Лабораторна робота №2

Бази даних NoSQL

“NoSQL”

Виконав студент групи ІПС-31

Міцкевич Костянтин Олександрович

Київ — 2025

Види NoSQL баз

Key–Value (Redis, Riak) - зберігають дані у вигляді пари ключ–значення. Дуже швидкі, підходять для кешування.

Документні (MongoDB, CouchDB) - дані зберігаються у вигляді JSON-подібних документів. Підходять для гнучких структур, які часто змінюються.

Колонкові (Cassandra, HBase) - оптимізовані для великих обсягів даних і аналітики.

Графові (Neo4j, ArangoDB) - для роботи зі складними зв’язками між сутностями — соцмережі, рекомендації.

У другій лабораторній я скористаюсь MongoDB, оскільки вже працював з нею і маю досвід з її використання. Для виконання 6 завдання я скористаюсь ...

Доцільність винесення частини даних у NoSQL

Проаналізувавши свою базу даних, я прийшов до висновку, що на даному етапі нема потреби виносити щось у MongoDB нема потреби, оскільки дані в моїй базі даних це структуровані транзакційні дані, які потребують:

- цілісності
- строгих зв’язків
- транзакційності
- уніфікованих структур

А дляля таких даних реляційна БД є найбільш оптимальним варіантом.

Доцільне використання MongoDB

Щоб використання NoSQL було виправданим, додаємо 3–4 сутності з даними, які:

- мають змінну структуру
- можуть бути дуже великими
- не потребують транзакційності
- часто читаються, але рідко змінюються

Додамо наступні таблиці:Recommendations, Product Reviews, Product Metadata, ActivityLog

Ці ж таблиці будуть реалізовані в SQL для перевірки швидкодії

Перевірка швидкодії:

Тестування продуктивності виконувалось за однаковим сценарієм для обох баз даних (PostgreSQL та MongoDB).

Мета: порівняти швидкодію реляційного та документного підходів для однакових структурованих та напівструктурованих даних.

Методика тестування була реалізована наступна:

Для кожної тестової сутності (Product Metadata, Product Reviews, Activity Log, Recommendations):

1) Було згенеровано однакові 500 записів

Використовувався єдиний генератор, тому SQL і Mongo отримували ідентичні дані.

Дані містили test_id і test_batch_id, щоб легко проводити вибірки і видалення.

2) Виконувалися однакові операції:

- INSERT (поелементне створення 500 записів)
- SELECT (10 випадково обраних елементів)
- UPDATE (оновлення 10 випадкових записів)
- AGGREGATION (GROUP BY / aggregation pipeline)
- JOIN або еквівалент у Mongo (lookup / embedded)
- DELETE (видалення всієї batch-сесії)

