурацій баз даних.

Очікувались наступні результати (від найкращого до найгіршого):

– Microsoft SQL Server з використанням FILESTREAM виявиться найшвидшим за рахунок окремого зберігання файлів від таблиць баз даних;

– Microsoft SQL Server зі зберіганням файлів на окремому сховищі буде повільнішим за використання FILESTREAM через необхідність реалізації окремого доступу до файлів. Покаже кращий результат ніж аналогічний підхід з MongoDB через швидкодію роботи зі структурованими даними;

– MongoDB з окремим сховищем;

– MongoDB зі зберіганням зображень в базі даних буде краще ніж MS SQL Server з аналогічним підходом через оптимізацію нереляційних СКБД для роботи з неструктурованими даними;

– Microsoft SQL Server зі зберіганням зображень в базі даних покаже найгірший результат через вплив фрагментації на сторінки пам'яті.

Для тестів використано ноутбук з процесором AMD Ryzen 5 5500u з 6 фізичними ядрами, 12 потоками. Базова частота: 2.1 ГГц, частота Boost: 4 ГГц. 16ГБ DDR4 3200 МГц оперативної пам'яті.

Технології використані для розробки та підтримки програмної системи та виконання експерименту:

– система Microsoft Windows 10 22H2;

– .NET Core 9.0.201, C# 13.0;

– середа розробки Microsoft Visual Studio 2022;

– СКБД Microsoft SQL Server 2022 та MongoDB;

– SQL Server Management Studio;

– MongoDB Atlas;

– ORM (Object-Relational Mapping) Microsoft Entity Framework Core;

– OpenAPI Swagger UI – інтерфейс для документації, тестування та взаємодії з API застосунку;

– BenchmarkDotNet – фреймворк для виконання тестів продуктивності програмного застосунку.

Бібліотека для бенчмаркінгу автоматично знаходить та видаляє викиди (outliers) – значення що значно відрізняються від інших спостережень.

Операції читання та запису виконуються порціями/сторінками (batch/page) по 20 записів. Загалом виконується 1000 чи 10000 операції запису в залежності від експерименту. Для читання та фільтрації операції виконуються повторно, допоки середній час виконання не стабілізується, чи не буде досягнуто ліміт кількості операцій.

Після цього розраховується середній час кожної з операції.

$$T(1) = \frac{T(N)}{N},$$

де $T(1)$ – середній час виконання однієї операції, $T(N)$ – загальний час виконання, N – кількість операцій.

Нижче наведено середній час виконання однієї операції запису, читання та фільтрації в мілісекундах.

InsertOne	1000FHD	10000FHD	1000UHD	10000UHD
MSSQLLocal	9.99	22.42	51.86	84.94
MSSQLFilestream	9.98	10.69	47.25	179.01
MSSQLFilestorage	3.21	2.86	20.41	19.38
MongoDBLocal	15.13	16.66	173.50	175.90
MongoDBFilestorage	3.68	3.20	13.98	20.88
ReadOne	1000FHD	10000FHD	1000UHD	10000UHD
MSSQLLocal	2.737	2.922	1369.51	1462.08
MSSQLFilestream	3.925	4.8206	2599.22	3192.304
MSSQLFilestorage	0.4782	1.983	0.503	4.712
MongoDBLocal	0.8455	1.392	18.62	305
MongoDBFilestorage	0.5905	1.943	7.86	18.84
FilterOne	1000FHD	10000FHD	1000UHD	10000UHD
MSSQLLocal	7.746	9.354	1472	1502
MSSQLFilestream	5.441	6.892	1569	1987.419
MSSQLFilestorage	1.506	3.592	13.47	17.76
MongoDBLocal	114.2	1968	3748	52224
MongoDBFilestorage	5.063	27.68	13.13	41.76

Рис. 1. Час викорання операцій

Де:

– 1000FHD, 10000FHD, 1000UHD, 10000UHD – експерименти з 1000 та 10000 записів FullHD та UHD відповідно;

– InsertOne – час однієї операції запису;

– ReadOne – час однієї операції читання;
– FilterOne – час однієї операції фільтрації;
– MSSQLLocal – підхід з локальним зберіганням зображень в базі даних Microsoft SQL Server;
– MSSQLFilestream – Microsoft SQL Server з використанням технології FILESTREAM;
– MSSQLFilestorage – Microsoft SQL Server зі зберіганням зображень в окремому сховищі;
– MongoDBLocal – підхід з локальним зберіганням зображень в базі даних MongoDB;
– MongoDBFilestorage – MongoDB зі зберіганням зображень в окремому сховищі.

