Лабораторна робота №2

**ТЕМА: Влаштування бази даних. Документи**

Джерела: https://metanit.com/nosql/mongodb/2.3.php

https://www.mongodb.com/json-and-bson

https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON

**Проблеми**

Якщо минуло разу ви забулися зупинити БД, то можете отримати проблеми із зупуском mongod (серверу БД). Для їх вирішення, в командному рядку (від Адміністратора) наберіть наступне:

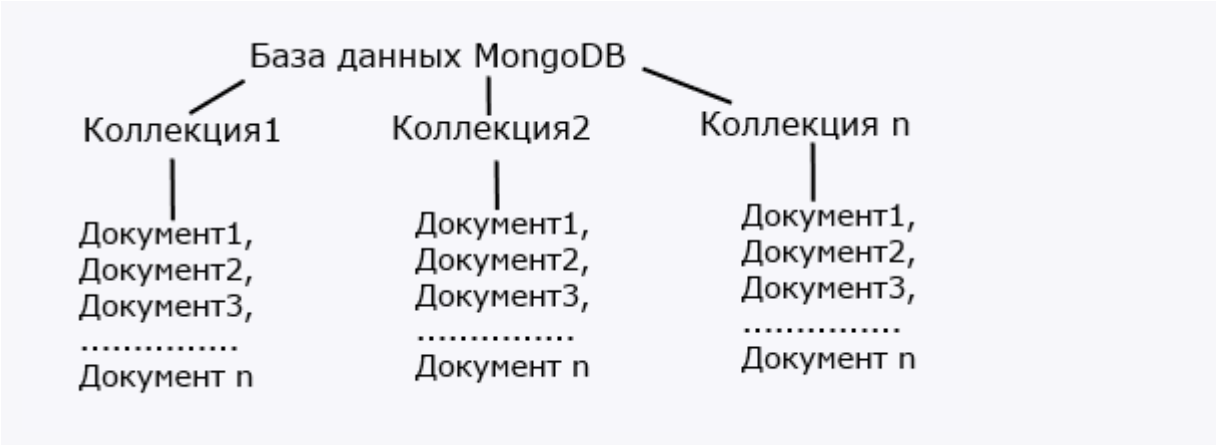
```
mongod --repair
```

або

```
mongod --dbpath /data/db --repair
```

**Влаштування бази даних**

Всю модель пристрою бази даних MongoDB можна представити таким чином:




Якщо у реляційних бд вміст складають таблиці, то mongodb база даних складається з колекцій.

Кожна колекція має своє унікальне ім'я - довільний ідентифікатор, що складається з не більш ніж 128 різних алфавітно-цифрових символів та підкреслення.

На відміну від реляційних баз даних MongoDB не використовує табличний пристрій із чітко заданою кількістю стовпців та типів даних. MongoDB є документоорієнтованою системою, в якій центральним поняттям є документ.

Документ можна подати як об'єкт, який зберігає певну інформацію. У певному сенсі він подібний до рядків у реляційних субд, де рядки зберігають інформацію про окремий елемент. Наприклад, типовий документ:

```
{
    "name": "Tom",
    "surname": "Smith",
    "age": "37",
    "company": {
        "name" : "Microsoft",
        "salary" : "100"
    }
}
```

Документ представляє набір пар ключ-значення. Наприклад, у виразі "name": "Tom" "name" представляє ключ, а "Tom" - значення.

Ключі представляють рядки. Значення можуть відрізнятися за типом даних. У цьому випадку майже всі значення також представляють рядковий тип, і лише один ключ (company) посилається на окремий об'єкт. Усього є такі типи значень:

**String:** рядковий тип даних, як у наведеному вище прикладі (для рядків використовується кодування UTF-8)

**Array (масив):** тип даних для зберігання масивів елементів

**Binary data (двійкові дані):** тип зберігання даних у бінарному форматі

**Boolean:** булевий тип даних, що зберігає логічні значення TRUE або FALSE, наприклад, {"married": FALSE}

**Date:** зберігає дату у форматі часу Unix

**Double:** числовий тип даних для зберігання чисел з плаваючою точкою

**Integer:** використовується для зберігання цілих значень розміром 32 біти, наприклад, {"age": 29}

**Long:** використовується для зберігання цілих значень розміром 64 біти

**JavaScript:** тип даних для зберігання коду javascript

**Min key/Max key:** використовуються для порівняння значень із найменшим/найбільшим елементів BSON

**Null:** тип даних для зберігання Null

**Object:** рядковий тип даних, як у наведеному вище прикладі

**ObjectId:** тип даних для зберігання id документа

**Regular expression:** застосовується для зберігання регулярних виразів

**Decimal128:** тип даних для зберігання десяткових дробових чисел розміром 128 біт, які дозволяють вирішити проблеми з проблемою точності обчислень при використанні дробових чисел, які представляють тип Double.

**Timestamp:** застосовується для зберігання часу

На відміну від рядків, документи можуть містити різнорідну інформацію. Так, поруч із документом, описаним вище, в одній колекції може бути інший об'єкт, наприклад:

```
{
    "name": "Bob",
    "birthday": "1985.06.28",
    "place" : "Berlin",
    "languages" :[
        "english",
        "german",
        "spanish"
    ]
}
```

Ще пара важливих зауважень: в MongoDB запити мають реєстрозалежність і строгу типізацію. Тобто наступні два документи не будуть ідентичними:

```
1 {"age" : "28"}
2 {"age" : 28}
```

Якщо в першому випадку для ключа age визначено значенням рядка, то в другому випадку значенням є число.

**Ідентифікатор документа**

Для кожного документа MongoDB визначено унікальний ідентифікатор, який називається _id. При додаванні документа до колекції цей ідентифікатор створюється автоматично. Однак розробник може сам явно задати ідентифікатор, а не покладатися на автоматично генеровані, вказавши відповідний ключ і його значення в документі.

Це поле має мати унікальне значення в межах колекції. І якщо ми спробуємо додати до колекції два документи з однаковим ідентифікатором, то буде додано лише один із них, а при додаванні другого ми отримаємо помилку.

Якщо ідентифікатор не заданий явно, MongoDB створює спеціальне бінарне значення розміром 12 байт. Це значення складається з декількох сегментів: значення типу timestamp розміром 4 байта (що представляє кількість секунд з моменту початку епохи

Unix), випадкове число з 5 байт і лічильник з 3 байт, який ініціальований випадковим числом. Така модель побудови ідентифікатора гарантує з високою ймовірністю, що він матиме унікальне значення.

**Адміністрування mongodb**

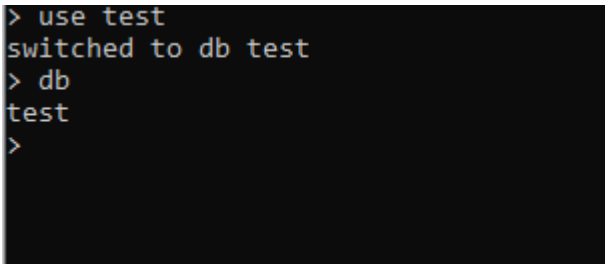
Отже, запустимо консольну оболонку mongo.exe і введемо наступну команду: *use test*. Тепер як поточна буде встановлена БД test.

Якщо ви раптом не впевнені, а чи вже існує база даних з такою назвою, то за допомогою команди show dbs можна вивести назви всіх наявних бд на консоль.

Для бази даних можна встановити будь-яке ім'я, однак є деякі обмеження. Наприклад, в імені не повинно бути символів /, \, ., ", *, <, >, :, |, ?, $. Крім того, імена баз даних обмежені 64 байтами.

Також є зарезервовані імена, які можна використовувати: local, admin, config. Ці імена є базами даних, які вже маються за замовчуванням на сервері і призначені для службових цілей.

Якщо ми хочемо дізнатися, яка бд використовується зараз, то ми можемо скористатися командою db:




Окрім баз даних, ми можемо переглянути список усіх колекцій у поточній бд за допомогою команди show collections.

