

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ЗВІТ З ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«СУЧАСНІ ПАРАДИГМИ ЗБЕРЕЖЕННЯ ДАНИХ»

Виконав: студент групи КН-22-1(а)

Панцюк К.В.

Перевірив: Андреев П.І.

КРЕМЕНЧУК 2025

Лабораторна робота № 2

Тема. Розробка моделей даних в MongoDB

Мета: навчитися створювати та використовувати моделі даних у MongoDB для зберігання й обробки інформації.

Порядок виконання роботи

1. Отримати індивідуальний варіант завдання.
2. Спроекувати схему БД.
3. Реалізувати схему у MongoDB.
4. Створити представлення (view), яке об'єднує створені колекції.

Виконання роботи

1.Індивідуальне завдання:

Індивідуальне завдання

Розробити схему бази даних для інтернет-магазину електроніки у бізнес-доміні Електронна комерція (E-commerce). Система має задовольняти наступні вимоги:

1. Кожен товар належить до певної категорії та підкатегорії
2. Товари мають різні технічні характеристики залежно від категорії
3. Користувачі можуть додавати товари до кошика та створювати замовлення
4. Користувачі можуть залишати відгуки та рейтинги для товарів
5. Система має зберігати історію переглядів товарів
6. Адміністратори можуть керувати складом та цінами товарів
7. Система знижок та промокодів
8. Управління доставкою та відділеннями

Бізнес-домен: Електронна комерція (E-commerce)

Ця предметна область охоплює онлайн-продаж електронних товарів, управління каталогом продуктів, обробку замовлень та взаємодію з клієнтами.

2.Проектуємо схему бази даних:

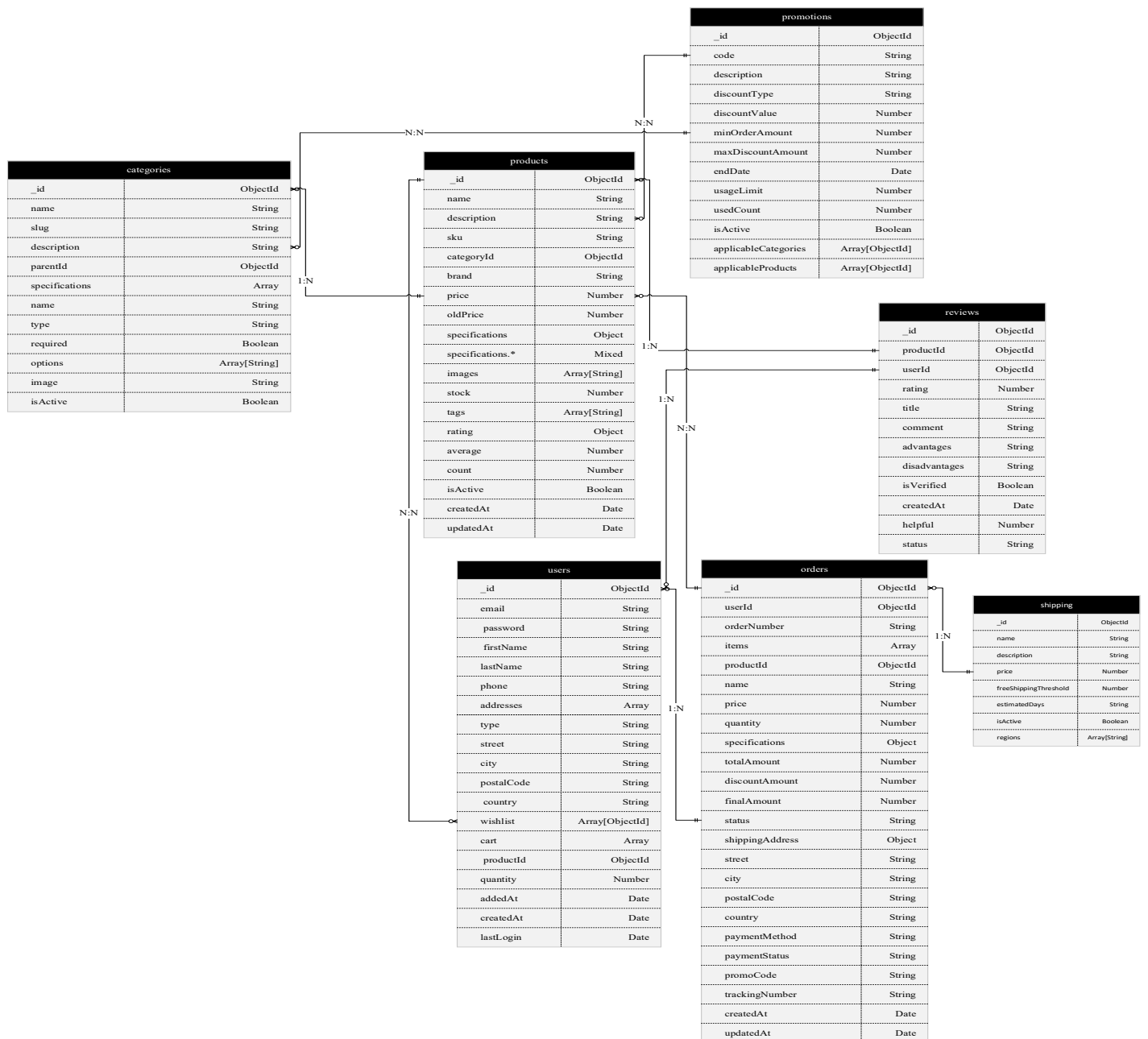
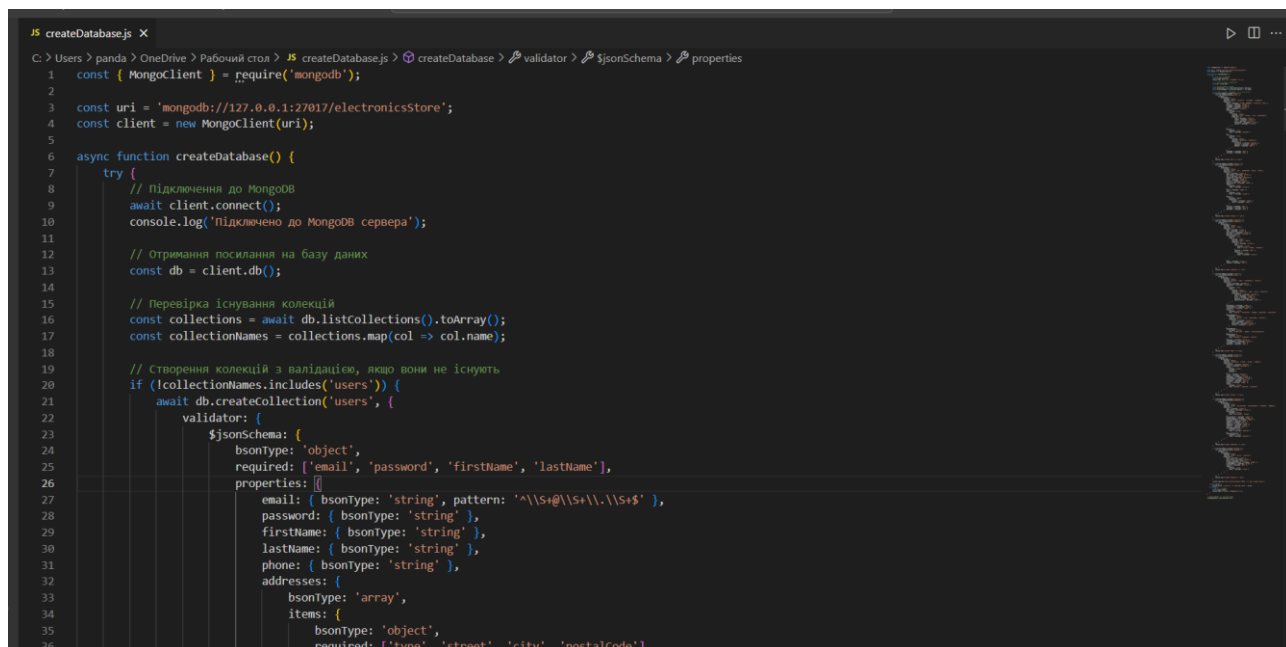


Рисунок 2.1—Схема бази даних відповідно до поставленої задачі

3.Реалізуємо схему у MongoDB:

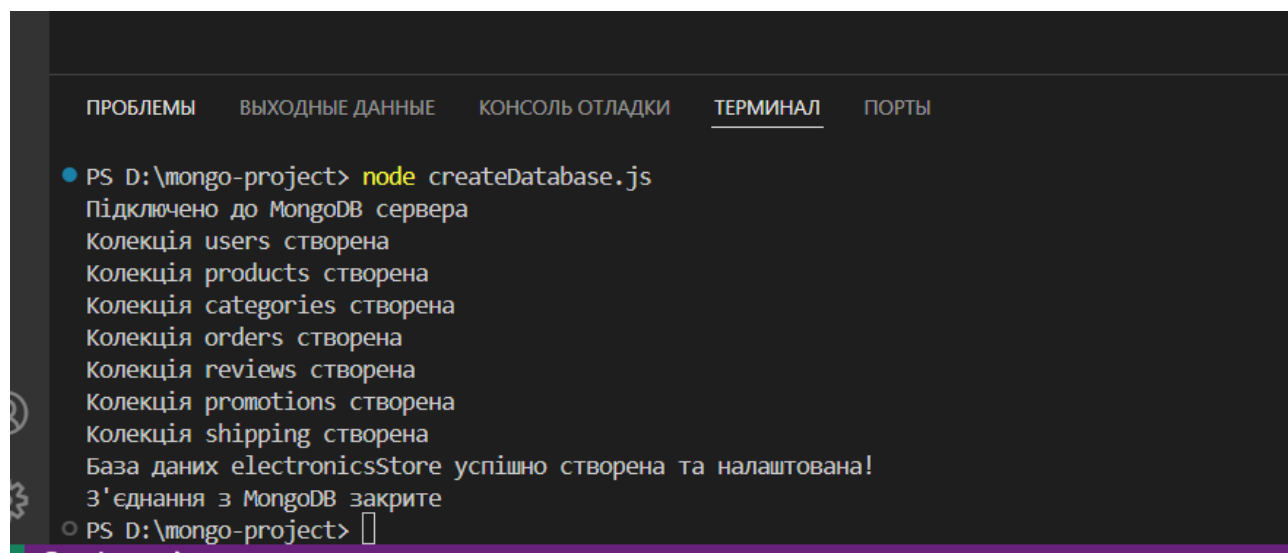
Для цього напишемо скрипт для створення бази даних та відповідних колекцій зі схеми зазначеної вище.



```
1 const { MongoClient } = require('mongodb');
2
3 const uri = 'mongodb://127.0.0.1:27017/electronicsStore';
4 const client = new MongoClient(uri);
5
6 async function createDatabase() {
7   try {
8     // Підключення до MongoDB
9     await client.connect();
10    console.log('Підключено до MongoDB сервера');
11
12    // Отримання посилання на базу даних
13    const db = client.db();
14
15    // Перевірка існування колекцій
16    const collections = await db.listCollections().toArray();
17    const collectionNames = collections.map(col => col.name);
18
19    // Створення колекцій з валідацією, якщо вони не існують
20    if (!collectionNames.includes('users')) {
21      await db.createCollection('users', {
22        validator: {
23          $jsonSchema: {
24            bsonType: 'object',
25            required: ['email', 'password', 'firstName', 'lastName'],
26            properties: {
27              email: { bsonType: 'string', pattern: '^\\S+@\\S+\\.\\S+$' },
28              password: { bsonType: 'string' },
29              firstName: { bsonType: 'string' },
30              lastName: { bsonType: 'string' },
31              phone: { bsonType: 'string' },
32              addresses: {
33                bsonType: 'array',
34                items: {
35                  bsonType: 'object',
36                  required: ['type', 'street', 'city', 'postalCode'],
```

Рисунок 2.2–Частина скрипту для створення БД

У терміналі отримуємо наступний результат :



```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ
● PS D:\mongo-project> node createDatabase.js
Підключено до MongoDB сервера
Колекція users створена
Колекція products створена
Колекція categories створена
Колекція orders створена
Колекція reviews створена
Колекція promotions створена
Колекція shipping створена
База даних electronicsStore успішно створена та налаштована!
З'єднання з MongoDB закрито
PS D:\mongo-project>
```

Рисунок 2.3–Результат виконання скрипта для створення БД

Заповнимо БД інформацією:

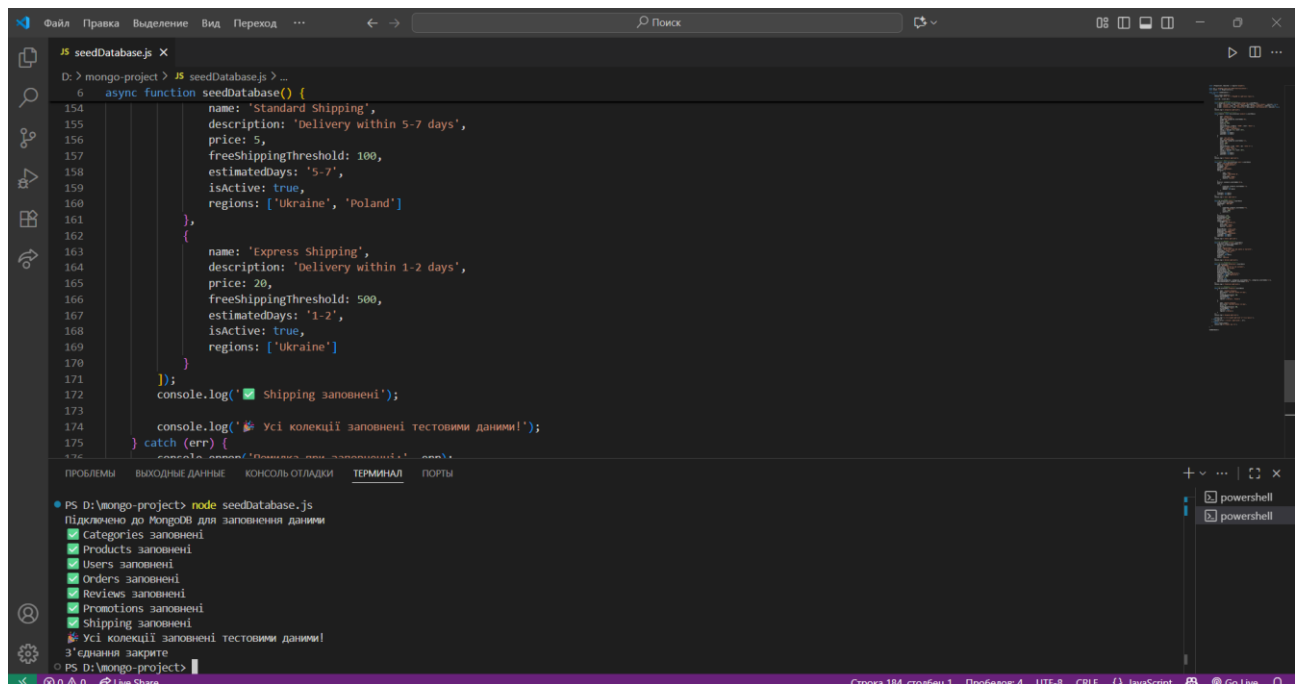


Рисунок 2.4—Результат виконання скрипта для заповнення БД інформацією

4. Створюємо представлення (view), яке об'єднує створені колекції.

Представлення `orderDetails` показує всі замовлення разом з інформацією про користувача та товари в замовленні. Воно містить номер замовлення, користувача, список товарів із їх ціною та кількістю, фінансові підсумки замовлення, статус, адресу доставки, метод оплати та промокод. Фактично об'єднує колекції `orders`, `users` та `products`, щоб мати всі потрібні дані для перегляду замовлення в одному місці.

Представлення `productDetails` показує всі продукти разом із категоріями та рейтингом. Воно містить назву, опис, артикул, бренд, ціну, характеристики, зображення, кількість на складі, теги та статус активності. Об'єднує інформацію з колекцій `products`, `categories` та `reviews`, щоб можна було бачити повний профіль продукту в одному документі.

Представлення `userActivity` показує активність користувачів. Воно містить ім'я, прізвище, email користувача, кількість замовлень, останній вхід, кількість товарів у корзині та в списку бажань. Об'єднує колекції `users` та `orders`, щоб швидко бачити, хто і що замовляв та наскільки активний.

```

48 console.log('Колекція shipping створена');
49 }
50
51 // ===== Створення представлень (view) =====
52 const views = [
53   {
54     name: "orderDetails",
55     viewOn: "orders",
56     pipeline: [
57       { $lookup: { from: "users", localField: "userId", foreignField: "_id", as: "user" } },
58       { $unwind: "$user" },
59       { $lookup: { from: "products", localField: "items.productId", foreignField: "_id", as: "products" } },
60       { $project: { orderNumber: 1, status: 1, totalAmount: 1, finalAmount: 1, createdAt: 1, "user.firstName": 1, "user.lastName": 1, "user.email": 1, "prod
61     ]
62   },
63   {
64     name: "productDetails",
65     viewOn: "products",
66     pipeline: [
67       { $lookup: { from: "categories", localField: "categoryId", foreignField: "_id", as: "category" } },
68       { $unwind: "$category" },
69       { $lookup: { from: "reviews", localField: "_id", foreignField: "productId", as: "reviews" } },
70       { $project: { name: 1, sku: 1, brand: 1, price: 1, stock: 1, "category.name": 1, reviewsCount: { $size: "$reviews" }, avgRating: { $avg: "$reviews.rat
71     ]
72   },
73   {
74     name: "userActivity",
75     viewOn: "users",
76     pipeline: [
77       { $lookup: { from: "orders", localField: "_id", foreignField: "userId", as: "orders" } },
78       { $project: { firstName: 1, lastName: 1, email: 1, ordersCount: { $size: "$orders" } } }
79     ]
80   }
81 ]

