Лабораторная работа №3

Тема: документальные базы данных

Цель работы: Получение практических навыков использования документальных баз данных на примере СУБД MongoDB.

Задание: разработать согласно варианту документальную базу данных и в ней:

- разработать не менее 5 коллекций, одна из коллекций должна быть ограничена,
- создать в коллекциях документы:
  - в первой вставить документы (insertOne, insertMany, insert), в документах должны быть и вложенные поля,
  - во вторую вставить документы из JSON-файла (подготовить файл с данными для импорта),
  - в третью вставить из CSV-файла (подготовить файл с данными для импорта),
- коллекции должны ссылаться друг на друга ($ref),
- выполнить модификацию данных:
  - изменить (обновить) данные (replaceOne, updateOne, updateMany),
  - удалить документ (deleteOne, deleteMany),
  - вставить значение в массив ($push, $addToSet),
  - удалить значение из массива ($pop),
  - удалить коллекцию (drop),
- выполнить фильтрацию (find):
  - по нескольким полям,
  - по отсутствующим свойствам,
  - по элементам массива,
  - проекцию,
  - по вложенным объектам,
  - с ограничением выводимых строк и пропуском первых строк,
  - с сортировкой,
  - с условным оператором,
  - с логическим оператором,
  - с условием для массива,

    - с условием на поле,
  - создать 2 индекса, затем один удалить,
  - создать агрегаты (3 разных),
  - создать резервные копии базы и одной коллекции.

Отчет по лабораторной работе должен содержать:

1. Фамилию и номер группы учащегося, задание
2. Краткое описание базы данных (предметная область, выделенные коллекции, документы, ссылки, выбранные индексы).
3. Скрины всех выполненных операций.
4. Код БД

Справочная информация по работе с MongoDB https://metanit.com/nosql/mongodb/.

Документация по MongoDB (на английском) https://docs.mongodb.com/manual/introduction/.

Справочные материалы

Команды MongoDB

Синтаксис команды	Описание	Пример
use ИМЯ_БД	создаётся новая БД, если она ещё не создана. Если же она уже создана, то переключимся на неё.	use projectdb
db	получение имени текущей БД	db
show dbs	получение списка БД	show dbs
Db.ИМЯ_БД.insetr(запись)	Вставка записи в БД	db.projectdb.insert({"name":"proselyte.net"})
db.dropDatabase()	Удаление текущей БД	db.dropDatabase()
db.createCollection(имя_коллекции, свойства_коллекции)	Создание коллекции	db.createCollection("developers", {capped: true, autoIndexID: true, size: 371400, max: 10})
show collections	Получение списка коллекций	show collections
db.ИМЯ_КОЛЛЕКЦИИ.drop()	Удаление коллекции	db.proselyte.drop()
db.ИМЯ_КОЛЛЕКЦИИ.insert(документ)	Вставка документа	db.developers.insert({title: 'Eugene Suleimanov', specialty: 'Java Developer', skills: ['Java', 'Hibernate', 'Spring']})
db.ИМЯ_КОЛЛЕКЦИИ.save({_id:ObjectId(), НОВЫЕ_ДАННЫЕ})	Вставка или перезапись документа (при наличии соответствующего _id)	db.developers.save({title: 'Eugene Suleimanov', specialty: 'Java Developer', skills: ['Java', 'Hibernate', 'Spring']})
db.ИМЯ_КОЛЛЕКЦИИ.update(КРИТЕРИЙ_ВЫБОРА, ИЗМЕНЁННЫЕ_ДАННЫЕ)	Изменение документов по критерию	db.developers.save({ "_id": ObjectId("5815f4f535ee883c37ac8b4f"), "title" : "Nikolay Nikolaev", "specialty": "C++", "skills": ["C++", "STL", "SQL"], "salary" : 3500 })
db.ИМЯ_КОЛЛЕКЦИИ.remove(КРИТЕРИЙ_УДАЛЕНИЯ)	Удаление документов по критерию	db.test_collection.remove({"title": "Document3"}) db.test_collection.remove({})
db.ИМЯ_КОЛЛЕКЦИИ.remove(КРИТЕРИЙ_УДАЛЕНИЯ, 1)	Удаление только одного документа	db.test_collection.remove({"title": "Document3"},1)
db.ИМЯ_КОЛЛЕКЦИИ.find()	запрос элементов коллекции	db.developers.find()
db.ИМЯ_КОЛЛЕКЦИИ.find().pretty()	Структурированный вывод данных коллекции	db.developers.find().pretty()
db.ИМЯ_КОЛЛЕКЦИИ.find({}, {АТТРИБУТ: 1\0})	Проекция 1 – означает, что данный атрибут будет выведен, 0 – нет.	db.developers.find({}, {_id: 0, "specialty": 1}) db.developers.find({}, {"specialty": 1})

