



{ “curso”: “MBA Ciência de Dados”, “tipo”: “DB”, “autor”: “Edson
Guimarães”, “instituicao”: “Unip - Bauru” }

Agenda

- 1 MongoDB - Overview
- 2 Conceitos iniciais: execução, coleções, documentos e comandos básicos;
- 3 Relacionamentos: conversão de um MER para um modelo de documentos;
- 4 Métodos avançados de recuperação de informações no MongoDB.

Agenda

1

MongoDB - Overview;

2

Conceitos iniciais: execução, coleções, documentos e comandos básicos;

3

Relacionamentos: conversão de um MER para um modelo de documentos;

4

Métodos avançados de recuperação de informações no MongoDB.


Introdução: O que é o MongoDB?

- ◆ O MongoDB é um **SGBD NOSQL** open-source e orientado a documentos.
- ◆ Alguns de seus diferenciais são:
 - ◆ Alto desempenho: documentos **embutidos** e **índices** atuando sobre eles;
 - ◆ Rica linguagem de consulta: permite operações **CRUD**, **agregações** de dados, busca por **texto** e consultas **geoespaciais**;
 - ◆ Alta disponibilidade: *replica set*;
 - ◆ Escalabilidade horizontal: **sharding**.

O MongoDB é realmente utilizado?

- ◆ Com a popularidade e a consolidação da linguagem SQL no mercado, este tipo de questionamento é comum.
- ◆ DB-ENGINES RANKING: ranking de popularidade dos SGBD mais utilizados, atualizado mensalmente.
 - ◆ Pode ser acessado em: <https://db-engines.com/en/ranking>;
 - ◆ Considera uma série de critérios para obter uma pontuação capaz de classificar os SGBD.

O Ranking do DB-ENGINES

Rank			DBMS	Database Model	Score		
Aug 2018	Jul 2018	Aug 2017			Aug 2018	Jul 2018	Aug 2017
1.	1.	1.	Oracle +	Relational DBMS	1312.02	+34.24	-55.85
2.	2.	2.	MySQL +	Relational DBMS	1206.81	+10.74	-133.49
3.	3.	3.	Microsoft SQL Server +	Relational DBMS	1072.65	+19.24	-152.82
4.	4.	4.	PostgreSQL +	Relational DBMS	417.50	+11.69	+47.74
5.	5.	5.	MongoDB + 	Document store	350.98	+0.65	+20.48
6.	6.	6.	DB2 +	Relational DBMS	181.84	-4.36	-15.62
7.	7.	↑ 9.	Redis +	Key-value store	138.58	-1.34	+16.68
8.	8.	↑ 10.	Elasticsearch +	Search engine	138.12	+1.90	+20.47
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	129.10	-3.48	+2.07
10.	10.	↓ 8.	Cassandra +	Wide column store	119.58	-1.48	-7.14

Agenda

- 1 MongoDB - Overview;
- 2 Conceitos iniciais: execução, coleções, documentos e comandos básicos;
- 3 Relacionamentos: conversão de um MER para um modelo de documentos;
- 4 Métodos avançados de recuperação de informações no MongoDB.

Executando o MongoDB

- ◆ O MongoDB possui versões para as plataformas **MacOS**, **Linux** e **Windows**.
 - ◆ Maiores detalhes sobre os procedimentos necessários para realizar a instalação deste SGBD podem ser encontrados em:
<https://docs.mongodb.com/manual/administration/install-community/>.
- ◆ Após instalado, é necessário rodar o servidor do SGBD, localizado no executável **mongod**.
 - ◆ Com o servidor rodando, o shell do MongoDB é acessado através do executável **mongo**.
 - ◆ No MacOS ou no Linux, basta digitar o nome dos executáveis no terminal.

Criando um Banco de Dados

- ◆ O MongoDB **abstrai** diversos comandos **DDL**.
 - ◆ Estruturas são criadas conforme estas se tornam necessárias.
- ◆ Para criar um banco de dados, basta você usar o comando para **acessar um banco que ainda não existe**.
 - ◆ Assim que um registro for inserido neste banco, ele será **criado e persistido automaticamente**.

SQL

```
JP=# CREATE DATABASE db_aula;
```

SQL

```
> use db_aula
```

Coleções e Documentos (Insert)

- ◆ Como em outros modelos orientado a documentos, o MongoDB organiza os dados em **coleções** de **documentos**.
 - ◆ Cada documento possui um atributo identificador (`_id`) e uma **quantidade** **qualquer** de outros atributos.
 - ◆ Não é necessário (Mas é possível) especificar o ID dos documentos!
 - ◆ Não é necessário especificar o tipo dos atributos!
 - ◆ Documentos diferentes que fazem parte de uma mesma coleção podem ter atributos diferentes!

Coleções e Documentos (Insert)

- ◆ Para criar uma coleção, basta inserir um documento nela.
- ◆ Existem duas operações de inserção no MongoDB:
 - ◆ Inserção de um único documento: **insertOne**.
 - ◆ Recebe como parâmetro um único documento.
 - ◆ Inserção de múltiplos documentos de uma só vez: **insertMany**.
 - ◆ Recebe como parâmetro um vetor de documentos.

