Banco de dados II

Queries complexas

No MySQL existem operadores e funções que nos permitem criar consultas mais específicas, como SUM, MAX, AVG, LIKE e BETWEEN.

No MongoDB também utilizamos operadores e funções, que permitem a realização de consultas complexas em seus dados. As boas práticas orientam incluir, em uma única entidade, todos ou quase todos os dados que necessitamos. Assim conseguimos, na maioria das vezes, utilizar uma única consulta por _id, que é a maneira mais eficiente de se recuperar dados, reduzindo o número de requisições e garantindo a performance.

Operadores e funções desempenham um papel fundamental nesse contexto. Você encontra mais informações no site oficial, clicando aqui.

Confira abaixo alguns exemplos.

Funções de agregação

$group

A função $group é equivalente ao group by do MySQL.

Neste exemplo, utilizamos a agregação para trazer a quantidade de filmes por gênero:

```
db.filmes.aggregate([{$group: {_id: "$genero",Total: {$sum:
1}}}])
{ _id: 'Drama', Total: 3 }
{ _id: 'Suspense', Total: 1 }
```

```
{ _id: 'Infantil', Total: 2 }
{ _id: 'Aventura', Total: 3 }
{ _id: 'Animação', Total: 4 }
{ _id: 'Ficção Científica', Total: 5 }
{ _id: 'Comédia', Total: 1 }
```

Mas, com o MongoDB, podemos fazer mais do que isso.

Que tal criarmos uma consulta que retorne o gênero, a quantidade de filmes e o nome dos filmes?

Vamos utilizar a função agregadora $group e adicionar mais um ingrediente nesta fórmula: $push.

Observe a consulta abaixo:

```
db.filmes.aggregate([ {$group:{ _id:"$genero", total : {$sum:1},
acervo: {$push: "$titulo"} } }])
{ _id: 'Drama',
  total: 3,
  acervo: [ 'Titanic', 'A vida é bela', 'I am Sam' ] }
{ _id: 'Infantil',
  total: 2,
  acervo: [ 'O Rei Leão', 'Hotel Transylvania' ] }
{ _id: 'Ficção Científica',
  total: 5,
  acervo:
   [ 'Avatar',
     'Star Wars: Episodio VI',
     'Star Wars: Episodio VII',
     'Jurassic Park',
     'Transformers: o lado escuro da Lua' ] }
{ _id: 'Animação',
  total: 4,
  acervo: [ 'Procurando Nemo', 'Toy Story', 'Toy Story 2',
'Divertidamente' ] }
{ _id: 'Aventura',
  total: 3,
```

```
  acervo:
   [ 'Harry Potter e a Pedra Filosofal',
     'Harry Potter e a Câmara Secreta',
     'Big' ] }
{ _id: 'Comédia', total: 1, acervo: [ 'Meu pobre anjinho' ] }
{ _id: 'Suspense',
  total: 1,
  acervo: [ 'Harry Potter e as Relíquias da Morte - Parte 2' ] }
```

O operador $push agregou uma matriz de todos os valores resultantes do campo título à nossa consulta. E o resultado foi surpreendente.

Os operadores de atualização permitem modificar ou adicionar os dados em seu banco de dados. São eles:

$addToSet - Adiciona elementos distintos a uma matriz.

$pop - Remove o primeiro ou o último item de uma matriz.

$pull - Remove todos os elementos da matriz que correspondem a uma consulta especificada.

$push - Adiciona um item a uma matriz.

$pullAll - Remove todos os valores correspondentes de uma matriz.

Função $lookup

Esta função é equivalente ao left join do MySQL. Executa uma associação externa esquerda a outra coleção no mesmo banco de dados. Para cada documento de entrada, o estágio $lookup adiciona um novo campo de matriz cujos elementos são os documentos correspondentes da coleção "juntada". O estágio $lookup passa esses documentos reformulados para o próximo estágio.

Exemplo:

```
db.atorFilme.aggregate([
    {
        '$lookup': {
            'from': 'filmes',  // coleção externa
            'localField': 'filme',  // nome do campo atual a comparar com o campo da coleção externa
            'foreignField': 'titulo',  // nome do campo da coleção externa
            'as': 'genero' // nome do novo campo que vai receber os valores pesquisados.
             }
         }
    ]
)
```

O resultado é:

```
{ _id: 1,
  nome: 'Sam',
  sobrenome: 'Worthington',
  filme: 'Avatar',
```

Dados da coleção **atorFilme** (coleção atual)

```
genero:

 [ { _id: 1,
     titulo: 'Avatar',
     avaliacao: 7.9,
     premios: 3,
    data_lancamento:
2010-04-10T03:00:00.000Z,
     duracao: 120,
     genero: 'Ficção
Científica' } ] }
```

o campo **genero** foi o nome dado para o novo campo que receberá o resultado da pesquisa.

Observe que o resultado da pesquisa retorna todos os dados da coleção externa.

Logo, **genero** recebe como array todos os dados da coleção **filmes**

Agora, para capturar apenas o campo gênero do array, vamos utilizar a função **$unwind.**

Função $unwind

Esta função desconstrói um campo de matriz (array) e o transforma em objeto, permitindo a captura do campo da tabela externa.

Exemplo:

```
{$unwind: {
 path: '$genero'
}
```

o campo **gênero**, neste caso, é o novo campo que criamos para receber o resultado da consulta à coleção externa, como um array.

O **$unwind** destrói esse array e o transforma em objeto, facilitando, assim, a captura do campo desejado.

```
genero: {
 $addToSet: '$genero.genero'
}
```

E, finalmente, criamos um novo campo **genero**, recuperamos o **gênero** da coleção **filmes** e finalizamos nossa consulta.

```
{$group: {
 _id: '$filme',
 atores: {
  $sum: 1
 },
 elenco: {
  $push: {
   $concat: [
    '$nome',
    ' ',
    '$sobrenome'
   ]
  }
 },

}
```

Neste novo estágio, agrupamos nossa consulta por filmes e trazemos o título do filme, a quantidade de atores e listamos todos os atores, concatenando o nome e o sobrenome.

```
{ _id: 'Avatar',
  atores: 3,
 elenco: [ 'Sam Worthington', 'Zoe Saldana', 'Sigourney Weaver' ],
genero: [ 'Ficção Científica' ] }
```

// título do filme;
// quantidade de atores do filme;
// os nomes dos atores

// e o gênero que buscamos da coleção filmes.

Não é extraordinário? Basicamente fizemos um left join e recuperamos os dados, utilizando funções e operadores.

Exists

O operador $exists verifica se na coleção existe ou não determinado campo.

No exemplo abaixo, verificamos se o campo genero existe. Se existir, pedimos que liste os filmes que não pertençam aos gêneros citados.

```
db.filmes.find( { genero: { $exists: true, $nin: [ "Ficção Científica", "Suspense", "Drama", "Aventura", "Animação" ] } } )
{ _id: 10,
  titulo: 'O Rei Leão',
  avaliacao: 9.1,
  premios: 3,
  data_lancamento: 2000-04-02T03:00:00.000Z,
  genero: 'Infantil' }
```

```
{ _id: 15,
  titulo: 'Meu pobre anjinho',
  avaliacao: 3.2,
  premios: 1,
  data_lancamento: 1989-04-01T03:00:00.000Z,
  duracao: 120,
  genero: 'Comédia' }
{ _id: 19,
  titulo: 'Hotel Transylvania',
  avaliacao: 7.1,
  premios: 1,
  data_lancamento: 2012-04-05T03:00:00.000Z,
  duracao: 90,
  genero: 'Infantil' }
```

Like

Para listarmos os filmes cujos gêneros contenham %ra%, em MongoDB substituímos o % por / /

No exemplo abaixo, listamos todos os filmes cujo gênero comece com a sequência In:

```
db.filmes.find({genero: /^In/})
{ _id: 10,
  titulo: 'O Rei Leão',
  avaliacao: 9.1,
  premios: 3,
  data_lancamento: 2000-04-02T03:00:00.000Z,
```

```
  genero: 'Infantil' }
{ _id: 19,
  titulo: 'Hotel Transylvania',
  avaliacao: 7.1,
  premios: 1,
  data_lancamento: 2012-04-05T03:00:00.000Z,
  duracao: 90,
  genero: 'Infantil' }
```

Como resultado, foi listado apenas o gênero “Infantil”,
db.filmes.find({genero: /^In/}) é equivalente a:
SELECT * FROM filmes WHERE genero LIKE ‘In%’

Se quisermos listar todos os gêneros que possuem em algum lugar da palavra a sílaba “ra”, usamos: db.filmes.find({"genero": /ra/}).
Esta consulta é equivalente a ‘ %ra%’.

Vamos, agora, conhecer alguns operadores:

Operadores de Comparação

Nome	Descrição
`$eq`	Corresponde a valores que são iguais a um valor especificado.
`$gt`	Corresponde a valores maiores que um valor especificado.
`$gte`	Corresponde a valores maiores ou iguais a um valor especificado.

`$in`	Corresponde a qualquer um dos valores especificados em uma matriz.
`$lt`	Corresponde a valores menores que um valor especificado.
`$lte`	Corresponde a valores menores ou iguais a um valor especificado.
`$ne`	Corresponde a todos os valores que não são iguais a um valor especificado.
`$nin`	Não corresponde a nenhum dos valores especificados em uma matriz.

Operadores Lógicos

Nome	Descrição
`$and`	Retorna todos os documentos que correspondem às condições de ambas as cláusulas.
`$not`	Retorna documentos que *não* correspondem à expressão de consulta.
`$nor`	Junta cláusulas de consulta e lógica `e` retorna todos os documentos que não correspondem a ambas as cláusulas.
`$or`	Une cláusulas de consulta e lógica `e` retorna todos os documentos que correspondem às condições de qualquer uma das cláusulas.