Работа на лекции 1

Цель работы: получить практические навыки работы с документо-ориентированной базой данных MongoDB путем выполнения основных операций по управлению данными.

Задачи работы:

1. Изучить основы работы с MongoDB:
   - Подключение к базе данных.
   - Создание коллекций.
   - Добавление документов.
2. Освоить основные операции с документами:
   - Поиск документов.
   - Обновление документов.
   - Удаление документов.
3. Научиться работать с индексами:
   - Создание индексов.
   - Использование индексов для оптимизации.
4. Освоить базовые принципы агрегации данных:
   - Группировка данных.
   - Подсчет статистики.
   - Сортировка результатов.
5. Научиться выполнять текстовый поиск:
   - Создание текстовых индексов.
   - Поиск по текстовым полям.

Ход работы

Предполагаем, что платформа devops_dba_25.ova https://disk.yandex.ru/d/gagWU_zn1erR8g запущена и доступна.

Подключение к среде MongoDB

Использование утилиты командной строки MongoDB

Можно найти утилиту командной строки mongo внутри контейнера Docker MongoDB, работающего как часть платформы. Подключитесь к хосту Docker и выполните следующую команду docker exec

```
sudo docker exec -ti mongo-1 mongosh -u "root" -p "abc123!"
```

Это позволит вам подключиться к контейнеру mongo и запустить оболочку mongo внутри него.

Вы должны увидеть вывод, аналогичный приведенному ниже.

```
bigdata@bigdata:~$ docker exec -ti mongo-1 mongosh -u "root" -p "abc123!"
Current Mongosh Log ID: 67b215162e08a3633d544ca6
Connecting to: mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.8
Using MongoDB:        7.0.16
Using Mongosh:        2.3.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
```

```
To help improve our products, anonymous usage data is collected and sent to MongoDB
periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2025-02-16T16:10:32.061+00:00: Using the XFS filesystem is strongly recommended with the
WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2025-02-16T16:10:33.459+00:00: vm.max_map_count is too low
------
test>
```

Теперь вы находитесь в командной строке MongoDB, готовой к выполнению любых операторов MongoDB. Также можно увидеть версию сервера MongoDB и оболочки MongoDB.

Оболочка запускает JavaScript. Есть несколько глобальных команд, которые вы можете выполнить, например help или exit. Команды, которые вы выполняете для текущей базы данных, выполняются для объекта db, например db.help() или db.stats().

Команды, которые выполняются для определенной коллекции, выполняются для объекта db.COLLECTION_NAME, например db.movies.help() или db.movies.countDocuments().

Выполнив команду db.help(), получите список команд, которые вы можете выполнить для объекта db.

Примечание: Поскольку это оболочка JavaScript, если выполните метод и опустите скобки (), увидите тело метода, а не выполнение метода. Первый раз, когда выполните запрос и получите ответ, который начинается с function (...){, вас это не удивляло. Например, если введете db.help (без скобок), увидите внутреннюю реализацию метода help.

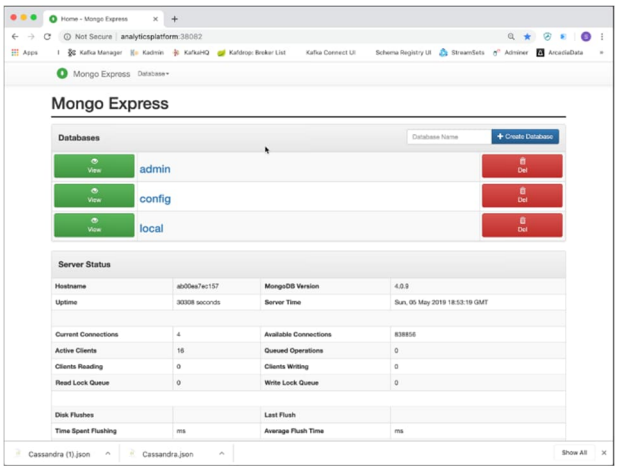
Использование браузерного графического интерфейса

Вместо работы в командной строке и, следовательно, необходимости подключаться к Docker Host, можем использовать браузерный графический интерфейс для доступа к MongoDB. В рамках платформы доступны две браузерные утилиты.