Coleções e Documentos (Insert)

SQL →

```
JP — psql -p5432
db_aula=# CREATE TABLE timesfutebol (
id int not null,
nome varchar(30) not null,
pais varchar(30) not null
);
CREATE TABLE
db_aula=# INSERT INTO timesfutebol VALUES
(1, 'Cruzeiro', 'Brasil'),
(2, 'Barcelona', 'Espanha');
INSERT 0 2
```

Coleções e Documentos (Insert)


```
JP — mongo — 80x24
> db.timesfutebol.insertOne({"_id": 1, "nome": "Cruzeiro", "pais": "Brasil"})
{ "acknowledged" : true, "insertedId" : 1 }
> db.timesfutebol.insertMany([{"nome": "Barcelona", "pais": "Espanha"}, {"nome": "Palmeiras", "mundial": 0}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5911b28009702042baa255e2"),
    ObjectId("5911b28009702042baa255e3")
  ]
}
```

The logo consists of a red square with a white upward-pointing arrow inside it. Below the arrow, the word "Mongo" is written in white text.

Mongo

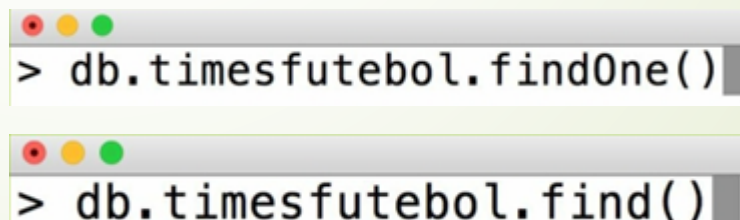
Coleções e Documentos (Select)

- ◆ O MongoDB possui dois métodos principais para retornar informações de documentos.
 - ◆ O método **find()** retorna um ponteiro para todos os documentos que atendem aos critérios especificados.
 - ◆ O método **findOne()** retorna um único documento que atende aos critérios especificados.
 - ◆ Caso exista mais de um documento atendendo aos critérios, o método **findOne()** retorna apenas o primeiro.



```
db_aula=# SELECT * FROM timesfutebol;
```

SQL



```
> db.timesfutebol.findOne()  
  
> db.timesfutebol.find()
```

MONGO

Coleções e Documentos (Select)

- ◆ Ambos os métodos `find()` e `findOne()` permitem especificar, da mesma forma, critérios de **seleção** e **projeção** para o resultado.

```
JP — psql -p5432 — 80x24  
db_aula=# SELECT nome, pais FROM timesfutebol WHERE nome = 'Cruzeiro';
```

SQL

Projeção

Seleção

```
JP — mongo — 80x24  
> db.timesfutebol.find({"nome": "Cruzeiro"}, {"_id": 0, "nome": 1, "pais": 1})
```

MONGO

Seleção

Projeção

Coleções e Documentos (Update)

- ◆ O MongoDB possui três métodos para atualização de dados em um documento.
 - ◆ Os métodos `updateOne()` e `updateMany()` localizam o documento segundo os critérios especificados e fazem as alterações descritas.
 - ◆ Diferença: **quantidade** de documentos afetada.
 - ◆ Enquanto o `updateOne()` afeta somente um documento que atenda os critérios, o `updateMany()` afeta todos.
 - ◆ O método `replaceOne()` localiza um único documento que atenda aos critérios especificados e o substitui por um novo documento.
 - ◆ O atributo `_id` do documento permanece o mesmo.

Coleções e Documentos (Update)

SQL

```
db_aula=# UPDATE timesfutebol  
SET nome = 'Atlético MG', pais = 'Brasil'  
WHERE id = 2;
```

SELEÇÃO

PROJEÇÃO

Coleções e Documentos (Update)

```
JP — mongo — 80x24  
> db.timesfutebol.updateOne({"_id": ObjectId("5911b28009702042baa255e2")}, {$set: {"nome": "Atlético MG", "pais": "Brasil"}})
```

PROJEÇÃO

SELEÇÃO

```
JP — mongo — 80x24  
> db.timesfutebol.updateMany({"pais": "Brasil"}, {$unset: {"pais": "Brasil"}})
```

Seleção

Alteração

```
JP — mongo — 80x24  
> db.timesfutebol.replaceOne({"mundial": 0}, {"nome": "Palmeiras"})
```

MONGO

Seleção

Alteração

Coleções e Documentos (Delete)

- ◆ O MongoDB possui dois métodos para a remoção de documentos.
 - ◆ Os métodos `deleteOne()` e `deleteMany()` localizam o documento segundo os critérios especificados e removem da base de dados.
 - ◆ Diferença: quantidade de documentos afetada.
 - ◆ Enquanto o `deleteOne()` afeta somente um documento que atenda os critérios, o `deleteMany()` afeta todos.

