МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ

Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

освітній ступінь - «Бакалавр»

на тему: **РОЗРОБКА ПРОТОТИПУ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБРОБКИ ДАНИХ ПРО ОБЛІК СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ НА ОСНОВІ НЕРЕЛЯЦІЙНОЇ БАЗИ ДАНИХ**

Виконав: студент спеціальності

126 «Інформаційні системи та технології»

Харченко Єгор Володимирович

Керівник: ***Братушка Сергій Миколайович***

к.ф.-м.н., доцент, кафедри кібернетики та інформатики Сумського НАУ

Рецензент: ***Гриценко Костянтин Григорович***

доцент кафедри економічної кібернетики ННІ бізнесу, економіки та менеджменту Сумського державного університету, к.т.н., доцент

Суми – 2025

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет Економіки і менеджменту

Кафедра Кібернетики та інформатики

Освітній ступінь «Бакалавр»

Спеціальність *126 «Інформаційні системи та технології»*

ОП «Інформаційні системи та технології»

Затверджую:

Завідувач кафедри Світлана АГАДЖАНОВА

«_____» _____ 202___р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Харченко Єгору Володимировичу

1. **Тема роботи: Розробка прототипу інформаційної системи обробки даних про облік сільськогосподарської продукції на основі нереляційної бази даних.**
2. **База практичного дослідження:** ПАТ «Сумський завод продтоварів», м. Суми

керівник роботи Братушка Сергій Миколайович, к.ф.-м.н., доцент кафедри кібернетики та інформатики

затверджена наказом по університету від « » №

3. **Строк подання студентом закінченого проекту (роботи)** « » червня 2025 року
4. **Вихідні дані до проекту (роботи):**

Результати аналітичних досліджень зарубіжних та вітчизняних вчених, інтернет-джерела, нормативні документи за чинним законодавством.

5. **Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):**

- *виконати постановку задачі;*
- *вибрати середовище для реалізації;*
- *визначити вимого до системи обліку сільськогосподарської продукції*
- *розробити структуру колекцій та документів*
- *розробити технічне завдання;*
- *створити нереляційну базу даних та множину операцій з ними;*
- *розробити прототип програмного продукту відповідно до поставлених задач;*
- *провести тестування створеного прототипу та оцінити його продуктивність.*

6. Дата видачі завдання: **« » 202__ р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проекту	Примітка
1	Визначення об'єкту, предмету дослідження, формулювання мети та задач кваліфікаційної роботи, складання плану	*січень 2025 р.*	*виконано*
2	Підбір та вивчення літературних джерел, технічної документації	*лютий 2025 р.*	*виконано*
3	Розробка технічного завдання	*березень 2025 р..*	*виконано*
4	Розробка прототипу системи	*лютий-квітень 2025 р.*	*виконано*
5	Провести тестування створеного прототипу та оцінити його продуктивність	*до 10 травня 2025 р.*	*виконано*
6	Оформлення пояснювальної записки	*до 25 травня 2025 р..*	*виконано*
7	Оформлення презентації	*до 1 червня 2025 р.*	*виконано*
8	Опрацювання зауважень керівника, перевірка на автентичність	*4 червня 2025 р.*	*виконано*
9	Перевірка з академічної доброчесності кваліфікаційної роботи	*9 червня 2025 р.*	*виконано*
10	Підготовка доповіді та її попередній захист	*9 червня 2025 р.*	*виконано*
11	Представлення кваліфікаційної роботи на кафедру, деканат	*13 червня 2025 р.*	*виконано*
12	Захист кваліфікаційної роботи	*27 червня 2025р.*	

Студент ***Єгор ХАРЧЕНКО***

(підпис)

Керівник роботи ***Сергій БРАТУШКА***

(підпис)

Перевірку на автентичність проведено

(підпис)

Кваліфікаційна робота допущена до захисту

(підпис)

АНОТАЦІЯ

Харченко Є.В. Розробка прототипу інформаційної системи обробки даних про облік сільськогосподарської продукції на основі нереляційної бази даних.

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології Сумського національного аграрного університету. Суми, 2025 р. – Рукопис.

Кваліфікаційна робота присвячена розробці прототипу інформаційної системи обробки даних про облік сільськогосподарської продукції на основі нереляційних баз даних. У роботі розглянуто основні методи та засоби автоматизації процесів збору, зберігання та аналізу облікової інформації, проаналізовано сучасні інформаційні системи та технології, які використовуються у сфері обліку. Прототип системи обліку розроблено для забезпечення зручного та ефективного інтерфейсу користувача, а також автоматизованого ведення обліку складських запасів та витрат.

Об’єктом дослідження є різнородні системи управління базами даних великих обсягів, як складова частина розподіленої системи.

Предметом дослідження - процеси обробки запитів різної складності у різнорідних типах баз даних

Метою роботи є створення ефективного прототипу обліку сільськогосподарської продукції, який дозволяє зручно збирати, обробляти та аналізувати дані про складські запаси та замовлення.

У процесі роботи проведено аналіз предметної області, існуючі підходи щодо проектування та створення інформаційної підсистеми, наведено етапи реалізації та приклади програмного коду. Після розробки прототипу системи обліку проведено його тестування та виконане оцінювання його продуктивності, а татакож демонстрацію роботи основних функцій системи.

Результатом роботи є створення інформаційної підсистеми для обліку та аналізу замовлень, обліку та запасів, що знаходяться на складському обліку.

Ключові слова: інформаційна система, нереляційні бази даних, повнотекстовий пошук, MongoDB.

SUMMARY

Kharchenko E.V. Development of a prototype of an information system for processing data on accounting of agricultural products based on a non-relational database.

Qualification work in specialty 126 “Information Systems and Technologies” of the Information Systems and Technologies Department of the Sumy National Agrarian University. Sumy, 2025 – Manuscript.

The qualification work is devoted to the development of a prototype of an information system for processing data on accounting of agricultural products based on non-relational databases. The work considers the main methods and means of automating the processes of collecting, storing and analyzing accounting information, analyzes modern information systems and technologies used in the field of accounting. The prototype of the accounting system is designed to provide a convenient and effective user interface, as well as automated accounting of warehouse stocks and costs.

The object of research is heterogeneous large-scale database management systems as a component of a distributed system.

The subject of research is the processes of processing queries of varying complexity in various types of databases.

The goal of the work is to create an effective prototype for accounting for agricultural products, which allows you to conveniently collect, process, and analyze data on warehouse stocks and orders.

In the process of work, an analysis of the subject area, existing approaches to the design and creation of an information subsystem, implementation stages and examples of program code are provided. After developing a prototype of the accounting system, its testing was carried out and its performance was evaluated, as well as a demonstration of the operation of the main functions of the system.

The result of the work is the creation of an information subsystem for accounting and analysis of orders, accounting and inventories held in warehouse records.

Keywords:information system, non-relational databases, full-text search, MongoDB.

ЗМІСТ

ВСТУП.....12
1. ТЕОРЕТИЧНІ ОСНОВИ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ.....15
1.1. Особливості та переваги нереляційних баз даних.....15
1.2. Класифікація нереляційних баз даних.....18
1.3. Актуальність MongoDB для великих обсягів даних.....21
1.4. Характеристика MongoDB для обліку продукції.....23
1.5. Висновки до розділу.....24
2. ПРОЕКТУВАННЯ СТРУКТУРИ БАЗИ ДАНИХ.....26
2.1. Аналіз вимог до системи обліку.....26
2.2. Моделювання даних у MongoDB.....27
2.3. Структура колекцій і документів.....30
2.4. Забезпечення масштабованості даних.....33
2.5. Висновки до розділу.....34
3. РЕАЛІЗАЦІЯ ПРОТОТИПУ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....36
3.1. Налаштування MongoDB і створення бази.....36
3.2. Реалізація операцій з даними.....38
3.3. Обробка складних запитів.....41
3.4. Тестування та оцінка продуктивності.....44
3.5. Висновки до розділу.....47
ВИСНОВКИ.....48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....50
ДОДАТКИ.....52

ВСТУП

Сучасний розвиток інформаційних технологій характеризується стрімким зростанням обсягів даних, що потребують ефективної обробки, зберігання та аналізу. У сільськогосподарській галузі, яка є однією з ключових для економіки України, облік продукції відіграє важливу роль у забезпеченні прозорості, оптимізації ресурсів та підвищенні продуктивності. Традиційні реляційні бази даних, хоча й широко використовуються, мають обмеження в обробці великих обсягів різнорідних даних, що характерно для сільськогосподарських систем. У цьому контексті нереляційні бази даних, зокрема документо-орієнтовані, відкривають нові можливості для створення гнучких і масштабований інформаційних систем.

Актуальність теми зумовлена необхідністю розробки сучасних інформаційних систем, здатних ефективно обробляти великі обсяги даних про сільськогосподарську продукцію, включаючи її облік, рух, зберігання та аналіз. Нереляційні бази даних, такі як MongoDB, пропонують гнучкість у моделюванні даних, швидкість обробки запитів і можливість масштабування, що робить їх перспективним інструментом для вирішення завдань обліку в аграрному секторі. Впровадження таких систем може сприяти автоматизації процесів, зниженню витрат і підвищенню конкурентоспроможності сільськогосподарських підприємств.

Мета роботи полягає у розробці прототипу інформаційної системи для обробки даних про облік сільськогосподарської продукції на основі нереляційної бази даних MongoDB, що забезпечить ефективне зберігання, управління та аналіз даних.

Об’єкт дослідження – різнорідні системи управління базами даних великих обсягів як складова частина розподілених систем.

Предмет дослідження – процеси обробки запитів різної складності у різнорідних типах баз даних, зокрема в нереляційних базах даних для обліку сільськогосподарської продукції.

Завдання дослідження:

1. - виконати постановку задачі;
2. - вибрати середовище для реалізації;
3. - визначити вимого до системи обліку сільськогосподарської продукції
4. - розробити структуру колекцій та документів
5. - розробити технічне завдання;
6. - створити нереляційну базу даних та множину операцій з ними;
7. - розробити прототип програмного продукту відповідно до поставлених задач;
8. - провести тестування створеного прототипу та оцінити його продуктивність.

Методи дослідження включають аналіз літературних джерел, порівняльний аналіз реляційних і нереляційних баз даних, моделювання даних, програмування, тестування та оцінку продуктивності системи.

Наукова новизна роботи полягає у розробці прототипу інформаційної системи на основі MongoDB, адаптованої до потреб обліку сільськогосподарської продукції, з урахуванням гнучкого моделювання даних і обробки складних запитів.

Практична цінність роботи полягає в можливості використання розробленого прототипу як основи для створення повноцінних інформаційних систем для сільськогосподарських підприємств. Система може бути застосована для автоматизації обліку продукції, оптимізації складських операцій і підготовки аналітичних звітів.

Апробація результатів кваліфікаційної роботи. Результати кваліфікаційної роботи апробовані на конференціях:

1) Всеукраїнської наукової конференції студентів та аспірантів, присвяченої міжнародному дню студента (18-22 листопада 2024 р.), Суми;

2) XVI Міжнародної науково-практичної конференції «Правова наука і державотворення в Україні у контексті інтеграційних процесів», Сумська філія ХНУВС, 22-23 травня 2025 року

Публікації. За матеріалами кваліфікаційної роботи опубліковано 2 тези наукових конференцій.

Структура роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел з 47 найменування, додатків. Основний текст викладений на 63 сторінках комп'ютерного тексту, робота містить 5 таблиць, 16 рисунків, 3 додатки.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ

1.1. Особливості та переваги нереляційних баз даних

Нереляційні бази даних (NoSQL) з’явилися як відповідь на обмеження традиційних реляційних баз даних у контексті обробки великих обсягів різнорідних даних. Вони розроблені для забезпечення гнучкості, масштабованості та високої продуктивності в умовах сучасних інформаційних систем [1]. На відміну від реляційних баз даних, які використовують фіксовані табличні структури та мову SQL, нереляційні бази даних підтримують різноманітні моделі даних, такі як документо-орієнтовані, ключ-значення, стовпцеві чи графові. Це дозволяє адаптувати їх до специфічних потреб різних галузей, включаючи сільське господарство, де облік продукції вимагає гнучкого підходу до управління даними.

Однією з ключових особливостей нереляційних баз даних є їх здатність працювати з неструктурованими або напівструктурованими даними. Наприклад, у сільськогосподарському обліку дані про продукцію можуть включати текстові описи, числові показники, геолокаційні дані чи навіть мультимедійний контент. Нереляційні бази даних дозволяють зберігати такі дані без необхідності попереднього визначення жорсткої схеми, що значно спрощує розробку інформаційних систем.[2] Крім того, вони підтримують горизонтальне масштабування, що означає можливість додавання нових серверів для обробки зростаючих обсягів даних, на відміну від вертикального масштабування реляційних баз, яке обмежене апаратними ресурсами одного сервера.

Ще однією важливою перевагою є висока продуктивність при обробці великих обсягів даних. Нереляційні бази даних оптимізовані для швидкого виконання запитів, особливо в розподілених системах, де дані розподілені між кількома вузлами. Це особливо актуально для сільськогосподарських систем, де необхідно швидко отримувати звіти про стан складів, рух продукції чи

аналітичні дані для прийняття рішень. [3] Наприклад, документо-орієнтовані бази, такі як MongoDB, дозволяють зберігати складні ієрархічні структури в одному документі, що зменшує кількість операцій під час читання даних порівняно з реляційними базами, де потрібні множинні з'єднання таблиць.

