Лабораторна робота №3

ТЕМА: Вибірка із БД

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

http://matphys.rpd.univ.kiev.ua/downloads/courses/programming_for_ce/RegExp.pdf

Фільтрування даних

Якщо нам треба отримати не всі документи, а лише ті, які задовольняють певній вимогі, то використовуємо функцію find(). Виведемо всі документи, у яких name=Tom:

```
> db.users.find({name:Tom})
2022-02-07T11:03:02.510+0200 ReferenceError: Tom is not defined
> db.users.find({name:"Tom"})
{ "_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" ] }
>
```

Тепер складніший запит: нам треба вивести ті об'єкти, які мають name=Tom і одночасно age=32. Тобто мовою SQL це міг би виглядати так: SELECT * FROM Table WHERE Name='Tom' AND Age=32. Даному критерію відповідає останній доданий об'єкт. Тоді ми можемо написати наступний запит:

```
> db.users.find({name: "Bill", age: 32})
{ "_id" : ObjectId("61f79a9d38c8ff5ef4d80da8"), "name" : "Bill", "age" : 32, "languages" : [ "english", "french" ] }
>
```

Фільтрування за відсутніми властивостями

Якісь документи можуть мати певну властивість, інші можуть не мати. Що якщо ми хочемо отримати документи, у яких немає певної властивості? І тут для якості передається значення null. Наприклад, знайдемо всі документи, де відсутня властивість languages:

```
> db.users.find({languages:null})
>
```

Або знайдемо всі документи, де name="Tom", але властивість languages не визначено.

```
> db.users.find({name: "Tom", languages:null})
>
```

Фільтрування за елементами масиву

Також легко знайти по елементу в масиві. Наприклад, наступний запит виводить усі документи, у яких у масиві languages є english:

> db.users.find({languages: "english"})

Ускладнимо запит і отримаємо ті документи, у яких в масиві languages одночасно дві мови: "english" і "german":

> db.users.find({languages: ["english", "german"]})

Причому саме у цьому порядку, де "english" визначено першим, а "german" - другим.

Тепер виведемо всі документи, в яких "english" у масиві languages знаходиться на першому місці:

> db.users.find({"languages.0": "english"})

Зверніть увагу, що "languages.0" надає складну властивість і тому береться в лапки. Відповідно, якщо нам треба вивести документи, де english на другому місці (наприклад, ["german", "english"]), то замість нуля ставимо одиницю: "languages.1".

Проекція

Документ може мати безліч полів, але не всі ці поля нам можуть бути потрібними і важливими при запиті. І в цьому випадку ми можемо включити у вибірку лише потрібні поля, використовуючи проекцію. Наприклад, виведемо тільки порцію інформації, наприклад, значення полів "age" у всіх документів, у яких name=Tom:

```
> db.users.find({name: "Tom"},{age:1})
{ "_id" : 123457, "age" : 28 }
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "age" : 28 }
{ "_id" : 23457, "age" : 28 }
>
```

І зворотна ситуація: ми хочемо знайти всі поля документа, крім властивості age. В цьому випадку як параметр вказуємо 0:

```
> db.users.find({name: "Tom"},{age:0})
{ "_id" : 123457, "name" : "Tom", "languages" : [ "english", "spanish" ] }
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "name" : "Tom", "languages" : [ "english", "spanish" ] }
{ "_id" : 23457, "name" : "Tom", "languages" : [ "english", "spanish" ] }
>
```

При цьому треба враховувати, що навіть якщо ми зауважимо, що ми хочемо отримати тільки поле name, поле _id також буде включено до результуючої вибірки. Тому, якщо ми не хочемо бачити це поле у вибірці, то треба явно вказати: {"_id":0}

Альтернативно замість 1 та 0 можна використовувати true та false:

```
> db.users.find({name: "Tom"},{age:true, _id: false})
{ "age" : 28 }
{ "age" : 28 }
{ "age" : 28 }
>
```

Якщо ми не хочемо при цьому конкретизувати вибірку, а хочемо вивести всі документи, можна залишити перші фігурні дужки порожніми:

```
> db.users.find({},{age:true, _id: false})
{ "age" : 28 }
{ "age" : 28 }
{ "age" : 28 }
{ "age" : 32 }
>
```

Запит до вкладених об'єктів

Попередні запити застосовувалися до найпростіших об'єктів. Але документи можуть бути дуже складними за структурою. Наприклад, додамо до колекції users наступний документ:

```
> db.users.insert({"name": "Alex", "age": 28, company: {"name":"Microsoft", "country":"USA"}})
WriteResult({ "nInserted" : 1 })
```

Тут визначається вкладений об'єкт із ключем company. І щоб знайти всі документи, у яких у ключі company вкладена властивість name=microsoft, нам потрібно використовувати оператор точку:

```
> db.users.find({"company.name": "Microsoft"})
{ "_id" : ObjectId("6200e506f2681a8dc1af12f6"), "name" : "Alex", "age" : 28, "company" : { "name" : "Microsoft", "countr
y" : "USA" } }
>
```

РЕГУЛЯРНІ ВИРАЗИ

Метасимвол (англ. metacharacter) – це символ або комбінація символів, яка має спеціальне значення, якщо вона зустрічається у складі регулярного виразу. Найпростіший приклад метасимволів – символи узагальнення.