Mongo Express

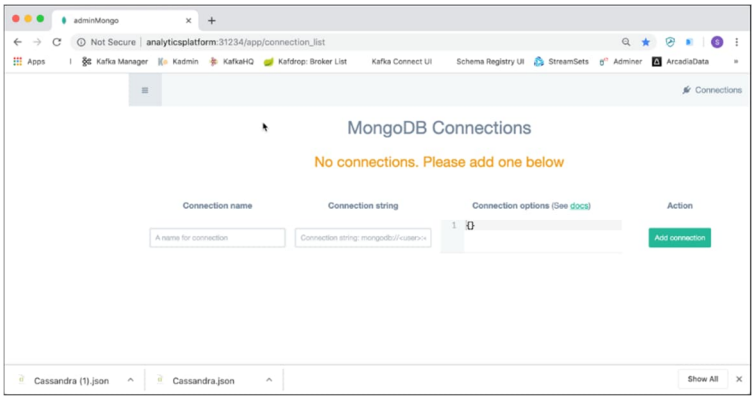
Первый — Mongo Express, веб-интерфейс администратора MongoDB, написанный с помощью Node.js, Express и Bootstrap3.

В окне браузера перейдите на http://localhost:28203/, с именем пользователя admin и паролем pass, переходим на главный экран, как показано ниже.

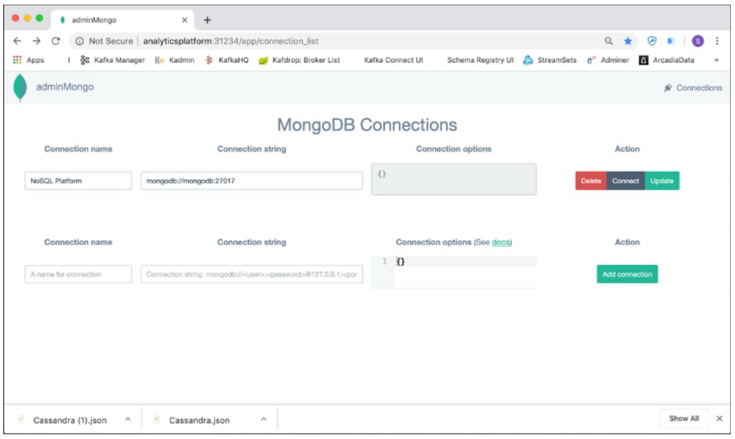


Admin Mongo

Второй — Admin Mongo, открытый исходный интерфейс администратора для MongoDB. В окне браузера перейдите на http://localhost:28204/ и войдите с именем пользователя admin и паролем pass, и вы должны увидеть главный экран, как показано ниже.



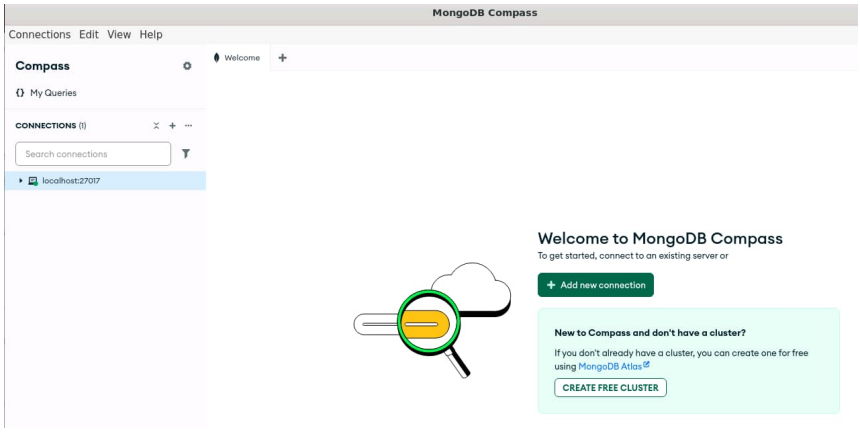
Чтобы подключиться к экземпляру MongoDB, добавьте новое подключение к Admin Mongo. Введите Data Platform в поле **Имя подключения** и mongodb://mongo-1:27017 в поле **Строка подключения** и нажмите **Добавить подключение**. Должно появиться сообщение о том, что подключение было успешно добавлено.



