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: асист. каф. Андреєв П.І.

КРЕМЕНЧУК 2025

Лабораторна робота №10

Тема. Потоки змін в MongoDB

Мета роботи: навчитися створювати потоки змін для відстеження змін у базах даних у режимі реального часу.

Порядок виконання роботи

1.Отримати індивідуальний варіант завдання.

2.Створити Change Stream з прослуховуванням події change.

3.Відстежити подію з типом операції insert, потім виконати вставку документа.

4.Відстежити подію update, потім оновити документ.

5.Відстежити подію delete, потім видалити документ.

Виконання роботи

1.Відповідно до варіанту розробленого у 2–й лабораторній роботі продовдуємо роботу з потоками змін

Відповідно до завдання лабораторної роботи було розроблено наступний код:

```
const { MongoClient, ObjectId } = require('mongodb');
const uri = 'mongodb://127.0.0.1:27017/electronicsStore?replicaSet=rs0';
const client = new MongoClient(uri);
/**
 * Функція для налаштування та запуску прослуховувача потоку змін.
 * @param {Db} db - Об'єкт бази даних MongoDB.
 * @returns {ChangeStream} - Повертає екземпляр потоку змін
 */
async function setupChangeStream(db) {
   console.log("Налаштування потоку змін для колекції 'products'...");
   // Обираємо колекцію
   const collection = db.collection('products');
   // Ми також додамо 'fullDocument: "updateLookup"'
```

```
        // щоб бачити повний документ *після* оновлення.
        const changeStream = collection.watch([], { fullDocument: "updateLookup" });
        // Обробляємо подію 'change'
        changeStream.on('change', (change) => {
          console.log('\n--- Зфіксовано Зміну! ---');

          // Аналізуємо тип операції
          switch (change.operationType) {
            case 'insert':
              console.log('Тип операції: insert');
              console.log('Вставлений документ:', change.fullDocument);
              break;
            case 'update':
              console.log('Тип операції: update');
              console.log('ID документа:', change.documentKey._id);
              console.log('Оновлені поля:', change.updateDescription.updatedFields);
              console.log('Повний документ ПІСЛЯ оновлення:', change.fullDocument);
              break;
            case 'delete':
              console.log('Тип операції: delete');
              console.log('ID видаленого документа:', change.documentKey._id);
              break;
            default:
              console.log(`Інша операція: ${change.operationType}`);
          }
          console.log('-----------------------------');
```

```
        });
        changeStream.on('error', (error) => {
            // Ми перевіримо, чи це "очікувана" помилка при закритті
            if (error.codeName !== 'Interrupted' && error.constructor.name !==
'MongoClientClosedError') {
                console.error('Помилка в потоці змін:', error);
            }
        });
        console.log("Потік змін активний. Очікування операцій...");
        // === ЗМІНА 1: Повертаємо потік ===
        return changeStream;
    }
    /**
     * Функція, що виконує операції CUD (Create, Update, Delete)
     * @param {Db} db - Об'єкт бази даних MongoDB.
     */
    async function performOperations(db) {
        const collection = db.collection('products');
        let insertedId;
        try {
            // Затримка, щоб слухач встиг ініціалізуватися
            await new Promise(resolve => setTimeout(resolve, 2000));
            console.log("\n--- Виконання операцій ---");
            // Відстежити подію insert
            console.log("\n1. Виконання INSERT...");
            // Використовуємо ObjectId з вашого коду для сумісності
            const mockCategoryId = new ObjectId();
            const insertResult = await collection.insertOne({
                name: "Magic Mouse",
                sku: "MM-001",
```

```
            categoryId: mockCategoryId,
            price: 79.99,
            stock: 150,
            brand: "Apple",
            isActive: true,
            createdAt: new Date()
        });
        insertedId = insertResult.insertedId;
        console.log(`Документ вставлено з ID: ${insertedId}`);
        await new Promise(resolve => setTimeout(resolve, 1500)); // Пауза для наочності
        // Відстежити подію update
        console.log("\n2. Виконання UPDATE...");
        await collection.updateOne(
            { _id: insertedId },
            {
                $set: { price: 74.99, stock: 149 },
                $currentDate: { updatedAt: true }
            }
        );
        console.log("Документ оновлено.");
        await new Promise(resolve => setTimeout(resolve, 1500)); // Пауза для наочності
        // Відстежити подію delete
        console.log("\n3. Виконання DELETE...");
        await collection.deleteOne({ _id: insertedId });
        console.log("Документ видалено.");
    } catch (error) {
        // Перевірка на помилку валідації
        if (error.code === 121) {
```

```
            console.error(
              '\n--- ПОМИЛКА ВАЛІДАЦІЇ ---' +
              '\nНе вдалося вставити документ. Переконайтеся, що у вашому ' +
              'тестовому документі (Magic Mouse) є всі *required* поля,' +
              '\nвизначені у валідаторі колекції `products`.' +
              '\nПоточна помилка:', error.errInfo.details
            );
          } else {
            console.error("Помилка під час виконання операцій:", error);
          }
        }
      }
      /**
       * Головна функція
       */
      async function main() {
        // === ЗМІНА 2: Оголошуємо змінну для потоку ===
        let changeStreamInstance = null;
        try {
          await client.connect();
          console.log('Підключено до MongoDB (Replica Set: rs0)');
          const db = client.db('electronicsStore'); // Явно вказуємо БД
          // === ЗМІНА 3: Зберігаємо екземпляр потоку ===
          // Запускаємо прослуховувач
          changeStreamInstance = await setupChangeStream(db);
          // Виконуємо операції
          await performOperations(db);
          // Дамо час потоку обробити останню подію перед закриттям
```

```
            console.log("\nЗавершення... Очікуємо 3 секунди і закриваємо з'єднання.");
            await new Promise(resolve => setTimeout(resolve, 3000));
        } catch (error) {
            if (error.codeName === 'FailedToSatisfyReadPreference') {
                console.error(
                    '\n--- ПОМИЛКА: Не вдалося підключитися. ---' +
                    '\nПереконайтеся, що ваш MongoDB запущено як НАБІР РЕПЛІК (Replica Set)' +
                    '\ni що ваш URI містить `?replicaSet=rs0`'
                );
            } else {
                console.error('Виникла непередбачувана помилка:', error);
            }
        } finally {
            // === ЗМІНА 4: "Граціозне" закриття ===
            console.log("\nЗакриття ресурсів...");
            if (changeStreamInstance) {
                await changeStreamInstance.close();
                console.log('Потік змін успішно закрито.');
            }
            await client.close();
            console.log('З\'єднання з MongoDB закрито.');
        }
    }

    // Запуск
main().catch(console.error);
```

```
З'єднання з MongoDB закрито.
PS D:\Mongo_db_course\Mongo_db_course\mongo-project> node runLab.js
Підключено до MongoDB (Replica Set: rs0)
Налаштування потоку змін для колекції 'products'...
Потік змін активний. Очікування операцій...

--- Виконання операцій ---

1. Виконання INSERT...
Документ вставлено з ID: 690fa20492a709b4715012fe

--- Зфіксовано Зміну! ---
Тип операції: insert
Вставлений документ: {
  _id: new ObjectId('690fa20492a709b4715012fe'),
  name: 'Magic Mouse',
  sku: 'MM-001',
  categoryId: new ObjectId('690fa20492a709b4715012fd'),
  price: 79.99,
  stock: 150,
  brand: 'Apple',
  isActive: true,
  createdAt: 2025-11-08T20:03:16.577Z
}
------------------------------
```