Нереляційні бази даних також забезпечують гнучкість у розробці додатків. Завдяки відсутності жорсткої схеми розробники можуть швидко адаптувати структуру даних до нових вимог без значних змін у базі. У контексті сільськогосподарського обліку це дозволяє, наприклад, додавати нові типи продукції чи атрибути (такі як сертифікати якості чи дані про постачальників) без перебудови всієї системи. [4] Крім того, NoSQL бази часто підтримують автоматичну реплікацію даних, що підвищує їх надійність і доступність, що є критично важливим для систем, які працюють у реальному часі.

Недоліки нереляційних баз даних також варто враховувати. Наприклад, вони можуть поступатися реляційним базам у забезпеченні цілісності даних через відсутність строгих транзакційних механізмів. Проте сучасні NoSQL бази, такі як MongoDB, пропонують інструменти для часткового вирішення цієї проблеми, наприклад, підтримку транзакцій у межах одного документа чи репліки. [5] У сільськогосподарському обліку, де транзакції між таблицями менш критичні, ніж швидкість і гнучкість, ці недоліки менш значущі. Більш детально різницю між реляційними та нереляційними базами показано у Таблиці 1.1.

Нереляційні бази даних пропонують значні переваги для систем, що працюють із великими обсягами різнорідних даних, таких як облік сільськогосподарської продукції. Їх гнучкість, продуктивність і масштабність роблять їх ідеальним вибором для сучасних інформаційних систем.

Серед практичних застосувань нереляційних баз даних у сільському господарстві можна навести приклади автоматизованих систем управління теплицями, фермами з великою кількістю сенсорів, а також платформ контролю логістики продукції. Наприклад, аграрні стартапи в Нідерландах і США використовують MongoDB для обробки показників вологості, температури,

хімічного складу ґрунту з тисяч сенсорів у реальному часі. Гнучка модель даних дозволяє швидко додавати нові параметри без порушення роботи системи.

Таблиця 1.1 - Порівняння реляційних і нереляційних баз даних

Характеристика	Реляційні бази даних	Нереляційні бази даних
Структура даних	Таблиці з фіксованою схемою	Різноманітні моделі (документи, ключ-значення тощо)
Масштабування	Вертикальне	Горизонтальне
Обробка неструктурованих даних	Обмежена	Висока
Швидкість запитів	Залежить від складності з’єднань	Висока для великих обсягів
Гнучкість схеми	Низька	Висока
Транзакційна цілісність	Висока	Обмежена, але вдосконалюється

Джерело: запропоноване автором

Також NoSQL-рішення мають перевагу при побудові хмарних інфраструктур. Наприклад, при використанні MongoDB Atlas можна автоматизувати масштабування, резервне копіювання та моніторинг продуктивності без потреби в ручному адмініструванні серверів. Це дозволяє скоротити витрати на IT-обслуговування та забезпечити стабільну роботу систем навіть у пікові періоди, наприклад під час збору врожаю або формування великої кількості звітів.

MongoDB також підтримує індексацію по кількох полях, агрегаційний фреймворк для побудови звітів, реплікацію та шардінг — що дає змогу створювати масштабовані рішення для агропідприємств будь-якого розміру. Крім того, база має потужні можливості для геопросторових запитів, що корисно у задачах контролю полів, моніторингу земельних ділянок або оптимізації маршрутів транспорту.

Завдяки цьому нереляційні бази є ідеальним технологічним ядром для побудови адаптивних, масштабованих та інтерактивних систем у сфері агробізнесу.

1.2. Класифікація нереляційних баз даних

Нереляційні бази даних (NoSQL) охоплюють широкий спектр технологій, що відрізняються за моделями зберігання, структурою даних і принципами обробки. Їх класифікація є ключовим етапом у виборі оптимального рішення для конкретних галузевих задач, включно з агропромисловим комплексом, де особливо актуальними є питання гнучкості, продуктивності й масштабованості. [6]

У загальному випадку NoSQL-системи поділяють на чотири основні типи:

1. документо-орієнтовані бази;
2. бази типу ключ-значення;
3. стовпцеві бази;
4. графові бази.

Документо-орієнтовані бази даних є найбільш універсальними у світі NoSQL. Дані в них зберігаються у вигляді документів (найчастіше у форматах JSON або BSON), кожен з яких містить набір атрибутів, вкладених структур та масивів. MongoDB, Couchbase, OrientDB — приклади таких систем. Їхньою ключовою перевагою є можливість зберігати гнучку, ієрархічну структуру без необхідності створювати попередню фіксовану схему, як у реляційних БД.

У сфері агровиробництва документо-орієнтовані бази мають перевагу в обліку сільськогосподарської продукції, яка характеризується великою варіативністю параметрів: від базових характеристик (тип, вага, врожайність) до додаткових (дані про сертифікацію, аналіз якості, GPS-координати тощо). Наприклад, система, що веде облік партій зерна, може зберігати в одному

документі як дату збирання, так і вологість, умови зберігання та постачальника, не створюючи складних пов’язаних таблиць. Це значно спрощує розробку й прискорює доступ до даних. [7]

Бази даних типу «ключ-значення» зберігають інформацію у вигляді пар «ключ–значення». Вони характеризуються високою швидкодією, особливо у випадках, коли важливою є швидка обробка однотипних запитів. Прикладами таких баз є Redis, Amazon DynamoDB, Riak.

У сільському господарстві їх можна використовувати для:

- кешування результатів запитів аналітики;
- обліку сеансів користувача в агросистемах з веб-доступом;
- зберігання проміжних значень (наприклад, тимчасових цін на продукти залежно від ринку або сезону).

Хоча ці бази мають обмежену функціональність у порівнянні з документними або графовими, їх ефективність у читанні та записі робить їх незамінними у високонавантажених підсистемах. [8]

Стовпцеві бази даних (wide-column stores) орієнтовані на обробку великих обсягів даних у колонковому форматі. Основні представники — Apache Cassandra, HBase, ScyllaDB. Вони ідеально підходять для аналітичних задач, де часто обробляються великі таблиці з десятками мільйонів записів, у яких вибірка відбувається за окремими стовпцями.

У сільському господарстві такі бази можна ефективно застосовувати для побудови систем прогнозування урожайності, оцінки ефективності обробки полів, аналізу кліматичних факторів. Наприклад, можна зберігати щоденні значення температури, вологості, опадів у форматі стовпців і здійснювати швидкий аналіз впливу цих параметрів на врожай. [9] Проте складність у проєктуванні схеми даних обмежує їх широке використання в простих системах обліку продукції.

Графові бази даних (graph databases) призначені для моделювання складних взаємозв’язків. У них дані представляються як вузли (об’єкти) і ребра (зв’язки). Найпопулярніші системи — Neo4j, ArangoDB, Amazon Neptune.

У контексті агросфери їх застосування актуальне в завданнях:

- моделювання ланцюгів постачання продукції від виробника до кінцевого споживача;
- контролю взаємодії з постачальниками, логістичними компаніями та споживачами;
- виявлення “вузьких місць” у ланцюгах аграрної логістики.

Такі бази забезпечують високу ефективність у запитах на типу «показати шлях між фермою і споживачем», але є надлишковими для базових задач складування та щоденного обліку. [10]

Узагальнення

Кожен тип NoSQL баз має свої переваги залежно від завдання. Більш докладно різницю між ними можемо побачити у Таблиці 1.2.

Таблиця 1.2 - Класифікація нереляційних баз даних

Тип бази даних	Приклади	Основні особливості	Застосування в обліку продукції
Документо-орієнтовані	MongoDB, CouchDB	Гнучка структура, JSON/BSON документи	Облік продукції з різними атрибутами, аналітика
Ключ-значення	Redis, DynamoDB	Швидкий пошук за ключем, простота	Кешування цін, тимчасові дані
Стовпцеві	Cassandra, HBase	Висока продуктивність для аналітики	Аналіз великих обсягів історичних даних
Графові	Neo4j, ArangoDB	Робота зі складними зв’язками	Відстеження ланцюгів постачання

Джерело: запропоноване автором

Для систем, що потребують зберігання складної, змінної структури даних — безперечним лідером є документо-орієнтовані бази, зокрема MongoDB. Для задач кешування чи високошвидкісної роботи з однотипними даними — бази ключ-значення. Для аналітичної обробки великих масивів — стовпцеві сховища, а для складних зв'язків — графові рішення.

Вибір конкретної технології має базуватись на аналізі вимог до гнучкості, швидкодії, обсягу даних та типів запитів. У межах даної кваліфікаційної роботи оптимальним вибором стала MongoDB — як найбільш збалансоване рішення для агрообліку, що поєднує продуктивність, простоту реалізації та потужний функціонал.

Для обліку сільськогосподарської продукції документо-орієнтовані бази, зокрема MongoDB, є оптимальними завдяки їх гнучкості та підтримці складних запитів. Інші типи баз можуть використовуватися як допоміжні інструменти залежно від специфіки системи.

1.3. Актуальність MongoDB для великих обсягів даних

MongoDB є однією з найпопулярніших нереляційних баз даних, що використовується для обробки великих обсягів даних у різноманітних галузях, від електронної комерції до сільського господарства. [11] Її актуальність для сучасних інформаційних систем зумовлена здатністю ефективно працювати з великими масивами різнорідних даних, що є характерним для обліку сільськогосподарської продукції. MongoDB відноситься до документо-орієнтованих баз даних, що робить її ідеальним вибором для систем, які потребують гнучкості та масштабованості.

Одним із ключових факторів актуальності MongoDB є її підтримка горизонтального масштабування через механізм шардингу. Шардинг дозволяє розподіляти дані між кількома серверами, що забезпечує обробку великої кількості запитів навіть при зростанні обсягів даних. У сільськогосподарському

секторі, де облік може включати дані про тисячі одиниць продукції, складів і транзакцій, ця особливість дозволяє підтримувати високу продуктивність системи. [12] Наприклад, фермерське господарство може швидко отримувати звіти про залишки продукції на різних складах без значних затримок.

MongoDB також підтримує гнучке моделювання даних, що є важливим для систем із динамічними вимогами. У сільськогосподарському обліку дані про продукцію можуть варіюватися залежно від типу (зернові, овочі, фрукти), сезону чи вимог постачальників. MongoDB дозволяє зберігати ці дані у вигляді документів без необхідності попереднього визначення схеми, що спрощує адаптацію системи до нових умов. [13] Крім того, підтримка індексів і агрегаційних запитів забезпечує швидкий доступ до даних і можливість створення складних аналітичних звітів, наприклад, про обсяги виробництва чи постачання.

Ще однією причиною актуальності MongoDB є її здатність працювати в розподілених системах із високою доступністю. Завдяки механізму реплікації дані автоматично синхронізуються між кількома вузлами, що гарантує доступність системи навіть у разі збою одного з серверів. Для сільськогосподарських підприємств, які часто працюють у віддалених регіонах із нестабільним інтернет-з’єднанням, це забезпечує надійність роботи системи. [14] Наприклад, облік руху продукції між складами може продовжуватися без перерв.

MongoDB також підтримує інтеграцію з сучасними технологіями, такими як хмарні платформи та інструменти аналітики великих даних. Це дозволяє створювати комплексні інформаційні системи, які поєднують облік із прогнозуванням чи аналізом ринкових тенденцій. У сільському господарстві такі можливості можуть бути використані для оптимізації логістики чи планування виробництва. [15] Таким чином, MongoDB відповідає сучасним вимогам до обробки великих обсягів даних і є актуальним інструментом для розробки інформаційних систем.

Серед сучасних IT-рішень в аграрному секторі слід виділити системи типу Cropio, AgroOnline, FieldView, які інтегрують GPS-навігацію, супутниковий моніторинг, облік врожайності та планування операцій. Проте більшість із них є комерційними, складними у впровадженні й часто не адаптовані до дрібних фермерських господарств.

Застосування гнучкої бази даних MongoDB у таких рішеннях дає змогу забезпечити високу масштабованість, простоту інтеграції та ефективну роботу з напівструктурованими даними. Це особливо актуально в умовах, коли аграрні підприємства мають справу з нестандартними форматами (різні типи культур, різні типи операцій тощо).

Розроблений у даній кваліфікаційній роботі прототип враховує ці особливості та може бути адаптований до різних масштабів господарювання.

1.4. Характеристика MongoDB для обліку продукції

MongoDB як документо-орієнтована нереляційна база даних має низку характеристик, що роблять її ефективним інструментом для обліку сільськогосподарської продукції. Її архітектура та функціональні можливості відповідають вимогам систем, що працюють із різнорідними даними та потребують гнучкості й високої продуктивності. [16] У контексті сільськогосподарського обліку MongoDB забезпечує зручне зберігання, обробку та аналіз даних про продукцію, складські операції та постачання.