Нажатие кнопки **Подключить** открывает страницу сведений об администрировании Mongo для подключения.

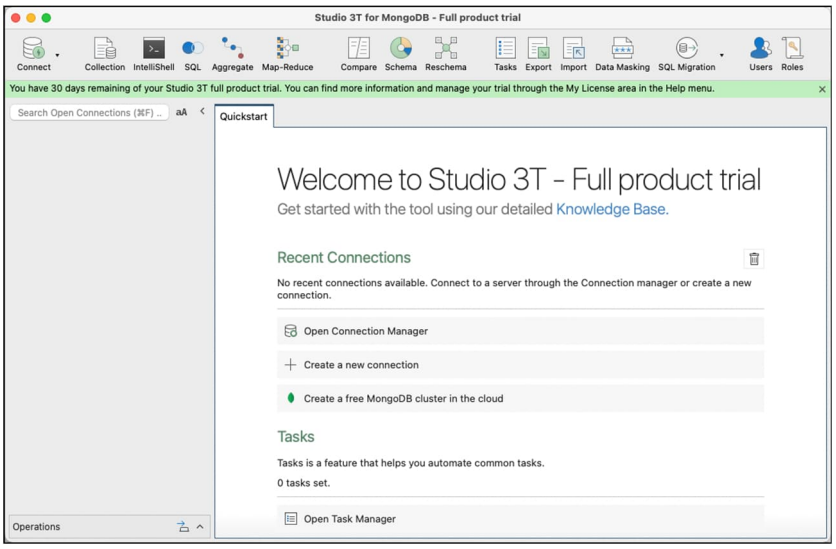
Использование настольных приложений

Существуют также различные настольные приложения для управления и администрирования MongoDB, которые можно загрузить и установить на рабочем столе, например, Mongo Compass. Оттуда можете подключиться как к локальному, так и к удаленному экземпляру Mongo.



Studio 3T (**ранее известная как** Robo 3T **или** Robomongo)

Studio 3T - приложение, встраивающее оболочку MongoDB. Оно доступно для Windows, Mac и Linux.



Работа с MongoDB

Рассмотрим основные механизмы работы с MongoDB. Это основа для понимания MongoDB.

6 **Основных Концепций** MongoDB

1. **База данных** (Database)
   - Верхний уровень организации данных
   - Аналог схемы в традиционных СУБД
   - Содержит коллекции данных
   - В одном экземпляре MongoDB может быть несколько баз данных
2. **Коллекции** (Collections)
   - Аналог таблиц в реляционных БД
   - Группа связанных документов
   - Динамическая схема
   - Пример: users, products, orders
3. **Документы** (Documents)
   - Аналог строк в таблице
   - Хранятся в формате BSON

- 📌 Каждый документ имеет уникальный _id
- ✨ Пример:

```
{
  "_id": ObjectId("5099803df3f4948bd2f98391"),
  "name": "iPhone",
  "price": 699,
  "category": "Electronics"
}
```

4. **Поля** (Fields)
   - 📋 Аналог столбцов в таблице
   - 🔤 Могут содержать различные типы данных
   - 🔗 Поддерживают вложенные структуры
   - 🎯 Пример:

```
{
  "name": "String",
  "age": "Number",
  "address": {
    "street": "String",
    "city": "String"
  },
  "hobbies": ["Array", "of", "Strings"]
}
```

5. **Индексы** (Indexes)
   - 🚀 Ускоряют поиск документов
   - 📑 Поддерживают различные типы индексов
   - ⚠️ Влияют на производительность записи
   - 💻 Пример создания:

```
db.collection.createIndex({ "field": 1 })  // 1 для возрастания, -1 для убывания
```

6. **Курсоры** (Cursors)
   - 👆 Указатель на результат запроса
   - 🔄 Позволяют итерировать по результатам
   - 📊 Поддерживают методы обработки данных
   - 🎮 Пример использования:

```
const cursor = db.collection.find()
cursor.forEach(doc => {
  console.log(doc)
})
```