Для підходу MSSQLLocal помітно значне зниження продуктивності зі збільшенням кількості та розміру зображень. Це пов'язано з тим, що буферний пул Microsoft SQL Server знаходиться в оперативній пам'яті. Коли розмір бази даних перебільшує обсяг вільної оперативної пам'яті, збільшується частота операцій вивільнення пам'яті та кількість звернень до жорсткого диску для дозавантаження нових сторінок пам'яті. Жоден інший підход, окрім цього, не піднімав використання оперативної пам'яті до 100%.

Неочікуванно, у випадку з використанням FILESTREAM продуктивність була ще нижчою. Відсутність кешування файлів в буферному пулі ще збільшило кількість звертань до диску з боку серверу бази даних, що ще більше підвищило час виконання операцій читання та запису. Використання цього підходу значно збільшило навантаження на сервер бази даних через накладні витрати на синхронізацію та координацію транзакцій між базою даних та файловою системою.

Хоч MongoDB і оптимізована для неструктурованих даних, робота з великими бінарними даними значно зменшує продуктивність роботи бази даних.

Методи MSSQLFilestorage та MongoDBFilestorage показали найкращі результати через відсутність проблем з фрагментацією даних. Відсутність файлів в базах даних дає можливість завантажити всю базу даних в оперативну пам'ять, що ще більше пришвидшило роботу СКБД. Також відсутні додаткові накладні витрати на транзації з боку бази даних.

Отже, методи гібрідного сховища показали найкращу швидкість роботи програмної системи.

5. Математичне моделювання та автоматизація аналізу

Для формалізації експериментальних результатів та підвищення об'єктивності порівняння різних підходів до зберігання візуальних даних було застосовано математичне моделювання основних процесів у досліджуваній системі. Це дозволяє не лише кількісно описати залежності між параметрами системи, а й автоматизувати аналіз продуктивності при зміні обсягів даних чи архітектури.

Зокрема, час виконання основних операцій (T) (запису, читання, фільтрації) моделюється як функція кількості (n) та середнього розміру (s) зображень:

$$T(n, s) = \alpha \cdot n + \beta \cdot \log(n) + \gamma \cdot s + \delta$$

де n – кількість записів, s – середній розмір одного зображення (МБ), α, β, γ, δ – емпіричні коефіцієнти, що визначаються шляхом апроксимації експериментальних даних.

Використання цієї моделі дозволяє прогнозувати час виконання операцій для різних обсягів даних без необхідності проведення додаткових експериментів. Для автоматизації аналізу модель реалізовано у вигляді функцій у програмному забезпеченні (наприклад, мовою C#).

```
public double PredictExecutionTime(int n, double s,
double alpha, double beta, double gamma, double delta)
{
  return alpha * n + beta * Math.Log(n) + gamma * s + delta;
}
```

Або у вигляді збереженої процедури в MS SQL Server.

```
CREATE PROCEDURE PredictExecutionTime
   @n INT, @s FLOAT, @alpha FLOAT, @beta FLOAT,
@gamma FLOAT, @delta FLOAT
AS
BEGIN
   SELECT @alpha * @n + @beta * LOG(@n) + @
gamma * @s + @delta AS PredictedTime
END
```

Параметри моделі (α, β, γ, δ) зберігаються у відповідній таблиці бази даних та можуть бути оновлені за результатами нових експериментів. Це забезпечує гнучкість та адаптивність моделі до змін у конфігурації системи чи апаратного забезпечення.

Застосування математичних моделей у поєднанні з програмною реалізацією забезпечує автоматизований аналіз продуктивності та дозволяє оперативно приймати рішення щодо оптимізації архітектури системи зберігання візуальних даних.

6. Масштабованість методів зберігання візуальних даних

Масштабованість – здатність системи справлятися зі збільшенням навантаження. Чим нижче втрата продуктивності методу зберігання даних, тим краще система працює під навантаженням.