Символ узагальнення (англ. wildcard character) – символ, який дає змогу робити у рядку підстановку інших символів, щоб в одному запиті шукати, наприклад, одразу багато файлів.

Зазвичай з цією метою вживають символи «*» (заміняє довільне число інших символів) та «?» (заміняє один довільний символ).

Регулярний вираз (англ. regular expression, regexp) – це зразок (англ. pattern) (послідовність символів та метасимволів), який відповідає (або не відповідає) послідовностям символів у тексті. Як правило, регулярні вирази використовуються для того, щоб задати правило пошуку підрядка у тексті.

Регулярні вирази використовуються не самі по собі, а разом з якоюсь мовою програмування або застосуванням. Синтаксис регулярних виразів може дещо відрізнятися, в залежності від того, як саме вони використовуються Деякі метасимволи:

. (крапка) – один довільний символ (крім символу переходу на новий рядок)

\t – табуляція

\n – новий рядок

\\ – власне \ (backslash)

\s – один довільний пробільний символ (пробіл, табуляція, новий рядок)

\S – один довільний символ, який не входить у перелічені для \ s

\d – одна довільна цифра (digit)

\D – один довільний символ, який не може бути цифрою

\w – одна довільна літера або цифра або знак підкреслювання (word character), те саме що [A-Za-z0-9_]

\W – один довільний символ, який не входить у перелічені для \w

- – позначає діапазон(у класі символів) або власне цей символ, якщо це перший символ у класі ([-abc])

$ – кінець рядка

^ – початок рядка або заперечення, якщо це перший символ у класі символів ([^abc])

^$ – пустий рядок

| - логічне АБО (використовується у групі символів)

\< – початок слова

\> – кінець слова

\b – межа слова (початок або кінець) або символ backspace, якщо він знаходиться у класі символів

\B – позиція, що не є межею слова

\. – власне крапка

\$ – власне символ $

\[, \] – власне квадратні дужки

\(, \) – власне круглі дужки

\{, \} – власне фігурні дужки

Групування та повторення

* – нуль або більше разів повторений попередній символ (або група символів)

+ – один або більше разів повторений попередній символ (або група символів)

? – нуль або один раз повторений попередній символ (або група символів)\

() – групують символи (всі, що присутні у дужках, можливе застосування логічного АБО (символ |))

[] – визначають клас (або множину) символів – неупорядковану групу символів, з якої для відповідного регулярного виразу обирається один довільний

{n} – рівно n разів повторений попередній символ (або група символів)

{n, m} – попередній символ (або група символів) повторений від n до m разів

{n, } – n або більше разів повторений попередній символ (або група символів)

При визначенні діапазонів символів слід враховувати, що вважається, що символи беруться з таблиці ASCII або Unicode (з якої саме, буде визначатися конкретним випадком використання).

Приклади:

[A-Za-z] – один довільний символ латинської абетки, незалежно від регістру (тут не можна писати [A-z], бо між літерами у великому та малому регістрах знаходяться інші символи)

[0-9] або (0|1|2|3|4|5|6|7|8|9) або \d – одна довільна цифра

\(\d{3}\) \d{3}-\d{4} – номер телефону у форматі (044) 123-4567

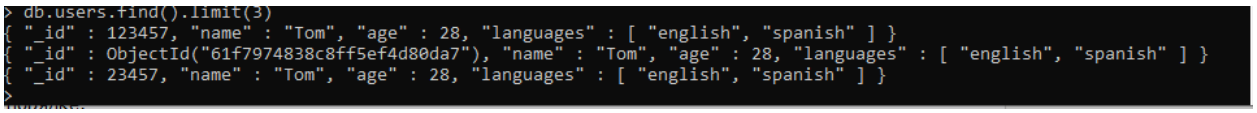
#[0-9a-fA-F]{6} – шістнадцятковий код кольору (наприклад, #12CCAA)

Використання регулярних виразів

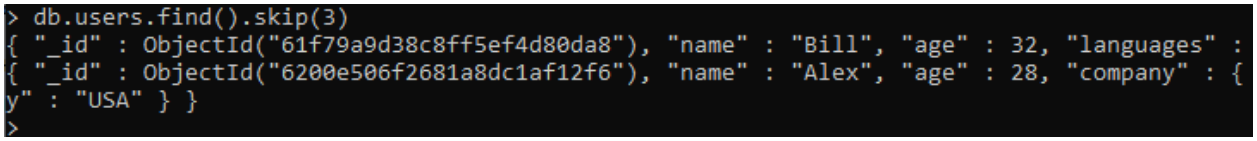
Ще однією чудовою можливістю при побудові запитів є використання регулярних виразів. Наприклад, знайдемо всі документи, у яких значення ключа name починається з літери B:

Налаштування запитів та сортування

MongoDB представляє ряд функцій, які допомагають керувати вибіркою із бд. Одна з них – функція limit. Вона визначає максимально допустиму кількість одержуваних документів. Кількість передається як числового параметра. Наприклад, обмежимо вибірку трьома документами:



В даному випадку ми отримаємо перші три документи (якщо в колекції 3 та більше документів). Але що, якщо ми хочемо зробити вибірку не спочатку, а пропустивши якусь кількість документів? У цьому допоможе функція skip. Наприклад, пропустимо перші три записи:



MongoDB дозволяє відсортувати отриманий з бд набір даних за допомогою функції sort. Передаючи в цю функцію значення 1 або -1, ми можемо вказати, в якому порядку сортувати: за зростанням (1) або за спаданням (-1). Багато в чому ця функція аналогічна виразу ORDER BY в SQL. Наприклад, сортування за зростанням по полю name:

```
> db.users.find().sort({name:1})
{ "_id" : ObjectId("6200e506f2681a8dc1af12f6"), "name" : "Alex", "age" : 28
y" : "USA" } }
{ "_id" : ObjectId("61f79a9d38c8ff5ef4d80da8"), "name" : "Bill", "age" : 32
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "s
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "name" : "Tom", "age" : 28,
{ "_id" : 23457, "name" : "Tom", "age" : 28, "languages" : [ "english", "sp
>
```

Ну і в кінці треба відзначити, що ми можемо поєднувати всі ці функції в одному ланцюжку:

```
> db.users.find().sort({name:1}).skip(3).limit(3)
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : 23457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
>
```

Пошук одиночного документа

Якщо всі документи виймаються функцією find, то одиночний документ витягується функцією findOne. Її дія аналогічна тому, якби ми використовували функцію limit(1), яка також отримує перший документ колекції. А комбінація функцій skip та limit витягне документ за потрібним місцезнаходженням.

Параметр $natural

Якщо раптом нам потрібно відсортувати обмежену колекцію, ми можемо скористатися параметром $natural. Цей параметр дозволяє встановити сортування: документи передаються в тому порядку, в якому вони були додані в колекцію, або у зворотному порядку.

Наприклад, відберемо останні п'ять документів:

```
> db.users.find().sort({$natural:-1}).limit(5)
{ "_id" : ObjectId("6200e506f2681a8dc1af12f6"), "name" : "Alex", "age" :
y" : "USA" } }
{ "_id" : ObjectId("61f79a9d38c8ff5ef4d80da8"), "name" : "Bill", "age" :
{ "_id" : 23457, "name" : "Tom", "age" : 28, "languages" : [ "english",
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "name" : "Tom", "age" :
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english",
>
```

Оператор $slice

$slice є певною мірою комбінацією функцій limit і skip. Але, на відміну від них, $slice може працювати з масивами.

Оператор $slice приймає два параметри. Перший параметр вказує на загальну кількість документів, що повертаються. Другий параметр необов'язковий, але якщо він використовується, тоді перший параметр вказує на зсув відносно початку (як функція skip), а другий - на обмеження кількості документів, що витягуються.

Наприклад, у кожному документі визначено масив мов для зберігання мов, на яких говорить людина. Їх може бути і 1, і 2, і 3 та більше. І ми хочемо при виведенні документів зробити так, щоб у вибірку попадала тільки одна мова з масиву languages, а не весь масив:

```
> db.users.find ({name: "Tom"}, {languages: {$slice : 1}})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english" ] }
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "name" : "Tom", "age" : 28, "languages" : [ "english" ] }
{ "_id" : 23457, "name" : "Tom", "age" : 28, "languages" : [ "english" ] }
```

Даний запит під час вилучення документа залишить у результаті лише першу мову з масиву languages, тобто у цьому випадку english.

Назад: нам треба залишити в масиві також один елемент, але не з початку, а з кінця. У цьому випадку необхідно передати до параметра негативне значення:

```
> db.users.find ({name: "Tom"}, {languages: {$slice : -1}})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "spanish" ] }
{ "_id" : ObjectId("61f7974838c8ff5ef4d80da7"), "name" : "Tom", "age" : 28, "languages" : [ "spanish" ] }
{ "_id" : 23457, "name" : "Tom", "age" : 28, "languages" : [ "spanish" ] }
>
```

ЗАВДАННЯ

Варіанти обираються за номером в списку підгрупи

ВАРІАНТИ ПРЕДМЕТНИХ ОБЛАСТЕЙ

1	Сховище «Деканат», в якому зберігається інформація про студентів
2	БД для пошукової системи, в якій зберігається інформація про документи, розміщені в мережі Internet
3	Сховище стандартів мов програмування
4	Сховище, в якому зберігається інформація про художні фільми
5	Сховище, в якому зберігається інформація про комп’ютерні ігри

6	Сховище, в якому зберігаються інструкції по роботі програмного забезпечення
7	Сховище для централізованої системи контролю версій (типу Git)
8	БД інструкцій до гаджетів
9	Сховище стандартів баз даних
10	Сховище для соціальної мережі типу “TIK-TOK”

1. Створити БД
2. Для своєї БД виконати всі запити, які надані в теоретичній частині цієї л/р (для регулярних виразів створити запит, який може бути характерним для вашої ПО).