Рисунок 1.1 – Результат виконання insert

```
------------------------------

2. Виконання UPDATE...
Документ оновлено.

--- Зфіксовано Зміну! ---
Тип операції: update
ID документа: new ObjectId('690fa20492a709b4715012fe')
Оновлені поля: { price: 74.99, stock: 149, updatedAt: 2025-11-08T20:03:18.110Z }
Повний документ ПІСЛЯ оновлення: {
  _id: new ObjectId('690fa20492a709b4715012fe'),
  name: 'Magic Mouse',
  sku: 'MM-001',
  categoryId: new ObjectId('690fa20492a709b4715012fd'),
  price: 74.99,
  stock: 149,
  brand: 'Apple',
  isActive: true,
  createdAt: 2025-11-08T20:03:16.577Z,
  updatedAt: 2025-11-08T20:03:18.110Z
}
------------------------------
```

Рисунок 1.2 – Результат виводу оновлення

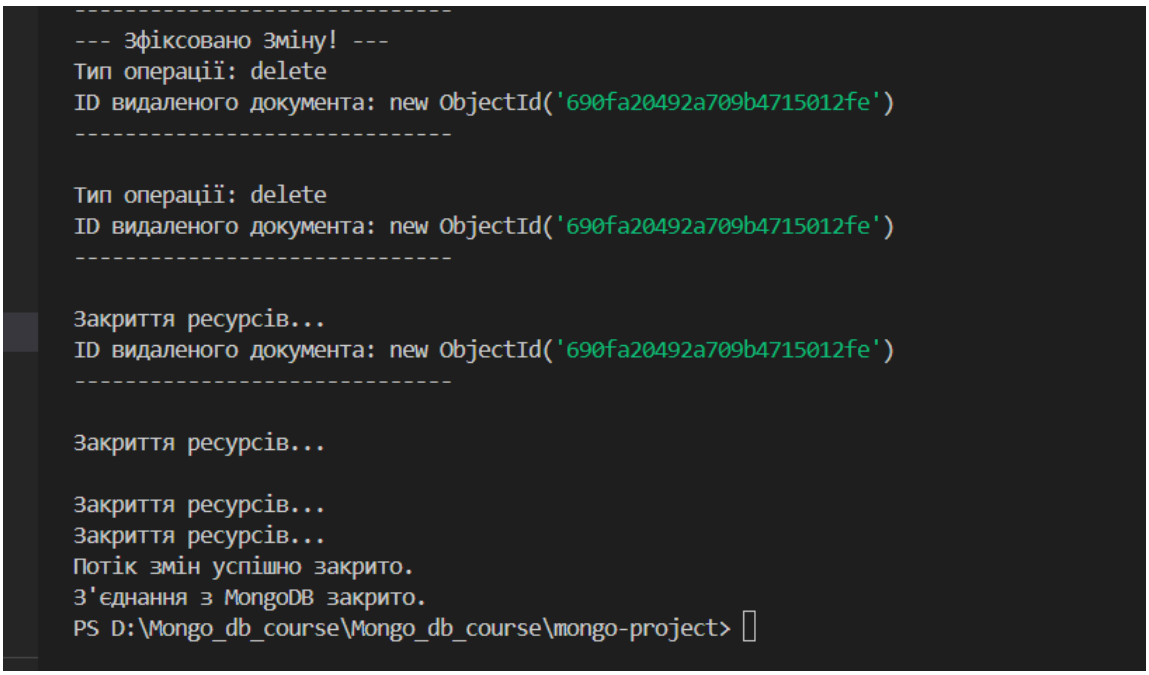

Рисунок 1.3 – Результат операції видалення

Обгрунтування

Звісно. Обґрунтування вибору коду базується на вашому початковому завданні та структурі бази даних, яку ви самі створили.

Ми обрали підключення до бази даних electronicsStore і вказали ?replicaSet=rs0, оскільки це ваша робоча база даних, а rs0 – це технічна вимога для роботи Change Streams, яку ми налаштували.

Для відстеження змін ми обрали колекцію products. Це було логічне рішення, оскільки це центральна колекція магазину, яка добре підходить для демонстрації всіх трьох операцій: insert, update і delete. Ми могли б слухати всю базу (db.watch()), але моніторинг конкретної колекції (collection.watch()) є більш точним для цього завдання.

У налаштуваннях потоку ми використали опцію { fullDocument: "updateLookup" }. Це було важливо. За замовчуванням, подія update показує лише що змінилося. Ця опція змушує MongoDB показати повний документ після оновлення, що є набагато інформативнішим для демонстрації.

У функції, що виконує операції, ми створювали тестовий документ ("Magic Mouse") таким чином, щоб він відповідав вашому валідатору для колекції products. Ми включили всі обов'язкові поля, які ви вказали (name, sku, price, stock, categoryId).

Оскільки ваш валідатор вимагає categoryId типу ObjectId, ми згенерували його на льоту (const mockCategoryId = new ObjectId()), щоб тест пройшов валідацію.

Операція updateOne була обрана, щоб показати класичний сценарій e- commerce (зміна ціни та залишку). Ми також додали $currentDate: { updatedAt: true }, щоб коректно оновити поле updatedAt, яке було у вашій схемі.

Операція deleteOne видаляла той самий документ, який ми створили. Це "прибирало за собою" і гарантувало, що наш потік змін успішно зафіксує третій тип операції – delete.

Відповіді на контрольні питання

1. Що таке потоки змін (Change Streams) у MongoDB?

Відповідь: Це API, яке дозволяє додаткам у реальному часі "підписуватися" на зміни даних (такі як вставка, оновлення, видалення) в колекції, базі даних або у всьому кластері.

2. Яка умова потрібна для роботи Change Streams у MongoDB?

Відповідь: Головна умова – MongoDB має бути розгорнутий як набір реплік (replica set) або шардований кластер (sharded cluster).

3. Який метод використовується для створення Change Stream на колекції?

Відповідь:Метод`.watch()`.Наприклад: `db.collection('myCollection').watch()`.

4. Яка подія обробляється методом `.on()` для отримання змін у Change Stream?

Відповідь: Подія `'change'`. Наприклад: `myStream.on('change', (data) => { ... })`.

5. Які типи операцій відслідковують Change Streams?

Відповідь: Вони відстежують `insert`, `update`, `delete`, `replace` (заміна), `drop` (видалення колекції), `rename` (перейменування) та `invalidate` (сигнал про закриття потоку).

6. Чи можна використовувати Change Streams на standalone сервері MongoDB?

Відповідь Ні, не можна.Вони покладаються на журнал операцій (oplog), який є частиною механізму реплікації.

7. Який об’єкт містить інформацію про конкретну зміну у Change Stream?

Відповідь:Документ про зміну (change event document). Це об'єкт, який передається в обробник події та містить такі поля, як `operationType`, `fullDocument`, `documentKey` тощо.