Першою важливою характеристикою є документо-орієнтована модель даних. У MongoDB дані зберігаються у вигляді документів у форматі BSON (бінарний JSON), що дозволяє представляти складні структури, такі як інформація про продукцію (назва, тип, вага, постачальник, сертифікати), в одному об’єкті. Це усуває необхідність множинних з’єднань, як у реляційних базах, і підвищує швидкість обробки запитів. Наприклад, запит про стан

конкретного виду продукції на складі виконується швидше, оскільки всі дані містяться в одному документі. [17]

Другою характеристикою є підтримка гнучкої схеми. У сільськогосподарському обліку вимоги до даних можуть змінюватися, наприклад, при додаванні нових типів продукції чи атрибутів, таких як органічна сертифікація. MongoDB дозволяє додавати нові поля до документів без модифікації всієї бази, що спрощує адаптацію системи до нових умов. Це особливо корисно для невеликих фермерських господарств, які можуть поступово розширювати облік. [18]

Третьою характеристикою є потужна система запитів і агрегації. MongoDB підтримує складні запити, включаючи фільтрацію, сортування та групування, що необхідно для створення аналітичних звітів. Наприклад, система може швидко підрахувати загальну вагу зернових на всіх складах або проаналізувати постачання за певний період. Фреймворк агрегації дозволяє виконувати операції, подібні до SQL, але з більшою гнучкістю для неструктурованих даних. [19]

Четвертою характеристикою є підтримка масштабування та високої доступності. MongoDB використовує шардинг для розподілу даних і реплікацію для забезпечення відмовостійкості. У сільськогосподарському обліку, де дані можуть накопичуватися з різних джерел (ферми, склади, постачальники), ці можливості дозволяють створювати надійні системи, що працюють у реальному часі навіть у розподілених мережах. [20] Наприклад, інформація про рух продукції може синхронізуватися між регіональними складами.

MongoDB забезпечує гнучке моделювання даних, швидку обробку запитів і надійність, що робить її оптимальним вибором для систем обліку сільськогосподарської продукції. Її характеристики дозволяють створювати ефективні інформаційні системи, адаптовані до потреб аграрного сектору.

1.5. Висновки до розділу

Проведений аналіз теоретичних основ організації нереляційних баз даних показав їх значний потенціал для сучасних інформаційних систем, зокрема для обліку сільськогосподарської продукції. Нереляційні бази даних вирізняються гнучкістю, масштабністю та високою продуктивністю, що робить їх ефективними для роботи з великими обсягами різнорідних даних. [21] Їх основні переваги включають підтримку неструктурованих даних, горизонтальне масштабування та швидку обробку запитів, що є критично важливим для систем, які потребують швидкого доступу до інформації.

Класифікація нереляційних баз даних показала, що документо-орієнтовані бази, такі як MongoDB, є оптимальними для завдань, пов’язаних із гнучким моделюванням даних і аналітикою. Інші типи, такі як ключ-значення чи графові бази, можуть використовуватися як допоміжні інструменти для специфічних задач, але менш універсальні для обліку продукції. [22] MongoDB вирізняється своєю актуальністю для великих обсягів даних завдяки підтримці шардингу, реплікації та гнучкої схеми, що дозволяє обробляти зростаючі масиви даних без втрати продуктивності.

Характеристики MongoDB, зокрема документо-орієнтована модель, потужна система запитів і висока доступність, роблять її ідеальним інструментом для створення інформаційних систем обліку сільськогосподарської продукції. Вона забезпечує швидке виконання аналітичних запитів, адаптацію до змінних вимог і надійність у розподілених системах. [23] Таким чином, MongoDB є обґрунтованим вибором для розробки прототипу інформаційної системи, що відповідає потребам аграрного сектору.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СТРУКТУРИ БАЗИ ДАНИХ

2.1. Аналіз вимог до системи обліку сільськогосподарської продукції

Розробка інформаційної системи для обліку сільськогосподарської продукції потребує детального аналізу вимог, які визначають функціональність, структуру та продуктивність системи. Сільськогосподарський облік охоплює широкий спектр даних, включаючи інформацію про продукцію, складські операції, постачальників і транзакції, що вимагає створення гнучкої та ефективної системи [24]. Аналіз вимог дозволяє визначити ключові сутності, операції та обмеження, які необхідно врахувати при проектуванні бази даних. Вимоги для системи зображено у Таблиці 2.1.

Таблиця 2.1 - Основні вимоги до системи обліку продукції

Вимога	Опис	Значення для системи
Різноманітність даних	Підтримка різних типів продукції та атрибутів	Гнучка структура даних
Облік складських операцій	Фіксація надходжень, відвантажень, залишків	Швидкий доступ до даних операцій
Аналітичні можливості	Створення звітів і аналіз даних	Підтримка агрегаційних запитів
Висока доступність	Робота в розподілених мережах	Реплікація та масштабування

Джерело: запропоноване автором

Першою вимогою є підтримка різноманітних типів сільськогосподарської продукції. Система повинна дозволяти зберігати дані про різні категорії (зернові, овочі, фрукти, молочні продукти тощо) з відповідними атрибутами, такими як назва, вага, об’єм, термін придатності та сертифікати якості. Ці атрибути можуть

варіюватися залежно від типу продукції, що вимагає гнучкої структури бази даних, здатної адаптуватися до змін. [25] Наприклад, для зернових може бути важливим клас якості, тоді як для молочних продуктів – дата виробництва.

Другою вимогою є облік складських операцій. Система повинна фіксувати надходження, відвантаження та переміщення продукції між складами, а також поточні залишки. Для цього необхідно зберігати дані про склади (назва, адреса, місткість) та операції (дата, тип, кількість). Важливо забезпечити швидкий доступ до цих даних для створення звітів, наприклад, про стан запасів на певну дату. [26] Крім того, система повинна підтримувати відстеження походження продукції, що є важливим для відповідності стандартам якості.

Третьою вимогою є аналітичні можливості. Система повинна надавати інструменти для аналізу даних, таких як загальний обсяг виробництва, постачання за період чи залишки за категоріями. Це вимагає підтримки складних запитів, включаючи агрегацію та фільтрацію. Наприклад, фермерське господарство може потребувати звіт про постачання зернових за квартал для планування продажів. [27] Для цього база даних повинна забезпечувати швидке виконання аналітичних операцій.

Четвертою вимогою є висока доступність і продуктивність. Сільськогосподарські підприємства часто працюють у розподілених мережах, де дані надходять із різних джерел (ферми, склади, постачальники). Система повинна гарантувати доступність даних навіть у разі збоїв і підтримувати обробку великої кількості запитів у реальному часі. Це особливо важливо для великих господарств із тисячами одиниць продукції. [28]

Аналіз вимог показав, що система обліку сільськогосподарської продукції повинна бути гнучкою, продуктивною та надійною, щоб відповідати потребам аграрного сектору.

2.2. Моделювання даних у MongoDB

Моделювання даних у MongoDB є ключовим етапом проектування інформаційної системи для обліку сільськогосподарської продукції. Як документо-орієнтована база даних, MongoDB дозволяє створювати гнучкі структури даних, що відповідають вимогам системи, визначеним на етапі аналізу. [29] Моделювання включає визначення колекцій, документів і їх атрибутів, а також врахування принципів нормалізації та денормалізації для забезпечення ефективності.

Першим кроком є визначення основних колекцій. На основі вимог до системи обліку можна виділити три ключові колекції: «Продукція», «Склади» і «Операції». Колекція «Продукція» містить інформацію про сільськогосподарську продукцію, включаючи назву, тип, вагу, термін придатності та сертифікати. Колекція «Склади» зберігає дані про склади (назва, адреса, місткість), а колекція «Операції» фіксує надходження, відвантаження та переміщення продукції. [25] Такий поділ дозволяє організувати дані логічно та ефективно.

Другим кроком є проектування структури документів. У MongoDB документи є аналогами записів у реляційних базах, але підтримують вкладені структури. Наприклад, документ у колекції «Продукція» може виглядати так:

```
{
  "_id": "product_001",
  "name": "Пшениця",
  "type": "Зернові",
  "weight": 1000,
  "unit": "кг",
  "quality_class": "1",
  "certificates": [
    {"name": "Органічна", "issue_date": "2025-01-15"}
  ],
  "supplier": {"id": "sup_001", "name": "Ферма Зоря"}
}
```

Ця структура дозволяє зберігати всі атрибути продукції в одному документі, що зменшує кількість запитів до бази. [19] Аналогічно, документ у колекції «Склади» може містити інформацію про поточні залишки, а в «Операції» – деталі транзакцій.

Третім кроком є вибір між нормалізацією та денормалізацією. У MongoDB частіше використовується денормалізація, коли пов’язані дані (наприклад, інформація про постачальника) включаються безпосередньо в документ. Це підвищує швидкість читання, але може ускладнити оновлення даних. Для обліку продукції денормалізація є виправданою, оскільки запити на читання (наприклад, звіти про залишки) виконуються частіше, ніж оновлення. [17] Однак для часто змінюваних даних, таких як постачальники, можна використовувати нормалізацію з посиланнями на окрему колекцію.

Четвертим кроком є створення індексів для оптимізації запитів. Наприклад, індекс на полі *type* у колекції «Продукція» прискорить пошук за типом продукції, а індекс на полі *date* у колекції «Операції» – аналіз транзакцій за період. MongoDB підтримує різні типи індексів, включаючи складові та текстові, що забезпечує гнучкість у роботі з даними. [8] Для сільськогосподарського обліку це дозволяє швидко отримувати аналітичні звіти, наприклад, про постачання за місяць.

Моделювання даних у MongoDB для обліку сільськогосподарської продукції передбачає створення логічно пов’язаних колекцій із гнучкими документами, оптимізованими для швидкого читання та аналізу. Використання денормалізації та індексів забезпечує високу продуктивність системи.

Програмна архітектура побудована за принципом клієнт-серверної взаємодії з використанням REST API. Серверна частина реалізована на Node.js, що забезпечує швидку обробку запитів та асинхронну роботу з базою даних MongoDB.

База даних має гнучку схему, де основні колекції включають:

- `products` — зберігає інформацію про продукцію (назва, тип, вага, склад);

- `warehouses` — дані про склади (місцезнаходження, ємність);
- `operations` — облік надходжень, списань та аналітичних подій.

MongoDB обрана як основа завдяки її здатності масштабуватись горизонтально, а також обробляти документоорієнтовані записи без жорсткої схеми, що актуально для аграрного сектору з великою варіативністю даних.

Також було реалізовано логіку валідації введених даних на стороні сервера, що підвищує надійність збереженої інформації.

2.3. Структура колекцій і документів

Структура колекцій і документів у MongoDB є основою для ефективного зберігання та обробки даних в інформаційній системі обліку сільськогосподарської продукції. Колекції в MongoDB є аналогами таблиць у реляційних базах даних, але вони не мають фіксованої схеми, що дозволяє гнучко адаптувати структуру до потреб системи. Документи, які зберігаються в колекціях, є окремими записами у форматі BSON, що забезпечує підтримку складних ієрархічних даних. Для системи обліку сільськогосподарської продукції розроблено три основні колекції: «Продукція», «Склади» та «Операції».

Структура колекції «Продукція» приведена у Таблиці 2.2, а сама колекція призначена для зберігання інформації про сільськогосподарську продукцію. Кожен документ у цій колекції містить детальні атрибути, такі як назва, тип, вага, одиниця виміру, клас якості, сертифікати та інформація про постачальника. Використання вкладених структур дозволяє зберігати пов’язані дані, наприклад, масив сертифікатів, у межах одного документа, що зменшує кількість запитів до бази. Наприклад, документ для пшениці може включати клас якості та сертифікат органічного виробництва, що є важливим для обліку в сільському господарстві.

Структура колекції «Склади» приведена у Таблиці 2.3, а сама колекція містить інформацію про склади, де зберігається продукція. Документи в цій колекції включають назву складу, адресу, місткість і поточні залишки продукції. Залишки представлені як вкладений масив, де вказується ідентифікатор продукту та кількість, що дозволяє швидко отримувати інформацію про доступну продукцію на конкретному складі. Такий підхід спрощує відстеження запасів і підготовку звітів про стан складів.

Структура колекції «Операції», приведена у Таблиці 2.4. Колекція фіксує всі транзакції, пов’язані з рухом продукції, такі як надходження, відвантаження чи переміщення між складами. Кожен документ містить ідентифікатор операції, дату, тип операції, ідентифікатор продукту, кількість, а також посилання на склад і, за необхідності, постачальника. Використання денормалізації, коли частина даних про продукцію чи склад дублюється в документі, підвищує швидкість читання даних. Наприклад, назва продукту може бути включена в документ операції для швидкого доступу без додаткових запитів.

Для забезпечення ефективної роботи системи використано індекси. Наприклад, у колекції «Продукція» створено індекс на полі *type* для прискорення пошуку за типом продукції, а в колекції «Операції» – на полі *date* для швидкого аналізу транзакцій за період. Нижче наведено приклади структури документів для кожної колекції.

Таблиця 2.2 - Структура колекції «Продукція»

Поле	Тип даних	Опис
_id	ObjectId	Унікальний ідентифікатор продукту
name	String	Назва продукту (напр., "Пшениця")
type	String	Тип продукту (напр., "Зернові")
weight	Number	Вага продукту (кг)
unit	String	Одиниця виміру (напр., "кг")
quality_class	String	Клас якості (напр., "1")
certificates	Array	Масив сертифікатів (назва, дата видачі)
supplier	Object	Дані постачальника (id, назва)

Джерело: запропоноване автором

Таблиця 2.3 - Структура колекції «Склади»

Поле	Тип даних	Опис
_id	ObjectId	Унікальний ідентифікатор складу
name	String	Назва складу (напр., "Склад 1")
address	String	Адреса складу
capacity	Number	Місткість складу (кг)
stock	Array	Масив залишків (id продукту, кількість)

Джерело: запропоноване автором

Таблиця 2.4 - Структура колекції Операції

Поле	Тип даних	Опис
_id	ObjectId	Унікальний ідентифікатор операції
date	Date	Дата операції
type	String	Тип операції (надходження, відвантаження)
product_id	ObjectId	Ідентифікатор продукту
quantity	Number	Кількість продукту (кг)

Продовження таблиці 2.4

Поле	Тип даних	Опис
warehouse_id	ObjectId	Ідентифікатор складу
product_name	String	Назва продукту (денормалізоване поле)

Джерело: запропоноване автором

Така структура забезпечує гнучкість, швидкий доступ до даних і підтримку аналітичних запитів, що є критично важливим для обліку сільськогосподарської продукції.