Coleções e Documentos (Delete)

SQL

```
db_aula=# DELETE FROM timesfutebol WHERE nome = "Palmeiras";
```

Seleção

MONGO

```
> db.timesfutebol.deleteOne({"nome": "Palmeiras"})  
{ "acknowledged" : true, "deletedCount" : 1 }  
> db.timesfutebol.deleteMany({})  
{ "acknowledged" : true, "deletedCount" : 2 }
```

SELEÇÃO

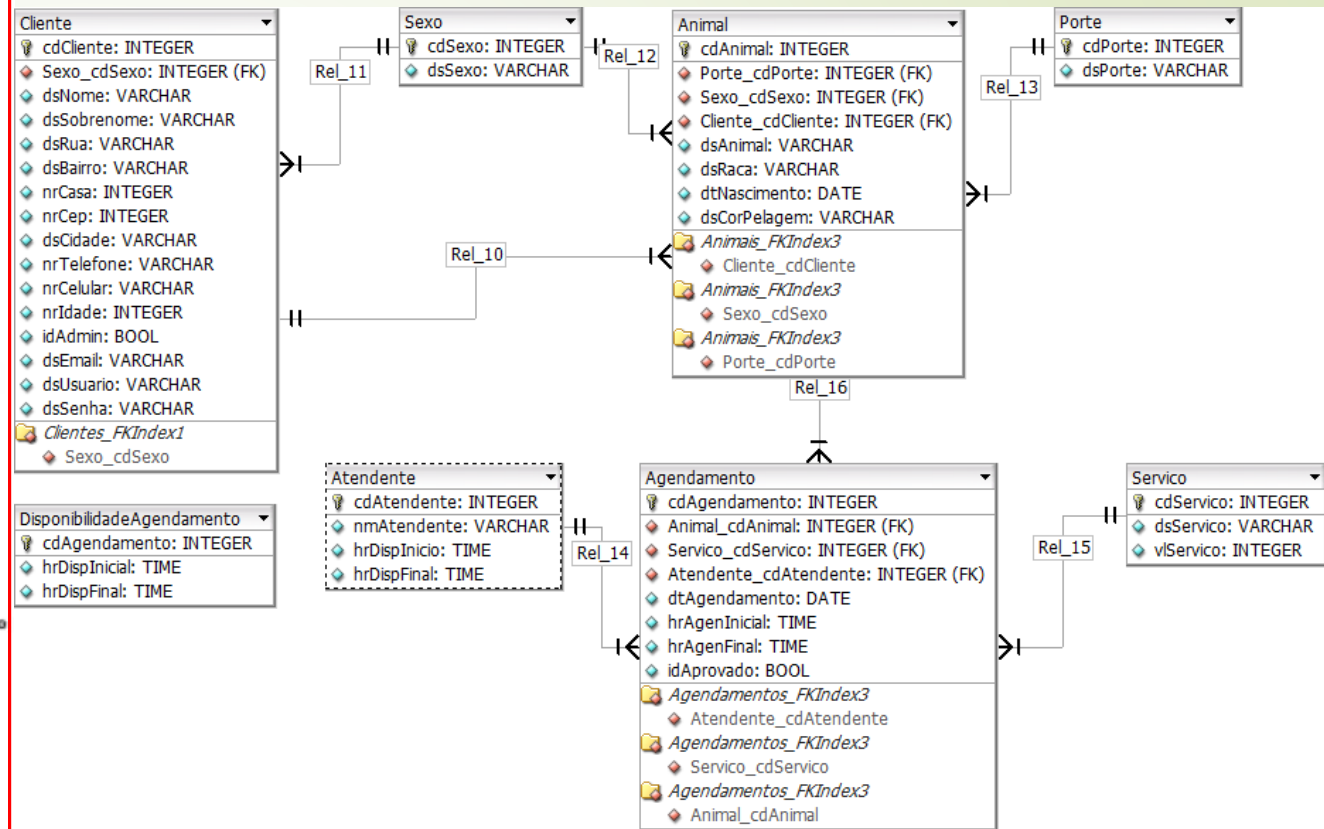
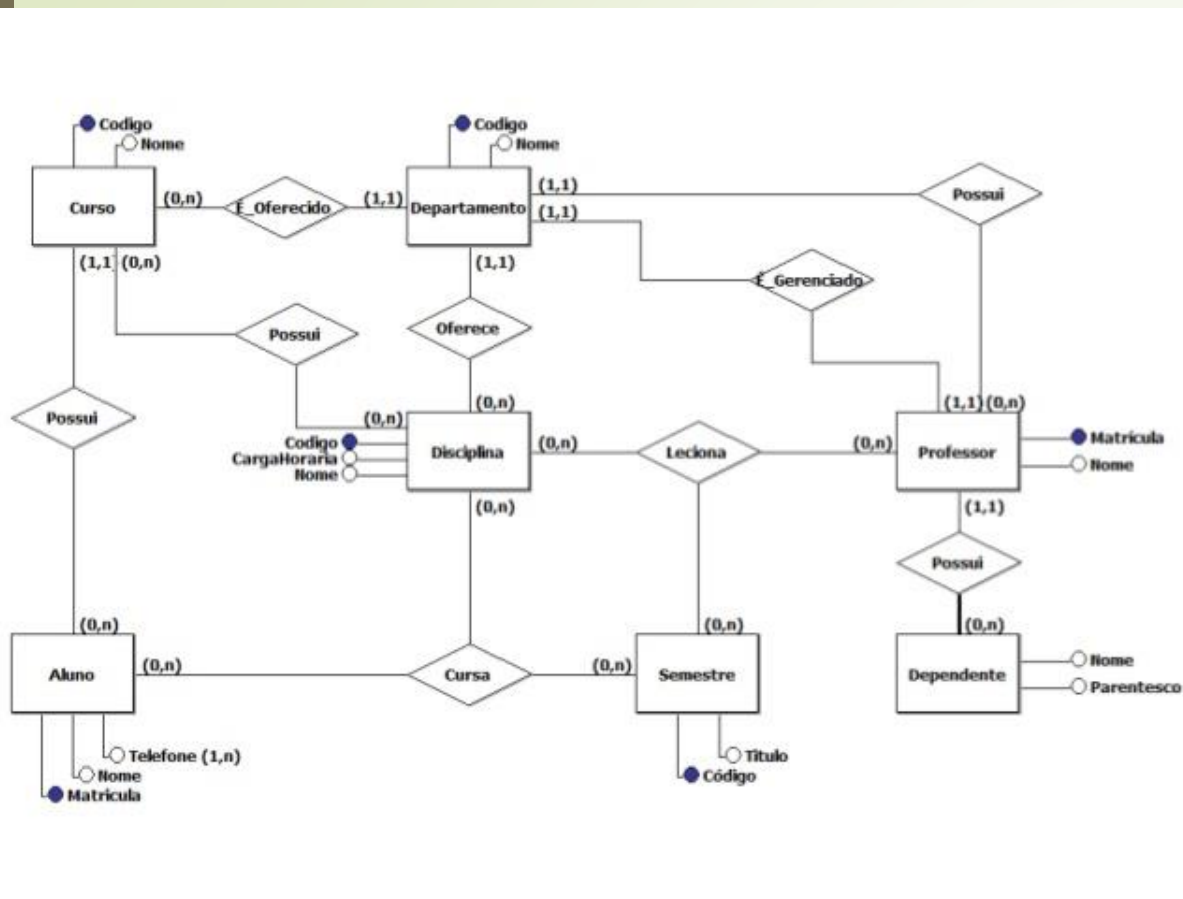
SELEÇÃO