Практический пример всех концепций:

```
// База данных: shop
use shop
```

```
// Коллекция: products
db.products.insertOne({
  name: "Laptop",
  price: 999.99,
  specs: {
    cpu: "Intel i7",
    ram: "16GB",
    storage: "512GB SSD"
  },
  tags: ["electronics", "computers"]
})

// Создание индекса
db.products.createIndex({ "name": 1 })

// Использование курсора
const cursor = db.products.find({ price: { $gt: 500 } })
while(cursor.hasNext()) {
  printjson(cursor.next())
}
```

Важные замечания:

- 🔑 Каждый документ должен иметь уникальный _id
- 📈 Индексы улучшают чтение, но замедляют запись
- 🔄 Курсоры автоматически закрываются через 10 минут
- 📝 Коллекции не требуют предварительного определения схемы
- 💡 MongoDB оптимизирована для частых операций чтения/записи

Структура и Терминология MongoDB

Основная Структура

Сравнение с Реляционными БД

MongoDB	**Реляционные БД**	**Ключевое отличие**
База данных	База данных	Аналогичны
Коллекция	Таблица	Коллекция не имеет фиксированной схемы
Документ	Строка	Документ может иметь уникальную структуру
Поле	Столбец	Поля определяются на уровне документа

Практический Пример

```
// Переключение на базу данных
use filmdb

// Проверка существующих коллекций
```

```
db.getCollectionNames()  // Вернёт: []

// Создание документа в новой коллекции
db.movies.insertOne({
   title: "Inception",
   year: 2010,
   director: "Christopher Nolan",
   genres: ["Sci-Fi", "Action"],
   ratings: {
      imdb: 8.8,
      metacritic: 74
   }
})
```

Важные Особенности

Гибкая Схема

- Каждый документ может иметь свою структуру
- Нет необходимости предварительно определять схему
- Можно добавлять новые поля "на лету"

Создание Структур

```
// Документ 1
{
   title: "Inception",
   year: 2010
}

// Документ 2 в той же коллекции
{
   title: "The Matrix",
   director: "Wachowski",
   cast: ["Keanu Reeves", "Laurence Fishburne"]
}
```

Работа с Данными

```
// Создание индекса
db.movies.createIndex({ "title": 1 })

// Использование курсора
const movieCursor = db.movies.find()
while(movieCursor.hasNext()) {
   printjson(movieCursor.next())
}
```

Примечания

1. База данных создаётся автоматически при создании первой коллекции.
2. Коллекции создаются автоматически при добавлении первого документа.

3. Каждый документ может иметь свой набор полей.
4. Курсоры оптимизируют работу с большими наборами данных.

Команды для Начала Работы

```
use filmdb                // Переключение/создание БД
db.getCollectionNames()   // Список коллекций
db.movies.insertOne({...})  // Добавление документа
```

Практическая работа 1. **Создание документов в** MongoDB

1. **Подключение к существующей** MongoDB
   1. Откройте MongoDB Compass
   2. В поле URI введите:

mongodb://root:abc123!@localhost:27017

ИЛИ заполните поля отдельно:

- Hostname: localhost
- Port: 27017
- Authentication: Username/Password
- Username: root
- Password: abc123!

   3. Нажмите "Connect"

2. **После успешного подключения**
   1. Вы увидите список баз данных
   2. Создадим новую базу данных:
      - Нажмите "Create Database"
      - Database Name: filmdb
      - Collection Name: movies
      - Нажмите "Create Database"

3. **Вставка Документа в** MongoDB Compass. **Вставка документа фильма в коллекцию** movies

