|  |  |
| --- | --- |
| **mongo** | запуск mongo db  Запуск сервера монго дб в первой консоле  cd C:\Program Files\MongoDB\Server\4.2\bin  mongod.exe --dbpath d:\Mongo\_Base  Запуск клиента монго дб во второй консоле  cd C:\Program Files\MongoDB\Server\4.2\bin  mongo.exe |
| **db.version** | версия программы |
| **show users** | список пользователей в текущей БД |
| **db.getMongo()** | список текущих подключений |
| **db.addUser(name, password)** | добавить пользователя(старый вариант) |
| **db.createUser(**  **{**  **user: 'name',**  **pwd: 'password',**  **roles: [ ]**  **})** | добавить пользователя |
| **db.removeUser(name)** | удалить пользователя(старый вариант) |
| **db.dropUser(**  **'reportUser1',**  **{**  **w: 'majority',**  **wtimeout: 5000**  **})** | удалить пользователя(новый вариант) |
| **db.currentOp()** | показать операции в текущей БД |
| **db.killOp()** | убить текущую операцию в БД |
| **db.repairDtabase()** | восстановление БД после краха |
| **db.shutdownServer()** | остановить сервер БД |
|  |  |
|  |  |
| **mongodump** | создает дамп всех БД (в текущей директории создаст каталог dump)  -d <name> БД для дампа  -o <путь> директория для сохранения БД |
| **mongorestore**  **--verbose**  **--db <name>** | восстановление БД из дампа |
| **db.printCollectionStats()** | статистическая информация о коллекции (с форматированием) |
| **db.stats()**  **db.users.stats()** | статистическая информация о коллекции (укороченная) |
| **use test** | переход в БД или её создание (отобразится в списке БД только после добавления данных), текущая БД |
| **db.getName()**  **db** | посмотреть название текущей БД |
| **show dbs** | вывести все БД |
| **show collections**  **show tables** | вывести все коллекции выбранной БД |
| **db.prof.find()**  **db.prof.find().pretty()**  **db.prof.findOne()** | выводит данные коллекции  db.<имя коллекции>.find({name: "Tom", age: 32})  db.users.find({languages: ["english", "german"]})  db.users.find({"languages.0": "english"})  # English в массиве на 1 месте (индекс 0)  db.users.find({name: "Tom"}, {age: 1})  # вывести только значение поля age  db.users.find({name: "Tom"}, {age: 0})  # вывести все поля кроме age  db.users.find({name: "Tom"}, {age: true, \_id: false})  # вместо 0 и 1 true и false  db.users.find({}, {age: 1, \_id: 0})  db.users.find({"company.name": "microsoft"})  # для сложных вложенных объектов(словарь в  словаре)  db.getCollection('prof').find({name:'Vasya'})  db.prof.find().pretty() # в улучшенном виде  db.users.find("this.name=='Tom'")  db.users.find({name:/T\w+/i}) # начинается с Т  fn = function() { return this.name=="Tom"; }  db.users.find(fn) # применение JS |
| **limit()** | кол-во выведенных документов  db.users.find().limit(3) |
| **skip()** | кол-во пропускаемых документов  db.users.find().skip(3) |
| **sort()** | сортировка 1 по возрастанию -1 по убыванию  db.users.find().sort({name: 1})  db.users.find().sort({name: 1}).skip(3).limit(3) |
| **$natural** | сортировка в порядке добавления документов в коллекцию 1, или в обратном порядке -1  db.users.find().sort({ $natural: -1 }).limit(5) |
| **$slice** | комбинация функций limit и skip, может работать с массивами.  $slice принимает 2 параметра (skip (необяз), limit)  db.users.find ({name: "Tom"}, {languages: {$slice : -1}})  db.users.find({name:"Tom"},{languages:{$slice:[-1,1]}}) |
| **insertOne()**  **insertMany()**  **insert()** | добавление данных в БД  названия ключей могут вводится в кавычках и без кавычек.  db.users.insertOne({"name": "Tom", "age": 28, languages: ["english", "spanish"]})  db.users.insertOne({"\_id": 123457, "name": "Tom", "age": 28, languages: ["english", "spanish"]})  db.users.insertMany([{"name": "Bob", "age": 26, languages: ["english", "frensh"]},  {"name": "Alice", "age": 31, languages:["german", "english"]}])  document=({"name": "Bill", "age": 32, languages: ["english", "french"]})  db.users.insert(document) |
| **count()** | число элементов в коллекции  db.users.count()  db.users.find({name: "Tom"}).count()  db.users.find({name: "Tom"}).skip(2).count(true)  # count(true) - можно использовать с limit и skip |
| **distinct()** | вывести только уникальные значения  db.users.distinct("name") |