**Отримання статистики**

Використовуючи команду db.stats(), можна отримати статистику поточної бази даних. Наприклад, у нас як поточна встановлена база даних test:

```
> db.stats()
{
        "db" : "test",
        "collections" : 3,
        "objects" : 5,
        "avgObjSize" : 73.6,
        "dataSize" : 368,
        "storageSize" : 24576,
        "numExtents" : 3,
        "indexes" : 1,
        "indexSize" : 8176,
        "fileSize" : 67108864,
        "nsSizeMB" : 16,
        "dataFileVersion" : {
                "major" : 4,
                "minor" : 5
        },
        "extentFreeList" : {
                "num" : 0,
                "totalSize" : 0
        },
        "ok" : 1
}
>
```

**Що таке JSON?**

Нотація об’єктів JavaScript, більш відома як JSON, була визначена як частина мови JavaScript на початку 2000-х років творцем JavaScript Дугласом Крокфордом, хоча формат був офіційно визначений лише у 2013 році.

Об’єкти JavaScript – це прості асоціативні контейнери, де ключ рядка зіставляється зі значенням (яким може бути число, рядок, функція або навіть інший об’єкт). Ця проста властивість мови дозволила об’єктам JavaScript бути надзвичайно просто представленими в тексті:

```
{
  "_id": 1,
  "name" : { "first" : "John", "last" : "Backus" },
  "contribs" : [ "Fortran", "ALGOL", "Backus-Naur Form", "FP" ],
  "awards" : [
    {
      "award" : "W.W. McDowell Award",
      "year" : 1967,
      "by" : "IEEE Computer Society"
    }, {
      "award" : "Draper Prize",
      "year" : 1993,
      "by" : "National Academy of Engineering"
    }
  ]
}
```

```
[
  {
    "name": "Molecule Man",
    "age": 29,
    "secretIdentity": "Dan Jukes",
    "powers": [
      "Radiation resistance",
      "Turning tiny",
      "Radiation blast"
    ]
  },
  {
    "name": "Madame Uppercut",
    "age": 39,
    "secretIdentity": "Jane Wilson",
    "powers": [
      "Million tonne punch",
      "Damage resistance",
      "Superhuman reflexes"
    ]
  }
]
```

Оскільки JavaScript став мовою за замовчуванням для веб-розробки на стороні клієнта, JSON почав жити власним життям. Завдяки тому, що він зчитається як людиною, так і машиною, а також порівняно простий у реалізації підтримки іншими мовами, JSON швидко вийшов за межі веб-сторінки.

JSON використовується в багатьох різних випадках:

API

Конфігураційні файли

Журнал повідомлень

Зберігання бази даних

JSON швидко обігнав XML, важчий для читання людиною, значно більш докладний і менш ідеально підходить для представлення об’єктних структур, які використовуються в сучасних мовах програмування.

Підключення MongoDB JSON

MongoDB була розроблена з самого початку як ідеальна платформа даних для розробки сучасних додатків. Повсюдність JSON зробила його очевидним вибором для представлення структур даних в інноваційній моделі даних документів MongoDB.

Однак є кілька проблем, які роблять JSON менш ідеальним для використання всередині бази даних.

JSON — це текстовий формат, і синтаксичний аналіз тексту дуже повільний

Читабельний формат JSON далекий від використання всюди, ще одна проблема з базою даних

JSON підтримує лише обмежену кількість основних типів даних

Щоб зробити JSON швидшим, але все ще високопродуктивним і універсальним, був винайдений BSON, щоб подолати прогалини: було розроблене двійкове подання для зберігання даних у форматі JSON, оптимізоване для швидкості та гнучкості.

Що таке BSON?

BSON просто означає «двійковий JSON», і це саме те, для чого його придумали. Двійкова структура BSON кодує інформацію про тип і довжину, що дозволяє аналізувати її набагато швидше.