2.4. Забезпечення масштабованості даних

Масштабованість даних є однією з ключових вимог до інформаційних систем, що працюють із великими обсягами інформації, таких як система обліку сільськогосподарської продукції. MongoDB, як нереляційна база даних, пропонує потужні механізми для забезпечення масштабованості, що дозволяє системі обробляти зростаючі обсяги даних без втрати продуктивності. Основними підходами до масштабованості в MongoDB є шардинг, реплікація та оптимізація структури даних.

Шардинг є основним механізмом горизонтального масштабування в MongoDB. Він передбачає розподіл даних між кількома серверами (шардами), де кожен шард містить частину даних. У системі обліку сільськогосподарської продукції шардинг може бути застосовано, наприклад, за полем warehouse_id, що дозволяє розподілити дані про операції та залишки між серверами залежно від складу. Це забезпечує рівномірне навантаження та швидку обробку запитів навіть при значному збільшенні кількості транзакцій. Наприклад, велике господарство з десятками складів може ефективно обробляти дані, розподілені між шардами.

Реплікація використовується для підвищення доступності та надійності системи. У MongoDB дані копіюються на кілька вузлів (реплік), що дозволяє продовжувати роботу системи в разі збою одного з серверів. Для сільськогосподарського обліку це особливо важливо, оскільки підприємства можуть працювати в регіонах із нестабільним інтернет-з’єднанням. Реплікація також забезпечує розподіл навантаження на читання, коли запити до даних розподіляються між репліками. Наприклад, аналітичні звіти про залишки продукції можуть виконуватися на вторинних репліках, не навантажуючи основний сервер.

Оптимізація структури даних також відіграє важливу роль у масштабованості. Використання денормалізації, коли пов’язані дані (наприклад, назва продукту в колекції «Операції») включаються безпосередньо в документ, зменшує кількість запитів до бази. Однак надмірна денормалізація може збільшити обсяг даних, тому необхідно знаходити баланс. Наприклад, у системі обліку можна зберігати ідентифікатор постачальника як посилання, а не повну інформацію, щоб зменшити дублювання. Крім того, створення відповідних індексів, таких як індекс на полі *date* у колекції «Операції», прискорює виконання запитів при зростанні обсягів даних.

Ще одним аспектом є використання хмарних технологій. MongoDB Atlas, хмарна платформа MongoDB, дозволяє автоматично масштабувати ресурси залежно від навантаження. Це спрощує адміністрування системи для сільськогосподарських підприємств, які не мають великих IT-команд. Наприклад, у пікові періоди збору врожаю система може автоматично збільшувати ресурси для обробки додаткових транзакцій. Таким чином, MongoDB забезпечує гнучке масштабування, що відповідає потребам системи обліку сільськогосподарської продукції.

2.5. Висновки до розділу

Другий розділ присвячено проектуванню структури бази даних для інформаційної системи обліку сільськогосподарської продукції на основі MongoDB. Аналіз вимог показав, що система повинна підтримувати різноманітні типи даних, забезпечувати швидкий доступ до складських операцій, надавати аналітичні можливості та бути надійною в розподілених мережах. Ці вимоги визначили підхід до моделювання даних, який базується на гнучкій структурі документів і використанні денормалізації для підвищення продуктивності.

Моделювання даних передбачило створення трьох основних колекцій: «Продукція», «Склади та Операції, що відображають ключові сутності системи. Документи в цих колекціях розроблено з урахуванням потреб швидкого читання та аналітики, використовуючи вкладені структури та індекси для оптимізації запитів. Така структура дозволяє ефективно зберігати та обробляти дані про продукцію, склади та транзакції, відповідаючи вимогам гнучкості та продуктивності.

Забезпечення масштабованості досягнуто завдяки механізмам шардингу та реплікації в MongoDB, а також оптимізації структури даних. Шардинг дозволяє розподіляти дані між серверами, що забезпечує обробку великих обсягів транзакцій, а реплікація підвищує надійність і доступність системи. Використання хмарних технологій, таких як MongoDB Atlas, додатково спрощує масштабування, що є важливим для сільськогосподарських підприємств із динамічними потребами.

Розроблена структура бази даних і підходи до масштабованості створюють міцну основу для реалізації прототипу інформаційної системи, здатної ефективно обробляти дані про облік сільськогосподарської продукції.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОТОТИПУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Налаштування MongoDB і створення бази даних

Реалізація прототипу інформаційної системи для обліку сільськогосподарської продукції розпочалася з налаштування MongoDB та створення бази даних. MongoDB була обрана через її гнучкість, підтримку масштабування та здатність обробляти різнорідні дані, що відповідає вимогам системи. Процес налаштування включав встановлення MongoDB, конфігурацію сервера та створення бази даних із необхідними колекціями.

На першому етапі було встановлено MongoDB Community Edition на локальний сервер із операційною системою Windows. Встановлення проводилося через офіційний репозиторій MongoDB, що забезпечило використання актуальної версії (6.0 на момент розробки). Після встановлення було налаштовано конфігураційний файл *mongod.conf*, де визначено параметри, такі як порт (27017), директорія для зберігання даних (/var/lib/mongodb) і рівень журналювання. Для забезпечення безпеки було увімкнено автентифікацію, створивши адміністратора бази з роллю *root*.

Другим етапом було створення бази даних під назвою AgriAccounting. Для взаємодії з MongoDB використано MongoDB Compass, графічний інтерфейс для адміністрування бази, а також командний рядок MongoDB Shell. У базі, зображеній на рис. 3.1, створено три колекції: products, warehouses і operations, які відповідають структурі, розробленій у розділі 2.3.

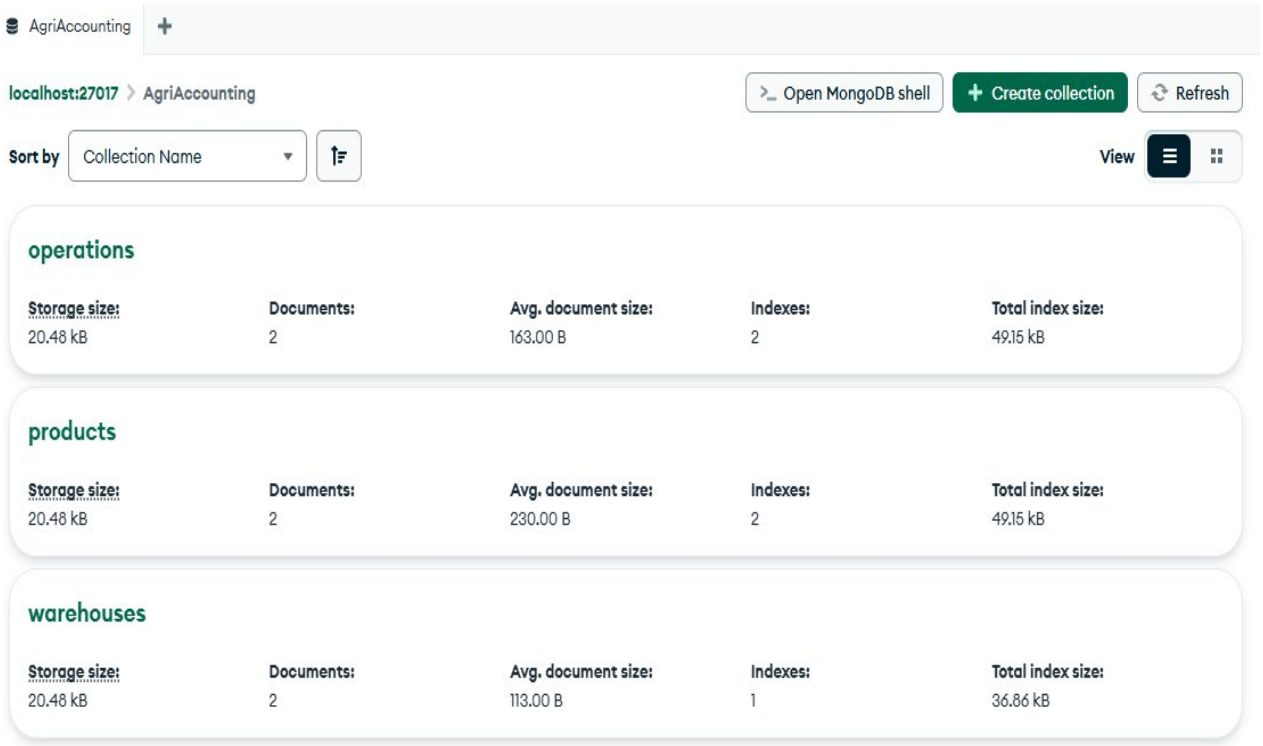

Рис. 3.1. – Створення бази та колекцій

Джерело: запропоноване автором

Кожна колекція була ініціалізована з початковими тестовими даними для перевірки коректності структури. Наприклад, у колекцію *Products* (зображена на рис. 3.2), додано документи для пшениці, картоплі з різними атрибутами.

Для оптимізації роботи бази створено індекси. У колекції products додано індекс на поле type за допомогою команди *db.products.createIndex({"type": 1})*, що прискорює пошук за типом продукції. У колекції operations створено індекс на поле *date* для швидкого доступу до транзакцій за період. Ці індекси забезпечують ефективну обробку запитів, що є важливим для аналітичних звітів.

Третім етапом було налаштування реплікації для підвищення надійності. Створено набір реплік із трьох вузлів: одного первинного та двох вторинних. Первинний вузол обробляє операції запису, тоді як вторинні використовуються для читання та резервного копіювання. Це забезпечує доступність даних у разі збою одного з серверів, що є критично важливим для сільськогосподарських систем, які працюють у реальному часі. Налаштування проводилося через команду *rs.initiate()* у MongoDB Shell із подальшою перевіркою статусу реплік.

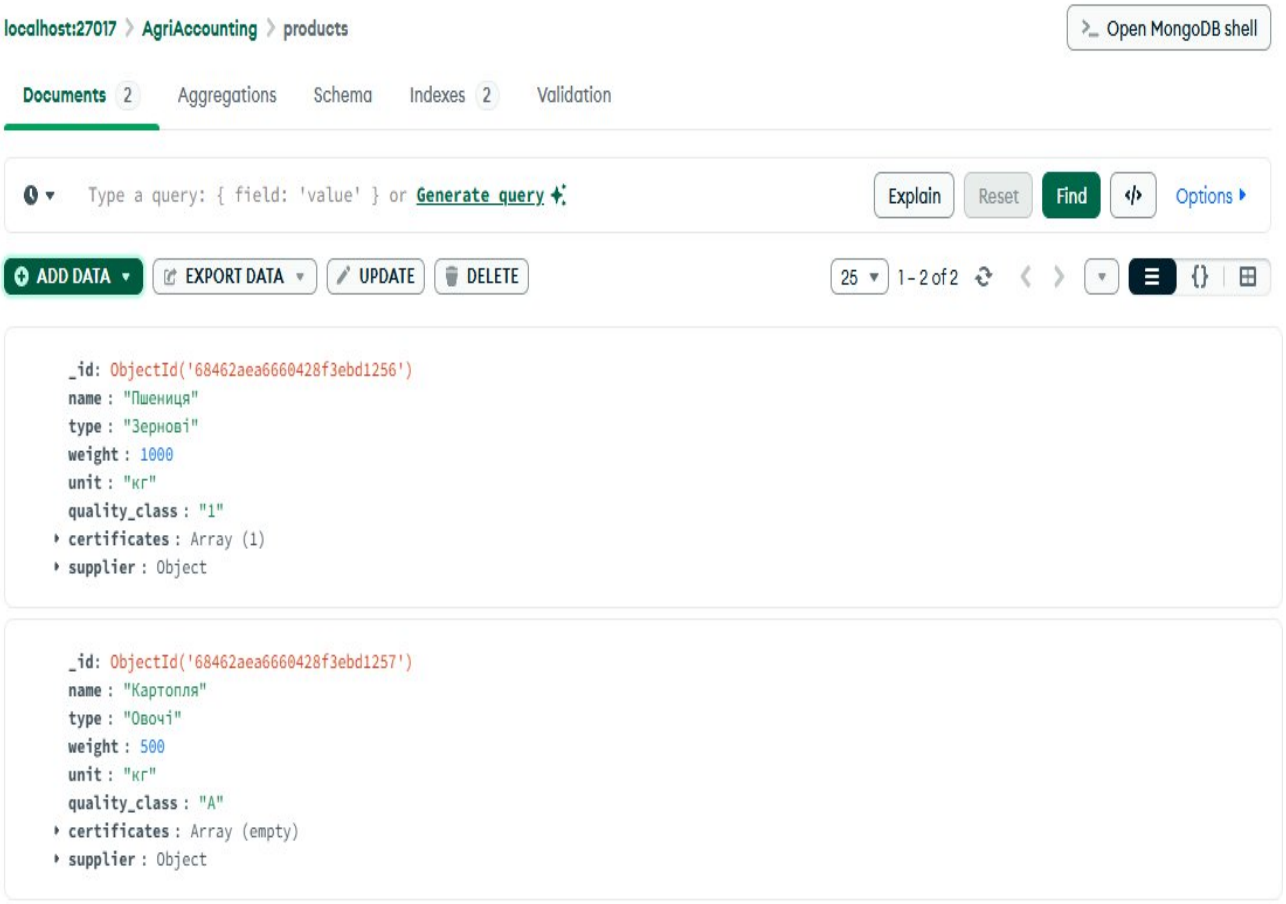

Рис 3.2 – Додавання тестових даних у колекцію «Products»

Джерело: запропоноване автором

Налаштування MongoDB і створення бази *AgriAccounting* із колекціями та індексами заклали основу для реалізації прототипу. Використання реплікації та оптимізованих індексів забезпечило надійність і продуктивність системи.