| **group()** | группировка  key: ключ, по которому проводится группировка  initial: инициализирует поля документа, который будет представлять группу документов  reduce: функция, возвращающая количество элементов, принимает в качестве аргументов два параметра: текущий элемент и агрегатный результирующий документ для текущей группы  keyf: (необяз) вместо key - представляет функцию, которая возвращает объект key  cond: (необяз) условие, которое должно возвращать true, иначе документ не участвует в группировке. Если данный параметр не указан, то в группировке участвуют все документы  finalize: (необяз) функция, которая срабатывает перед тем, как будут возвращены результаты группировки.  db.users.group ({key: {name : true}, initial: {total : 0},  reduce : function (curr, res){res.total += 1}}) |
| **Условные операторы** | $eq (равно)  $ne (не равно)  $gt (больше чем)  $lt (меньше чем)  $gte (больше или равно)  $lte (меньше или равно)  $in определяет массив значений, одно из которых должно иметь поле документа  $nin определяет массив значений, которые не должно иметь поле документа  db.users.find ({age: {$lt : 30}}) # age < 30  db.users.find ({age: {$gt : "30"}}) # сравнение будет  производиться как с цифрами  db.users.find ({age: {$gt : 30, $lt: 50}})  db.users.find ({age: {$in : [22, 32]}}) |
| **Логические операторы** | $or: или  $and: и  $not: не  $nor: не соответствие обоим условиям  db.users.find ({$or : [{name: "Tom"}, {age: 22}]})  db.users.find ({$or : [{name: "Tom"}, {age: {$gte:30}}]}) |
| **Операторы для поиска по массивам** | $all: определяет набор значений, которые должны иметься в массиве  $size: количество элементов в массиве  $elemMatch: определяет условие, которым должны соответствовать элементы в массиве  db.users.find ({languages: {$all : ["english", "french"]}})  db.grades.find({courses: {$elemMatch: {name: "MongoDB", grade: {$gt: 3}}}})  db.users.find ({languages: {$size:2}}) |
| **$exists** | извлекает документы, в которых определенный ключ присутствует или отсутствует.  db.users.find ({company: {$exists:true}}) |
| **$type** | извлекает документы, в которых определенный ключ имеет значение определенного типа  db.users.find ({age: {$type:"string"}})  db.users.find ({age: {$type:"number"}}) |
| **$regex** | задает регулярное выражение, которому должно соответствовать значение поля  db.users.find ({name: {$regex:"b"}})  db.users.find ({name: {$regex:"om$"}}) |
| **update()**  **updateOne()**  **updateMany()** | обновление данных в БД  query: принимает запрос на выборку документа, который надо обновить  objNew: документ с новой информацией, который заместит старый при обновлении  options: определяет дополнительные параметры при обновлении документов. Может принимать два аргумента: upsert и multi.  upsert true, что mongodb будет обновлять документ, если он найден, и создавать новый, если такого документа нет  upsert false, то mongodb не будет создавать новый документ, если запрос на выборку не найдет ни одного документа.  multi указывает, должен ли обновляться первый элемент в выборке (используется по умолчанию, если данный параметр не указан) или же должны обновляться все документы в выборке.  db.users.update({name : "Tom"}, {name: "Tom", age : 25}, {upsert: true}) |
| **save()** | обновление данных в БД  можно передать параметр \_id. Если метод находит документ с таким значением \_id, то документ обновляется. Если же с подобным \_id нет документов, то документ вставляется.  Если параметр \_id не указан, то документ вставляется, а параметр \_id генерируется автоматически как при обычном добавлении через функцию insert  db.users.save({name: "Eugene", age : 29, languages: ["english", "german", "spanish"]}) |
| **$set()** | обновление определенных ключей в поле, если документ не содержит обновляемое поле, то оно создается. Обновляется только один документ, первый в выборке. Указав значение multi:true, мы можем обновить все документы выборки:  db.users.update({name : "Tom", age: 29}, {$set: {age : 30}})  db.users.update({name : "Tom", age: 29}, {$set: {salary : 300}})  db.users.update({name : "Tom"}, {$set: {name: "Tom", age : 25}}, {multi:true}) |
| **$unset()** | удаление отдельного ключа  db.users.update({name : "Tom"}, {$unset: {salary: 1, age: 1}}) |