db.ИМЯ_КОЛЛЕКЦИИ.find().limit(КОЛИЧЕСТВО_ДОКУМЕНТОВ)	Ограничение выводимых документов	db.developers.find().pretty().limit(2)
db.ИМЯ_КОЛЛЕКЦИИ.find().limit(КОЛИЧЕСТВО_ДОУМЕНТОВ).skip(КОЛИЧЕСТВО_ДОКУМЕНТОВ)	Пропуск документов	db.developers.find().pretty().limit(2).skip(1)
db.ИМЯ_КОЛЛЕКЦИИ.find().sort({АТРИБУТ: 1\-1})	Сортировка по возрастанию используется число ‘1’, для сортировки по убыванию – ‘-1’	db.developers.find().pretty().sort({"title": 1}) db.developers.find().pretty().sort({"title": -1})
db.ИМЯ_КОЛЛЕКЦИИ.ensureIndex({KEY: 1})	создание индекса ‘1’ – порядок по возрастанию, по убыванию число – ‘-1’	db.developers.ensureIndex({"title": 1}) db.developers.ensureIndex({"title": 1, "specialty": -1}) db.members.createIndex({ "**user_id**": 1 }, { unique: true })
db.ИМЯ_КОЛЛЕКЦИИ.aggregate(ОПЕРАЦИЯ_АГРЕГАЦИИ)	Получение агрегатов	db.developers.aggregate([{$group :{ _id : "Developers", total_salary: { $sum : "$salary" }}}])
mongodump	Создание резервной копии	mongodump –dbpath /data/db/ –out /data/backup/ mongodump –collection developers –db projectdb
mongorestore	Восстановление	mongorestore
mongostat	проверяет статус всех работающих экземпляров mongod и возвращает счётчики операций базы данных	
mongotop	отслеживает и делает отчёты о чтении и записи данных в коллекции экземпляра MongoDB. По умолчанию, **mongotop** возвращает информацию за каждую секунду, можно изменить временной параметр.	mongotop mongotop 10

Типы данных

- String: строковый тип данных, как в приведенном выше примере (для строк используется кодировка UTF-8)
- Array (массив): тип данных для хранения массивов элементов
- Binary data (двоичные данные): тип для хранения данных в бинарном формате
- Boolean: булевый тип данных, хранящий логические значения `TRUE` или `FALSE`, например, `{"married": FALSE}`
- Date: хранит дату в формате времени Unix
- Double: числовой тип данных для хранения чисел с плавающей точкой
- Integer: используется для хранения целочисленных значений размером 32 бита, например, `{"age": 29}`

- Long: используется для хранения целочисленных значений размером 64 бита
- JavaScript: тип данных для хранения кода javascript
- Min key/Max key: используются для сравнения значений с наименьшим/наибольшим элементов BSON
- Null: тип данных для хранения значения `Null`
- Object: строковый тип данных, как в приведенном выше примере
- ObjectId: тип данных для хранения id документа
- Regular expression: применяется для хранения регулярных выражений
- Decimal128: тип данных для хранения десятичных дробных чисел размером 128 бит, которые позволяют решить проблемы с проблемой точности вычислений при использовании дробных чисел, которые представляют тип Double.
- Timestamp: применяется для хранения времени

Свойства коллекции

Поле	**Тип**	**Описание**
capped	Boolean	Создаёт ограниченную по размеру коллекцию. Если количество элементов в коллекции достигло максимума, то каждая следующая запись будет перезаписывать более старые данные.
autoIndexID	Boolean	Автоматически создаёт индекс для поля _id field.s
size	number	Определяет максимальный размер (в байтах) ограниченной (capped) коллекции. Если коллекция ограничена, то необходимо обязательно указать максимальный размер.
max	number	Определяет максимальное количество документов в ограниченной коллекции.