Agenda

- 1 MongoDB - Overview;
- 2 Conceitos iniciais: execução, coleções, documentos e comandos básicos;
- 3 **Relacionamentos: conversão de um MER para um modelo de documentos;**
- 4 Métodos avançados de recuperação de informações no MongoDB.

Relacionamentos no MongoDB

- ◆ O MongoDB não implementa integridade referencial e nem operações de junção!
 - ◆ Logo, não existe o conceito de chave estrangeira para documentos.
- ◆ Existem duas maneiras de se expressar relacionamentos entre documentos no MongoDB.
 - ◆ Referências entre documentos: é possível guardar o `_id` de um documento como um atributo em outro documento.
 - ◆ Não é o mesmo que guardar uma chave estrangeira!
 - ◆ Documentos embutidos: o MongoDB permite guardar um documento inteiro como um atributo em um documento (Sub-Documents).



MER - MODELO ENTIDADE-RELACIONAMENTO

```
{  
  "FUNCIONÁRIO"  
  "_id" : 0,  
  "nome" : "riv roca",  
  "sexo" : "F",  
  "escolaridade" : "MC",  
  "cpf" : "462.726.237-04",  
  "datanasc" : {  
    "dia" : 13,  
    "mes" : 11,  
    "ano" : 1985  
  }  
}
```

```
{  
  "DEPARTAMENTO"  
  "_id" : 0,  
  "nome" : "DEPT0",  
  "funcionario_id" : [0,25000,50000,75000]  
}
```

```
{  
  "PRODUTO"  
  "_id" : 0,  
  "departamento_id" : 0,  
  "nome" : "PRODUTO00",  
  "valor" : 75.4866714477539  
}
```

```
{  
  "DISTRIBUIDORA"  
  "_id" : 0,  
  "nome" : "DIST0",  
  "produto_cidade_id" : [[0,0],[50000,0]]  
}
```

```
{  
  "CIDADE"  
  "_id" : 0,  
  "nome" : "CIDADE0",  
  "qtdhab" : 1231043  
}
```

MODELO DE DOCUMENTOS (REFERÊNCIAS)

