Нереляционные базы данных и MongoDB

Урок 2. Введение в MongoDB

docker run --name skillbox-mongodb -d mongo:3.1 – запустим контейнер

docker exec -it skillbox-mongodb mongo – соединяемся с серверов внутри контейнера

docker ps –a – покажет доступные запущенные процессы docker

docker rm {id процесса}

docker cp posts.json skillbox-mongodb:/tmp

docker exec -it skillbox-mongodb mongoimport --db skdb --collection posts --file /tmp/posts.json – копируем посты юзеров

docker cp users.json skillbox-mongodb:/tmp

docker exec -it skillbox-mongodb mongoimport --db skdb --collection users --file /tmp/users.json – копируем юзеров

docker exec -it skillbox-mongodb mongo

show dbs – показать доступные базы

use skdb – переключиться на базу

show collections – показать доступные коллекции

db.users.find({"\_id": "vjaniya@example.ru"}) – в базе данных users найдем id со значением в фигурных скобках

{ "\_id" : "vjaniya@example.ru", "first\_name" : "Vinnie", "last\_name" : "Janiya", "registration\_date" : ISODate("2021-09-11T00:00:00Z"), "birth\_date" : ISODate("1909-04-06T00:00:00Z"), "visits" : 645, "top\_tags" : { "was" : 9, "as" : 1, "alice" : 8, "on" : 1, "own" : 8, "a" : 9 }, "karma" : -43 }

db.users.find({"karma": {$lt: -10}, "first\_name": /.\*an.\*/}) – найдем записи с указанными параметарми

{ "\_id" : "danni@example.com", "first\_name" : "Deandra", "last\_name" : "Anni", "registration\_date" : ISODate("2020-01-27T00:00:00Z"), "birth\_date" : ISODate("1975-01-28T00:00:00Z"), "visits" : 541, "top\_tags" : { "to" : 9, "of" : 3, "watch" : 10, "rabbit" : 10, "use" : 4, "by" : 10 }, "karma" : -25 }

{ "\_id" : "jathene@example.ru", "first\_name" : "Janey", "last\_name" : "Athene", "registration\_date" : ISODate("2020-04-17T00:00:00Z"), "birth\_date" : ISODate("1932-01-10T00:00:00Z"), "visits" : 803, "top\_tags" : { "stupid" : 1, "to" : 8, "tired" : 5 }, "karma" : -58 }

db.users.find({"karma": {$lt: -10}, "first\_name": /.\*an.\*/}).limit(1) – вывести только одну запись с указанными параметрами

{ "\_id" : "danni@example.com", "first\_name" : "Deandra", "last\_name" : "Anni", "registration\_date" : ISODate("2020-01-27T00:00:00Z"), "birth\_date" : ISODate("1975-01-28T00:00:00Z"), "visits" : 541, "top\_tags" : { "to" : 9, "of" : 3, "watch" : 10, "rabbit" : 10, "use" : 4, "by" : 10 }, "karma" : -25 }

db.users.updateMany({"karma": {$lt: -10}, "first\_name": /.\*an.\*/}, {$set: {"karma": 0}}) – обновили (изменили карму на значение 0)

{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }

db.users.insert({"first\_name": "Test", "karma": 100, "admin": true}) – добавляем новую запись

WriteResult({ "nInserted" : 1 })

db.users.find({"admin": true}) – найдем добавленную запись

{ "\_id" : ObjectId("66e87f44b367115d9bc8459b"), "first\_name" : "Test", "karma" : 100, "admin" : true }

db.users.insertMany([{"first\_name": "Test 2", "karma": 100, "admin": true}, {"first\_name": "Test 3", "karma": 100, "admin": true}]) – добавляем сразу несколько записей (массив)

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("66e88127b367115d9bc8459c"),

ObjectId("66e88127b367115d9bc8459d")

]

}

db.users.find({"last\_name": {$exists: false}}) – найдем все записи, у которых last\_name пустые

{ "\_id" : ObjectId("66e87f44b367115d9bc8459b"), "first\_name" : "Test", "karma" : 100, "admin" : true }

{ "\_id" : ObjectId("66e88127b367115d9bc8459c"), "first\_name" : "Test 2", "karma" : 100, "admin" : true }

{ "\_id" : ObjectId("66e88127b367115d9bc8459d"), "first\_name" : "Test 3", "karma" : 100, "admin" : true }

db.users.remove({"last\_name": {$exists: false}})

WriteResult({ "nRemoved" : 3 }) – удалим все записи, у которых last\_name пустые

db.misc.insert({"textValue": "simple text на любом языке с поддержкой utf-8 :)"})

