Самченко ІТІНФ-20-1 В-21

**1. Запити до елементів масиву. Наведіть приклади.**

Припустимо, є у нас колекція food і ми туди вставляємо документ з масивом фруктів.

> db.food.insert( { "fruit" : [ "apple", "banana", "peach" ] } );

То такий запит

> db.food.find({"fruit": "banana"});

успішно його знайде.

Якщо потрібно вибрати документи більше, ніж по одному елементу масиву, ми можемо використовувати оператор $all

> db.food.find( { fruits: { $all : [ "apple", "banana" ] } } );

Такий запит поверне всі документи, в масиві фруктів яких є і яблука, і банани.

Отримати документи щодо повного збігу елементів у масиві можемо так:

> db.food.find( { fruits: [ "apple", "banana", "peach"] } );

Є у нас блог, у ньому зберігаються коментарі. Ми хочемо отримати перші 10 коментарів. На допомогу нам приходить оператор $slice:

> db.blog.posts.findOne( { }, { "comments" : { "$slice" : 10 } } );

findOne - працює аналогічно find, але повертає перший збіг документ.

Якщо потрібно отримати останні 10 коментарів пишемо ось так:

> db.blog.posts.findOne( { }, { "comments" : { "$slice" : -10 } } );

Також $slice вміє отримувати документи із середини:

> db.blog.posts.findOne( { }, { comments : { "$slice" : [ 23, 10 ] } });

у цьому випадку буде пропущено 23 початкові елементи і повернуться елементи з 24 по 34, якщо це можливо

Уявімо таку ситуацію, що нам потрібно перевірити поле, яке у свою чергу є масивом, на входження до нього елемента. Це досить просто, mongoDB сама зрозуміє, що ми робимо запит до масиву і знайде в ньому входження:

> db.users.find({ comments: { name: 'Rock' } })

Знайде у нашій базі поле comments у якому може бути масив.

Якщо нам потрібно порівняти лише один певний елемент масиву, ми можемо скористатися наступним запитом:

> db.users.find({ 'comments.2': { name: 'John' } })

mongoDB знатиме, що ми шукаємо в масиві і вибере за нас другий елемент і порівняє його з переданим об'єктом. Вкладеність можна продовжувати скільки завгодно.

> db.users.find({ 'comments.2.name': 'John' })

Також можна шукати за кількістю елементів у масиві за допомогою оператора $size

> db.users.find({ comments: { $size: 2} })

**2. Акумулятори: $sum, $avg, $max, $min, $first та $last.**

$sum - Повертає суму всіх чисельних полів.

У наступному прикладі обчислюється загальна кількість проданої кави в колекції продажів:

db.sales.aggregate([

{

$group: {

\_id: null,

totalQty: { $sum: '$quantity' },

},

},

]);

$avg — Розраховує середнє значення між числовими полями.

У наступному прикладі документи згруповано за полем елемента та використовується $avg для обчислення середньої кількості для кожної групи:

db.sales.aggregate([

{

$group: {

\_id: '$item',

averageQty: { $avg: '$quantity' },

},

},

]);

$min - отримає мінімальне значення з числових полів

У наступному прикладі $min використовується для визначення мінімальної кількості з усіх документів:

db.sales.aggregate([

{

$group: {

\_id: null,

maxQty: { $min: '$quantity' },

},

},

{

$project: {

\_id: 0,

},

},

]);

$max — Отримати максимальне значення з числових полів

У наступному прикладі $max використовується для визначення максимальної кількості з усіх документів:

db.sales.aggregate([

{

$group: {

\_id: null,

maxQty: { $max: '$quantity' },

},

},

{

$project: {

\_id: 0,

},

},

]);

$first — Отримує лише перший документ із згрупованих, зазвичай використовується із сортуванням.

Групуючи документи за полем товару, наступна операція використовує накопичувач $first для обчислення дати першого продажу для кожного товару:

db.sales.aggregate(

[

{ $sort: { item: 1, date: 1 } },

{

$group:

{

\_id: "$item",

firstSale: { $first: "$date" }

}

}

]

)

$last — Отримує останній документ

Наступна операція спочатку сортує документи за елементом і датою, а потім на наступному етапі $group групує відсортовані документи за полем елемента та використовує накопичувач $last для обчислення останньої дати продажу для кожного товару:

db.sales.aggregate(

[

{ $sort: { item: 1, date: 1 } },

{

$group:

{

\_id: "$item",

lastSalesDate: { $last: "$date" }

}

}

]

)