Було розраховано втрату продуктивності для всіх методів зберігання даних для виконаних експериментів.

$$\eta = \frac{T_1}{T_2},$$

де η – втрата продуктивності, T_1 та T_2 середній час виконання експериментів.

Розраховано втрату продуктивності запису, читання та фільтрації для наступних змін

– 1000FHD to 10000FHD – збільшення кількості записів FullHD файлів;

– 1000FHD to 1000UHD – збільшення розміру зображення з FullHD до 8K UltraHD;

– 1000UHD to 10000UHD – збільшення кількості записів 8K UltraHD файлів;

– 1000FHD to 10000UHD – загальна втрата продуктивності зі збільшенням розміру та кількості файлів.

При записі даних MSSQLFilestream показав відносно непогану стійкість до збільшення розміру файлів з FullHD до 8K UHD, але при цьому продуктивність також падала й від кількості операцій (рис. 2).

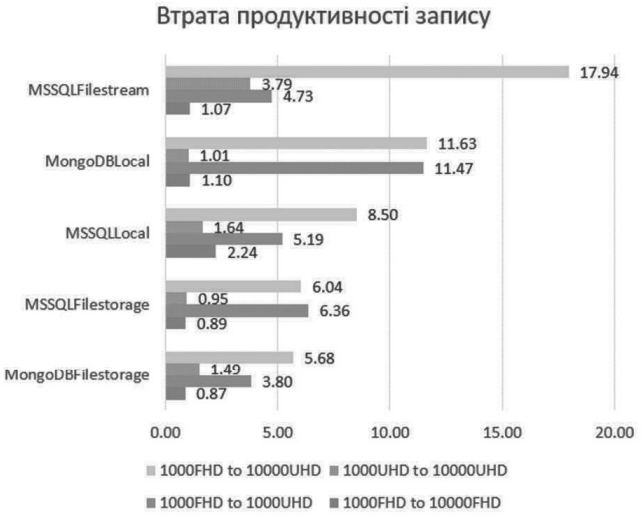

Рис. 2. Втрата продуктивності запису даних

У випадку читання даних, у всіх методах окрім MSSQLFilestorage продуктивність більше знизилась через збільшення розміру файлу. Для MSSQLFilestorage розмір файлу майже не вплинув на час виконання операцій читання. При цьому основний вплив мала саме кількість записів, що свідчить про те, що при ще більшому обсязі даних MongoDBFilestorage може показати кращі результати (рис. 3).

Результати тестування фільтрації даних показують слабкість MongoDB до зберігання великих обсягів неструктурованих даних, де продуктивність операцій сильно знижувалась як від кількості, так і від обсягу файлів. В цьому випадку MongoDBLocal показав навіть гіршу масштабованість, ніж MSSQLFilestream (рис. 4).

Рис. 3. Втрата продуктивності читання даних

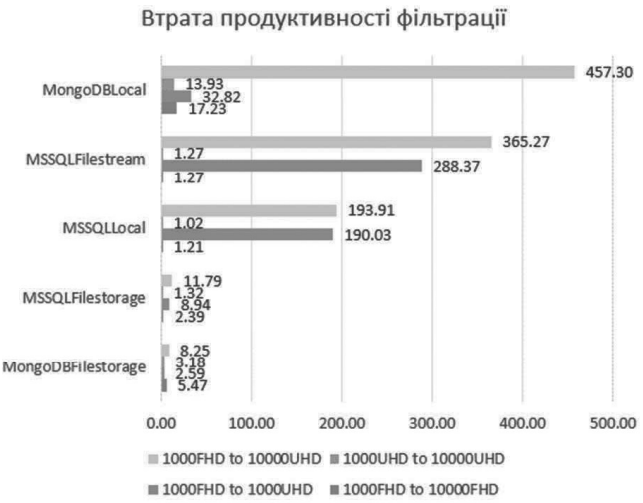

Рис. 4. Втрата продуктивості фільтрації даних

З отриманих результатів можна побачити, що і реляційна СКБД MS SQL Server, і нереляційна MongoDB, показали низький рівень масштабованості при зберіганні великих обсягів візуальних даних прямо в базі даних.

З урахуванням базового часу операції до 5 мс для гібридних методів сховища, сповільнення в 8 разів у гіршому випадку є прийнятним з урахуванням збільшення обсягу приблизно в 123 рази (Загальний обсяг даних в тесті 1000FHD був 580 МБ, 577 МБ з яких складали файли. Для тесту 10000 UHD розмір даних – 71500 МБ).