| **$push()** | добавить еще одно значение в массив к уже существующему  db.users.updateOne({name : "Tom"}, {$push: {languages: "russian"}})  db.users.update({name : "Tom"}, {$push: {languages: {$each: ["russian", "spanish", "italian"]}}}) |
| **$position()**  **и**  **$slice()** | $position задает позицию в массиве для вставки элементов, а $slice указывает, сколько элементов оставить в массиве после вставки.  db.users.update({name : "Tom"}, {$push: {languages: {$each: ["german", "spanish", "italian"], $position:1, $slice:5}}}) |
| **$addToSet()** | добавляет данные, если их нет в массиве  db.users.update({name : "Tom"}, {$addToSet: {languages: "russian"}}) |
| **pop()** | удаляет элемент из массива  Указывая для ключа languages значение 1, мы удаляем первый элемент с конца. Чтобы удалить первый элемент сначала массива, надо передать отрицательное значение  db.users.update({name:"Tom"}, {$pop: {languages: 1}})  db.users.update({name:"Tom"}, {$pop: {languages:-1}}) |
| **pull()** | удаляет каждое вхождение элемента в массив  db.users.update({name : "Tom"}, {$pull: {languages: "english"}}) |
| **$pullAll()** | удаляет каждое вхождение нескольких элементов в массив  db.users.update({name : "Tom"}, {$pullAll: {languages: ["english", "german", "french"]}}) |
| **remove()** | удаление документов  db.users.remove({name : "Tom"})  db.users.remove({name : /T\w+/i})  db.users.remove({age: {$lt : 30}})  db.users.remove({name : "Tom"}, true) # удалит  только первый документ  db.users.remove({}) # удалит все док. из коллекции |
| **drop()** | удаление коллекции и БД  db.users.drop()  db.dropDatabase() |
| **db.dropDatabase()** | удаление БД |
| **load("D:/users.js")** | загрузка файла в текущую БД |
| **createCollection()** | создание коллекции  db.createCollection(name, options)  db.createCollection("accounts") |
| **renameCollection()** | переименование коллекции  db.users.renameCollection("новое\_название") |
| **Ограниченная коллекция** | коллекция, в которой документы располагаются в порядке добавления, коллекция имеет фиксиров размер, если коллекция переполняется, то старые данные стираются, а новые записываются. Из таких коллекций нельзя удалять данные  db.createCollection("profile", {capped:true, size:9500})  db.createCollection("profile", {capped:true, size:9500, max: 150}) # или кол-во документов 150 |
| **Подколлекции** | db.users.profiles  db.users.accounts  т. о. существуют 3 различных несвязанным между собой коллекции(users, profiles, accounts) |
| **Ручная установка ссылок** | присвоение значения поля \_id одного документа полю другого документа  db.companies.insert({"\_id" : "microsoft", year: 1974})  db.users.insert({name:'Tom',age:28,company:'microsoft'})  db.companies.findOne({\_id: user.company}) |
| **Автоматическое связывание** | функциональность DBRef устанавливает автомат связывание между документами, сохранение идет с помощью метода save, не insert. Метод save при добавлении нового документа генерирует \_id.  **{ "$ref" : название\_коллекции, "$id": значение [, "$db" : название\_бд ]}**  Первый параметр $ref указывает на коллекцию, где хранится связанный документ. Второй параметр -на значение, к-рое будет представлять что-то типа внешнего ключа. Третий необяз параметр - на БД.  При тестировании в качестве запроса на выборку указывается выражение \_id: steve.company.$id. Так как person.company представляет теперь объект new DBRef('companies', apple.\_id)}), то нам надо конкретизировать параметр steve.company.$id  apple=({"name" : "apple", "year": 1976})  db.companies.save(apple)  steve = ({"name": "Steve", "age": 25, company: new DBRef('companies', apple.\_id)})  db.users.save(steve)  db.companies.findOne({\_id: steve.company.$id}) |
| **Индексы** | db.users.createIndex({"name" : 1}, {"unique" : true})  # создание индекса по полю name, флаг unique  устанавливается, если нам необходимо, чтобы в  поле name были только уникальные значения  db.users.createIndex({"name" : 1, "age" : 1}, {"unique" : true})  db.system.indexes.find() # выводит индексы и  связанную с ними информацию  db.users.getIndexes() # инфо об индексах в  конкретной коллекции  db.users.dropIndex("name\_1") # удаление индекса |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |