

Департамент образования города Москвы  
Государственное автономное образовательное учреждение высшего  
образования города Москва «Московский Городской Педагогический  
Университет»

Институт цифрового образования  
Департамент информатики, управления и технологий

ЛАБОРАТОРНАЯ РАБОТА №2.1 (Вар.19)  
по дисциплине «Инструменты для хранения и обработки больших данных»  
Тема: «Изучение методов хранения данных на основе NoSQL»  
Направление подготовки 38.03.05 – «бизнес-информатика»  
Профиль подготовки «Аналитика данных и эффективное управление» (очная  
форма обучения)

Выполнил:  
Студент группы АДЭУ-221  
Черенков Иван Романович

Преподаватель:  
Босенко Т. М.  
к.т.н., доц. департамента  
информатики, управления и  
технологий

Москва  
2025

Цель работы – изучение и практическое освоение трех видов NoSQL баз данных: документо-ориентированной, графовой и типажа ключ-значение. Приобретение навыков создания, заполнения, анализа данных в каждой из вышеописанных систем. Выполнение запросов для извлечения нужной информации для проработки навыка работы с нереляционными моделями данных.

Вариант индивидуального самостоятельного задания – №19.

### **Выполнение индивидуального задания**

Постановка задачи:

1. Обновить все документы, где year меньше 1970, установив им новое поле era со значением "Old School" (MongoDB)
2. Найти актеров, которые снимались и в "The Matrix", и в "Cloud Atlas" (Neo4j)
3. Используя SETNX, попытаться установить ключ lock:resource:123. Проверить результат. Попытаться установить его еще раз. (Redis)

#### *1.1 Задание в MongoDB*

Задание №1. Обновить все документы, где year меньше 1970, установив им новое поле era со значением "Old School"

Сначала я решил вывести количество фильмов со значением «year» меньше 1970, используя тег «lower then» – «\$lt» и метод «.count()» (рисунок 1).

```
|  
filmdb> db.movies.find({year: {$lt: 1970}}).count()  
10  
filmdb>
```

Рисунок 1. Результат выполнения запроса на количество всех фильмов младше 1970 года

Далее, используя метод «updateMany» определил новый столбец и установил в него значение Old School для всех документов, в которых поле «year» меньше значения 1970 и вывел их названия с количеством (рисунок 2 и 3).

```
filmdb> db.movies.updateMany({year: {$lt: 1970}}, {$set: {era: "Old School"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 10,
  modifiedCount: 10,
  upsertedCount: 0
}
filmdb>
```

Ln 203, Col 30 Spa

Рисунок 2. Результат выполнения задания №1 в системе MongoDB

```
filmdb> db.movies.find({era: "Old School"}, {title: 1}).count()
10
filmdb> db.movies.find({era: "Old School"}, {title: 1})
[
  { _id: ObjectId('68bf29dfcf1f59882afa335b'), title: '12 Angry Men' ✓ },
  {
    _id: ObjectId('68bf29dfcf1f59882afa335e'),
    title: 'The Good, the Bad and the Ugly' ✓
  },
  { _id: ObjectId('68bf29dfcf1f59882afa3368'), title: 'Seven Samurai' ✓ },
  {
    _id: ObjectId('68bf29dfcf1f59882afa336d'),
    title: "It's a Wonderful Life" ✓
  },
  { _id: ObjectId('68bf29dfcf1f59882afa3375'), title: 'Psycho' ✓ },
  { _id: ObjectId('68bf29dfcf1f59882afa3377'), title: 'City Lights' ✓ },
  { _id: ObjectId('68bf29dfcf1f59882afa3378'), title: 'Casablanca' ✓ },
  {
    _id: ObjectId('68bf29dfcf1f59882afa3379'),
    title: 'Once Upon a Time in the West' ✓
  },
  { _id: ObjectId('68bf29dfcf1f59882afa337b'), title: 'Modern Times' ✓ },
  { _id: ObjectId('68bf29dfcf1f59882afa3382'), title: 'Rear Window' ✓ }
]
filmdb>
```

Ln 203, Col 30

Рисунок 3. Проверка выполнения задания №1 в MongoDB

Используемые команды:

```
// Количество фильмов младше 1970 года
db.movies.find({year: {$lt: 1970}}).count()

// Добавление столбца
db.movies.updateMany({year: {$lt: 1970}}, {$set: {era: "Old School"}})

// Проверка
db.movies.find({era: "Old School"}, {title: 1}).count()
db.movies.find({era: "Old School"}, {title: 1})
```

### 1.2 Задание в Neo4j

Задание №2. Найти актеров, которые снимались и в "The Matrix", и в "Cloud Atlas" (Neo4j).