WriteResult({ "nInserted" : 1 })

db.misc.insert({"textValue": "simple text на любом языке с поддержкой utf-8 😀)"}) – добавим в текст с эмодзи

WriteResult({ "nInserted" : 1 })

db.misc.find() – выведем все записи

{ "\_id" : ObjectId("66e88314b367115d9bc8459e"), "textValue" : "simple text на любом языке с поддержкой utf-8 :)" }

{ "\_id" : ObjectId("66e88394b367115d9bc8459f"), "textValue" : "simple text на любом языке с поддержкой utf-8 😀)" }

db.misc.insertOne({doubleValue: 9.99, integerValue: 9}); - добавим запись с числами

{

"acknowledged" : true,

"insertedId" : ObjectId("66e88458b367115d9bc845a0")

}

db.misc.insertOne({nowTs: new Timestamp(), created: ISODate("2020-01-11T00:00:00Z")}); - добавим дату

{

"acknowledged" : true,

"insertedId" : ObjectId("66e89101b367115d9bc845a1")

}

db.misc.find({"\_id": ObjectId("66e89101b367115d9bc845a1")}) – выведем добавленную ранее запись

{ "\_id" : ObjectId("66e89101b367115d9bc845a1"), "nowTs" : Timestamp(1726517505, 1), "created" : ISODate("2020-01-11T00:00:00Z") }

db.misc.insertMany([{tags: ["mongodb", "mysql"]}, {tags: ["mangodb", "postgresql"]} ]); - добавим еще запись

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("66e89236b367115d9bc845a2"),

ObjectId("66e89236b367115d9bc845a3")

]

}

db.misc.find({tags: "mysql"}) – найдем добавленный объект

{ "\_id" : ObjectId("66e89236b367115d9bc845a2"), "tags" : [ "mongodb", "mysql" ] }

db.misc.find({tags: "mysql"})

{ "\_id" : ObjectId("66e89236b367115d9bc845a2"), "tags" : [ "mongodb", "mysql" ] }

> db.misc.insert({"message": 'text', 'author': DBRef("misc", ObjectId('66e89236b367115d9bc845a3'))})

WriteResult({ "nInserted" : 1 })

> db.misc.insert({"message": 'text', 'author': DBRef("misc", "aaa")})

WriteResult({ "nInserted" : 1 })

После открытия новой сессии терминала нужно

docker ps –a – скопировать ip процесса

docker start 4bfd03de4e8e – запустить процесс

docker exec -it skillbox-mongodb mongo – выполнить команду для входа в mongodb

db.articles.find({"comments": {"$elemMatch": {"\_id": ObjectId("66e89236b367115d9bc845a3")}}}, {'comments.$': 1})

db.articles.find({"comments": {"$elemMatch": {"\_id": ObjectId("66e89236b367115d9bc845a3")}}}, {'comments.$': 1}).explain() – добавление метода explain в конце запроса, позволяет увидеть план его выполнения

db.posts.find({"author": "denvoy@example.ru"}).explain()

{

"waitedMS" : NumberLong(0),

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "skdb.posts",

"indexFilterSet" : false,

"parsedQuery" : {

"author" : {

"$eq" : "denvoy@example.ru"

}

},

"winningPlan" : {

"stage" : "COLLSCAN",

"filter" : {

"author" : {

"$eq" : "denvoy@example.ru"

}

},

"direction" : "forward"

},

"rejectedPlans" : [ ]

},

"serverInfo" : {

"host" : "4bfd03de4e8e",

"port" : 27017,

"version" : "3.1.9",

"gitVersion" : "9c317d92145a77ab582acb215b3ed41af2f1b33c"

},

"ok" : 1

}

db.posts.createIndex({"author": 1}) – создадим индекс дерева

db.posts.find({"author": "denvoy@example.ru"}).explain() – теперь не происходит сканирования всей коллекции

{

"waitedMS" : NumberLong(0),

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "skdb.posts",

"indexFilterSet" : false,

"parsedQuery" : {

"author" : {

"$eq" : "denvoy@example.ru"

}

},

"winningPlan" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"author" : 1

},

"indexName" : "author\_1",

"isMultiKey" : false,

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 1,

"direction" : "forward",

"indexBounds" : {

"author" : [

"[\"denvoy@example.ru\", \"denvoy@example.ru\"]"

]

}

}

},

"rejectedPlans" : [ ]

},

"serverInfo" : {

"host" : "4bfd03de4e8e",

"port" : 27017,

"version" : "3.1.9",

"gitVersion" : "9c317d92145a77ab582acb215b3ed41af2f1b33c"

},

"ok" : 1

}

db.posts.createIndex({author: 1, score: 1}) – создадим композитный индекс

db.places.save( {locs : [[55.5 , 42/3 ] , [ -74 , 44.74 ] , {lng : 55.5, lat : 42.3 } ] }) – сохраним данные геолокации