Структура документа movies

```
{
  // Основная информация
  "id": String,          // ID фильма
  "title": String,       // Название
  "year": Number,        // Год выпуска
  "runtime": Number,     // Продолжительность в минутах

  // Массивы данных
  "languages": [String],  // Список языков
  "genres": [String],     // Список жанров

  // Рейтинги
  "rating": Number,      // Оценка
  "votes": Number,       // Количество голосов

  // Текстовое описание
```

```
  "plotOutline": String,   // Сюжет

  // Медиа
  "coverUrl": String,      // URL постера

  // Вложенные массивы объектов
  "actors": [{           // Актёры
    "actorID": String,
    "name": String
  }],
  "directors": [{        // Режиссёры
    "directorID": String,
    "name": String
  }],
  "producers": [{        // Продюсеры
    "producerID": String,
    "name": String
  }]
}
```

3.1. **Создание Документа**

1. В MongoDB Compass:
   - Выберите базу данных filmdb
   - Выберите коллекцию movies
   - Нажмите "Add Data" → "Insert Document"
2. В окне редактора вставьте JSON:


```
{
  "id": "0110912",
  "title": "Pulp Fiction",
  "year": 1994,
  "runtime": 154,
  "languages": ["en", "es", "fr"],
  "rating": 8.9,
  "votes": 2084331,
  "genres": ["Crime", "Drama"],
  "plotOutline": "Jules Winnfield (Samuel L. Jackson) and Vincent Vega (John Travolta) are two hit men who are out to retrieve a suitcase stolen from their employer, mob boss Marsellus Wallace (Ving Rhames). Wallace has also asked Vincent to take his wife Mia (Uma Thurman) out a few days later when Wallace himself will be out of town. Butch Coolidge (Bruce Willis) is an aging boxer who is paid by Wallace to lose his fight. The lives of these seemingly unrelated people are woven together comprising of a series of funny, bizarre and uncalled-for incidents.",
  "coverUrl": "https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBlZi00MTRlLWFjM2ItYzViMjE3YzI5MjIjXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_SY150_CR1,0,101,150_.jpg",
```

```
    "actors": [
      { "actorID": "0000619", "name": "Tim Roth"},
      { "actorID": "0001625", "name": "Amanda Plummer"},
      { "actorID": "0522503", "name": "Laura Lovelace"},
      { "actorID": "0000237", "name": "John Travolta"},
      { "actorID": "0000168", "name": "Samuel L. Jackson"},
      { "actorID": "0482851", "name": "Phil LaMarr"},
      { "actorID": "0001844", "name": "Frank Whaley"},
      { "actorID": "0824882", "name": "Burr Steers"},
      { "actorID": "0000246", "name": "Bruce Willis"},
      { "actorID": "0000609", "name": "Ving Rahmes"},
      { "actorID": "0000235", "name": "Uma Thurman"},
      { "actorID": "0000233", "name": "Quentin Tarantino"}
    ],
    "directors": [
      { "directorID": "0000233", "name": "Quentin Tarantino"}
    ],
    "producers": [
      { "producerID": "0004744", "name": "Lawrence Bender"},
      { "producerID": "0000362", "name": "Danny DeVito"},
      { "producerID": "0321621", "name": "Richard N. Gladstein"},
      { "producerID": "0787834", "name": "Michael Shamberg"},
      { "producerID": "0792049", "name": "Stacey Sher"},
      { "producerID": "0918424", "name": "Bob Weinstein"},
      { "producerID": "0005544", "name": "Harvey Weinstein"}
    ]
  }
```

3. Нажмите "Insert"

4. **Проверка Вставки**

После вставки:

1. Обновите вид коллекции
2. Вы должны увидеть новый документ
3. MongoDB автоматически добавит поле _id

5. **Структура Документа**

Документ содержит:

- Простые поля (id, title, year, etc.)
- Массивы (languages, genres)
- Вложенные массивы объектов (actors, directors, producers)

6. **Поиск Документа**

В Compass:

1. Перейдите во вкладку "Find"
2. Введите фильтр:

```
{
```

```
  "title": "Pulp Fiction"
}
```

3. Нажмите "Find"

7. **Возможные Проблемы**

Если возникает ошибка:

- Проверьте формат JSON (все кавычки должны быть двойными)
- Убедитесь, что все скобки закрыты
- Проверьте, что нет trailing comma (запятой после последнего элемента)

8. **Проверка данных**

- В коллекции movies должен появиться новый документ
- Можно использовать вкладку "Find" для поиска данных
- Попробуйте фильтр: {title: "Pulp Fiction"}

Дополнительно

- Используйте вкладку "Indexes" для создания индексов
- "Explain Plan" поможет оптимизировать запросы
- "Aggregation" для сложных запросов
- "Schema" покажет структуру документов

Результат вставки

```
{
  acknowledged: true,
  insertedId: ObjectId('66cb796c8137d3ecf9c76a8d')
}
```

Важные моменты:

1. MongoDB автоматически создаёт _id, если не указан
2. Документ может содержать:
    - Простые типы (строки, числа)
    - Массивы
    - Вложенные объекты
    - Массивы объектов

Проверка вставки

```
// Поиск вставленного документа
db.movies.findOne({ "title": "Pulp Fiction" })

// Подсчёт документов в коллекции
db.movies.countDocuments()
```

Полная версия размещена на портале Github

https://github.com/BosenkoTM/nosql-workshop/blob/main/04-working-with-mongodb/README.md

Варианты самостоятельной работы.

Вариант 1.

1. Найдите все фильмы жанра "Drama" с рейтингом выше 9.0.
2. Подсчитайте количество фильмов для каждого года выпуска.
3. Увеличьте на 1000 количество голосов у фильма "The Matrix".
4. Создайте индекс по полю year.
5. Найдите всех актеров, снявшихся более чем в 2 фильмах.

Вариант 2.

1. Найдите все фильмы жанра "Action" или "Sci-Fi" после 2000 года.
2. Обновите рейтинг фильма "Fight Club" на 8.5.
3. Создайте текстовый индекс по полям title и plotOutline.
4. Найдите фильмы, содержащие слово "love" в описании или названии.
5. Подсчитайте средний рейтинг фильмов по жанрам.

Вариант 3.

1. Найдите все фильмы с рейтингом между 8.5 и 9.0.
2. Добавьте новое поле views со значением 0 всем фильмам.
3. Удалите все фильмы с рейтингом ниже 8.0.
4. Создайте составной индекс по полям year и rating.
5. Найдите топ-5 фильмов по рейтингу для каждого жанра.

Вариант 4.

1. Найдите фильмы, где Bruce Willis играл главную роль.
2. Подсчитайте количество фильмов каждого режиссера.
3. Обновите язык всех фильмов старше 1990 года на "en".
4. Создайте уникальный индекс по полю id.
5. Найдите все фильмы, где есть более трех режиссеров.

Вариант 5.

1. Найдите все фильмы жанра "Comedy" с продолжительностью больше 120 минут.
2. Увеличьте рейтинг на 0.1 для всех фильмов после 2015 года.
3. Удалите поле runtime у всех фильмов до 1980 года.
4. Создайте индекс по полю votes.
5. Подсчитайте количество фильмов в каждом десятилетии.

Вариант 6.

1. Добавьте новый фильм "Inception 2" с произвольными данными, используя структуру существующих фильмов.
2. Найдите все фильмы, где снимались одновременно Bruce Willis и Samuel L. Jackson.
3. Обновите описание (plotOutline) для фильма "The Matrix".
4. Создайте индекс по полю actors.name.
5. Подсчитайте средний рейтинг фильмов для каждого актера.

Вариант 7.

1. Создайте новую коллекцию "awards" и добавьте в неё информацию о наградах для 3 фильмов.
2. Найдите все фильмы жанра "Thriller" с рейтингом выше 8.5.
3. Добавьте поле "boxOffice" со значением 0 всем фильмам после 2000 года.
4. Создайте составной индекс по полям genres и year.
5. Найдите количество фильмов по языкам.

Вариант 8.

1. Добавьте нового актера в коллекцию persons с полной биографией.
2. Найдите все фильмы длительностью более 150 минут.
3. Обновите поле votes для всех фильмов Quentin Tarantino.
4. Создайте текстовый индекс по полю tradeMarks в коллекции persons.
5. Подсчитайте количество актеров в каждом фильме.

Вариант 9.

