{ "name": "Mongo", "type": "DB", "authors": ["João Pedro Castro", "Cristina Ciferri"] }

Roteiro

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.

Roteiro

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.

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**.

{ "fonte": "https://docs.mongodb.com/manual/introduction/" }

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.

{ "fonte": "https://db-engines.com/en/ranking_definition" }

Critérios do DB-ENGINES

- Menções do SGBD em **mecanismos de busca**;
- **Interesse geral** no SGBD (Google Trends);
- Frequência de **discussões técnicas** sobre o SGBD (Stack Overflow e DBA Stack Exchange);
- Número de **ofertas de emprego** relacionadas ao SGBD;
- Número de **perfis em redes profissionais** onde o SGBD é mencionado (Linkedin e Upwork);
- Relevância em **redes sociais** (Twitter).

{ "fonte": "https://db-engines.com/en/ranking_definition" }

O Ranking do DB-ENGINES

Rank			DBMS	Database Model	Score		
May 2017	Apr 2017	May 2016			May 2017	Apr 2017	May 2016
1.	1.	1.	Oracle	Relational DBMS	1354.31	-47.68	-107.71
2.	2.	2.	MySQL	Relational DBMS	1340.03	-24.59	-31.80
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1213.80	+9.03	+70.98
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	365.91	+4.14	+58.30
5.	5.	↓ 4.	MongoDB	Document store	331.58	+6.16	+11.36
6.	6.	6.	DB2	Relational DBMS	188.84	+2.18	+2.88
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	129.87	+1.69	-1.70
8.	8.	↓ 7.	Cassandra	Wide column store	123.11	-3.07	-11.39
9.	9.	9.	Redis	Key-value store	117.45	+3.09	+9.21
10.	10.	10.	SQLite	Relational DBMS	116.07	+2.27	+8.81

{ "fonte": "https://db-engines.com/en/ranking" }

O MongoDB no Mercado

PROGRAMADOR (JavaScript, HTML, NodeJs) (v1500721)

Código da vaga:	v1500721
Nível hierárquico:	Pleno
Local:	Curitiba / PR / BR
Quantidade de vagas:	1

REQUISITOS OBRIGATÓRIOS:
- Ensino superior completo
- SQL
- JavaScript
- HTML
- NodeJs

REQUISITOS DESEJÁVEIS:
- MongoDB
- Conhecimentos de Design
- Bootstrap

EDITORA POSITIVO

ATIVIDADES:
- Manutenção da base de dados (SQL Server).
- Criação de projetos web (front end e back end).

{ "fonte": "https://www.vagas.com.br/vagas/v1500721/programador-javascript-html-nodejs" }

O MongoDB no Mercado

Arquiteto Técnico - Scrum Master (SP) (v1510244)

Código da vaga:	v1510244
Nível hierárquico:	Supervisão/Coordenação
Local:	São Paulo / SP / BR

Os requisitos para esta posição são:

- Experiência na indústria de serviços financeiros;
- Profundo conhecimento no desenvolvimento e criação de arquiteturas para projetos complexos usando tecnologias e frameworks como: SAFe, Node, Bootstrap, MongoDB, .NET e JavaScript;
- Certificado Scrum Master (CSM);
- Superior completo;
- Inglês intermediário.

avanade®
Results Realized

{ "fonte": "https://www.vagas.com.br/vagas/v1510244/arquiteto-tecnico-scrum-master-sp" }

O MongoDB no Mercado

Consultor Especialista (v1501921)

Código da vaga:	v1501921
Nível hierárquico:	Sênior
Local:	São Paulo / SP / BR
Quantidade de vagas:	1
Data de expiração:	13 de Maio de 2017

Experiência em ambientes HortonWorks/Azure HDInsights (Hadoop, Hive, Spark 1.6 e 2.0), administração de ambiente ETL com Talend (TAC).
Conhecimentos em Gestão/Modelagem de Dados Estruturado e Não Estruturado.
Administração de ambientes no SQL: MongoDB e Cassandra.

Desejável:
Conhecimento e visão arquitetônica em ambientes;
Conhecimento de esteiras de CI/CD com Jenkins.

Atos

{ "fonte": "https://www.vagas.com.br/vagas/v1501921/consultor-especialista" }

O MongoDB no Mercado

Analista Desenvolvedor Magento

hoje

De R$ 4.001,00 a R$ 5.000,00

1 vaga: Valinhos - SP (1)

Atuar em fábrica de software com desenvolvimento na tecnologia Magento, integração de sistemas e modelagem de programação.