WriteResult({ "nInserted" : 1 })

db.places.createIndex( { "locs": "2d" } ) – создадим двухмерный индекс по двум полям

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

db.places.createIndex( { text : "text" } ) – создадим текстовый индекс

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 2,

"numIndexesAfter" : 3,

"ok" : 1

}

db.collection.createIndex( { author: "hashed" } ) – создание хэш индекса

{

"createdCollectionAutomatically" : true,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

db.collection.createIndex( { "first\_name": 1, "last\_name": 1}, {unique: true }) – добавим индекс на уникальность полей

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 2,

"numIndexesAfter" : 3,

"ok" : 1

}

db.eventlog.createIndex( { "lastModifiedDate@: 1}, { expireAfterSeconds: 3600 } ) - создадим индекс, который удалиться через 3600 секунд

2024-09-17T18:22:14.601+0000 E QUERY [thread1] SyntaxError: unterminated string literal @(shell):1:27

> db.eventlog.createIndex( { "lastModifiedDate": 1}, { expireAfterSeconds: 3600 } )

{

"createdCollectionAutomatically" : true,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

db.users.aggregate({$match: {'\_id': /example.info/}}) – выполним фильтрацию данных с содержанием example.info

{ "\_id" : "dozay@example.info", "first\_name" : "Dore", "last\_name" : "Ozay", "registration\_date" : ISODate("2021-07-01T00:00:00Z"), "birth\_date" : ISODate("1990-07-24T00:00:00Z"), "visits" : 969, "top\_tags" : { "would" : 8, "and" : 6, "pictures" : 6, "be" : 8, "nor" : 1 }, "karma" : -46 }

{ "\_id" : "etxp@example.info", "first\_name" : "Elsi", "last\_name" : "Txp", "registration\_date" : ISODate("2020-01-02T00:00:00Z"), "birth\_date" : ISODate("1975-05-18T00:00:00Z"), "visits" : 472, "top\_tags" : { "nor" : 8, "a" : 10, "her" : 9, "as" : 3, "have" : 9 }, "karma" : 120 }

{ "\_id" : "jcolleen@example.info", "first\_name" : "Jovan", "last\_name" : "Colleen", "registration\_date" : ISODate("2020-02-25T00:00:00Z"), "birth\_date" : ISODate("1919-05-15T00:00:00Z"), "visits" : 329, "top\_tags" : { "watch" : 1, "remarkable" : 3, "rabbit" : 3, "at" : 5, "as" : 2, "a" : 4 }, "karma" : 106 }

{ "\_id" : "kdelphine@example.info", "first\_name" : "Katya", "last\_name" : "Delphine", "registration\_date" : ISODate("2020-03-18T00:00:00Z"), "birth\_date" : ISODate("2003-04-14T00:00:00Z"), "visits" : 188, "top\_tags" : { "bank" : 1, "seemed" : 4 }, "karma" : 97 }

{ "\_id" : "lluc@example.info", "first\_name" : "Leny", "last\_name" : "Luc", "registration\_date" : ISODate("2021-03-11T00:00:00Z"), "birth\_date" : ISODate("1995-09-05T00:00:00Z"), "visits" : 495, "top\_tags" : { "a" : 3, "her" : 9, "book" : 3 }, "karma" : -33 }

{ "\_id" : "mmickey@example.info", "first\_name" : "Mamie", "last\_name" : "Mickey", "registration\_date" : ISODate("2020-01-12T00:00:00Z"), "birth\_date" : ISODate("1960-07-11T00:00:00Z"), "visits" : 805, "top\_tags" : { "a" : 5, "book" : 5, "trouble" : 8 }, "karma" : 80 }

{ "\_id" : "oray@example.info", "first\_name" : "Oorschot", "last\_name" : "Ray", "registration\_date" : ISODate("2021-04-23T00:00:00Z"), "birth\_date" : ISODate("1968-07-21T00:00:00Z"), "visits" : 907, "top\_tags" : { "of" : 8, "whether" : 9, "pictures" : 8, "the" : 10, "it" : 8, "but" : 7 }, "karma" : -3 }

db.users.aggregate([{$match: {'\_id': /example.info/}}, {$project: {karma: "$karma", year: {$year: "$birth\_date"}}}]) – добавим дополнительные условия для фильтрации