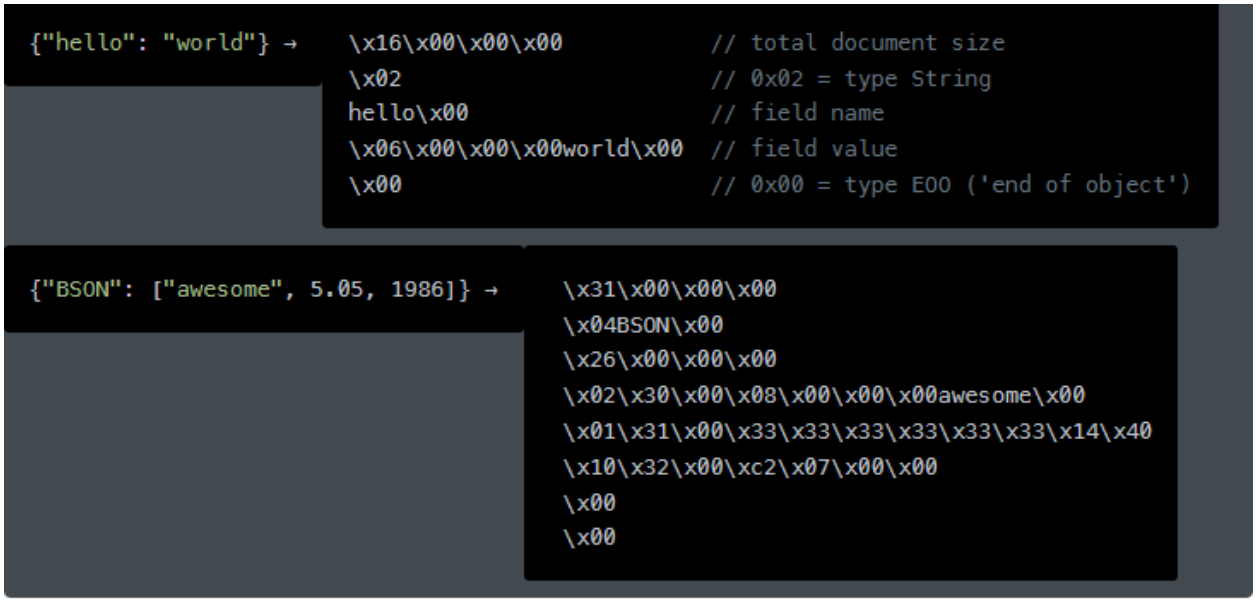
З моменту свого початкового формулювання BSON було розширено, щоб додати деякі додаткові типи даних, які не належать до JSON, як-от дати та двійкові дані, без яких MongoDB не мав би цінної підтримки.

Мови, які підтримують будь-який вид складної математики, зазвичай мають цілі числа різного розміру (ints або long) або різні рівні десяткової точності (float, double, decimal128 тощо).

Це не тільки корисно відображати ці відмінності в даних, що зберігаються в MongoDB, це також дозволяє проводити порівняння та обчислення безпосередньо з даними способами, які спрощують використання коду програми.

MongoDB зберігає дані у форматі BSON як всередині, так і через мережу, але це не означає, що ви не можете вважати MongoDB базою даних JSON. Все, що ви можете представити в JSON, можна зберігати в MongoDB і так само легко отримати в JSON.

Нижче наведено деякі приклади документів (у синтаксисі стилю JavaScript/Python) та відповідні їм представлення BSON.




**Додавання даних**

Всі дані зберігаються в БД у форматі BSON, який близький до JSON, тому нам також потрібно вводити дані в цьому форматі. І хоча у нас, можливо, на даний момент немає жодної колекції, але при додаванні до неї даних вона автоматично створюється.

Як раніше говорилося, ім'я колекції – довільний ідентифікатор, що складається з не більше ніж 128 різних алфавітно-цифрових символів та знака підкреслення. У той же час ім'я колекції не повинне починатися з префіксу system. І також ім'я не повинно містити знак долара - $.