Experiência em PHP Magento, MongoDB, ZendFramework, jQuery, CSS, Sit, JavaScript, WebServices e Json.

enviar currículo 7 dias grátis

BENEFÍCIOS

Tíquete-alimentação

HORÁRIO

De segunda a sexta, das 9h às 18h.

REGIME DE CONTRATAÇÃO

CLT (Efetivo)

{ "fonte": "http://www.catho.com.br/vagas/analista-desenvolvedor-magento/11659583/" }

O MongoDB no Mercado

Engenheiro de Software

quarta, 19/04

De R$ 10.001,00 a R$ 15.000,00
1 vaga: São Paulo - SP (1)

Desenvolvimento de novos produtos com arquiteturas resilientes, escaláveis e em tempo real utilizando tecnologia de ponta. Desenvolvimento de novas features e manutenção de softwares em C# e NodeJS. Participar ativamente de todo o ciclo de desenvolvimento, desde a coleta de requisitos, arquitetura, desenvolvimento e testes. Participar de discussão e propor arquiteturas e tecnologias que se adequem ao objetivo do projeto/demanda baseando-se em dados e cases. Nosso stack é composto por tecnologia Microsoft, porém, estamos totalmente abertos e valorizamos os poliglotas tecnológicos. Experiência em arquitetura e desenvolvimento de software resilientes, escaláveis e com alto volume de informações. Conhecimento em desenvolvimento orientado a testes e habilidade na resolução de problemas do negócio. Experiência em desenvolvimento .NET/C#, Node.JS. Experiência com microservices e arquitetura REST. Orientação a objetos. Banco de dados relacionais/não-relacionais. Código limpo e técnicas de refatoração. Metodologias e práticas ágeis.

INFORMAÇÕES ADICIONAIS

Tecnologias que utilizamos: .NET/C#, Node.Js, Java. AWS (EC2, ELB, SQS, Beanstalk, RDS, SNS, S3). Aurora. Oracle. Elastic Search. MongoDB. Tableau. RedShift. Jenkins. New Relic. Kafka. Docker. Ansible.

{ "fonte": "http://www.catho.com.br/vagas/engenheiro-de-software/11769216/" }

Roteiro

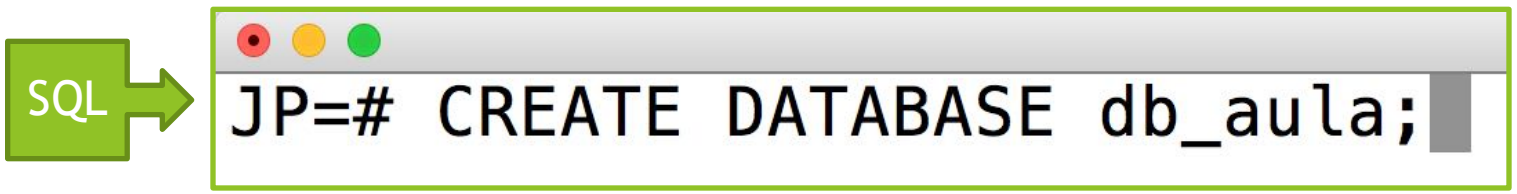
1. O MongoDB e sua participação no mercado;
2. Conceitos iniciais: execução, coleções, documentos e comandos básicos;
3. Relacionamentos: conversão de um MER para um modelo de documentos;
4. Métodos avançados de recuperação de informações no MongoDB.

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**.



Coleções e Documentos (Insert)