Отже, з урахуванням масштабованості, гібридне сховище є оптимальним підходом для зберігання великого обсягу візуальних даних.

Висновки

Швидкодія роботи є одним з найважливіших параметрів програмної системи. На продуктивність можуть вплинути як програмна реалізація, так і архітектурні рішення, зроблені під час проектування

застосунку. Останні є значно складнішими для виправлення, потребують більше часу розробників.

І реляційні, і нереляційні бази даних не пристосовані для зберігання великих файлів. Це підвищує фрагментованість даних, що в свою чергу знижчує продуктивність баз даних, ефективність індексів, що неприпустимо для систем, які працюють з великими обсягами даних.

Використання Microsoft SQL Server FILESTREAM, технології, розробленої для роботи з файлами в базі даних, виявилось найгіршим з усіх протестованих методів з урахуванням часу виконання операцій та масштабованості. Цей метод втратив як і загальні переваги баз даних, такі як кешування актуальних даних в оперативній пам'яті, швидкодію індексів, так і переваги окремого зберігання файлів через накладні витрати ресурсів бази даних для роботи з цими файлами.

Зберігання файлів прямо в базах даних MS SQL Server та MongoDB показало відносно прийнятні результати для невеликого обсягу даних, але фрагментація даних та неефективність індексів негативно вплинули на масштабованість таких методів у випадку з більшим обсягом даних.

Отже, гібридне сховище виявилось оптимальним варіантом для програмних систем. Реалізація такого методу дозволяє ефективно зберігати файли окремо, що в свою чергу дозволяє розкрити потенціал як реляційних баз даних, коли необхідна висока продуктивність бази даних, присутні складні зв'язки між різними сутностями чи критично важливо підтримування ACID (atomicity, consistency, isolation, durability) властивостей, так і нереляційних баз даних, коли необхідна гнучкість та горизонтальна масштабованість програмної системи.

Таким чином, вибір оптимальної конфігурації програмної системи залежить від предметної області застосунку, від кількості та розміру неструктурованих, які необхідно зберігати. Доцільно провести додаткові дослідження цих двох гібридних підходів з різними предметними областями та побудовами баз даних.

Список використаної літератури:

[1] Data never sleeps. URL: https://www.domo.com/data-never-sleeps

[2] Data never sleeps 12. URL: https://www.domo.com/learn/infographic/data-never-sleeps-12

[3] Калашников П., Кириченко І. Гібридне сховище даних для оптимізації та покращення обробки медіафайлів. VI Всеукраїнська студентська наукова конференція «Експериментальні та теоретичні дослідження в контексті сучасної науки», 21 червня 2024 р., м. Рівне, 2024. С. 156–158. DOI: 10.62732/liga-ukr-21.06.2024

[4] FILESTREAM (SQL Server). Microsoft. URL: https://learn.microsoft.com/en-us/sql/relational-databases/blob/filestream-sql-server

[5] Mhereeg, M., & Tawil, A. G. (2015). Analysis and Design of a Filestream Based English Language Learning System. Journal of Computers, 10(4), pp. 268–283.

[6] Tawil, A., & Mhereeg, M. (2015). The Implementation of a Filestream based English Language Learning System. In The International Technology Management Conference (ITMC2015), p. 39.

[7] Mhereeg, M., Tawil, A., & Belghet. (2015) The Results and Evaluation of the Filestream Based English Language Learning System. International Journal of Computing, Communications & Instrumentation Engineering (IJCCIE), Vol. 2, Issue 1 (2015). ISSN 2349-1469, EISSN 2349-1477.

[8] Aditya. Why SQL databases are more vertically scalable than horizontally scalable. Medium. URL: https://aditya003-ay.medium.com/why-sql-databases-are-more-vertically-scalable-than-horizontally-scalable-ef3a3f5d5f05

[9] Horizontal vs. vertical scaling basics. MongoDB. URL: https://www.mongodb.com/resources/basics/horizontal-vs-vertical-scaling

[10] Advantages Of MongoDB. MongoDB. URL: https://www.mongodb.com/resources/compare/advantages-of-mongodb/

Надійшла до редколегії 21.02.2025