MongoDB

Banco de dados de documentos

Um registro no MongoDB é um documento, que é uma estrutura de dados composta por pares de campos e valores. Os documentos do MongoDB são semelhantes aos objetos JSON. Os valores dos campos podem incluir outros documentos, arrays e arrays de documentos.

As vantagens de usar documentos são:

- Os documentos correspondem a tipos de dados nativos em muitas linguagens de programação.
- Documentos e matrizes incorporados reduzem a necessidade de junções caras.
- Esquema dinâmico suporta polimorfismo fluente.

Coleções/Visualizações/Visualizações materializadas sob demanda

O MongoDB armazena documentos em coleções. As coleções são análogas às tabelas em bancos de dados relacionais.

Além das coleções, o MongoDB suporta:

- Exibições somente leitura (a partir do MongoDB 3.4).
- Exibições materializadas sob demanda (a partir do MongoDB 4.2).

Comparativo de Banco relacional x MongoDB:
https://www.mongodb.com/docs/manual/reference/sql-comparison/

Características principais

Alta performance

O MongoDB fornece persistência de dados de alto desempenho. Em particular,

- O suporte para modelos de dados incorporados reduz a atividade de E/S no sistema de banco de dados.
- Os índices oferecem suporte a consultas mais rápidas e podem incluir chaves de matrizes e documentos incorporados.

API de consulta

A API de consulta do MongoDB suporta operações de leitura e gravação (CRUD), bem como:

- Agregação de dados.
- Pesquisa de Texto e Consultas Geoespaciais.

Veja também:

- Gráfico de mapeamento SQL para MongoDB.
- Gráfico de mapeamento de SQL para agregação.
- Saiba mais sobre os recursos de linguagem de consulta mais recentes com o Linguagem de consulta do MongoDB: o que há de novo apresentação do **MongoDB.live 2020**.

Alta disponibilidade

O recurso de replicação do MongoDB, chamado replica set, fornece:

- failover *automático.*
- redundância de dados.

Um conjunto de réplicas é um grupo de servidores MongoDB que mantém o mesmo conjunto de dados, fornecendo redundância e aumentando a disponibilidade de dados.

Escalabilidade horizontal

O MongoDB fornece escalabilidade horizontal como parte de sua funcionalidade *principal* :

- A fragmentação distribui dados em um cluster de máquinas.
- A partir da versão 3.4, o MongoDB dá suporte à criação de zonas de dados com base na chave de fragmentação. Em um cluster balanceado, o MongoDB direciona leituras e gravações cobertas por uma zona apenas para os shards dentro da zona. Consulte a página do manual Zonas para obter mais informações.

Suporte para vários mecanismos de armazenamento

O MongoDB suporta vários mecanismos de armazenamento:

- Mecanismo de armazenamento WiredTiger (incluindo suporte para criptografia em repouso)
- Mecanismo de armazenamento na memória.

Além disso, o MongoDB fornece uma API de mecanismo de armazenamento conectável que permite que terceiros desenvolvam mecanismos de armazenamento para o MongoDB.

Alternar banco de dados

Dentro do shell, `db` refere-se ao seu banco de dados atual. Digite `db` para exibir o banco de dados atual.

Para alternar bancos de dados ou criar se estiver inexistente, mas um banco para realmente ser criado ele precisa ter dados (coleções), digite `use <db>`. Por exemplo, para alternar para o banco de dados `examples`.

Você não precisa criar o banco de dados antes de alternar. O MongoDB cria o banco de dados quando você armazena dados nesse banco de dados pela primeira vez (como criar a primeira coleção no banco de dados).

Preencher uma coleção (inserir)

O MongoDB armazena documentos em coleções . As coleções são análogas às tabelas em bancos de dados relacionais. Se uma coleção não existir, o MongoDB criará a coleção quando você armazenar os dados dessa coleção pela primeira vez.

O exemplo a seguir usa o `db.collection.insertMany()` método para inserir novos documentos na coleção de movies.

```
db.movies.insertMany([
   {
      title: 'Titanic',
      year: 1997,
      genres: [ 'Drama', 'Romance' ],
      rated: 'PG-13',
      languages: [ 'English', 'French', 'German', 'Swedish', 'Italian', 'Russian' ],
      released: ISODate("1997-12-19T00:00:00.000Z"),
      awards: {
         wins: 127,
         nominations: 63,
         text: 'Won 11 Oscars. Another 116 wins & 63 nominations.'
      },
      cast: [ 'Leonardo DiCaprio', 'Kate Winslet', 'Billy Zane', 'Kathy Bates' ],
      directors: [ 'James Cameron' ]
   },
   {
      title: 'The Dark Knight',
      year: 2008,
      genres: [ 'Action', 'Crime', 'Drama' ],
      rated: 'PG-13',
      languages: [ 'English', 'Mandarin' ],
      released: ISODate("2008-07-18T00:00:00.000Z"),
      awards: {
         wins: 144,
         nominations: 106,
         text: 'Won 2 Oscars. Another 142 wins & 106 nominations.'
      },
      cast: [ 'Christian Bale', 'Heath Ledger', 'Aaron Eckhart', 'Michael Caine' ],
      directors: [ 'Christopher Nolan' ]
   }
```

A operação retorna um documento que contém o indicador de confirmação e um array que contém o `_id` de cada documento inserido com sucesso.

Restrições:

> ## Diferenciação de maiúsculas e minúsculas do nome do banco de dados

Os nomes de banco de dados diferenciam maiúsculas de minúsculas no MongoDB. Por exemplo: se o banco de dados RubensDB já existir, o MongoDB retornará um erro se você tentar criar um banco de dados chamado RubensDB.

> ## Restrições sobre nomes de banco de dados para Windows

Para implantações do MongoDB em sistemas operacionais Windows, os nomes de banco de dados não podem conter nenhum dos seguintes caracteres:

- /."$*<>:|?

> ## Restrições sobre nomes de banco de dados para sistemas Unix e Linux

Para implantações do MongoDB em em sistemas operacionais Unix e Linux, os nomes de banco de dados não podem conter nenhum dos seguintes caracteres:

- /."$

Além disso, os nomes de banco de dados não podem conter o caractere nulo.

> ## Comprimento dos nomes de banco de dados

Os nomes de banco de dados não podem estar vazios e devem ter menos de 64 caracteres.

Assim como para criar um banco de dados, também existem restrições para se criar uma coleção aqui no MongoDB, que são:

- Os nomes das coleções devem começar com um sublinhado ou um caractere de letra.
- Não podem:

- Conter o $.
- Ser uma string vazia (por exemplo "",).
- Conter o caractere nulo.
- Começar com o system.prefixo. (Reservado para uso interno).

> Assim, como para criar um banco de dados e coleções, também existem restrições ao se criar um documento:

- O nome do campo `_id` é reservado para uso como chave primária. Seu valor deve ser único na coleção, é imutável e pode ser de qualquer tipo que não seja um array.
- Os nomes dos campos não podem conter o caractere **NULL**.

Os documentos BSON, também possuem restrições de tamanho:

- O tamanho máximo de um documento BSON é 16 megabytes.

Selecione todos os documentos

Para selecionar os documentos de uma coleção, você pode usar o método `db.collection.find()`. Para selecionar todos os documentos da coleção, passe um documento vazio como documento de filtro de consulta para o método.

Exemplo de consulta para retornar todos os documentos da coleção de movies:

```
db.movies.find( { } )
```

Criação Explícita

O MongoDB fornece o `db.createCollection()` método para criar explicitamente uma coleção com várias opções, como definir o tamanho máximo ou as regras de validação da documentação. Se você não estiver especificando essas opções, não precisará criar explicitamente a coleção, pois o MongoDB cria novas coleções quando você armazena dados para as coleções pela primeira vez.

Para modificar essas opções de coleta, consulte `collMod`.

Validação de documento

Por padrão, uma coleção não exige que seus documentos tenham o mesmo esquema; ou seja, os documentos em uma única coleção não precisam ter o mesmo conjunto de campos e o tipo de dados de um campo pode diferir entre os documentos de uma coleção.

A partir do MongoDB 3.2, no entanto, você pode impor regras de validação de documento para uma coleção durante as operações de atualização e inserção. Consulte Validação de esquema para obter detalhes.

Identificadores exclusivos

As coleções recebem um UUID imutável . O UUID da coleção permanece o mesmo em todos os membros de um conjunto de réplicas e fragmentos em um cluster fragmentado.

Para recuperar o UUID de uma coleção, execute o listaColeçõescomando ou o `db.getCollectionInfos()` método.

Filtrar dados com operadores de comparação

Para uma correspondência de igualdade (`<field>` equals `<value>`), especifique `<field>: <value>` no documento do filtro de consulta e passe para o `db.collection.find()` método.

No shell, execute a seguinte consulta para encontrar filmes dirigidos por Christopher Nolan:

```
db.movies.find( { "directors": "Christopher Nolan" } );
```

Você pode usar operadores de comparação para realizar consultas mais avançadas:

Execute a seguinte consulta para retornar filmes lançados antes do ano `2000` :

```
db.movies.find( { "released": { $lt: ISODate("2000-01-01") } } );
```

Execute a seguinte consulta para retornar filmes que ganharam mais de `100` prêmios:

```
db.movies.find( { "awards.wins": { $gt: 100 } } );
```

Execute a seguinte consulta para retornar filmes em que a matriz de `languages` (idiomas) contém `japonês` ou `mandarim`:

```
db.movies.find( { "languages": { $in: [ "Japanese", "Mandarin" ] } } )
```

Estrutura básica do FIND:

```
db.collection.find(query, projection)
```

> **QUERY:** especificamos a condição que os nossos documentos devem atender para serem retornados na nossa consulta.

> **PROJECTION:** especificamos quais campos devem ou não ser retornados na nossa consulta.

Query Selectors:

Comparison:

> Para comparação de diferentes valores de tipo BSON, consulte a ordem de comparação BSON especificada.

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.

Nome	Descrição
`$in`	Corresponde a qualquer um dos valores especificados em uma matriz.
`$lt`	Corresponde a valores que são 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.

LOGICAL:

Nome	Descrição
`$and`	Junta cláusulas de consulta com um `AND` lógico retorna todos os documentos que correspondem às condições de ambas as cláusulas.
`$not`	Inverte o efeito de uma expressão de consulta e retorna documentos que não correspondem à expressão de consulta.
`$nor`	Junta cláusulas de consulta com um `NOR` retorno lógico de todos os documentos que não correspondem a ambas as cláusulas.
`$or`	Junta cláusulas de consulta com um `OR` retorno lógico de todos os documentos que correspondem às condições de qualquer uma das cláusulas.

> Para obter detalhes sobre um operador específico, incluindo sintaxe e exemplos, clique no link de cada operador para a página de referência do operador.

💡 Para saber mais sobre outros seletores de consulta clique no link: https://www.mongodb.com/docs/manual/reference/operator/query/#std-label-query-projection-operators-top

Especificar campos para retornar (projeção)

Para especificar os campos a serem retornados, passe um documento de projeção para o `db.collection.find(<query document>, <projection document>)` método. No documento de projeção, especifique:

- `<field>: 1` incluir um campo nos documentos devolvidos
- `<field>: 0` para excluir um campo nos documentos retornados

No shell, execute a seguinte consulta para retornar os campos `id`, `title`, `directors` e `year` de todos os documentos da `movies` coleção:

```
db.movies.find( { }, { "title": 1, "directors": 1, "year": 1 } );
```

Você não precisa especificar o `_id` campo para retornar o campo. Ele retorna por padrão. Para excluir o campo, defina-o no `0` documento de projeção. Por exemplo, execute a seguinte consulta para retornar apenas o `title` e os `genres` campos nos documentos correspondentes:

```
db.movies.find( { }, { "_id": 0, "title": 1, "genres": 1 } );
```

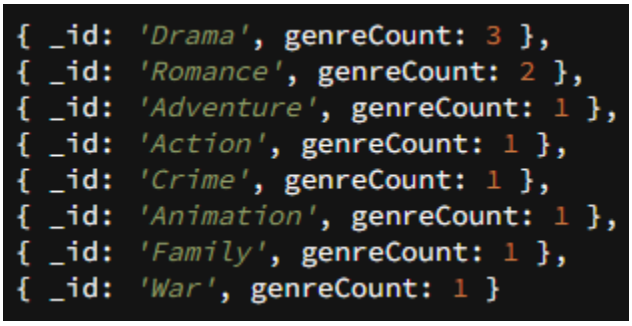
Dados agregados (`$group`)

Você pode usar a agregação para agrupar valores de vários documentos e retornar um único resultado. A agregação no MongoDB é realizada com um pipeline de agregação.

Embora `find()` as operações sejam úteis para recuperação de dados, o pipeline de agregação permite manipular dados, realizar cálculos e escrever consultas mais expressivas do que operações CRUD simples.

No shell, execute o seguinte pipeline de agregação para contar o número de ocorrências de cada `genre` valor:

```
db.movies.aggregate( [
   { $unwind: "$genres" },
   {
     $group: {
       _id: "$genres",
       genreCount: { $count: { } }
     }
   },
   { $sort: { "genreCount": -1 } }
] )
```



O pipeline usa:

- `$unwind` para produzir um documento para cada elemento na `genres` matriz.
- `$group` e o `$count` acumulador para contar o número de ocorrências de cada um `genre`. Este valor é armazenado no `genreCount` campo.
- `$sort` para classificar os documentos resultantes pelo `genreCount` campo em ordem decrescente.

💡 Clique nos comandos para saber mais dos mesmos.

UPDATE:

O MongoDB disponibiliza dois métodos para a atualização de documentos por linha de comando. Um deles é o **updateOne**, que atualiza um único documento por vez. O outro método é o **updateMany**, que atualiza vários documentos ao mesmo tempo.

```
db.users.updateMany(                 ← collection
  { age:  { $lt: 18 } },             ← update filter
    { $set:  { status: "reject" } }  ← update action
)
```

Além dos métodos **UpdateOne** e o **UpdateMany**, o MongoDB também disponibiliza o método **ReplaceOne**, que é utilizado para substituir um único documento na coleção com base no filtro.

O **ReplaceOne** possui a seguinte estrutura:

```
db.collection.replaceOne(
   <filter>,
   <replacement> )
```