Лабораторна робота №4

ТЕМА: Аналіз предметної області. Команди групування (mongodb) та оператори вибірки

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

https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/latest/44233/index.html

1. **Аналіз предметної області**

Опис. Аналіз предметної області складається з аналізу даних та аналізу завдань. Аналіз даних передбачає документування всіх атрибутів. Аналіз завдань може потребувати застосування різноманітних методів побудови діаграм для дослідження зв'язків і способів використання даних, подій, станів даних, а також детального опису алгоритмів.

Вивчається потреба в заходах із контролю та захисту даних, їхньому резервному копіюванні та відновленні. Має бути проведений детальний аналіз наявних систем та інших чинників, що впливають на процес впровадження системи. Потрібно виявити всі обмеження і припущення, що можуть вплинути на подальше проектування, використання ресурсів і терміни проведення робіт.

Підхід. На цьому етапі аналітики й користувачі працюють пліч-о-пліч, встановлюючи й перевіряючи вимоги. Аналіз предметної області передбачає:

- проведення бесід з користувачами;
- перегляд усіх документів та бланків, які обробляються і формуються організацією;
- аналіз потоків документів;
- аналіз способів вирішення завдань організації;
- фіксація правил, обмежень та законів, що діють у предметній області.

Результати:

- узгоджена діаграма сутностей і зв’язків;
- відомості про обсяги даних, частоту виконання завдань, очікуваний користувачем рівень продуктивності;
- деталізовані й узгоджені описи завдань:
- первинний варіант стратегії впровадження;
- опис заходів з ревізії і контролю даних, резервного копіювання й відновлення;
- загальний опис процедур, що не автоматизуються;
- критерії прийнятності, якості, гнучкості та продуктивності;
- попереднє оцінювання обсягів системи;
- узгоджений підхід до здійснення етапу проектування й фази реалізації;
- уточнений план розроблення системи.

Ключові чинники успіху:

- активна участь користувачів;
- ретельна перевірка достовірності, повноти й несуперечності даних;

- виявлення всіх питань та припущень, що мають ключове значення для проектування і впровадження;
- встановлення точних характеристик ключових завдань і даних; жорсткий контроль за ходом робіт, концентрація зусиль на виконанні календарних планів і дотриманні запланованих термінів

Системний аналіз предметної області

Приклад. Розробити базу даних «Навчання студентів».
Рішення.
Студенти навчаються на одному з факультетів, очолюваному деканатом, у функції якого входить контроль за навчальним процесом. У навчальному процесі беруть участь викладачі кафедр, що адміністративно належать до одного з факультетів. Кожному факультету можуть належати кілька кафедр. Студенти кафедр організовані групи. Викладачі кафедр характеризуються прізвищем ім'ям та по батькові,посадою, науковим званням, ставкою та стажем роботи, адресою проживання, віком. Кожна кафедра читає певний набір закріплених за нею дисциплін. Кожна дисципліна характеризується своєю повною назвою, вказівкою загальної кількості годин та форми контролю (залік, іспит). Наприкінці кожного семестру складаються екзаменаційно-залікові відомості, в яких вказуються дисципліни та для яких груп проводиться форма контролю, прізвище викладача та навчальний рік та семестр. У кожній такій відомості складається список студентів та виставляється оцінка.

2. Команди групування (mongodb) та оператори вибірки

Число елементів у колекції

За допомогою функції count() можна отримати кількість елементів у колекції:

```
> db.users.count()
1
```

Можна групувати параметри пошуку та функцію count, щоб підрахувати скільки певних документів, наприклад, у яких name=Tom:

```
> db.users.find({name: "Tom"}).count()
1
>
```

Більш того, ми можемо створювати ланцюжки функцій, щоб конкретизувати умови підрахунку:

```
> db.users.find({name: "Tom"}).skip(2).count(true)
0
```

Тут слід зазначити, що за замовчуванням функція count не використовується з функціями limit та skip. Щоб їх використати, як у прикладі вище, у функцію count треба передати булеве значення true

Функція distinct

Колекція може мати документи, які містять однакові значення для одного або декількох полів. Наприклад, у кількох документах визначено name: "Tom". І нам треба знайти тільки унікальні значення для одного з полів документа. Для цього ми можемо скористатися функцією distinct:

```
>  db.users.distinct("name")
[ "Tom", "Bob", "Alice" ]
```

3. Оператори вибірки

Умовні оператори

Умовні оператори задають умову, якій має відповідати значення поля документа:

$eq (рівно)

$ne (не рівно)

$gt (більше ніж)

$lt (менше ніж)

$gte (більше чи рівно)

$lte (менше або рівно)

$in визначає масив значень, одне з яких повинно мати поле документа

$nin визначає масив значень, які не повинно мати поле документа

Наприклад, знайдемо всі документи, у яких значення ключа age менше 30:

```
> db.users.find ({age: {$lt : 30}})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b69"), "name" : "Bob", "age" : 26, "languages" : [ "english", "french" ] }
>
```

Аналогічно буде використання інших операторів порівняння. Наприклад, той самий ключ, тільки більше 30:

```
> db.users.find ({age: {$gt : 30}})
{ "_id" : ObjectId("620d00d3ab384398dc000b6a"), "name" : "Alice", "age" : 31, "languages" : [ "german", "english" ] }
>
```

Зверніть увагу, що порівняння тут проводиться над цілими типами, а не рядками. Якщо ключ age є строковими значеннями, то відповідно треба проводити порівняння над рядками: db.users.find ({age: {$gt : "30"}}), однак результат буде тим самим.