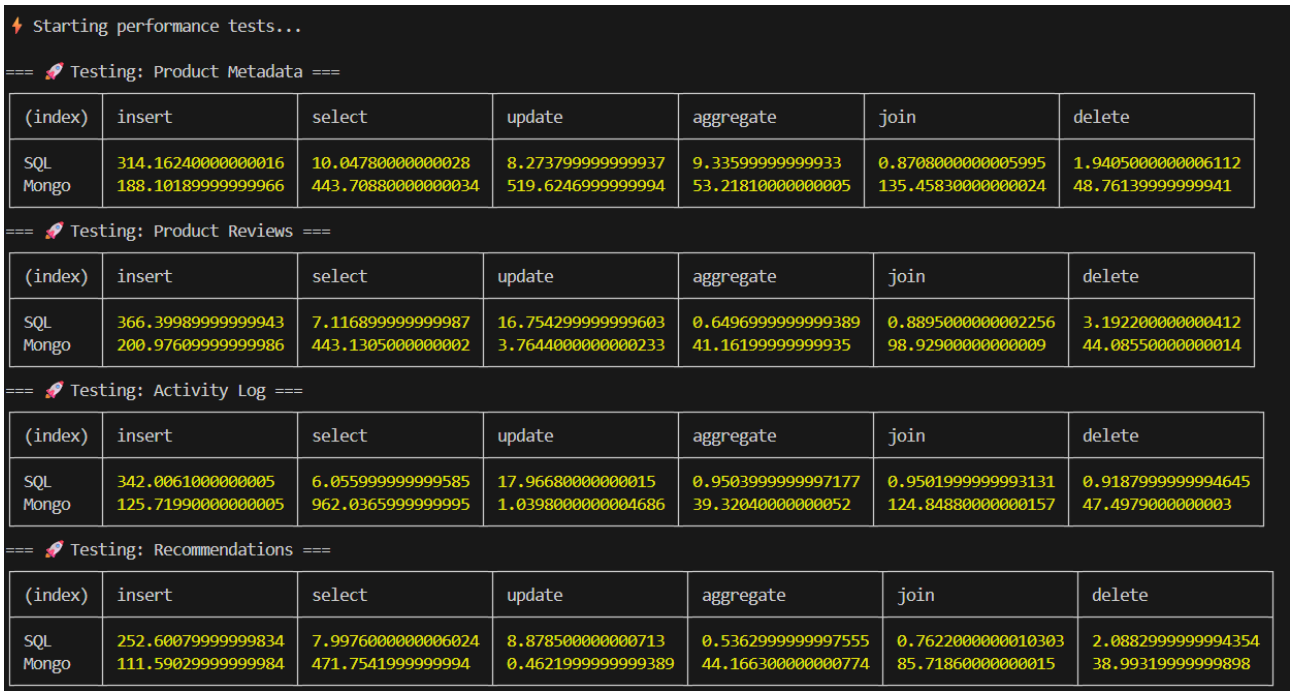
3) Вимірювання

Час кожної операції вимірювався за допомогою performance.now()

Підсумки виводилися у консоль та поверталися у JSON.

Таким чином, обидві БД працювали з однаковим навантаженням і однаковим об’ємом даних, що забезпечує коректність порівняння.

Результати:



Starting performance tests...

=== 🚀 Testing: Product Metadata ===

(index)	insert	select	update	aggregate	join	delete
SQL	314.16240000000016	10.04780000000028	8.273799999999937	9.33599999999933	0.8708000000005995	1.9405000000006112
Mongo	188.10189999999966	443.70880000000034	519.6246999999994	53.21810000000005	135.45830000000024	48.76139999999941

=== 🚀 Testing: Product Reviews ===

(index)	insert	select	update	aggregate	join	delete
SQL	366.39989999999943	7.116899999999987	16.754299999999603	0.6496999999999389	0.8895000000002256	3.192200000000412
Mongo	200.97609999999986	443.1305000000002	3.7644000000000233	41.16199999999935	98.92900000000009	44.08550000000014

=== 🚀 Testing: Activity Log ===

(index)	insert	select	update	aggregate	join	delete
SQL	342.0061000000005	6.055999999999585	17.966800000000015	0.9503999999997177	0.9501999999993131	0.9187999999994645
Mongo	125.71990000000005	962.0365999999995	1.0398000000004686	39.32040000000052	124.84880000000157	47.4979000000003

=== 🚀 Testing: Recommendations ===

(index)	insert	select	update	aggregate	join	delete
SQL	252.60079999999834	7.9976000000006024	8.878500000000713	0.5362999999997555	0.7622000000010303	2.0882999999994354
Mongo	111.59029999999984	471.7541999999994	0.4621999999999389	44.166300000000774	85.71860000000015	38.99319999999898

Підіб’ємо підсумок стосовно результатів:

INSERT

MongoDB майже у всіх випадках швидша за PostgreSQL. Очкуваний результат, оскільки MongoDB має менше накладних витрат на транзакційність та структуру даних.

SELECT

MongoDB значно повільніший близько.
Причина: у тестах виконувався пошук по неіндексованому полю test_id. SQL же має B-tree індекси за замовчуванням навіть для JSONB, що набагато швидше.

UPDATE

Mongo працює швидше у 2 із 4 випадків
(в Activity Log та Recommendations).
У складніших структурах SQL працює стабільніше.

AGGREGATION

SQL значно швидший.
PostgreSQL оптимізований під аналітичні GROUP BY, а MongoDB aggregation pipeline менш ефективний на великих наборах даних.

JOIN

SQL працює набагато швидше.
MongoDB через $lookup працює на порядок повільніше.

DELETE

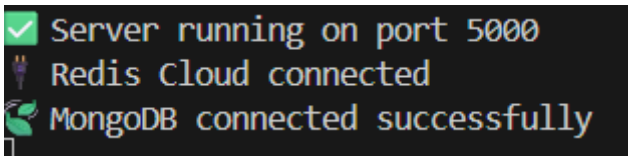
SQL видаляє швидше завдяки ефективній роботі з індексами.
Mongo видаляє також швидко, але повільніше через характер колекційної структури.

Перевірка швидкодії:

Для останньої частини роботи я обрав Redis, який належить до типу Key–Value NoSQL баз даних. Redis працює як високошвидкісне сховище пар "ключ–значення" і застосовується для кешування, лічильників, сесій, а також для роботи з даними, що не вимагають складної структури або транзакційних зв’язків.

У рамках роботи я:

1. Додав Redis до свого backend-проєкту та успішно підключився до нього через хмарний сервіс Redis Cloud, що дозволило працювати з базою даних без локального встановлення.



2. Реалізував базові операції (створення, отримання, оновлення та видалення) для двох сутностей: Product Metadata Recommendations