Параметры для индексов

Параметр	Тип	Описание
background	Boolean	Создаёт индекс в фоновом режиме таким образом, что данный индекс не блокирует другие процессы базы данных (далее – БД). Для активации необходимо установить **true**. По умолчанию – **false**.
unique	Boolean	Создаёт уникальный индекс таким образом, что коллекция не примет вставку документа, в котором ключ индекса, или ключи совпадают с уже существующими значениями. Для активации необходимо установить **true**. По умолчанию – **false**.
name	string	Имя индекса. Если не указан, то MongoDB генерирует путём конкантенации имён индексируемых полей.
dropDups	Boolean	Создаёт уникальный индекс для поля, которое может иметь дубликаты. MongoDB индексирует только первый случай ключа и удаляет все документы коллекции, которые содержат совпадения поля. Для активации необходимо установить **true**. По умолчанию – **false**.
sparse	Boolean	Индекс ссылается на документ по указанному полю. Данный вид индексов используют меньше памяти. Но, в некоторых ситуациях, ведут себя по-другому. Для активации необходимо установить **true**. По умолчанию – **false**.
expireAfterSeconds	integer	Определяет значение (в секундах) TTL для контроля того, как долго MongoDB сохраняет документ в коллекции.
v	index version	Номер версии индекса. По умолчанию, версия индекса зависит от версии MongoDB во время создания индекса.
weights	document	Число, которое ранжирует от 1 до 99,999 и обозначает важность соответствующего поля по отношению к другим индексируемым полям.
default_language	string	Для текстового индекса это язык, который определяет список стоп-слов и правил для сущностей stemmer и tokenizer. По умолчанию – **english**
language_override	string	Для текстового индекса определяет имя поля в документе, которое содержит, и переопределяет язык по умолчанию.

Операторы для фильтрации

Условные операторы задают условие, которому должно соответствовать значение поля документа:

Оператор	Назначение	Пример
`$eq`	Если значение равно	{marks:{$eq:20}}
`$gt`	Если значение больше	{marks: {$gt:20}}
`$lt`	Если значение меньше	{marks: {$lt:20}}
`$gte`	Если значение больше или равно	{marks:{$gte:22}}
`$lte`	Если значение меньше или равно	{marks:{$lte:22}}
`$ne`	Если значение не равно	{marks:{$ne:22}}
`$in`	Если значение равно одному из значений массива	{marks:{$in:[20,22]}}
`$nin`	Если значение не равно одному из значений массива	{marks:{$nin:[22,25]}

Логические операторы выполняются над условиями выборки:

- $or: соединяет два условия, и документ должен соответствовать одному из этих условий
- $and: соединяет два условия, и документ должен соответствовать обоим условиям
- $not: документ должен НЕ соответствовать условию
- $nor: соединяет два условия, и документ должен НЕ соответстовать обоим условиям

Операторы для работы с массивами:

- $all: определяет набор значений, которые должны иметься в массиве
- $size: определяет количество элементов, которые должны быть в массиве
- $elemMatch: определяет условие, которым должны соответствовать элемены в массиве

Операторы для проверки полей

- $type извлекает только те документы, в которых определенный ключ имеет значение определенного типа, например, строку или число
- $exists позволяет извлечь только те документы, в которых определенный ключ присутствует или отсутствует

Пример запросов с условиями на поля

Операция	Синтаксис	Пример
Равенство	{<key>:<value>}	db.developers.find({“specialty”:”Java”}).pretty()
Меньше, чем	{<key>:{$lt:<value>}}	db.developers.find({“salary”:{$lt:1000}}).pretty()
Меньше, либо равно	{<key>:{$lte:<value>}}	db.developers.find({“salary”:{$lte:1000}}).pretty()
Больше, чем	{<key>:{$gt:<value>}}	db.developers.find({“salary”:{$gt:1000}}).pretty()
Больше, либо равно	{<key>:{$gte:<value>}}	db.developers.find({“salary”:{$gte:1000}}).pretty()
Не равно	{<key>:{$ne:<value>}}	db.developers.find({“salary”:{$ne:1000}}).pretty()
И	{ $and: [{ключ1: значение1}, {ключ2:значение2}] }	db.developers.insert({title: 'Ivan Ivanov', specialty: 'Java', skills: ['Java', 'Hibernate', 'Spring'], salary: 3000})
ИЛИ	{ $or: [{ключ1: значение1},	db.developers.find({ $or: [{specialty: "C++"}, {salary: { $gte: 3000 }}] }).pretty()

	{ключ2:значение2}] }	db.developers.find({"salary": {$gte: 2000}, $or: [{"specialty": "Java"}, {"specialty": "Java Developer"}] }).pretty()