```

Рисунок 2.5 –Код для створення представлень

```

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ
PS D:\mongo-project> node checkViews.js
[
  {
    _id: new ObjectId('68c7e967f34f86910d0a8f9a'),
    orderNumber: 'ORD-1001',
    totalAmount: 1200,
    finalAmount: 1100,
    status: 'processing',
    createdAt: 2025-09-15T10:24:39.137Z,
    user: {
      email: 'john.doe@example.com',
      firstName: 'John',
      lastName: 'Doe'
    },
    products: [ [Object] ]
  }
]
--- productDetails ---
[
  {
    _id: new ObjectId('68c7e967f34f86910d0a8f97'),
    name: 'iPhone 15',
    sku: 'APL-IP15-256',
    brand: 'Apple',
    price: 1200,
    stock: 15,
    category: { name: 'Smartphones' },
    reviewsCount: 1,
    avgRating: 5
  },
  {
    _id: new ObjectId('68c7e967f34f86910d0a8f98'),
    name: 'Dell XPS 13',
    sku: 'DLL-XPS13-16GB',
    brand: 'Dell',
    price: 1500,
    stock: 7,
    category: { name: 'Laptops' },
    reviewsCount: 0,
    avgRating: null
  }
]

```

Рисунок 2.6 –Вивід представлень

	A	B	C	D	E	F	G
1							
2	Дія	Тип	Інформація	Частота	Пріоритет		
3	Створити нового користувача	Записування	email, пароль, ім'я, прізвище, телефон, адреса	50 на день	Високий		
4	Додати новий продукт	Записування	назва, опис, sku, категорія, бренд, ціна, stock	20 на день	Високий		
5	Створити категорію	Записування	назва, slug, опис, характеристики	5 на день	Середній		
6	Оформити замовлення	Записування	userId, items, totalAmount, статус, адреса доставки, метод оплати	200 на день	Високий		
7	Додати відгук	Записування	productId, userId, рейтинг, коментар	500 на день	Середній		
8	Переглянути продукт	Зчитування	назва, опис, характеристики, ціна, stock, рейтинг	10 000 на день	Високий		
9	Переглянути замовлення	Зчитування	orderNumber, користувач, items, суми, статус	2 000 на день	Високий		
10	Переглянути активність користувача	Зчитування	ім'я, email, кількість замовлень, останній вхід	500 на день	Середній		
11	Переглянути промоакції	Зчитування	код, тип знижки, значення, термін дії	1 000 на день	Низький		
12	Переглянути способи доставки	Зчитування	назва, ціна, безкоштовна доставка, регіони	1 000 на день	Низький		
13							

Рисунок 2.7—Таблиця дій їх типів та частоти виконання

Відповіді на контрольні питання

Як обрати між вбудованими документами та референсами при моделюванні даних у MongoDB?

Вибір залежить від шаблонів доступу до даних. Вбудовані документи кращі для даних, які часто зчитуються разом, мають невеликий розмір і рідко оновлюються. Референси кращі для окремих сутностей, які часто оновлюються, мають великий розмір або входять у зв'язки "багато-до-багатьох".

Що таке денормалізація в контексті MongoDB, і в яких випадках її варто застосовувати при проєктуванні моделі даних?

Денормалізація - це навмисне дублювання даних для покращення продуктивності читання. Її варто застосовувати для даних, які рідко оновлюються, але часто зчитуються разом, або коли потрібно уникнути дорогих операцій об'єднання.

Що таке View у MongoDB, як вона створюється, і які її обмеження?

View - це віртуальна колекція, результат агрегаційного запиту. Створюється командою `db.createView()`. Обмеження: лише для читання, не можна створювати індекси, продуктивність залежить від складності базового пайплайну.

Як змінити або видалити представлення?

Для видалення використовують `db.viewName.drop()`. Для зміни - видаляють і створюють нове представлення або використовують команду `collMod`.

Що таке обмежені колекції?

Це колекції фіксованого розміру, які автоматично видаляють найстаріші документи при заповненні. Використовуються для логування, кешування та черг повідомлень.

Як забезпечується узгодженість даних при дублюванні даних?

Через транзакції для критичних даних, паттерн "Versioned Documents", фонові процеси синхронізації, оптимістичне блокування та використання Change Streams для відстеження змін.