1. Создайте новую коллекцию "reviews" и добавьте несколько отзывов к фильмам.
2. Найдите все фильмы с более чем 5 продюсерами.
3. Добавьте поле "budget" всем фильмам после 1990 года.
4. Создайте индекс по полю directors.name.
5. Рассчитайте средний рейтинг фильмов по годам.

Вариант 10.

1. Добавьте информацию о саундтреках для 3 фильмов в новую коллекцию "soundtracks"
2. Найдите все фильмы, где режиссер также является актером
3. Обновите поле languages для всех фильмов жанра "Sci-Fi"
4. Создайте индекс по полю producers.name
5. Определите топ-3 самых продуктивных года по количеству фильмов

Вариант 11.

1. Создайте коллекцию "locations" с информацией о местах съёмок для 5 фильмов
2. Найдите все фильмы с exactly 2 режиссерами
3. Добавьте поле "remakes" в виде массива для фильмов до 1980 года
4. Создайте индекс по полю runtime
5. Посчитайте количество фильмов каждого жанра для каждого десятилетия

Вариант 12.

1. Добавьте данные о кассовых сборах для всех фильмов после 2010 года
2. Найдите фильмы, где один человек является и продюсером, и режиссером
3. Обновите headshot URL для всех актеров в коллекции persons
4. Создайте составной индекс по полям year и votes
5. Определите режиссеров с самым высоким средним рейтингом фильмов

Вариант 13.

1. Создайте коллекцию "trailers" с ссылками на трейлеры для новых фильмов
2. Найдите все фильмы с более чем 3 языками
3. Добавьте поле "sequels" для фильмов, имеющих продолжения
4. Создайте индекс по полю birthDate в коллекции persons
5. Рассчитайте процент фильмов каждого жанра от общего количества

Вариант 14.

1. Добавьте информацию о спецэффектах для фильмов жанра "Sci-Fi"
2. Найдите актеров, родившихся до 1960 года
3. Обновите поле rank для всех фильмов на основе их рейтинга
4. Создайте индекс по полю actedInMovies.movieId в коллекции persons
5. Определите самые популярные комбинации жанров

Вариант 15.

1. Создайте коллекцию "crew" с информацией о съемочной группе для 3 фильмов
2. Найдите все фильмы с одинаковым рейтингом
3. Добавьте поле "awards" со списком наград для оскароносных фильмов
4. Создайте составной индекс по полям rating и votes
5. Подсчитайте среднюю продолжительность фильмов по жанрам

Вариант 16.

1. Создайте коллекцию "merchandise" с данными о товарах по фильмам
2. Найдите все фильмы продолжительностью менее 100 минут
3. Добавьте поле "trivia" со списком интересных фактов для 5 фильмов
4. Создайте индекс по полю actedInMovies.title в коллекции persons
5. Определите жанры с наивысшим средним количеством голосов

Вариант 17.

1. Добавьте информацию о костюмах для исторических фильмов
2. Найдите фильмы без указанного языка
3. Обновите поле coverUrl для всех фильмов после 2020 года
4. Создайте составной индекс по полям languages и votes
5. Рассчитайте корреляцию между рейтингом и количеством голосов

Вариант 18.

1. Создайте коллекцию "quotes" с известными цитатами из фильмов
2. Найдите фильмы с максимальным количеством продюсеров
3. Добавьте поле "prequels" для фильмов, имеющих предыстории
4. Создайте индекс по полю tradeMarks в коллекции persons
5. Определите режиссеров с наибольшим количеством фильмов в базе

Вариант 19.

1. Добавьте данные о бюджете и кассовых сборах для фильмов 2000-2010 годов
2. Найдите актеров, сыгравших более чем в одном жанре
3. Обновите поле genres для фильмов с одним жанром
4. Создайте текстовый индекс по полю plotOutline
5. Подсчитайте среднее количество актеров в фильмах по жанрам

Вариант 20.

1. Создайте коллекцию "ratings_history" с историей изменения рейтингов
2. Найдите фильмы с самым большим количеством языков
3. Добавьте поле "remasterDate" для фильмов старше 30 лет
4. Создайте составной индекс по полям year и genres
5. Определите актеров с наивысшим средним рейтингом фильмов

Вариант 21.