MODELO DE DOCUMENTOS (REFERÊNCIAS/EMBTIDOS)

```
{
  "_id" : 0,
  "nome" : "DEPTO",
  "funcionarios" : [
    {
      "_id" : 0,
      "nome" : "riv roca",
      "sexo" : "F",
      "escolaridade" : "MC",
      "cpf" : "462.725.237-04",
      "datanasc" : { "dia" : 13, "mes" : 11, "ano" : 1985 }
    },
    {
      "_id" : 25000,
      "nome" : "cupi mafudaj",
      "sexo" : "F",
      "escolaridade" : "MC",
      "cpf" : "995.261.845-05",
      "datanasc" : { "dia" : 14, "mes" : 7, "ano" : 1955 }
    },
    {
      "_id" : 50000,
      "nome" : "bino pitibe",
      "sexo" : "M",
      "escolaridade" : "MC",
      "cpf" : "887.281.566-78",
      "datanasc" : { "dia" : 27, "mes" : 1, "ano" : 1968 }
    },
    {
      "_id" : 75000,
      "nome" : "lob cizame",
      "sexo" : "F",
      "escolaridade" : "MI",
      "cpf" : "188.733.432-58",
      "datanasc" : { "dia" : 8, "mes" : 6, "ano" : 1965 }
    }
  ]
}
```

```
{
  "_id" : 0,
  "nome" : "DISTR",
  "produto_cidade" : [
    {
      "_id" : 0,
      "departamento_id" : 0,
      "nome" : "PRODUTOS",
      "valor" : 75.4866714477539
    },
    {
      "_id" : 0,
      "nome" : "CIDADE",
      "qtddad" : 1231043
    }
  ],
  {
    {
      "_id" : 50000,
      "departamento_id" : 0,
      "nome" : "PRODUTOS",
      "valor" : 46.78875687299885
    },
    {
      "_id" : 0,
      "nome" : "CIDADE",
      "qtddad" : 1231043
    }
  ]
}
```

Agenda

- 1 O MongoDB e sua participação no mercado;
- 2 Conceitos iniciais: execução, coleções, documentos e comandos básicos;
- 3 Relacionamentos: conversão de um MER para um modelo de documentos;
- 4 Métodos avançados de recuperação de informações no MongoDB.

Consultas Complexas em MongoDB

- ◆ O MongoDB possui diversos **métodos e operadores** que permitem a realização de consultas complexas em seus dados.
 - ◆ Estes serão apresentados por meio de consultas escritas para os modelos de dados apresentados na **seção anterior**.
 - ◆ Frameworks como Map-Reduce não serão abordados nesta apresentação.
- ◆ As consultas apresentadas nesta seção foram retiradas de:
 - ◆ CASTRO, João Pedro de Carvalho. **Uma Análise de Técnicas de Recuperação de Informações em um SGBD NoSQL**. Monografia (Bacharelado em Sistemas de Informação) - Universidade Federal de Itajubá, Itajubá, 2015.

Operadores - MongoDB

Operadores MongoDB

\$eq	Retorna os valores iguais ao valor especificado
\$gt	Retorna os valores maior que o valor especificado
\$gte	Retorna os valores maiores ou iguais ao valor especificado
\$in	Retorna valores contidos no array especificado
\$lt	Retorna valores menores que o valor especificado
\$lte	Retorna os valores menores ou iguais ao valor especificado
\$ne	Retorna os valores diferentes do valor especificado
\$nin	Retorna valores que não contidos estão no array especificado

Operadores - MongoDB

Operadores Lógicos - MongoDB

\$and	Junção de Cláusulas da Consulta - Retorna todos os documentos que atendem às condições
\$not	Inverte o efeito de uma expressão de consulta - Retorna todos os documentos que não atendem à expressão da consulta
\$nor	Junção de Cláusulas da Consulta - Retorna todos os documentos que não atendem às condições
\$or	Junção de Cláusulas da Consulta - Retorna todos os documentos que atendem a pelo menos uma das condições

Operadores - MongoDB

Operadores "Elementos" - MongoDB

\$exists	Recupera documentos que tem um campo especificado
\$type	Seleciona documentos se um campo é de um tipo especificado

Operadores - MongoDB

Operadores "Avaliação" - MongoDB

\$expr	Recupera documentos que tem um campo especificado
\$jsonSchema	Valida o documento contra um dado "Schema JSON"
\$mod	Calcula o módulo do valor de um campo e seleciona documentos com um resultado especificado
\$regex	Seleciona documentos onde os valores são iguais a uma dada expressão regular
\$text	Realiza uma busca textual
\$where	Recupera documentos que satisfaçam a uma expressão JavaScript

Operadores - MongoDB

Operadores "Geo Espaciais" - MongoDB

\$geoIntersects	Seleciona a geometria que intercepta com uma geometria JSON
\$geoWithin	Seleciona geometrias dentro de uma geometria delimitadora GeoJSON
\$near	Retorna objetos geoespaciais na proximidade de um ponto. Requer um índice geoespacial.
\$nearSphere	Retorna objetos geoespaciais em proximidade a um ponto em uma esfera.

Operadores - MongoDB

Operadores "Array" - MongoDB

\$all	Recupera os arrays que contém os elementos especificado na consulta
\$elemMatch	Seleciona documentos se o elemento no campo do Array corresponder a todas as condições \$elemMatch especificadas.
\$size	Seleciona documentos se o campo do Array tiver um tamanho especificado.

Operadores - MongoDB

Operadores BitWise - MongoDB

\$bitsAllClear	Corresponde valores numéricos ou binários nos quais um conjunto de posições de bit tem um valor igual a 0
\$bitsAllSet	Corresponde valores numéricos ou binários nos quais um conjunto de posições de bit tem um valor de 1
\$bitsAnyClear	Corresponde valores numéricos ou binários nos quais qualquer bit de um conjunto de posições de bit tem um valor de 0
\$bitsAnySet	Corresponde valores numéricos ou binários nos quais qualquer bit de um conjunto de posições de bit tem um valor de 1

Operadores - MongoDB

Operador de Comentário - MongoDB

\$comments	Adiciona um comentário a um predicado de consulta
------------	---

Operadores - MongoDB

Operadores de Projeção - MongoDB

\$	Projeta o primeiro elemento em um array que corresponde à condição de consulta
\$elemMatch	Projeta o primeiro elemento em um array que corresponde à condição \$elemMatch especificada
\$meta	Projeta a pontuação do documento atribuída durante a operação \$text
\$slice	Limita o número de elementos projetados de um array. Suporta saltar e limitar fatias

Consulta 1: Operador LIKE

- ◆ Consulta em SQL:

```
SELECT * FROM FUNCIONARIO WHERE NOME LIKE 'nocaxi tepox';
```

- ◆ Consulta no MongoDB (Modelo com Referências):

```
db.getCollection("funcionario").find({nome: /nocaxi tepox/})
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("departamento").findOne({'funcionarios.nome': /nocaxi tepox/},  
{_id:0, 'funcionarios.$':1})
```

Consulta 2: Operador OR

- ◆ Consulta em SQL:

```
SELECT NOME, CPF FROM FUNCIONARIO WHERE ESCOLARIDADE = 'SI' OR ESCOLARIDADE = 'SC';
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
db.getCollection("funcionario").find({$or: [{escolaridade: "SC"}, {escolaridade: "SI"}]}, {_id: 0, nome: 1, cpf: 1})
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("departamento").aggregate([{ $unwind: "$funcionarios" }, { $match: { $or: [ { 'funcionarios.escolaridade': "SC" }, { 'funcionarios.escolaridade': "SI" } ] } }, { $project: { _id: 0, 'funcionarios.nome': 1, 'funcionarios.cpf': 1 } } ])
```

Consulta 3: Operador IN

- ◆ Consulta em SQL:

```
SELECT NOME, CPF FROM FUNCIONARIO WHERE ESCOLARIDADE IN ('SI', 'SC');
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
db.getCollection("funcionario").find({escolaridade: {$in: ["SC", "SI"]}}, {_id: 0, nome: 1, cpf: 1})
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("departamento").aggregate([{$unwind: "$funcionarios"}, {$match: {'funcionarios.escolaridade': {$in: ["SC", "SI"]}}}, {$project: {_id: 0, 'funcionarios.nome': 1, 'funcionarios.cpf': 1}}])
```


Consulta 4: Operadores >=/<=/!=

- ◆ Consulta em SQL:

```
SELECT CODIGO, NOME, CPF FROM FUNCIONARIO WHERE SEXO = 'M' AND ANONASC >= 1970 AND ANONASC <= 1990 AND ESCOLARIDADE != 'SI' AND ESCOLARIDADE != 'SC';
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
db.getCollection("funcionario").find({sexo: "M", 'datanasc.ano': {$gte: 1970, $lte: 1990}, $and: [{escolaridade: {$ne: "SC"}}, {escolaridade: {$ne: "SI"}}]}, {_id: 1, nome: 1, cpf: 1})
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("departamento").aggregate([{$unwind: "$funcionarios"}, {$match: {'funcionarios.sexo': "M", 'funcionarios.datanasc.ano': {$gte: 1970, $lte: 1990}, $and: [{'funcionarios.escolaridade': {$ne: "SC"}}, {'funcionarios.escolaridade': {$ne: "SI"}}]}}, {$project: {_id:0, 'funcionarios.id':1, 'funcionarios.nome':1, 'funcionarios.cpf':1}}])
```

Consulta 5: Operadores >=/<=/NOT IN

- ◆ Consulta em SQL:

```
SELECT CODIGO, NOME, CPF FROM FUNCIONARIO WHERE SEXO = 'M' AND ANONASC BETWEEN 1970 AND 1990 AND ESCOLARIDADE NOT IN ('SI', 'SC');
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
db.getCollection("funcionario").find({sexo: "M", 'datanasc.ano': {$gte: 1970, $lte: 1990}, escolaridade: {$nin: ["SC", "SI"]}}, {_id: 1, nome: 1, cpf: 1})
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("departamento").aggregate([{$unwind: "$funcionarios"}, {$match: {'funcionarios.sexo': "M", 'funcionarios.datanasc.ano': {$gte: 1970, $lte: 1990}, 'funcionarios.escolaridade': {$nin: ["SC", "SI"]}}}, {$project: {_id:0, 'funcionarios.id':1, 'funcionarios.nome':1, 'funcionarios.cpf':1}}])
```


Consulta 6: Função Agregada COUNT

- ◆ Consulta em SQL:

```
SELECT COUNT(*) FROM PRODUTO;
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
db.getCollection("produto").aggregate({$group: {"_id": null, "count": {$sum: 1}}})
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("distribuidora").aggregate([{$unwind: "$produto_cidade"},  
{ $unwind: "$produto_cidade", $match: {'produto_cidade.valor': {$exists: true}}},  
{ $group: {"_id": "$produto_cidade._id"}}, { $group: {"_id": null, "count": {$sum:  
1}}}]])
```

Consulta 7: Função Agregada AVG

- ◆ Consulta em SQL:

```
SELECT * FROM CIDADE WHERE QTDHAB <= (SELECT AVG(QTDHAB) FROM CIDADE);
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
var resultado = db.getCollection("cidade").aggregate({"$group": {"_id": null, "avgHab": {"$avg": "$qtdhab"}}}).next()  
db.getCollection("cidade").find({qtdhab: {$lte: resultado.avgHab}})
```

Consulta 7: Função Agregada AVG

- ◆ Consulta em SQL:

```
SELECT * FROM CIDADE WHERE QTDHAB <= (SELECT AVG(QTDHAB) FROM CIDADE);
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
var resultado = db.getCollection("distribuidora").aggregate([{$unwind:
"$produto_cidade"}, {$unwind: "$produto_cidade"}, {$match:
{'produto_cidade.qtdhab': {$exists: true}}}, {$group: {"_id":
"$produto_cidade._id", "qtdhab": {$addToSet: "$produto_cidade.qtdhab"}}}, {$unwind:
"$qtdhab"}, {$group: {"_id": null, "avgHab": {"$avg": "$qtdhab"}}}]).next()

db.getCollection("distribuidora").aggregate([{$unwind: "$produto_cidade"},
{$unwind: "$produto_cidade"}, {$match: {'produto_cidade.qtdhab': {$exists: true},
'produto_cidade.qtdhab': {$lte: resultado.avgHab}}}, {$project: {_id:0,
produto_cidade:1}}])
```


Consulta 8: Função Agregada SUM

- ◆ Consulta em SQL:

```
SELECT CODDEPT, SUM(V valor) FROM PRODUTO GROUP BY CODDEPT HAVING SUM(V valor) >= 200;
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
db.getCollection("produto").aggregate({$group: {_id: "$departamento_id", somaValor: {$sum: "$valor"}}}, {$match: {somaValor: {$gte: 200}}})
```

Consulta 8: Função Agregada SUM

- ◆ Consulta em SQL:

```
SELECT CODDEPT, SUM(V valor) FROM PRODUTO GROUP BY CODDEPT HAVING SUM(V valor) >= 200;
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("distribuidora").aggregate([{$unwind: "$produto_cidade"},  
{ $unwind: "$produto_cidade"}, {$match: {'produto_cidade.valor': {$exists: true}}},  
{ $group: {_id: "$produto_cidade._id", valor: {$addToSet: "$produto_cidade.valor"},  
departamento_id: {$addToSet: "$produto_cidade.departamento_id"}}}, {$unwind: "$valor"},  
{ $group: {_id: "$departamento_id", somaValor: {$sum: "$valor"}}},  
{ $match: {somaValor: {$gte: 200}}}]])
```

Consulta 9: Operação de Junção

- ◆ Consulta em SQL:

```
SELECT D.NOME, F.NOME FROM DEPARTAMENTO AS D, FUNCIONARIO AS F, FUNCDEPT AS FD  
WHERE D.CODIGO = FD.CODDEPT AND F.CODIGO = FD.CODFUNC AND F.SEXO = 'F' AND  
F.ANONASC >= 1970;
```

- ◆ Consulta no MongoDB (Modelo de Referências):

```
db.getCollection("departamento").find().forEach(function (newDept) {  
    newDept.funcionario_id = db.getCollection("funcionario").find({"_id": {$in:  
    newDept.funcionario_id}}).toArray();  
    db.getCollection("temp").insert(newDept);  
})  
  
db.getCollection("temp").aggregate([{$unwind: "$funcionario_id"}, {$match:  
{'funcionario_id.sexo': "F", 'funcionario_id.datanasc.ano': {$gte: 1970}}},  
{$project: {_id:0, nome:1, 'funcionario_id.nome': 1}}])  
  
db.getCollection("temp").drop()
```


Consulta 9: Operação de Junção

- ◆ Consulta em SQL:

```
SELECT D.NOME, F.NOME FROM DEPARTAMENTO AS D, FUNCIONARIO AS F, FUNCDEPT AS FD  
WHERE D.CODIGO = FD.CODDEPT AND F.CODIGO = FD.CODFUNC AND F.SEXO = 'F' AND  
F.ANONASC >= 1970;
```

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("departamento").aggregate([{$unwind: "$funcionarios"}, {$match:  
{'funcionarios.sexo': "F", 'funcionarios.datanasc.ano': {$gte: 1970}}}, {$project:  
{_id:0, nome:1, 'funcionarios.nome': 1}}])
```

Consulta 10: ...

- ◆ Consulta em SQL:

```
SELECT COUNT(DISTINCT(D.CODIGO)) FROM DISTRIBUIDORA AS D, PRODUTO AS P, CIDADE AS C, DISTCIDPROD AS DCP WHERE D.CODIGO = DCP.CODDIST AND P.CODIGO = DCP.CODPROD AND C.CODIGO = DCP.CODCID AND C.QTDHAB <= (SELECT AVG(QTDHAB) FROM CIDADE) AND P.VALOR >= (SELECT AVG(VALOR) FROM PRODUTO);
```

Consulta 10: ...

- ◆ Consulta no MongoDB (Modelo de Referências):

```
var resultado = db.getCollection("produto").aggregate({"$group": {"_id": null,
"avgValor": {"$avg": "$valor"}}}).next()

var resultado2 = db.getCollection("cidade").aggregate({"$group": {"_id": null,
"avgHab": {"$avg": "$qtdhab"}}}).next()

db.getCollection("distribuidora").aggregate([{$unwind: "$produto_cidade_id",
{$project: {_id:0, nome:1, produto_cidade_id:1, true_id: "$_id"}}, {$out: "temp"}])

db.getCollection("temp").find().forEach(function(newDist) {
    newDist.produto = db.getCollection("produto").findOne({_id:
    newDist.produto_cidade_id[0]});
    newDist.cidade = db.getCollection("cidade").findOne({_id:
    newDist.produto_cidade_id[1]});
    delete newDist.produto_cidade_id;
    db.getCollection("temp").save(newDist);
})

db.getCollection("temp").aggregate([{$match: {'cidade.qtdhab': {$lte:
resultado2.avgHab}, 'produto.valor': {$gte: resultado.avgValor}}}, {$group: {"_id":
"$true_id"}}, {$group: {"_id": null, "count": {$sum: 1}}}]

db.getCollection("temp").drop()
```


Consulta 10: ...

- ◆ Consulta no MongoDB (Modelo com Documentos Embutidos):

```
db.getCollection("distribuidora").aggregate([{$unwind: "$produto_cidade"},
{$project: {_id:0, nome:1, produto_cidade:1, true_id: "$_id"}}, {$out: "temp"}])

db.getCollection("temp").find().forEach(function(newDist) {
    newDist.produto = newDist.produto_cidade[0];
    newDist.cidade = newDist.produto_cidade[1];
    delete newDist.produto_cidade;
    db.getCollection("temp").save(newDist);
})

var resultado = db.getCollection("temp").aggregate([{$group: {"_id":
"$produto._id", "valor": {$addToSet: "$produto.valor"}}}, {$unwind: "$valor"},
{$group: {"_id": null, "avgValor": {"$avg": "$valor"}}}]).next()

var resultado2 = db.getCollection("temp").aggregate([{$group: {"_id":
"$cidade._id", "qtdhab": {$addToSet: "$cidade.qtdhab"}}}, {$unwind: "$qtdhab"},
{$group: {"_id": null, "avgHab": {"$avg": "$qtdhab"}}}]).next()

db.getCollection("temp").aggregate([{$match: {'cidade.qtdhab': {$lte:
resultado2.avgHab}, 'produto.valor': {$gte: resultado.avgValor}}}, {$group: {"_id":
"$true_id"}, {$group: {"_id": null, "count": {$sum: 1}}}}])

db.getCollection("temp").drop()
```

Consulta 10: Possível Solução

- ◆ Consulta no MongoDB (Modelo com Duplicatas, não mostrado na apresentação):

```
var resultado = db.getCollection("documentogeral").aggregate([{$group: {"_id": "$cidade._id", "qtdhab": {$addToSet: "$cidade.qtdhab"}}}, {$unwind: "$qtdhab"}, {$group: {"_id": null, "avgHab": {"$avg": "$qtdhab"}}}]).next()

var resultado2 = db.getCollection("documentogeral").aggregate([{$group: {"_id": "$produto._id", "valor": {$addToSet: "$produto.valor"}}}, {$unwind: "$valor"}, {$group: {"_id": null, "avgValor": {"$avg": "$valor"}}}]).next()

db.getCollection("documentogeral").aggregate([{$match: {'cidade.qtdhab': {$lte: resultado.avgHab}, 'produto.valor': {$gte: resultado2.avgValor}}}, {$group: {"_id": "$dist_id"}, {$group: {"_id": null, "count": {$sum: 1}}}]])
```