Список операций для агрегации документов

Выражение	Описание	Пример
$sum	Суммирует указанные значения всех документов в коллекции.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, total_salary : {$sum : “$salary”}}}])
$avg	Рассчитывает среднее значение указанного поля поля для всех документов коллекции.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, avg_salary : {$avg : “$salary”}}}])
$min	Получает минимальное значение указанного поля документа в коллекции	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, min_salary : {$min : “$salary”}}}])
$max	Получает максимальное значение указанного поля документа в коллекции	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, max_salary : {$max : “$salary”}}}])
$push	Вставляет значение в массив в результирующем документе.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, skills : {$push: “$skills”}}}])
$addToSet	Вставляет значение в массив в результирующем документе, но не создаёт дубликаты.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, skills : {$addToSet : “$skills”}}}])
$first	Получает первый документ из сгруппированных. Обычно используется вместе с сортировкой.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, first_skill : {$first : “$skills”}}}])
$last	Получает крайний документ из сгруппированных. Обычно используется вместе с сортировкой.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {_id : “$title”, last_skill : {$last : “$skills”}}}])

Опции резервирования

Синтаксис	Описание	Пример
mongodump –host ИМЯ_ХОСТА –port НОМЕР_ПОРТА	Данная команда создаст резервную копию всех БД указанного экземпляра mongod.	mongodump –host proselyte.net –port 27017
mongodump –dbpath ПУТЬ_К_БД –out ДИРЕКТОРИЯ_РЕЗЕРВНОЙ_КОПИИ	Данная команда создаст резервную копию указанной БД по указанному пути.	mongodump –dbpath /data/db/ –out /data/backup/
mongodump –collection КОЛЛЕКЦИЯ –db ИМЯ_БД	Данная команда создаст резервную копию только указанной коллекции указанной БД	mongodump –collection developers –db projectdb

Пример команд (командная строка MongoDB)

Запуск клиента:
```
sudo mongo
```

Список баз:
```
show dbs
```

Использовать базу (demo — имя базы):
```
use demo
```

Список collections:
```
show collections
```

Добавить пользователя в админы со всеми правами:
```
use admin
db.createUser({user:"root", pwd:"root", roles:[ "root"]})
```
Или с указанием ролей:
```
db.createUser({user:"admin", pwd:"password", roles:[ "userAdminAnyDatabase",
"dbAdminAnyDatabase", "readWriteAnyDatabase"]})
db.createUser({user:"admin", pwd:"password", roles:[ "dbOwner" ]})
```

Аутентифицироваться с командной строки:
```
sudo mongo -u root -p root --authenticationDatabase admin
sudo mongo -u admin -p admin --authenticationDatabase test
```

Системная информация о базе:
```
db.stats()
```

Системная информация о collections:
```
db.test.stats()
```

Список индексов в collection:
```
db.test.getIndexes()
```
Вставка документа в collection users:
```
db.users.insert({"name":"username"})
```

Выборка по условию (По одному полю):
```
db.test.find({"name":"Andre"})
```

Выборка по условию (И):
```
db.test.find({"name":"Andre","id":"2"})
```

Выборка по условию (Больше):
```
db.numbers.find({num:{"$gt": 60000}})
```

Выборка по условию (В диапазоне):
```
db.numbers.find({num:{"$gt": 60000, "$lt": 60010}})
```

Обновление записи:
```
db.test.update({"_id":ObjectId("5aae3be1b371d7112a438547")}, {$set:
{name:"Patya", id:3}})
db.users.update( { "favorites.movies": "Casblanca"},
    {$addtoSet: {"favorites.movies": "The Maltese Falcon"} },
    false,
    true )
```
Четвертый параметр true - говорит, что это множественное обновление. По умолчанию MongoDB обновит только первую запись подходящую по условию.

Обновление документа (SAVE) если нет поля _id, тогда будет создан новый документ:
```
db.test.save({"_id":ObjectId("5aae3be1b371d7112a438547")}, {$set:
{name:"Patya", id:3}})
```

Удаление документов:
```
db.test.remove({"_id":ObjectId("5aae3be1b371d7112a438547")})
```

Подсчет кол-ва документов:
```
db.numbers.count()
```

Удалить коллекцию со всеми данными и индексами:
```
db.test.drop()
```

Запрос с выводом информации о выполнении для дальнейшего анализа:
```
db.numbers.find({num:{"$gt": 60000}}).explain("executionStats")
```

Создать индекс по полю num по возрастанию:
```
db.numbers.ensureIndex({num: 1})
```

Экспорт (mongoexport) в JSON, CSV, TSV:
```
mongoexport --host=127.0.0.1 --port=27017 --db=test --collection=test --
type=json --out=test_test.json --jsonArray --pretty
```

Импорт (mongoimport) из JSON, CSV, TSV
```
mongoimport --host=127.0.0.1 --port=27017 --db=test --collection=test --
file=test_test.json --type=json --jsonArray
```

Информация и статистика
```
mongostat
```