1. Добавьте информацию о музыкальных композиторах для фильмов
2. Найдите всех режиссеров, снявших более 2 фильмов
3. Обновите поле runtime для фильмов без указанной продолжительности
4. Создайте индекс по полю producers.producerId
5. Рассчитайте распределение рейтингов по десятилетиям

Вариант 22.

1. Создайте коллекцию "deleted_scenes" с информацией о вырезанных сценах
2. Найдите фильмы с наибольшим количеством жанров
3. Добавьте поле "alternativeTitles" для международных фильмов
4. Создайте составной индекс по полям rating и year
5. Определите самые успешные года для каждого жанра

Вариант 23.

1. Добавьте информацию о локациях премьер для фильмов после 2015 года
2. Найдите всех актеров, сыгравших в фильмах определенного режиссера
3. Обновите поле actors для фильмов с неполным списком актеров
4. Создайте индекс по полю actedInMovies.year в коллекции persons
5. Подсчитайте количество фильмов для каждой комбинации жанров

Вариант 24.

1. Создайте коллекцию "awards_ceremonies" с данными о церемониях награждения
2. Найдите фильмы, где режиссер моложе главного актера
3. Добавьте поле "restoration" для фильмов до 1970 года
4. Создайте составной индекс по полям directors.name и year
5. Определите тренды в рейтингах фильмов по годам

Вариант 25.

1. Добавьте данные о рекламных кампаниях для блокбастеров
2. Найдите все фильмы с участием определенной киностудии
3. Обновите поле producers для фильмов последних 5 лет
4. Создайте индекс по полю rank
5. Рассчитайте статистику по продолжительности фильмов разных жанров

Критерии оценки и требования к отчету по практической работе с MongoDB

Что необходимо предоставить в отчете:

1. Титульный лист:
    - Название учебного заведения.
    - Название работы.
    - Номер варианта.
    - ФИО студента.
    - Группа.
    - ФИО преподавателя.
    - Год выполнения.
2. Содержание отчета:
    - Цель работы.
    - Задачи варианта.
    - Описание используемого программного обеспечения и версий.
    - Листинг команд с пояснениями для каждой задачи.
    - Скриншоты результатов выполнения команд.
    - Описание возникших проблем и способов их решения.
    - Выводы по работе.
3. Приложения (если есть):
    - Дополнительные скрипты.
    - Исходные данные.
    - Дополнительные материалы.

Критерии оценки

Оценка "Отлично" (90-100 баллов):
- Все 5 заданий выполнены правильно и полностью.
- Использованы оптимальные решения.
- Код хорошо структурирован и документирован.
- Отчет оформлен аккуратно и содержит все необходимые элементы.
- Продемонстрировано глубокое понимание материала.
- Работа сдана в срок.

Оценка "Хорошо" (70-89 баллов):
- Выполнено 4 задания полностью.
- Решения рабочие, но не оптимальные.
- Код в целом структурирован.
- Отчет содержит незначительные недочеты.
- Продемонстрировано понимание основных концепций.
- Работа сдана в срок.

Оценка "Удовлетворительно" (50-69 баллов):
- Выполнено 3 задания.
- Решения содержат ошибки или неоптимальны.
- Код недостаточно структурирован.
- Отчет содержит существенные недочеты.

- Базовое понимание материала
- Работа сдана с небольшим опозданием

Оценка "Неудовлетворительно" (0-49 баллов):

- Выполнено менее 3 заданий.
- Решения содержат критические ошибки.
- Код не структурирован.
- Отчет оформлен небрежно или отсутствует.
- Непонимание базовых концепций.
- Значительное опоздание сдачи работы.

Сроки сдачи

- Стандартный срок: 1 неделя с момента получения задания.
- Крайний срок: 2 недели (с понижением максимальной оценки).
- После крайнего срока работы принимаются только с разрешения преподавателя.

Защита работы

Студент должен:

1. Продемонстрировать работоспособность решений.
2. Объяснить использованные подходы.
3. Ответить на вопросы по теоретической части.
4. Показать понимание возможных альтернативных решений.