Для начала я решил сделать Cypher-запрос, который позволил бы увидеть всех актеров, которые участвовали в съемках хоть какого-либо фильма, таким образом получилось вывести следующий список (рисунок 4):



The screenshot shows the Neo4j Cypher query editor and results viewer. The query is:

```
1 MATCH (p:Person)-[:ACTED_IN]-(m:Movie)
2 RETURN p.name AS actor_name
```

The results are displayed in a table with the following data:

	actor_name
1	"Emil Eifrem"
2	"Hugo Weaving"
3	"Laurence Fishburne"
4	"Carrie-Anne Moss"
5	"Keanu Reeves"
6	"Hugo Weaving"
7	

At the bottom, a status message reads: "Started streaming 172 records after 7 ms and completed after 13 ms."

Рисунок 4. Результат запроса на вывод всех актеров, участвующих в съемках всех фильмов

Далее я начал потихоньку добавлять новые условия и отфильтровал список по фильмам, указанным в условиях (рисунок 5).



Рисунок 5. Результат Cypher-запроса на вывод всех актеров, участвующих в съемках двух фильмов (по условиям задачи)

Как видим, хоть запрос и выполнился, но все равно работает несколько неправильно. Cypher-запрос выводит актеров по двум фильмам, не группируя их, то есть, например актер «Hugo Weaving» повторяется в списке дважды. Немного видоизменив запрос, увидим новый результат (рисунок 6).



Рисунок 6. Результат выполнения запроса с проверкой уникальности имен актеров

Теперь вывод правильный, однако условия задачи еще не решены. Изменив запрос и добавив в него фильтрацию по количеству фильмов, увидим список тех актеров, кто снимался как в фильме "The Matrix", так и в "Cloud Atlas" (рисунок 7).