Для додавання до колекції можуть використовуватись три її методи:

insertOne(): додає один документ

insertMany(): додає кілька документів

insert(): може додавати як один, так і кілька документів

Отже, додамо один документ:

```
> db.users.insert({"name": "Tom", "age": 28, languages: ["english", "spanish"]})
WriteResult({ "nInserted" : 1 })
>
```

Документ представляє набір пар ключ-значення. У цьому випадку документ, що додається, має три ключі: name, age, languages, і кожному з них зіставляє певне значення. Наприклад, ключу languages в якості значення зіставляється масив.

Деякі обмеження під час використання імен ключів:

Символ $ не може бути першим символом у імені ключа

Ім'я ключа не може містити символ крапки.

При додаванні даних, якщо ми явно не надали значення для поля "_id" (тобто унікального ідентифікатора документа), воно генерується автоматично. Але, в принципі, ми можемо самі встановити цей ідентифікатор при додаванні даних:

```
> db.users.insert({"_id": 23457, "name": "Tom", "age": 28, languages: ["english", "spanish"]})
WriteResult({ "nInserted" : 1 })
```

Варто відзначити, що назви ключів можуть використовуватися в лапках, а можуть і без лапок. У разі успішного додавання на консоль буде виведено ідентифікатор доданого документа. І щоб переконатися, що документ у бд, ми його виводимо функцією find.

```
> db.users.find()
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : 23457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
>
```

Щоб вивести у більш читальному вигляді додамо метод pretty():

```
> db.users.find().pretty()
{
        "_id" : 123457,
        "name" : "Tom",
        "age" : 28,
        "languages" : [
                "english",
                "spanish"
        ]
}
{
        "_id" : ObjectId("61f7974838c8ff5ef4d80da7"),
        "name" : "Tom",
        "age" : 28,
        "languages" : [
                "english",
                "spanish"
        ]
}
{
        "_id" : 23457,
        "name" : "Tom",
        "age" : 28,
        "languages" : [
                "english",
                "spanish"
        ]
}
```

Є ще один спосіб додавання до бд документа, який включає два етапи: визначення документа (document = ( { ... } )) і власне його додавання:

```
> document=({"name": "Bill", "age": 32, languages: ["english", "french"]})
{ "name" : "Bill", "age" : 32, "languages" : [ "english", "french" ] }
> db.users.insert(document)
WriteResult({ "nInserted" : 1 })
>
```

## ЗАВДАННЯ

1. В текстовому редакторі написати у форматі JSON множини структур за варіантами (по три множини, щоб в одній був присутній масив об’єктів).
2. Створити БД TEST. Додати до неї множини із завдання 1.
3. Використати функцію pretty() при показі викладачу.

ВАРІАНТИ

| | |
|---|---|
| 1 | Структура: назва торгової організації, адреса, ПІБ директора, податковий номер, постачальники (може бути декілька) |
| 2 | Структура: ПІБ, посада (може бути декілька), номер відділу (може бути декілька), в якому працює, стать, адреса, дата народження. |
| 3 | Структура: ПІБ, паспортні дані, стать, вік, діти (може бути декілька) |
| 4 | Структура: назва туру, країна (може бути декілька), міста (може бути декілька), тип пересування (може бути декілька), тип харчування, ціна туру, тип проживання |
| 5 | Структура: прізвище, ім'я, по батькові, адреса, місто, телефон (може бути декілька). |
| 6 | Структура: назва кафедри, ПІБ завідувача, номер кімнати, номер корпусу, телефон, викладачі (може бути декілька). |
| 7 | Структура: назва факультету, ПІБ декана, номер кімнати, номер корпусу, телефон (може бути декілька) |
| 8 | Структура: назва відділу продаж, кількість прилавків, продавці (може бути декілька), номер залу (може бути декілька) |
| 9 | Структура: назва маршруту, транспорт, водій (може бути декілька), графік роботи. |
| 10 | Структура: назва відділення (хірургія, терапія, неврологія і т.д.), поверх, номери кімнат (може бути декілька), ПІБ завідувача. |