Але представимо ситуацію, коли нам треба знайти всі об'єкти зі значенням поля age більше 30, але менше 50. У цьому випадку ми можемо комбінувати два оператори:

```
> db.users.find ({age: {$gt : 30, $lt: 50}})
{ "_id" : ObjectId("620d00d3ab384398dc000b6a"), "name" : "Alice", "age" : 31, "languages" : [ "german", "english" ] }
>
```

Знайдемо користувачів, вік яких дорівнює 22:

```
> db.users.find ({age: {$eq : 22}})
>
```

Зворотня операція - знайдемо користувачів, вік яких не дорівнює 22:

```
>  db.users.find ({age: {$ne : 22}})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b69"), "name" : "Bob", "age" : 26, "languages" : [ "english", "french" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b6a"), "name" : "Alice", "age" : 31, "languages" : [ "german", "english" ] }
>
```

Оператор $in визначає масив можливих виразів і шукає ключі, значення яких є в масиві:

```
> db.users.find ({age: {$in : [22, 32]}})
>
```

Протилежним чином діє оператор $nin - він визначає масив можливих виразів і шукає ключі, значення яких відсутнє в цьому масиві:

```
> db.users.find ({age: {$nin : [22, 32]}})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b69"), "name" : "Bob", "age" : 26, "languages" : [ "english", "french" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b6a"), "name" : "Alice", "age" : 31, "languages" : [ "german", "english" ] }
>
```

Логічні оператори

Логічні оператори виконуються над умовами вибірки:

$or: з'єднує дві умови, і документ повинен відповідати одній з цих умов

$and: з'єднує дві умови, і документ повинен відповідати обом умовам

$not: документ повинен НЕ відповідати умові

$nor: з'єднує дві умови, і документ повинен НЕ відповідати обом умовам

Оператор $or

Оператор $or представляє логічну операцію АБО і визначає набір пар ключ-значення, які мають бути в документі. І якщо документ має хоч одну таку пару ключ-значення, він відповідає даному запиту і витягується з бд:

```
> db.users.find ({name: "Tom", $or : [{age: 22}, {languages: "german"}]})
>
```

Цей вираз поверне нам всі документи, у яких або name=Tom, або age=22.

Інший приклад поверне нам всі документи, у яких name=Tom, а age дорівнює 22, або серед значень languages є "german":

```
> db.users.find ({$or : [{name: "Tom"}, {age: 22}]})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
>
```

У підвиразах or можна застосовувати умовні оператори:

```
> db.users.find ({$or : [{name: "Tom"}, {age: {$gte:30}}]})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b6a"), "name" : "Alice", "age" : 31, "languages" : [ "german", "english" ] }
>
```

У цьому випадку ми вибираємо всі документи, де name="Tom" або поле age має значення 30 і вище.

Оператор $and

Оператор $and представляє логічну операцію І (логічне множення) і визначає набір критеріїв, яким обов'язково має відповідати документ. На відміну від оператора $or, документ повинен відповідати всім зазначеним критеріям. Наприклад:

```
> db.users.find ({$and : [{name: "Tom"}, {age: 32}]})
>
```

Тут документи, що вибираються, обов'язково повинні мати ім'я Tom і вік 32 - обидві ці ознаки.

Пошук по масивам

Ряд операторів призначений для роботи з масивами:

$all: визначає набір значень, які мають бути в масиві

$size: визначає кількість елементів, які мають бути в масиві

$elemMatch: визначає умову, якій повинні відповідати елементи в масиві

$all

Оператор $all визначає масив можливих виразів і вимагає, щоб документи мали весь набір виразів, що визначається. Відповідно він застосовується для пошуку за масивом. Наприклад, у документах є масив languages, що зберігає іноземні мови, якими говорить користувач. І щоб знайти всіх людей, які розмовляють одночасно і англійською, і французькою, ми можемо використовувати наступний вираз:

```
> db.users.find ({languages: {$all : ["english", "french"]}})
>
```

Оператор $ elemMatch

Оператор $elemMatch дозволяє вибрати документи, у яких масиви містять елементи, які під певні умови. Наприклад, нехай у базі даних буде колекція, яка містить оцінки користувачів за певними курсами. Додамо кілька документів:

```
> db.grades.insert ([{student: "Tom", courses:[{name: "Java", grade: 5}, {name: "MongoDB", grade: 4}]},
... {student: "Alice", courses:[{name: "C++", grade: 3}, {name: "MongoDB", grade: 5}]}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Кожен документ має масив курсів, який у свою чергу складається з вкладених документів.

Тепер знайдемо студентів, які для курсу MongoDB мають оцінку вище 3:

```
> db.grades.find({courses: {$elemMatch: {name: "MongoDB", grade: {$gt: 3}}}})
{ "_id" : ObjectId("620e6416203071e962aca317"), "student" : "Tom", "courses" : [ { "name" : "Java", "grade" : 5 },
me" : "MongoDB", "grade" : 4 } ] }
{ "_id" : ObjectId("620e6416203071e962aca318"), "student" : "Alice", "courses" : [ { "name" : "C++", "grade" : 3 },
ame" : "MongoDB", "grade" : 5 } ] }
```

Оператор $size

Оператор $size використовується для знаходження документів, у яких масиви мають число елементів, що дорівнює значенню $size. Наприклад, витягнемо всі документи, в яких у масиві laguages два елементи:

```
> db.users.find ({languages: {$size:2}})
{ "_id" : 123457, "name" : "Tom", "age" : 28, "languages" : [ "english", "spanish" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b69"), "name" : "Bob", "age" : 26, "languages" : [ "english", "french" ] }
{ "_id" : ObjectId("620d00d3ab384398dc000b6a"), "name" : "Alice", "age" : 31, "languages" : [ "german", "english" ] }
>
```

Оператор $exists

Оператор $exists дозволяє вилучити лише ті документи, в яких певний ключ присутній або відсутній. Наприклад, повернемо всі документи, в які є ключ компанії:

```
> db.users.find ({company: {$exists:true}})
>
```

Оператор $regex

Оператор $regex визначає регулярний вираз, якому має відповідати значення поля. Наприклад, нехай поле name обов'язково має букву "b":

```
>  db.users.find ({name: {$regex:"b"}})
{ "_id" : ObjectId("620d00d3ab384398dc000b69"), "name" : "Bob", "age" : 26, "languages" : [ "english", "french" ] }
>
```

Важливо розуміти, що $regex приймає не просто рядки, а саме регулярні вирази, наприклад: name: {$regex:"om$"} - значення name має закінчуватися на "om".

ЗАВДАННЯ

1. За своїм варіантом провести аналіз предметної області.
2. Написати запити (за своїм варіантом).

1	1. Порахувати всіх студентів на кафедрі. 2. Вивести всіх студентів, вік яких більше або дорівнює 20 рокам. 3. Вивести студента, якого звати Роман і якому 22 роки. 4. Самостійно скласти запит для виведення інформації із масивів.

2	1. Порахувати всі текстові документи у сховищі. 2. Вивести всі документи, розмір яких більше або дорівнює 158 КБ. 3. Вивести всі документи, які подані українською мовою, або менші ніж 300 КБ. 4. Самостійно скласти запит для виведення інформації із масивів.
3	1. Порахувати всі стандарти у сховищі. 2. Вивести всі стандарти по C++ після 1998 року. 3. Вивести всі стандарти по C# або по Java, прийняті раніше 2018 року. 4. Самостійно скласти запит для виведення інформації із масивів.
4	1. Порахувати всі фільми у сховищі. 2. Вивести всі фільми визначеного жанру 3. Вивести всі фільми декількох визначених жанрів 4. Самостійно скласти запит для виведення інформації із масивів.
5	1. Порахувати всі ігри у сховищі. 2. Вивести всі ігри визначеного жанру 3. Вивести всі ігри декількох визначених жанрів 4. Самостійно скласти запити для виведення інформації із масивів.
6	1. Порахувати всі ПЗ, інструкції до яких зберігаються у сховищі. 2. Вивести всі ПЗ для бухгалтерського обліку. 3. Вивести всі ПЗ для бухгалтерського обліку та 2020 року випуску. 4. Самостійно скласти запит для виведення інформації із масивів.
7	1. Порахувати всі проекти у сховищі, що написані мовою c#. 2. Вивести проекти, які додані останнього місяця або написані мовою c++.

	3. Вивести всі проекти, які використовують СУБД SQL Server або додані останнього року. 4. Самостійно скласти запити для виведення інформації із масивів.
8	1. Порахувати всі гаджети, інструкції до яких зберігаються у сховищі. 2. Вивести всі гаджети, які використовуються для спілкування в соціальних мережах. 3. Вивести всі гаджети, або 2020 або 2022 року випуску. 4. Самостійно скласти запит для виведення інформації із масивів.
9	1. Порахувати всі стандарти у сховищі. 2. Вивести всі стандарти по SQL після 1998 року. 3. Вивести всі стандарти по SQL або по mongodb, прийняті раніше 2018 року. 4. Самостійно скласти запит для виведення інформації із масивів
10	1. Порахувати всі відео у сховищі. 2. Вивести всі відео визначеного жанру 3. Вивести всі відео декількох визначених жанрів, які опубліковані за останній місяць та набрали більше 100 лайків. 4. Самостійно скласти запит для виведення інформації із масивів.