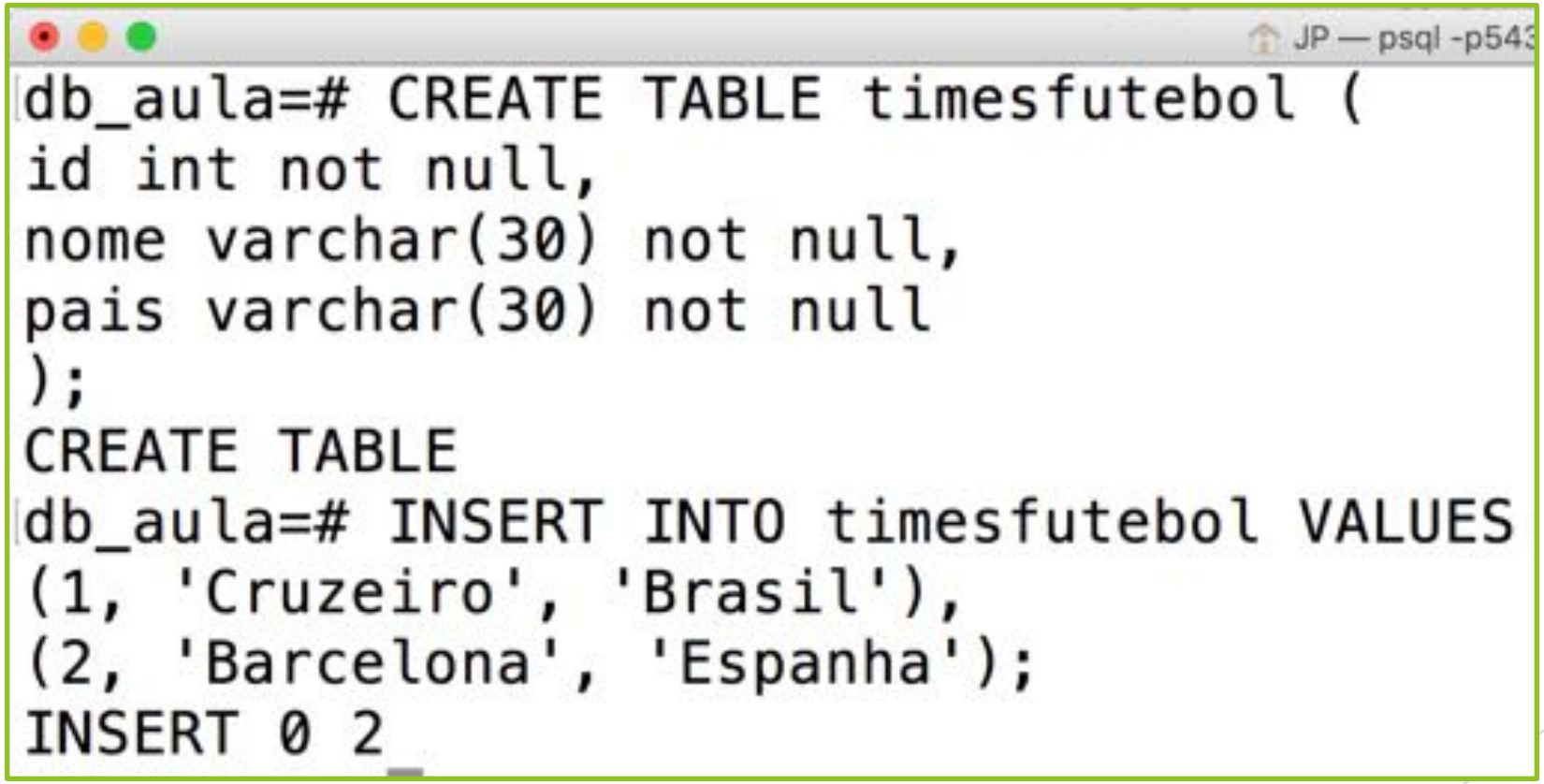
- Como em outros modelos orientado a documentos, o MongoDB organiza os dados em **coleções** de **documentos**.
  - Cada documento possui um atributo identificador (_id) e uma **quantidade qualquer** de outros atributos.
  - **Não é necessário (Mas é possível) especificar o ID dos documentos!**
  - **Não 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



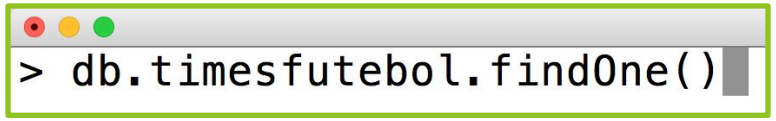
Coleções e Documentos (Insert)

```
> 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")
        ]
}
```

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**.