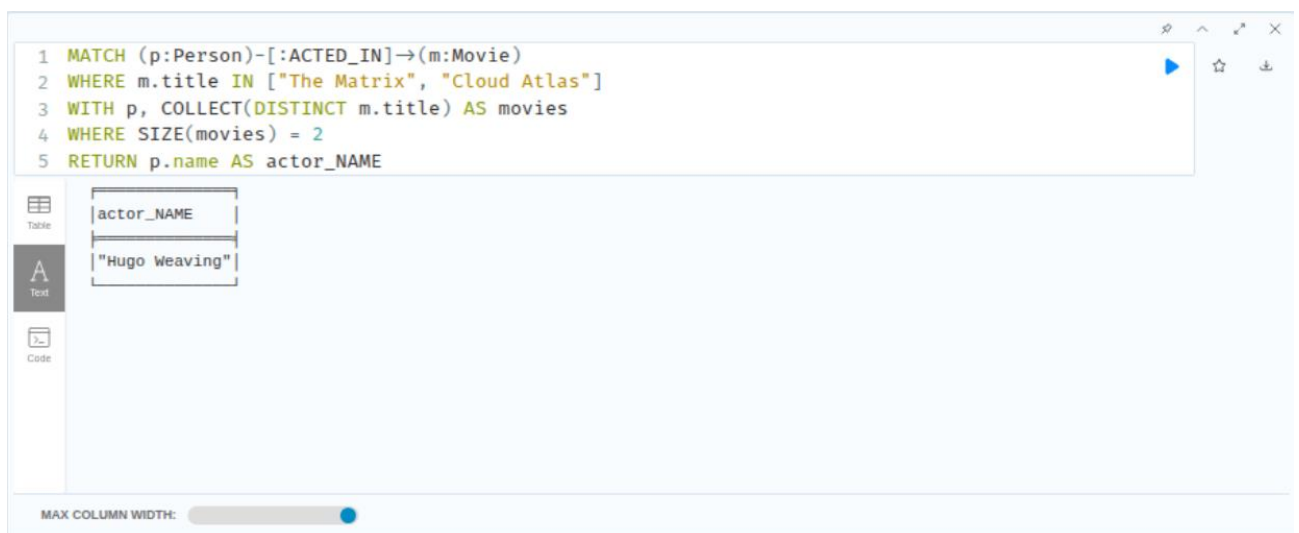


Рисунок 7. Итоговый результат выполнения задания Neo4j

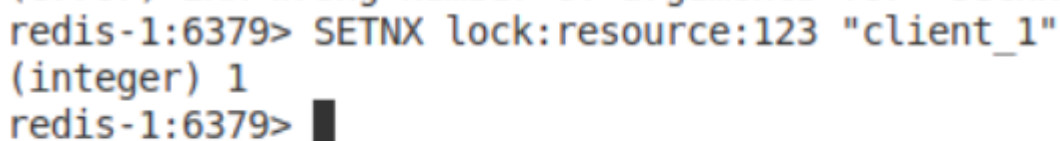
Используемые команды:

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.title IN ["The Matrix", "Cloud Atlas"]
WITH p, COLLECT(DISTINCT m.title) AS movies
WHERE SIZE(movies) = 2
RETURN p.name AS actor_name
```

### 1.3 Задание в Redis

Задание №3. Используя SETNX, попытаться установить ключ lock:resource:123. Проверить результат. Попытаться установить его еще раз. (Redis)

SETNX (Set if Not eXists) выполнение в первый раз изображено на рисунке 4.



```
redis-1:6379> SETNX lock:resource:123 "client_1"
(integer) 1
redis-1:6379> █
```

Рисунок 4. Результат выполнения SETNX в первый раз

Как видим при первом выполнении output вывел 1 (единицу), а значит set произошел успешно, так как lock:resource:123 не существовало до этого, но при выполнении этой команды повторно (но уже с другим значением), видим вывод «0», что значит, что команда не выполнилась, так как ключ-значение уже существует (рисунок 5).

```
redis-1:6379> SETNX lock:resource:123 "client_2"  
(integer) 0  
redis-1:6379> █
```

0 ▲ 0

Рисунок 5. Результат повторного выполнения SETNX

Теперь проведем проверку значения (рисунок 6).

```
(integer) 0  
redis-1:6379> GET lock:resource:123  
"client_1"  
redis-1:6379> █
```

0 ▲ 0

Рисунок 6. Проверка ключ-значения lock:resource:123

Как видим, ключ lock:resource:123 оставил в себе значение client\_1, которое было введено в первый раз.

Используемые команды:

```
SETNX lock:resource:123 "client_1"  
SETNX lock:resource:123 "client_1"  
GET lock:resource:123
```

### Выводы:

В ходе практической работы была развернута и изучена разнородная NoSQL-экосистема с использованием контейнеров Docker и пройдены следующие этапы работы:

- Изучена документоориентированная СУБД MongoDB
  - Структура данных: база данных, коллекции, документы, поля
  - Индексация
  - Встроенные типы данных



- CRUD-операции, теги и их семантика (insertOne, insertMany, find, findOne, \$gt, \$gte, \$lt, \$lte, \$ne, \$exists, \$or, \$and, updateOne, updateMany, deleteOne, deleteMany и так далее)
- Агрегация данных (\$match, \$group, \$unwind и др.)
- Текстовый поиск (через текстовый индекс)
- Изучена СУБД Redis
  - Утилита командной строки и веб-интерфейс
  - Структура данных: String, List, Set, Sorted Set, Hash
  - Изучены операции и их семантика (SET, GET, DEL, EXISTS, KEYS, INCR, DECR, MSET, MGET, EXPIRE, TTL, LLEN, LRange другие)
- Изучена СУБД Neo4j
  - Инструменты СУБД (веб-интерфейс, командная строка Cypher)
  - Ключевые концепции и структура данных (узлы, отношения, схема данных)
  - Операции и запросы Cypher (создание узлов и отношений CREATE, поиск данных MATCH, возврат результатов RETURN, сопоставление с условиями WHERE и так далее)

При выполнении индивидуальных заданий данной лабораторной работы возникали некоторые трудности. Так, например, из-за не слишком глубокого знания семантики работы с СУБД возникало непонимание и приходилось заходить в документацию СУБД и искать решение проблемы там (MongoDB), также была масштабная проблема, процессор моего ноутбука не поддерживает MongoDB версии 5.0+, а была установлена 7.7, в виду чего пришлось экстренно переносить образ виртуальной машины на персональный компьютер дома и настраивать локальную сеть между компьютером и ноутбуком для получения удаленного доступа к компьютеру (в итоге было настроено с помощью Radmin и подключения через Radmin Server)