Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«Работа с БД в СУБД MongoDB»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Камалов Руслан Олегович
Факультет прикладной информатики
Группа K3241
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

1. Цель работы:

Овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

2. Практическое задание/Выполне:

1. **Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени:**

db.unicorns.find({gender:"f"}).limit(3).sort({name: 1})
db.unicorns.find({gender:"m"}).sort({name: 1})

2. **Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций findOne и limit:**

db.unicorns.find({gender:"f", loves: "carrot"}).limit(1)

db.unicorns.findOne({gender:"f", loves: "carrot"}

3. **Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпотениях и поле:**

db.unicorns.find({gender:"m"}, {loves: 0, gender: 0})

4. **Вывести список единорогов в обратном порядке добавления:**

db.unicorns.find().sort({$natural: -1})

5. **Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор:**

db.unicorns.find({}, {_id: 0, loves: {$slice: 1}})

6. **Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора:**

db.unicorns.find({gender: "f", weight: {$gte: 500, $lte: 700}}, {_id: 0})

7. **Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора:**

db.unicorns.find({gender: "m", weight: {$gte: 500}, loves: {$all: ["grape", "lemon"]}}, {_id: 0})

8. **Найти всех единорогов, не имеющих ключ `vampires`:**

db.unicorns.find({vampires: {$exists: false}})

9. **Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении:**

db.unicorns.find({gender: "m"}, {_id: 0, name: 1, loves: {$slice: 1}}).sort({name: 1})

10. **Сформировать запрос, который возвращает список городов с независимыми мэрами (`party="I"`). Вывести только название города и информацию о мэре:**

db.towns.find({"mayor.party": "I"}, {_id: 0, name: 1, mayor: 1})

11. **Сформировать запрос, который возвращает список беспартийных мэров (`party` отсутствует). Вывести только название города и информацию о мэре:**

db.towns.find({"mayor.party": {$exists: false}}, {_id: 0, name: 1, mayor: 1})

12. **Сформировать функцию для вывода списка самцов единорогов:**

find_male = function() { return {gender: "m"}; }
db.unicorns.find(find_male())

13. **Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке:**

var cursor = db.unicorns.find(find_male()); null;

cursor.limit(2).sort({name: 1}); null;

14. Вывести результат, используя forEach:
cursor.forEach(function(obj) { print(obj.name); })

15. Вывести количество самок единорогов весом от полутонны до 600 кг:
db.unicorns.find({weight: {$gte: 500, $lte: 600}}).count()

16. Вывести список предпочтений:
db.unicorns.distinct("loves")

17. Посчитать количество особей единорогов обоих полов:
db.unicorns.aggregate({"$group": {_id: "$gender", count: {$sum: 1}}})

18. **Для самки единорога `Ayna` внести изменения в БД: теперь ее вес 800, она убила 51 вапмира:**
db.unicorns.updateOne({name: "Ayna"}, {$set: {weight: 800, vampires: 51}}, {upsert: true})

19. **Для самца единорога `Raleigh` внести изменения в БД: теперь он любит рэдбул:**
db.unicorns.updateOne({name: "Raleigh"}, {$set: {loves: ["redbull"]}}, {multi:true})

20. Всем самцам единорогов увеличить количество убитых вапмиров на 5:
db.unicorns.updateMany({gender: "m"}, {$inc: {vampires: 5}})

21. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный:
db.towns.updateOne({name: "Portland"}, {$unset: {"mayor.party": 1}})

22. Изменить информацию о самце единорога Pilot: теперь он любит и шоколад:
db.unicorns.updateOne({name: "Pilot"}, {$push: {loves: "chocolate"}})

23. Изменить информацию о самке единорога `Aurora`: теперь она любит еще и сахар, и лимоны:
db.unicorns.updateOne({name: "Aurora"}, {$addToSet: {loves: {$each: ["lemon", "sugar"]}}})

24. Удалите документы с беспартийными мэрами:
db.newTowns.deleteMany({"mayor.party": {$exists: false}})

25. Очистите коллекцию:
db.newTowns.deleteMany({})

26. Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание:
documents=[{ _id: "forest", name: "Enchanted Forest", description: "A mystical forest where unicorns graze on magical berries." },{ _id: "mountain", name: "Crystal Mountains", description: "High-altitude peaks where unicorns rest under the stars." },{ _id: "meadow", name: "Golden Meadow",description: "A sunlit field where unicorns play and run freely." }]

db.habitats.insert(documents)

27. Включите для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания:
db.unicorns.updateOne({ name: "Horny" },{ $set: { habitat: { $ref: "habitats", $id: "forest" } } })

db.unicorns.updateOne({ name: "Aurora" },{ $set: { habitat: { $ref: "habitats", $id: "mountain" }}})

db.unicorns.updateOne({ name: "Unicrom" },{ $set: { habitat: { $ref: "habitats", $id:"meadow"}}})

28. Проверьте содержание коллекции едиорогов:
db.unicorns.find({habitat: {$exists: true}})

29. Проверьте, можно ли задать для коллекции `unicorns` индекс для ключа `name` с флагом `unique`:
```
db.unicorns.ensureIndex({ name: 1 }, { unique: true })
```

30. Получите информацию о всех индексах коллекции `unicorns`:
```
db.unicorns.getIndexes()
```

31. **Удалите все индексы, кроме индекса для идентификатора:**
db.unicorns.dropIndex("name_1")

32. Попытайтесь удалить индекс для идентификатора:
db.unicorns.dropIndex("_id_")

MongoServerError[InvalidOptions]: cannot drop _id index

33. **Создайте объемную коллекцию `numbers`, задействовав курсор:**
for (let i = 0; i < 100000; i++) { db.numbers.insertOne({ value: i }) }

34. Выберите последних четыре документа:
db.numbers.find().sort({ value: -1 }).limit(4)

35. **Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра `executionTimeMillis`):**
db.numbers.explain("executionStats").find().sort({ value: -1 }).limit(4)
#executionTimeMillis: 63

36. **Создайте индекс для ключа `value`:**
db.numbers.createIndex({ value: 1 })

37. **Получите информацию о всех индексах коллекции `numbres`:**
db.numbers.getIndexes()

38. **Проанализируйте план выполнения запроса с установленным индексом.** Сколько потребовалось времени на выполнение запроса?
db.numbers.explain("executionStats").find().sort({ value: -1 }).limit(4)

#executionTimeMillis: 20

3. **Выводы:**

   В ходе практической работы была успешно освоена работа с CRUD-операциями (создание, чтение, обновление, удаление) в MongoDB, включая взаимодействие с вложенными объектами в коллекциях. Приобретены навыки агрегации данных, их модификации, а также работы со ссылками и индексами для оптимизации запросов. Ошибки, возникавшие в процессе выполнения операций, были выявлены и исправлены. В результате я овладел ключевыми практическими навыками эффективного управления данными в NoSQL-СУДБ MongoDB, что позволит применять их в реальных проектах.