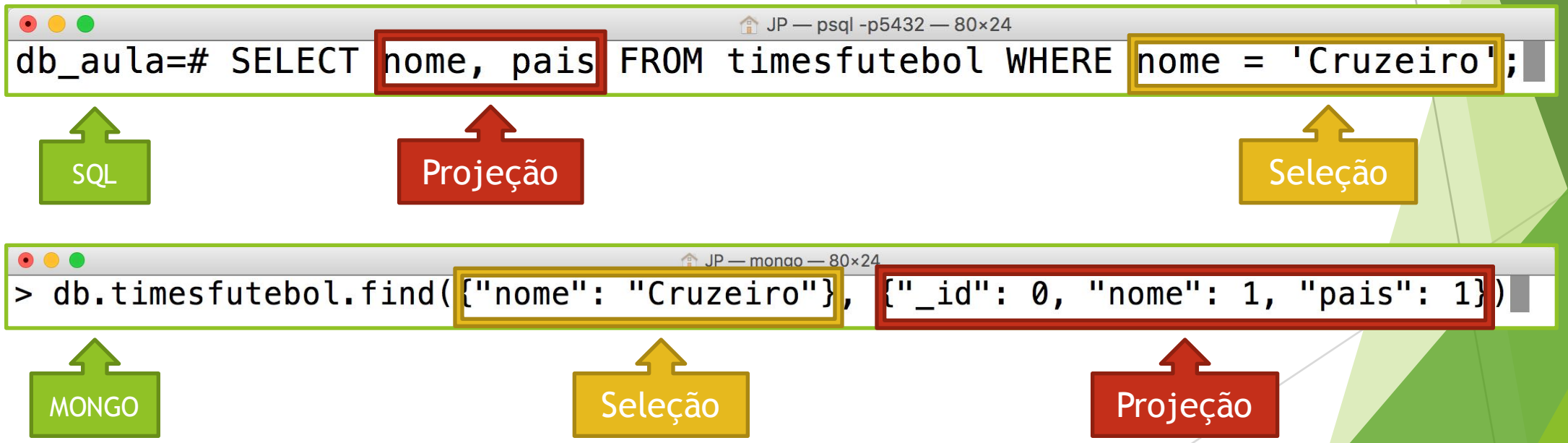
SQL



MONGO

Coleções e Documentos (Select)

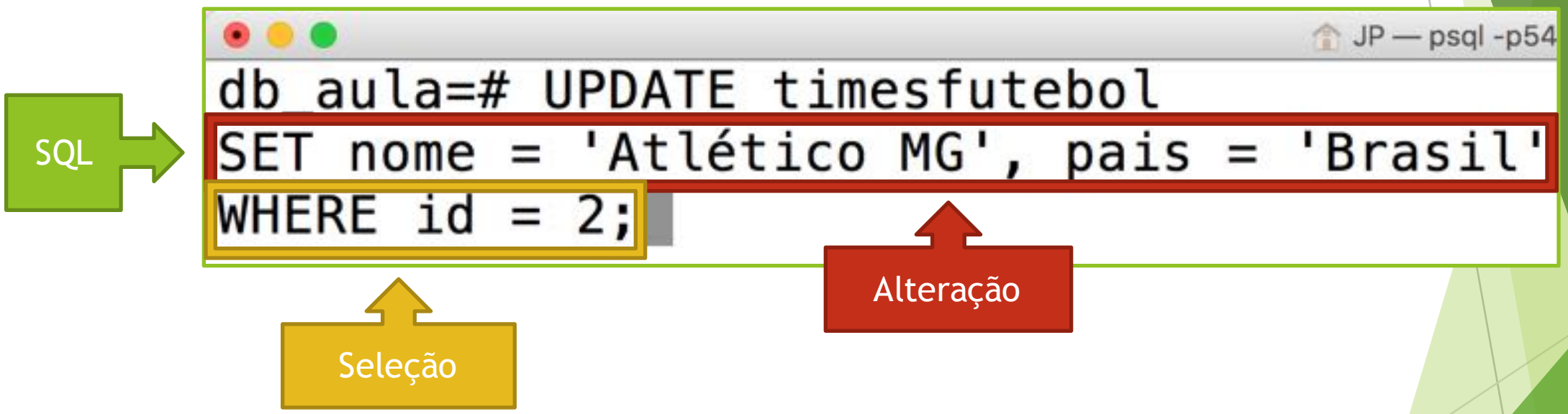
- Ambos os métodos **find()** e **findOne()** permitem especificar, **da mesma forma**, critérios de **seleção** e **projeção** para o resultado.