{ "\_id" : "dozay@example.info", "karma" : -46, "year" : 1990 }

{ "\_id" : "etxp@example.info", "karma" : 120, "year" : 1975 }

{ "\_id" : "jcolleen@example.info", "karma" : 106, "year" : 1919 }

{ "\_id" : "kdelphine@example.info", "karma" : 97, "year" : 2003 }

{ "\_id" : "lluc@example.info", "karma" : -33, "year" : 1995 }

{ "\_id" : "mmickey@example.info", "karma" : 80, "year" : 1960 }

{ "\_id" : "oray@example.info", "karma" : -3, "year" : 1968 }

db.users.aggregate([{$match: {'\_id': /example.info/}}, {$project: {karma: "$karma", year: {$year: "$birth\_date"}}}, { $group: {\_id: "$year", avg\_karma: { $avg: "$karma" } } }]) – дополнительные параметры группировки по значению года и кармы (среднее значение для кармы)

{ "\_id" : 1960, "avg\_karma" : 80 }

{ "\_id" : 1995, "avg\_karma" : -33 }

{ "\_id" : 2003, "avg\_karma" : 97 }

{ "\_id" : 1919, "avg\_karma" : 106 }

{ "\_id" : 1975, "avg\_karma" : 120 }

{ "\_id" : 1968, "avg\_karma" : -3 }

{ "\_id" : 1990, "avg\_karma" : -46 }

db.posts.aggregate({$unwind: "$topics"}, {$project: {"topic": "$topics"}}) – посчитает количество записей, в которые удовлетворяют условиям. unwind декомпозирует топики, делает из массива значение и перебирает весь массив.

{ "\_id" : ObjectId("66e879c1ff281cd59da15d34"), "topic" : "oh" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d34"), "topic" : "own" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d34"), "topic" : "having" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d34"), "topic" : "own" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d35"), "topic" : "use" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d35"), "topic" : "nor" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d35"), "topic" : "i" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d36"), "topic" : "once" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d36"), "topic" : "way" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d36"), "topic" : "use" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d36"), "topic" : "that" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d37"), "topic" : "tired" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d37"), "topic" : "sitting" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d37"), "topic" : "tired" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d38"), "topic" : "oh" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d38"), "topic" : "and" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d38"), "topic" : "would" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d39"), "topic" : "once" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d39"), "topic" : "oh" }

{ "\_id" : ObjectId("66e879c1ff281cd59da15d39"), "topic" : "actually" }

Type "it" for more

db.posts.aggregate({$unwind: "$topics"}, {$project: {"topic": "$topics"}}, {$group: {\_id: "$topic", cnt: {$sum: 1}}}) – добавим еще группировку по значению топика и подсчитаем сколько раз встречается тот или иной топик

{ "\_id" : "there", "cnt" : 8 }

{ "\_id" : "stupid", "cnt" : 6 }

{ "\_id" : "get", "cnt" : 7 }

{ "\_id" : "nothing", "cnt" : 4 }

{ "\_id" : "daisy", "cnt" : 6 }

{ "\_id" : "bank", "cnt" : 10 }

{ "\_id" : "reading", "cnt" : 12 }

{ "\_id" : "book", "cnt" : 9 }

{ "\_id" : "to", "cnt" : 8 }

{ "\_id" : "could", "cnt" : 5 }

{ "\_id" : "remarkable", "cnt" : 14 }

{ "\_id" : "when", "cnt" : 9 }

{ "\_id" : "whether", "cnt" : 9 }

{ "\_id" : "her", "cnt" : 9 }

{ "\_id" : "worth", "cnt" : 13 }

{ "\_id" : "on", "cnt" : 11 }

{ "\_id" : "into", "cnt" : 9 }

{ "\_id" : "by", "cnt" : 11 }

{ "\_id" : "thought", "cnt" : 9 }

{ "\_id" : "over", "cnt" : 9 }

Type "it" for more

db.posts.aggregate({$unwind: "$topics"}, {$project: {"topic": "$topics"}}, {$group: {\_id: "$topic", cnt: {$sum: 1}}}, {$sort: {cnt: -1}}, {$limit: 10}) – добавим еще сортировку в обратном порядке 10 топиков

{ "\_id" : "use", "cnt" : 20 }

{ "\_id" : "pink", "cnt" : 19 }

{ "\_id" : "having", "cnt" : 17 }

{ "\_id" : "nor", "cnt" : 17 }

{ "\_id" : "that", "cnt" : 17 }

{ "\_id" : "down", "cnt" : 17 }

{ "\_id" : "a", "cnt" : 17 }

{ "\_id" : "pleasure", "cnt" : 17 }

{ "\_id" : "made", "cnt" : 16 }

{ "\_id" : "hot", "cnt" : 16 }

db.system.js.save({\_id: "sum", value: function (x, y) { return x + y; }}); - запишем функцию для подсчета суммы

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" : "sum" })

db.eval("return sum(2, 3);"); - вызовем функцию и подсчитаем сумму чисел

WARNING: db.eval is deprecated

5