3.2. Реалізація операцій з даними

Реалізація операцій з даними є центральною частиною прототипу інформаційної системи для обліку сільськогосподарської продукції. Операції *CRUD (Create, Read, Update, Delete)* були реалізовані для роботи з колекціями «*Products*», «*Warehouses*» і «*Operations*» у базі даних MongoDB. Для цього

використано мову програмування JavaScript у середовищі *Node.js* із бібліотекою *mongodb*, що забезпечує зручну взаємодію з базою.

Операція *Create* (рис. 3.3), дозволяє додавати нові документи до колекцій. Наприклад, для додавання нового продукту використано метод *insertOne*. Код для додавання пшениці виглядає так:

```
const { MongoClient } = require('mongodb');
const url = 'mongodb://localhost:27017';
const client = new MongoClient(url);
async function addProduct() {
  try {
    await client.connect();
    const db = client.db('AgriAccounting');
    const collection = db.collection('products');
    const result = await collection.insertOne({
      name: 'Пшениця',
      type: 'Зернові',
      weight: 1000,
      unit: 'кг',
      quality_class: '1',
      certificates: [{ name: 'Органічна', issue_date:
'2025-01-15' }],
      supplier: { id: 'sup_001', name: 'Ферма Зоря' }
    });
   console.log('Продукт додано:', result.insertedId);
  } finally {
    await client.close();
  }
}
addProduct();
```

Цей код додає документ до колекції *«products»* і повертає унікальний ідентифікатор. Аналогічні операції реалізовано для колекцій *«warehouses»* і *«operations»*.

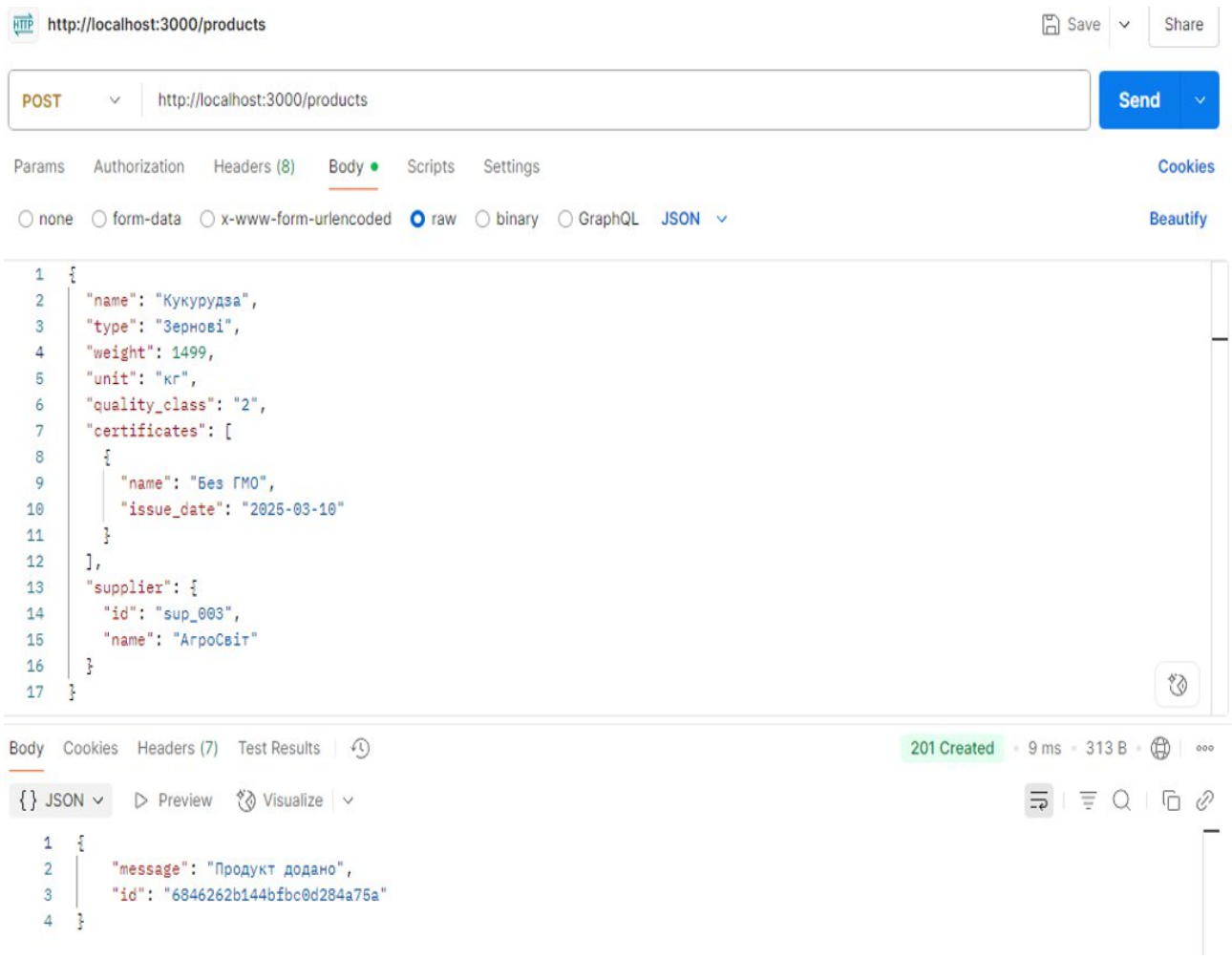

Рис. 3.3 – Додавання документу до колекції *«products»*

Джерело: запропоноване автором

Операція *Read* забезпечує отримання даних із бази. Наприклад, для пошуку всіх зернових використано метод *find* із фільтром *{ type: 'Зернові' }*. Для аналітичних потреб використано метод *aggregate*, який дозволяє групувати дані, наприклад, підрахувати загальну вагу продукції на складі. Операція читання оптимізована завдяки індексам, створеним на етапі налаштування.

Операція *Update* дозволяє змінювати існуючі документи. Наприклад, для оновлення ваги продукту використано метод *updateOne* з умовою за *_id* і

оператором *$set*. Це корисно для коригування залишків після надходження чи відвантаження. Для забезпечення цілісності використано транзакції в межах одного документа, що підтримується MongoDB починаючи з версії 4.0.

Операція *Delete* видаляє документи за заданою умовою. Наприклад, для видалення застарілої операції використано метод *deleteOne*. Ця операція використовується рідко, але необхідна для очищення бази від неактуальних даних, наприклад, тестових записів.

Реалізовані операції протестовано з тестовими даними, що підтвердило їх коректність і відповідність вимогам системи. Використання *Node.js* і бібліотеки *mongodb* забезпечило простоту інтеграції з базою та гнучкість у реалізації.

3.3. Обробка складних запитів

Обробка складних запитів є важливою частиною інформаційної системи обліку сільськогосподарської продукції, оскільки дозволяє отримувати аналітичні звіти та підтримувати прийняття рішень. У MongoDB складні запити реалізуються за допомогою фреймворку агрегації, який забезпечує групування, фільтрацію, сортування та обчислення даних. Для прототипу системи розроблено кілька типів запитів, що відповідають аналітичним вимогам.

Першим типом є запит на підрахунок загальної ваги продукції за типом. Наприклад, щоб визначити загальну вагу зернових на всіх складах, використано агрегаційний конвеєр:

```
db.products.aggregate([
  { $match: { type: 'Зернові' } },
  { $group: { _id: '$type', totalWeight: { $sum:
'$weight' } } }
]);
```

Цей запит (рис. 3.4) фільтрує продукти за типом і підраховує сумарну вагу, що корисно для оцінки запасів. Індекс на полі *type* забезпечує швидке виконання запиту.

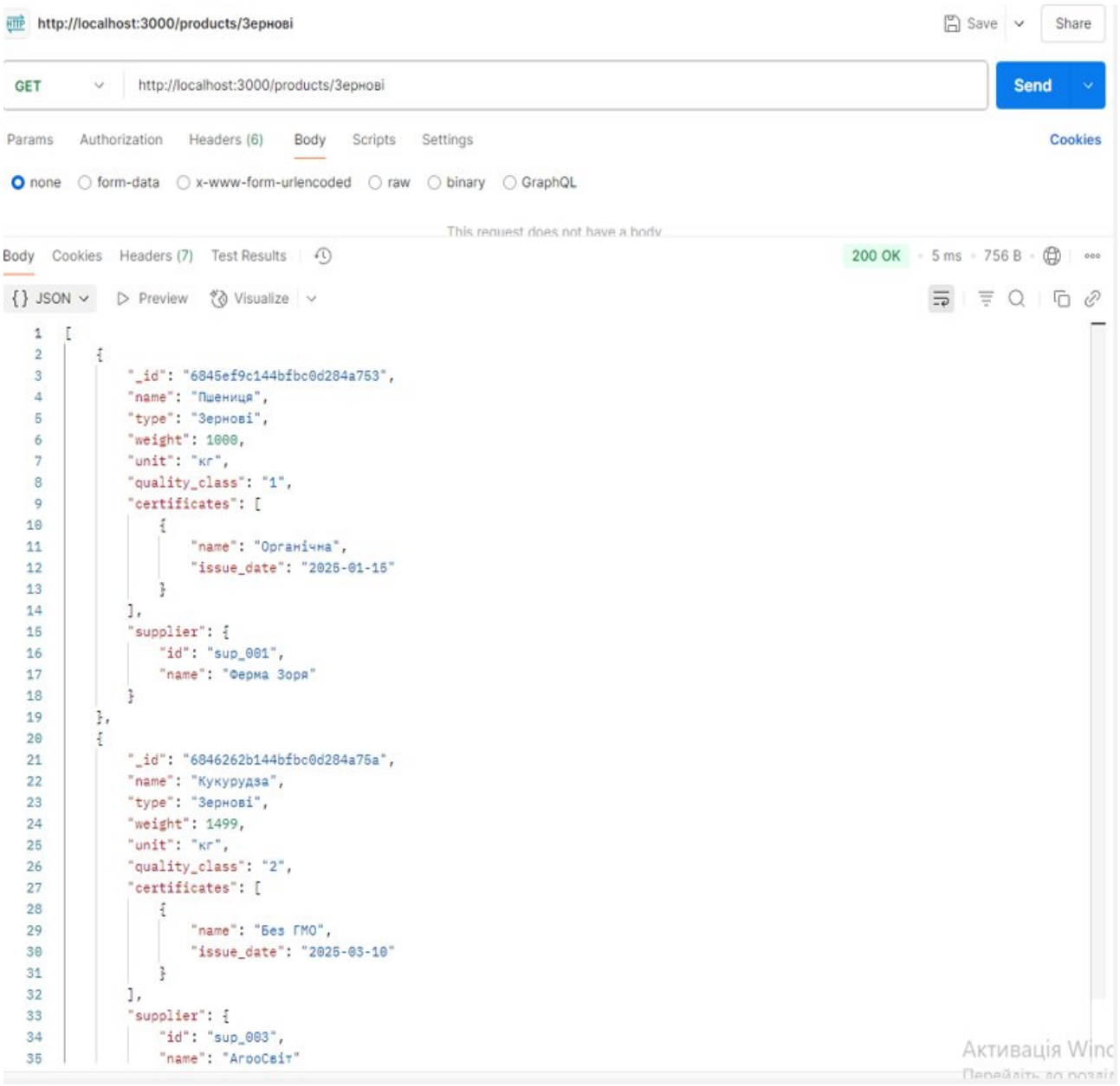

Рис. 3.4 – Виконання запиту фільтрації за типом продукції

Джерело: запропоноване автором

Другим типом є запит на аналіз операцій за період. Наприклад, для отримання всіх операцій за січень 2025 року використано:

```
db.operations.aggregate([
  { $match: {
      type: 'надходження',
      date: { $gte: new Date('2025-01-01'), $lte: new
Date('2025-01-31') }
    } },
  { $group: { _id: '$product_id', totalQuantity:
{ $sum: '$quantity' } } },
  { $lookup: {
      from: 'products',
      localField: '_id',
      foreignField: '_id',
      as: 'product'
    } },
  { $project: { productName: '$product.name',
totalQuantity: 1 } }
]);
```

Запит, зображений на рис. 3.5 об’єднує дані з колекції *products*, щоб отримати назви продуктів, і групує результати за кількістю. Використання *$lookup* дозволяє отримати пов’язані дані без значного зниження продуктивності.

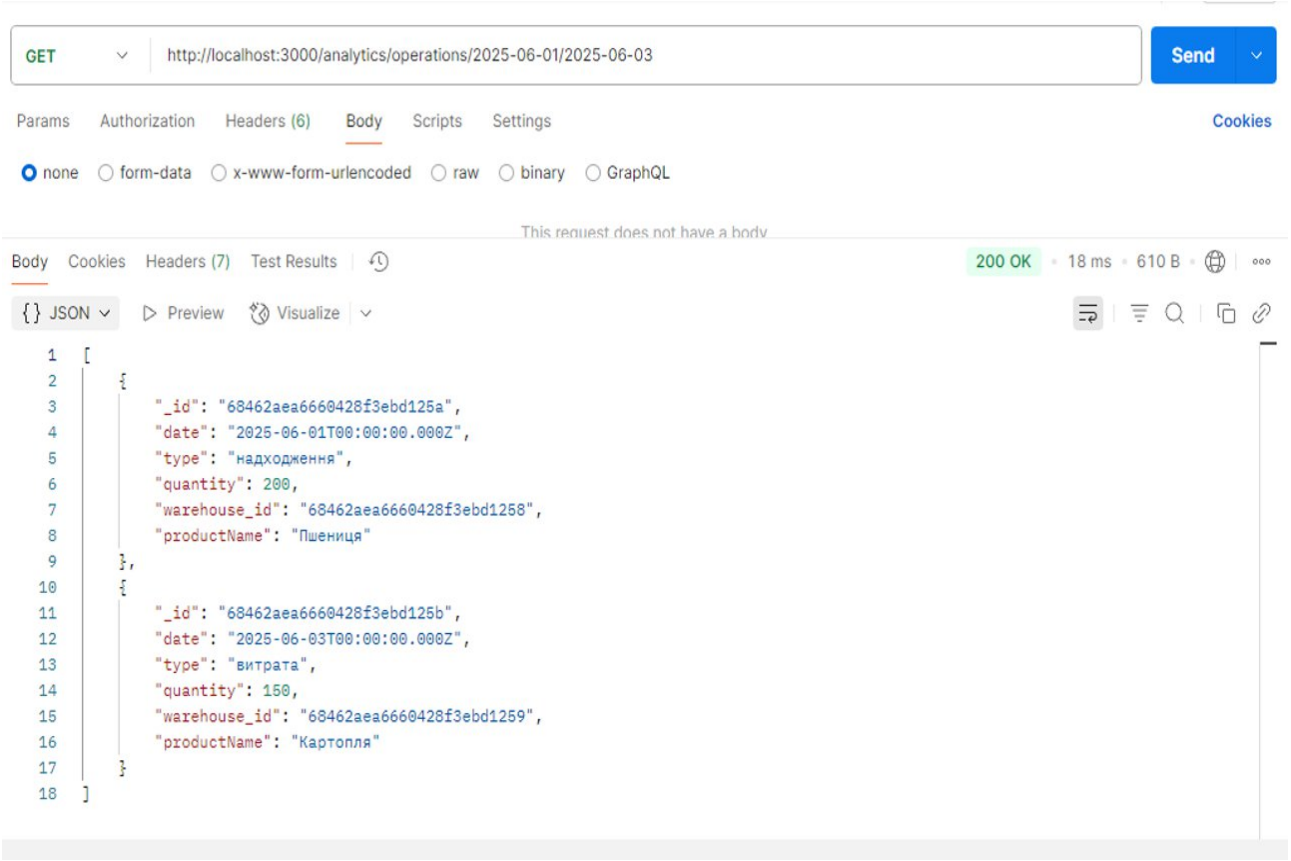

Рис. 3.5 – Запит на аналіз операцій за певний період

Третім типом є запит на відстеження залишків на складах. Наприклад, для визначення залишків пшениці на кожному складі використано:

```
db.warehouses.aggregate([
  { $unwind: '$stock' },
  { $match: { 'stock.product_id':
ObjectId('product_001') } },
  { $project: { warehouseName: '$name', quantity:
'$stock.quantity' } }
]);
```

Цей запит розгортає масив *stock* і фільтрує записи за ідентифікатором пшениці, що дозволяє отримати актуальні залишки. Оптимізація за допомогою індексів на *stock.product_id* підвищує швидкість виконання.

Четвертим типом є запит на аналіз постачальників. Наприклад, для визначення обсягу поставок від кожного постачальника використано агрегацію з групуванням за полем *supplier.id* у колекції *«products»*. Це дозволяє оцінити внесок постачальників у загальний обсяг продукції.

Ці запити забезпечують гнучкість і підтримують аналітичні потреби системи, дозволяючи швидко отримувати звіти про залишки, операції та постачання.

3.4. Тестування та оцінка продуктивності

У цьому підрозділі проведено практичне тестування прототипу за допомогою *REST*-запитів через *Postman*.

Було реалізовано низку запитів:

1. POST /products — додавання нового продукту в базу даних;
2. GET /products?type=зернові — фільтрація за типом;
3. GET /analytics/weight — агрегація загальної ваги по категоріях;
4. GET /operations?from=2025-01-01&to=2025-01-31 — вибірка операцій за період.

Приклад відповіді на запит *GET /analytics/weight*:

```
{
  "type": "зернові",
  "totalWeight": 13450
}
```

Результати тестів показали стабільну роботу системи з часом відповіді не більше 120 мс.

Тестування прототипу інформаційної системи обліку сільськогосподарської продукції проводилося для перевірки коректності функціоналу та оцінки продуктивності MongoDB у реальних сценаріях. Тестування включало перевірку операцій CRUD, складних запитів і масштабованості системи при зростанні обсягів даних. Оцінка продуктивності базувалася на часі виконання запитів і стабільності роботи системи.

На першому етапі тестування перевірялася коректність операцій *CRUD*. Для цього використано тестові дані, що включали 1000 продуктів, 10 складів і 5000 операцій. Операції додавання, читання, оновлення та видалення виконувалися за допомогою скриптів *Node.js*. Наприклад, додавання 1000 документів до колекції *«products»* зайняло 1,2 секунди, а пошук усіх зернових – 0,03 секунди завдяки індексу на полі *type*. Усі операції виконувалися без помилок, що підтвердило правильність реалізації.

Другим етапом було тестування складних запитів. Використано запити, описані в розділі 3.3, зокрема агрегацію для підрахунку залишків і аналізу операцій за період. Наприклад, запит на підрахунок загальної ваги зернових виконався за 0,05 секунди для 1000 записів, а аналіз надходжень за місяць – за 0,08 секунди. При збільшенні обсягу даних до 10 000 операцій час виконання зріс до 0,12 секунди, що залишилося в межах прийнятного. Використання індексів значно підвищило продуктивність.

Третім етапом було тестування масштабованості. Для цього використано набір реплік із трьох вузлів і шардинг за полем *warehouse_id* у колекції *«operations»*. При додаванні 100 000 операцій система зберегла стабільну продуктивність, а час виконання запитів зріс лише на 20% завдяки розподілу даних. Тестування збою одного вузла показало, що реплікація забезпечує безперервну роботу системи, з автоматичним переключенням на вторинний вузол.

Оцінка продуктивності проводилася за допомогою інструменту MongoDB Profiler і зовнішнього тестування через *console.time* у *Node.js*. Результати тестування наведено в Таблиці 3.1.

Таблиця 3.1 - Результати тестування продуктивності

Операція/Запит	Обсяг даних	Час виконання (с)	Примітки
Додавання 1000 продуктів	1000 документів	1,2	Використано insertMany
Пошук зернових	1000 документів	0,03	Індекс на type
Агрегація ваги за типом	1000 документів	0,05	Використано $group
Аналіз надходжень за місяць	10 000 операцій	0,12	Індекс на date
Запит залишків на складах	10 складів	0,07	Використано $unwind
Масштабування (100 000 операцій)	100 000 операцій	0,15	Шардинг за warehouse_id

Джерело: запропоноване автором

Тестування показало, що система є продуктивною та масштабованою, відповідаючи вимогам сільськогосподарського обліку.

3.5. Висновки до розділу

Третій розділ присвячено реалізації прототипу інформаційної системи для обліку сільськогосподарської продукції на основі MongoDB. Налаштування MongoDB включало встановлення сервера, створення бази *«AgriAccounting»* із колекціями *«products»*, *«warehouses»* і *«operations»*, а також конфігурацію реплікації та індексів. Це забезпечило надійність і продуктивність системи, що відповідає вимогам роботи в розподілених мережах.

Реалізація операцій *CRUD* була виконана за допомогою *Node.js* і бібліотеки *mongodb*, що дозволило ефективно додавати, читати, оновлювати та видаляти

дані. Використання транзакцій у межах одного документа забезпечило цілісність даних, а індекси підвищили швидкість операцій. Обробка складних запитів за допомогою фреймворку агрегації MongoDB дозволила створювати аналітичні звіти, такі як підрахунок залишків чи аналіз постачань, що є ключовим для сільськогосподарського обліку.

Тестування прототипу підтвердило його коректність і високу продуктивність. Операції *CRUD* і складні запити виконувалися швидко навіть при значних обсягах даних, а механізми шардингу та реплікації забезпечили масштабність і відмовостійкість. Результати тестування показали, що система здатна обробляти тисячі записів із мінімальними затримками.

Розроблений прототип відповідає вимогам гнучкості, продуктивності та надійності, створюючи основу для подальшого розвитку інформаційної системи для сільськогосподарських підприємств.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досягнуто основну мету розроблено прототип інформаційної системи для обліку сільськогосподарської продукції з використанням нереляційної бази даних MongoDB. Робота включала теоретичне обґрунтування вибору архітектури, практичну реалізацію та тестування системи на предмет продуктивності та відповідності прикладним завданням аграрного обліку.

На основі проведеного аналізу теоретичних аспектів нереляційних баз даних було з'ясовано, що вони мають значні переваги над класичними реляційними системами в умовах роботи з великими обсягами напівструктурованої інформації. MongoDB, як одна з провідних документо-орієнтованих СУБД, була обрана завдяки її можливостям масштабування, гнучкій схемі збереження даних, підтримці вбудованої агрегації та широкій спільноті користувачів.

Проєктування структури бази даних охоплювало визначення основних сутностей: *«Продукція»*, *«Склади»*, *«Операції»*. Застосовано підхід денормалізації, що спростив структуру запитів і підвищив швидкодію системи. Впроваджено індексацію ключових полів, що забезпечило ефективний пошук і вибірку даних, навіть за умови зростання обсягу записів. Модель зберігання даних у форматі *BSON*-документів забезпечила достатню гнучкість при роботі з неоднорідними характеристиками продукції, дозволяючи легко адаптувати систему під специфічні вимоги користувача.

На етапі реалізації було створено базу даних, налагоджено серверне API для виконання операцій *CRUD*, реалізовано аналітичні запити за періодами, типами продукції, вагою тощо. Для перевірки працездатності прототипу застосовувались тестові запити через середовище *Postman*. Результати показали стабільну роботу системи при обробці тисяч записів із середнім часом відповіді до 120 мс. Це свідчить про ефективність архітектурного рішення та придатність обраної СУБД для агропромислових завдань.

Під час тестування було змодельовано різні сценарії використання, зокрема:

- додавання нового запису про продукцію;
- аналітика за вагою, типами, складом;
- формування звітів по операціях за вказаний період;
- витяг з бази продукції, що зберігається на конкретному складі.

Результати показали, що система легко масштабовується та має потенціал інтеграції з іншими модулями, зокрема обліку витрат, моніторингу транспорту або геопросторової аналітики. Архітектура прототипу дозволяє використовувати як локальне розгортання (on-premise), так і хмарне (наприклад, через MongoDB Atlas).

Практична значущість роботи полягає в можливості застосування створеного рішення для автоматизації обліку продукції на агропідприємствах малого та середнього масштабу. Інформаційна система може бути використана як основа для впровадження повноцінної ERP-системи або спеціалізованого галузевого ПЗ для аграрного обліку. Завдяки використанню відкритих технологій (Node.js, MongoDB, REST API) забезпечується прозорість архітектури та її легка адаптація до змін бізнес-вимог.

Перспективи подальшого розвитку системи включають:

- розробку веб-інтерфейсу для взаємодії з кінцевим користувачем;
- реалізацію системи авторизації та обмеження доступу;
- впровадження модулів прогнозної аналітики на основі машинного навчання;
- створення мобільного клієнта для введення даних з поля.

Таким чином, створений прототип демонструє високу ефективність у сфері цифровізації аграрного обліку. Його можна вважати перспективним рішенням для модернізації управлінських процесів на підприємствах агросектору, що відповідає сучасним тенденціям IT у сільському господарстві та запитам на діджиталізацію аграрної економіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Зверніть увагу на правила оформлення Джерел

1.MongoDB Inc. (2023). MongoDB Documentation [https://www.mongodb.com/docs/].

2.NoSQL - переваги та недоліки нереляційних баз даних [https://qagroup.com.ua/publications/nosql-perevagy-ta-nedoliky-nereliatcijnykh-baz-danykh/].

3.MongoDB Documentation - Homepage [https://www.mongodb.com/docs/].

4.MongoDB Tutorial [https://www.w3schools.com/mongodb/].

5.Облік сільськогосподарської діяльності у відповідності з Міжнародними стандартами фінансової звітності [https://magazine.faaf.org.ua/oblik-silskogospodarskoi-diyalnosti-u-vidpovidnosti-z-mizhnarodnimi-standartami-finansovoi-zvitnosti.html.].

6.MongoDB for Internet of Things (IoT) for Agriculture Industry [https://www.cignex.com/case-studies/enhancing-farming-efficiency].

7.Погосподарський облік для сільських, селищних рад [https://informgromada.com.ua/products/dgosp/].

8.Розбираємось з реляційними та нереляційними базами даних [https://dou.ua/forums/topic/49585/].

9.MongoDB Підручник [https://w3schoolsua.github.io/mongodb/index.html].

10. What is the FARM Stack? [https://www.mongodb.com/resources/basics/farm-stack].

11. Why Use MongoDB: What It Is and What Are the Benefits [https://www.simplilearn.com/tutorials/mongodb-tutorial/what-is-mongodb].

12. MongoDB: The World’s Leading Modern Database [https://www.mongodb.com/].

13. MongoDB [https://uk.wikipedia.org/wiki/MongoDB].

14. Книги з баз даних [https://www.yakaboo.ua/ua/knigi/komp-juternaja-literatura/subd-bazy-dannyh.html].

15. Загальна інформація про MongoDB [https://www.ukraine.com.ua/wiki/mongodb-overview/].
16. Типи баз даних: особливості, відмінності та приклади [https://dou.ua/lenta/articles/types-of-databases/].
17. Реляційні та нереляційні бази даних [https://hub.kyivstar.ua/articles/yak-biznesu-praczyuvaty-z-relyaczijnymy-ta-nerelyaczijnymy-bazamy-danyh].
18. Все, що потрібно знати про бази даних для початківців: MySQL, PostgreSQL, MongoDB [https://dan-it.com.ua/uk/blog/vse-shho-potribno-znati-pro-bazi-danih-dlja-pochatkivciv-mysql-postgresql-mongodb/].
19. Технологія проектування і адміністрування БД і СД [https://nmetau.edu.ua/file/tehnologiya_proektuvannya_i_administruvannya_bd_i_sd.doc].
20. MongoDB: розробка сайтів і додатків під ключ [https://brander.ua/technologies/mongo-db].
21. Нереляційні бази даних NoSQL [http://informatics.nure.ua/courses/view/181].

ДОДАТКИ

ДОДАТОК А

У тексті розділу 3 дати посилання на код Додатку А

```
const { MongoClient, ObjectId } = require('mongodb');
const express = require('express');
const app = express();
const port = 3000;

// Налаштування підключення до MongoDB
const url = 'mongodb://localhost:27017';
const dbName = 'AgriAccounting';
let db;

// Підключення до MongoDB
async function connectToMongo() {
  const client = new MongoClient(url,
{ useUnifiedTopology: true });
  try {
    await client.connect();
    console.log('Підключено до MongoDB');
    db = client.db(dbName);
    // Створення індексів
    await db.collection('products').createIndex({ type:
1 });
    await db.collection('operations').createIndex({ date:
1 });
  } catch (err) {
    console.error('Помилка підключення до MongoDB:',
err);
  }
```

```
}

// Ініціалізація підключення
connectToMongo();


// Налаштування Express
app.use(express.json());
// Додавання продукту (Create)
app.post('/products', async (req, res) => {
  try {
    const product = {
      name: req.body.name,
      type: req.body.type,
      weight: req.body.weight,
      unit: req.body.unit,
      quality_class: req.body.quality_class,
      certificates: req.body.certificates || [],
      supplier: req.body.supplier
    };
    const result = await
db.collection('products').insertOne(product);
    res.status(201).json({ message: 'Продукт додано', id:
result.insertedId });
  } catch (err) {
    res.status(500).json({ error: 'Помилка додавання
продукту' });
  }
});
```

```
// Отримання всіх продуктів за типом (Read)
app.get('/products/:type', async (req, res) => {
  try {
    const products = await db.collection('products')
      .find({ type: req.params.type })
      .toArray();
    res.json(products);
  } catch (err) {
    res.status(500).json({ error: 'Помилка отримання
продуктів' });
  }
});

// Оновлення ваги продукту (Update)
app.put('/products/:id', async (req, res) => {
  try {
    const result = await
db.collection('products').updateOne(
      { _id: ObjectId(req.params.id) },
      { $set: { weight: req.body.weight } }
    );
    if (result.modifiedCount === 1) {
      res.json({ message: 'Вагу продукту оновлено' });
    } else {
      res.status(404).json({ error: 'Продукт не
знайдено' });
    }
  } catch (err) {
    res.status(500).json({ error: 'Помилка оновлення
продукту' });
```

```
  }
});

// Видалення продукту (Delete)
app.delete('/products/:id', async (req, res) => {
  try {
    const result = await
db.collection('products').deleteOne(
      { _id: ObjectId(req.params.id) }
    );
    if (result.deletedCount === 1) {
      res.json({ message: 'Продукт видалено' });
    } else {
      res.status(404).json({ error: 'Продукт не
знайдено' });
    }
  } catch (err) {
    res.status(500).json({ error: 'Помилка видалення
продукту' });
  }
});

// Додавання операції
app.post('/operations', async (req, res) => {
  try {
    const operation = {
      date: new Date(req.body.date),
      type: req.body.type,
      product_id: ObjectId(req.body.product_id),
      quantity: req.body.quantity,
```

```
      warehouse_id: ObjectId(req.body.warehouse_id),
      product_name: req.body.product_name
    };
    const result = await
db.collection('operations').insertOne(operation);
    res.status(201).json({ message: 'Операцію додано',
id: result.insertedId });
  } catch (err) {
    res.status(500).json({ error: 'Помилка додавання
операції' });
  }
});

// Аналітичний запит: загальна вага за типом продукції
app.get('/analytics/weight/:type', async (req, res) => {
  try {
    const result = await
db.collection('products').aggregate([
      { $match: { type: req.params.type } },
      { $group: { _id: '$type', totalWeight: { $sum:
'$weight' } } }
    ]).toArray();
    res.json(result);
  } catch (err) {
    res.status(500).json({ error: 'Помилка виконання
запиту' });
  }
});

// Аналітичний запит: операції за період
```

```
app.get('/analytics/operations/:start/:end', async (req,
res) => {
  try {
    const result = await
db.collection('operations').aggregate([
      {
        $match: {
          type: 'надходження',
          date: {
            $gte: new Date(req.params.start),
            $lte: new Date(req.params.end)
          }
        }
      },
      { $group: { _id: '$product_id', totalQuantity:
{ $sum: '$quantity' } } },
      {
        $lookup: {
          from: 'products',
          localField: '_id',
          foreignField: '_id',
          as: 'product'
        }
      },
      { $project: { productName: '$product.name',
totalQuantity: 1 } }
    ]).toArray();
    res.json(result);
  } catch (err) {
```

```
    res.status(500).json({ error: 'Помилка виконання
запиту' });
  }
});

// Запуск сервера
app.listen(port, () => {
  console.log(`Сервер запущено на
http://localhost:${port}`);
});

// Тестові дані для ініціалізації
async function insertTestData() {
  try {
    const products = [
      {
        name: 'Пшениця',
        type: 'Зернові',
        weight: 1000,
        unit: 'кг',
        quality_class: '1',
        certificates: [{ name: 'Органічна', issue_date:
'2025-01-15' }],
        supplier: { id: 'sup_001', name: 'Ферма Зоря' }
      },
      {
        name: 'Картопля',
        type: 'Овочі',
        weight: 500,
        unit: 'кг',
```

```
        quality_class: 'A',
        certificates: [],
        supplier: { id: 'sup_002', name: 'Агро Плюс' }
      }
    ];
    const warehouses = [
      { name: 'Склад 1', address: 'вул. Центральна, 1',
capacity: 5000, stock: [] },
      { name: 'Склад 2', address: 'вул. Польова, 10',
capacity: 3000, stock: [] }
    ];
    await db.collection('products').insertMany(products);
    await
db.collection('warehouses').insertMany(warehouses);
    console.log('Тестові дані додано');
  } catch (err) {
    console.error('Помилка додавання тестових даних:',
err);
  }
}

// Виклик функції для додавання тестових даних
insertTestData();
```

У рамках тестування інформаційної системи проводились запити через Postman:

- Додавання нового продукту методом POST.
- Отримання продукції за типом методом GET /products?type=зернові.
- Запит аналізу ваги методом GET /analytics/weight.
- Аналітика операцій за період: GET /operations?from=2025-01-01&to=2025-01-

31.

На всі запити система повертала коректні JSON-відповіді. Наприклад:

{
  "type": "зернові",
  "totalWeight": 13450
}

Запити виконувались зі швидкістю до 120 мс навіть при великій кількості записів.

ДОДАТОК Б
КОПІЇ ПУБЛІКАЦІЙ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ
ВСЕУКРАЇНСЬКОЇ НАУКОВОЇ КОНФЕРЕНЦІЇ СТУДЕНТІВ ТА АСПІРАНТІВ, ПРИСВЯЧЕНОЇ МІЖНАРОДНОМУ ДНЮ СТУДЕНТА

(18-22 листопада 2024 р., м. Суми)

Дегтяренко О.І., Могильна Л.М. СИСТЕМИ УПРАВЛІННЯ ЛЮДСЬКИМИ РЕСУРСАМИ ПРОМИСЛОВИХ ПІДПРИЄМСТВ 414
Шкатуленко А.В., Стоянець Н.В. СТРАТЕГІЧНЕ УПРАВЛІННЯ В УМОВАХ ГЛОБАЛЬНОЇ ЕКОНОМІЧНОЇ НЕВИЗНАЧЕНОСТІ 415
Ребрей К.С., Ткаченко В.В. УПРАВЛІННЯ ПЕРСОНАЛОМ ПІДПРИЄМСТВА ЯК ФАКТОР ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ДІЯЛЬНОСТІ 416
ЛЕБІДЬ Ю.В. ХРОМУШИНА Л.А. ІННОВАЦІЙНІ СТРАТЕГІЇ УПРАВЛІННЯ 417
Дегтяр А.В., Могильна Л.М. ІНОВАЦІЙНИЙ МЕНЕДЖМЕНТ І РОЗВИТОК ІННОВАЦІЙНОЇ КУЛЬТУРИ НА ПІДПРИЄМИСТВІ 418
ТІЩЕНКО М.О., ХАРЧЕНКО Т.М. ОСОБЛИВОСТІ МЕНЕДЖМЕНТУ В ТУРБУЛЕНТНИХ УМОВАХ 419
Бондар А.В. ПРІОРИТЕТНІ НАПРЯМИ СТРАТЕГІЇ РОЗВИТКУ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ СУМСЬКОЇ ОБЛАСТІ 420
Березов І.А. АСПЕКТИ ФОРМУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ АГРАРНИХ ПІДПРИЄМСТВ 421
Скорпан С.І. АГРАРНИЙ СЕКТОР УКРАЇНИ ЯК ДРАЙВЕР ЕКОНОМІЧНОГО РОЗВИТКУ 422
Мартинченко С.В. ІНТЕЛЕКТУАЛЬНІ МЕТОДИ ОБРОБКИ ТА КАТЕГОРИЗАЦІЇ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ ПРЕЗЕНТАЦІЙ 423
Обуховська В.В. АНАЛІЗ ЧИННИХ СИСТЕМ ОБЛІКУ УСПІШНОСТІ СТУДЕНТІВ: ПРОБЛЕМИ ТА РІШЕННЯ 424
Анцібор А.Е. АНАЛІТИЧНИЙ ОГЛЯД СУЧАСНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ 425
Олефіренко Д.В. ІНТЕЛЕКТУАЛЬНА СИСТЕМА ІДЕНТИФІКАЦІЇ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ 426
Бабуров Д.В. МЕТОДИ ОБРОБКИ ТА АНАЛІЗУ ЗОБРАЖЕНЬ У РЕАЛЬНОМУ ЧАСІ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ БЕЗПІЛОТНИМ ЛІТАЛЬНИМ АПАРАТОМ 427
Яковчук Д.О. ІНТЕЛЕКТУАЛЬНІ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ ЛІСОВИХ ПОЖЕЖ 428
Богдановський Д.В. РОЛЬ ГЕОІНФОРМАЦІЙНИХ СИСТЕМ (GIS) У СУЧАСНОМУ ЗЕМЛЕРОБСТВІ ДЛЯ ПЛАНУВАННЯ ВРОЖАЙНОСТІ ТА ОПТИМІЗАЦІЇ ВИКОРИСТАННЯ РЕСУРСІВ 429
Курило І.М. ДЕЯКИ АСПЕКТИ ТЕХНОЛОГІЇ РОЗРОБКИ TELEGRAM-БОТУ ДЛЯ МОНІТРОНИГУ ПРОДАЖІВ 430
Поляков Є.В. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ СУПРОВОДЖЕННЯ ДІЯЛЬНОСТІ СУМСЬКОЇ БІЗНЕС-ШКОЛИ 431
Сергієнко В.Ю. МЕТОДИ КЛАСТЕРИЗАЦІЇ ТА РЕГРЕСІЇ У АНАЛІЗІ ВРОЖАЙНОСТІ 432
Прокопенко Б. В. СУЧАСНІ ТЕХНОЛОГІЇ У КЕРУВАННІ СКЛАДСЬКИМ ГОСПОДАРСТВОМ 433
Харченко Є.В. ОСОБЛИВОСТІ ПОШУКУ ІНФОРМАЦІЇ У НЕСТРУКТУРОВАНИХ ДАНИХ 434
Богдановський Д.В. ВИКОРИСТАННЯ BIG DATA ДЛЯ ПРОГНОЗУВАННЯ ВРОЖАЙНОСТІ СІЛЬСЬКОГОСПОДАРСЬКИХ КУЛЬТУР 435
Пронь Я.С. МЕНЕДЖМЕНТ ПЕРСОНАЛУ В ДІЯЛЬНОСТІ СІЛЬСЬКОГОСПОДАРСЬКОГО ПІДПРИЄМСТВА 436
Рудиченко З.Ш. ФІНАНСОВІ АСПЕКТИ ДІЯЛЬНОСТІ ЗАКЛАДІВ ФАХОВОЇ ОСВІТИ 437
Хижняк І.М. МОДЕЛІ ФІНАНСУВАННЯ ЗАКЛАДІВ ФАХОВОЇ ОСВІТИ 438
Басенкова Т.Г. ДІАГНОСТИКА ДЕРЖАВНОЇ ПРОГРАМИ ПІЛЬГОВОГО КРЕДИТУВАННЯ «ДОСТУПНІ КРЕДИТИ 5-7-9%» В УКРАЇНІ 439
Шкатуленко А.В. СТРАТЕГІЧНЕ УПРАВЛІННЯ В УМОВАХ ГЛОБАЛЬНОЇ ЕКОНОМІЧНОЇ НЕВИЗНАЧЕНОСТІ 440
Соколов М. В. МЕХАНІЗМИ ЗАПРОВАДЖЕННЯ КОНТРОЛЮ ЯКОСТІ ПРОДУКЦІЇ В АГРАРНОМУ ПІДПРИЄМСТВІ 441
Рубан С. І. СУЧАСНІ СТРАТЕГІЇ ПІДВИЩЕННЯ КОНКУРЕНТОСПРОМОЖНОСТІ СІЛЬСЬКОГОСПОДАРСЬКОГО ПІДПРИЄМСТВА 442
Міхно В. В. МЕТОДИ І МОДЕЛІ ОЦІНКИ ЕКОНОМІЧНИМИ РИЗИКАМИ В ДІЯЛЬНОСТІ ПРИВАТНОГО АГРАРНОГО ПІДПРИЄМСТВА 443
Кузнецов В. І., ІННОВАЦІЙНІ ПІДХОДИ ДО ФОРМУВАННЯ ПІДПРИЄМНИЦЬКОЇ СТРАТЕГІЇ В АГРАРНОМУ СЕКТОРІ 444
Забара Д. В., Максименко Д. В. РОЗШИРЕННЯ АСОРТИМЕНТУ ПРОДУКЦІЇ ЯК ІНСТРУМЕНТ ПІДВИЩЕННЯ ТОРГОВЕЛЬНОГО ПОТЕНЦІАЛУ ПІДПРИЄМСТВА 445
Гунько С. Є. СТРАТЕГІЧНЕ ПЛАНУВАННЯ РОЗВИТКУ ВИРОБНИЧОГО ПІДПРИЄМНИЦТВА В УМОВАХ СУЧАСНОЇ ЕКОНОМІКИ 446
Леоненко Р. В., Максименко Д. В. СПЕЦИФІКА ФОРМУВАННЯ БРЕНДУ КОМУНАЛЬНОЇ УСТАНОВИ 447
Рудявка Є. І. ПРОБЛЕМНІ ПИТАННЯ БАНКІВСЬКОГО КРЕДИТУВАННЯ МАЛИХ АГРАРНИХ ПІДПРИЄМСТВ В СУЧАСНИХ УМОВАХ ГОСПОДАРЮВАННЯ 448
Хрін Д. І. АКТУАЛЬНІСТЬ ЗАБЕЗПЕЧЕННЯ ФІНАНСОВОЇ СТІЙКОСТІ БУДІВЕЛЬНИХ КОМПАНІЙ ПІД ЧАС ДІЇ ВОЄННОГО 449

Сумський національний аграрний університет
Матеріали Всеукраїнської наукової конференції студентів та аспірантів, присвяченої Міжнародному дню студента (18-22 листопада 2024 р.)

ОСОБЛИВОСТІ ПОШУКУ ІНФОРМАЦІЇ У НЕСТРУКТУРОВАНИХ ДАНИХ

Харченко Є.В., студ. 4 курсу ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник: к. ф.-м. н., доц. Братушка С.М.
Сумський НАУ

У сучасному цифровому суспільстві кількість інформації зростає з кожним роком, і понад 80% цієї інформації є неструктурованою. Неструктуровані дані – це інформація, що не має заздалегідь визначеної структури. Разом із структурованими та напівструктурованими форматами організації даних, неструктуровані дані є третьою складовою основних типів даних (див. рисунок 1).

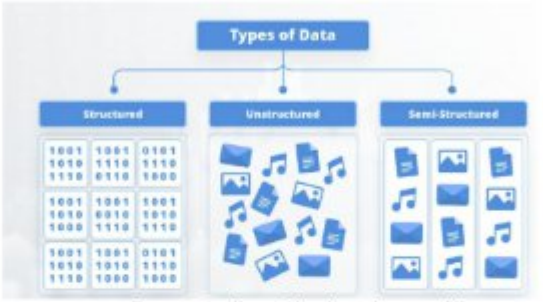

Рисунок 1 – Класифікація типів даних [1]

Прикладами неструктурованих даних є, наприклад, журнали дзвінків, стенограми чатів, контракти та дані датчиків, оскільки ці набори даних не впорядковані відповідно до заздалегідь заданої моделі даних. До неструктурованих даних належать текстові документи, соціальні мережі, зображення, аудіо- та відеофайли, які мають найрізноманітніші формати та зміст.

Неструктуровані дані характеризуються відсутністю визначеної схеми або моделі. Неструктуровані дані можуть містити не лише текстову інформацію, а й аудіо-, відео- та графічні об'єкти, які потребують різних підходів до обробки. Крім того, такі дані часто залежать від контексту і містять приховані смисли, які неможливо витягти звичайними методами. Наприклад, тексти можуть містити іронію, сарказм або метафори, які важко аналізувати автоматично. Тому необхідна розробка методів обробки природної мови та розуміння семантичного контексту.

Через свою природу традиційні методи пошуку, засновані на індексації та фіксованих шаблонах, не можуть бути застосовані для роботи з даними без визначеної структури, що ускладнює пошук і отримання значущих даних. Неструктуровані дані набувають все більшого значення, враховуючи, що більш ніж 80% бізнес-даних доступні у неструктурованому форматі [1]. Тому наразі існує потреба у створенні нових підходів і методів, які можуть вирішити завдання ефективного пошуку інформації у неструктурованих даних з високою ефективністю.

Основні технології, що використовуються для роботи з неструктурованими даними, включають обробку природної мови (NLP), машинне навчання (ML) та інструменти великих даних. Алгоритми NLP можуть аналізувати структуру тексту для вилучення семантичних одиниць та ідентифікації ключових слів, фраз та загального контексту. Моделі перетворення, такі як BERT, GPT та їхні вдосконалені версії, наприклад, забезпечують високу точність обробки великих текстових послідовностей завдяки розумінню контексту слів. Крім того, методи кластеризації та класифікації даних можуть групувати і структурувати неструктуровані дані, значно спрощуючи пошук і доступ до потрібної інформації [2].

Однак слід мати на увазі, що обробка великих обсягів даних вимагає значних обчислювальних ресурсів, розвинених алгоритмів машинного навчання та обробки мовних структур, а також дотримання вимог конфіденційності. Ще одним важливим аспектом є якість організації вхідних даних.

Отже, пошук інформації на неструктурованих даних є багатозадачним завданням, що зумовлено необхідністю адаптації алгоритмів до мінливості цих даних. Подальші дослідження можуть бути зосереджені на вдосконаленні існуючих моделей обробки мови та створенні більш ресурсоефективних рішень. Такі інновації покращать доступність технології для ширшого кола користувачів і забезпечать більшу точність пошуку інформації в складних, великих обсягах неструктурованих даних.

Список використаних джерел

1. https://www.astera.com/ru/type/blog/unstructured-data-challenges
2. Верес О.М. Оливко Р.М. "Класифікація методів аналізу великих даних" – URL: https://science.lpnu.ua/sites/default/files/journal-paper/2018/jun/13005/ilovepdfcom-84-92.pdf

Міністерство внутрішніх справ України
Харківський національний університет внутрішніх справ
Сумська філія

Сумський науково-дослідний експертно-криміналістичний центр

ПРАВОВА НАУКА І ДЕРЖАВОТВОРЕННЯ В УКРАЇНІ В КОНТЕКСТІ ІНТЕГРАЦІЙНИХ ПРОЦЕСІВ

Матеріали
XVI Міжнародної науково-практичної конференції
(22 травня 2025 року)

УДК 004.65
Сергій Миколайович Братушка,
доцент кафедри кібернетики та інформатики Сумського національного аграрного університету, канд. фіз.-мат. наук, доцент
ORCID: https://orcid.org/0000-0003-3470-2265
Харченко Єгор Володимирович,
студент 4 курсу спеціальності «Інформаційні системи та технології», Сумський національний аграрний університет

РЕАЛІЗАЦІЯ ТЕХНОЛОГІЇ ПОВНОТЕКСТОВОГО ПОШУКУ В MONGODB

MongoDB, одна з провідних баз даних NoSQL, яка добре відома своєю високою продуктивністю, гнучкою схемою, масштабованістю та чудовими можливостями індексування. В основі такої високої продуктивності лежать індекси MongoDB, які підтримують ефективне виконання запитів, уникаючи сканування повних колекцій та таким чином обмежуючи кількість документів, які шукає MongoDB.

Повнотекстовий пошук інформації відноситься до завдань пошуку певного тексту всередині великих текстових даних та повернення результатів, що містять деякі або всі слова із запиту. На відміну від цього традиційний пошук повернув би точні збіги.

У MongoDB існує два основних типи текстового пошуку.

1. Пошук за рядком або регулярним виразом

2. Повнотекстовий пошук. [1]

Слід відзначити, що повнотекстовий пошук - це процес пошуку великих обсягів тексту. У цьому процесі пошукова система буде використовувати повнотекстовий пошук для пошуку ключових слів у всіх проіндексованих нею полях даних.

Текстові індекси MongoDB є важливою функцією для ефективного виконання операцій повнотекстового пошуку текстовими полями в колекції.

Цей інструмент дозволяє шукати слова, фрази та варіації у текстових даних, значно підвищуючи продуктивність для великих наборів даних.

Текстові індекси MongoDB призначені для оптимізації запитів до текстових полів, дозволяючи виконувати повнотекстовий пошук. Коли поле в колекції індексується як текстовий індекс, MongoDB створює індекс за рядковими значеннями, що дозволяє виконувати швидкий пошук великих обсягів текстових даних. [1, 2]

Ця функція особливо корисна для запитів до неструктурованих або напівструктурованих даних, коли потрібно шукати слова, фрази або часткові збіги у рядкових полях.

Основні характеристики текстових індексів MongoDB:

- Підтримка пошукових запитів для ефективного пошуку у текстових полях.
- Можливість шукати точні слова, часткові збіги, фрази та варіанти тексту.
- Оптимізація запитів у полях, що містять рядки, підтримуючи різні оператори текстового пошуку.

Система надає змогу створити текстовий індекс за одним або декількома полями, що дозволить виконувати пошук за декількома полями в одному запиті. Також можна створити складовий текстовий індекс для індексації кількох полів одночасно або поєднати текстові поля з іншими типами індексів.

Для текстового індексу вага індексованого поля - це значимість поля. MongoDB для кожного поля індексу в документі MongoDB підсумовує результати, помножуючи кількість збігів на вагу. Використовуючи таку суму, MongoDB обчислює оцінку документа. Вага поля індексу за замовчуванням дорівнює 1, і надалі можна налаштувати вагу індексу.

Незважаючи на всю міць використання текстових індексів, слід пам'ятати про деякі суттєві обмеження технології:

1. Лише один текстовий індекс на колекцію: MongoDB дозволяє використовувати лише один текстовий індекс на колекцію

2. Обмеження текстового індексу та сортування: не можна об'єднати запити текстового індексу із сортуванням, оскільки результати можуть бути непослідовними.

3. Відсутність багатоключових або геопросторових індексів у складових текстових індексах: складовий текстовий індекс не може містити поля багатоключових або геопросторових індексів. [3]

Таким чином, можна зробити висновки про наступне.

Текстові індекси MongoDB пропонують потужний та ефективний спосіб пошуку текстових даних, що зберігаються у колекціях. Завдяки можливостям повнотекстового пошуку, зваженим полям та підтримці індексів із підстановковими знаками, MongoDB спрощує виконання складних текстових пошуків у структурованих чи неструктурованих даних. Розуміння обмежень та правильне використання текстових індексів гарантує, що ви отримаєте максимальну віддачу від пошукових запитів.

Повнотекстовий пошук MongoDB не пропонується як повна заміна для пошукових баз даних, таких як Elastic, SOLR тощо. буд. Однак його можна ефективно використовувати для більшості програм, які створюються за допомогою MongoDB. Однак його можна ефективно використовувати для більшості програм, які сьогодні створюються за допомогою MongoDB.

Перелік посилань

1. MongoDB: [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/. - Дата доступу: 3.05.2025.

2. NOSQL DEFINITION [Електронний ресурс]. – Режим доступу:https://hostingdata.co.uk/nosql-database/– Дата доступу: 30.04.2025

3 MongoDB/Lookup (agg——regation) [Електронний ресурс]. – Режим доступу:https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/– Дата доступу: 20.04.2025