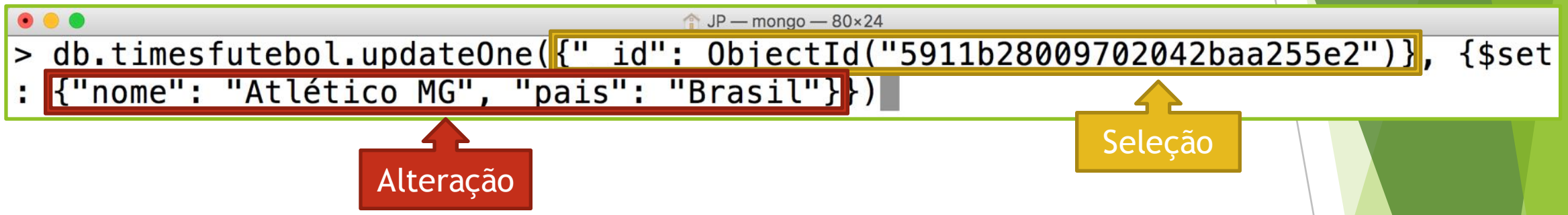
Coleções e Documentos (Update)

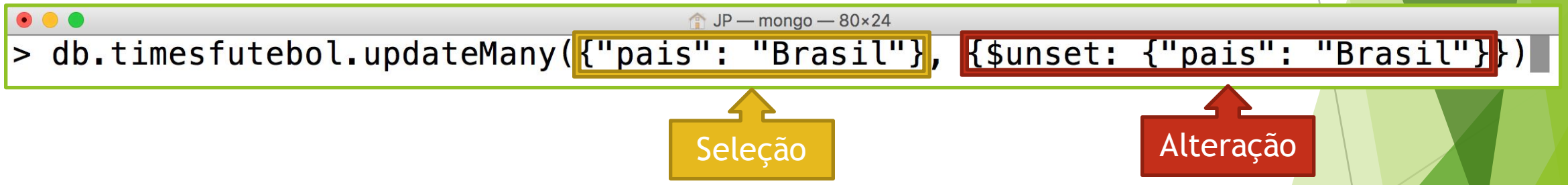
- O MongoDB possui três métodos para atualização de dados em um documento.
- Os médotos **updateOne()** e **updateMany()** localizam o documento segundo os critérios especificados e fazem as alterações descritas.
  - Diferença: **quantidade** de documentos afetada.
  - Enquanto o **updateOne()** afeta 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.

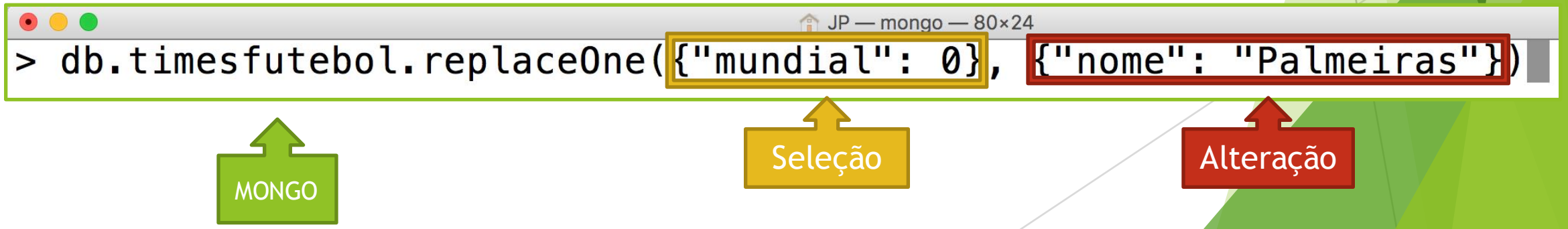
Coleções e Documentos (Update)



Coleções e Documentos (Update)





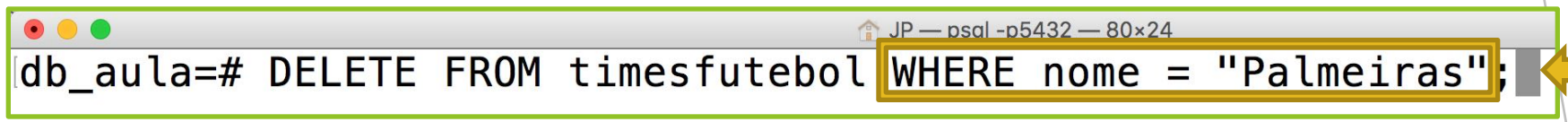


Coleções e Documentos (Delete)

- O MongoDB possui dois métodos para a remoção de documentos.
- Os médotos **deleteOne()** e **deleteMany()** localizam o documento segundo os critérios especificados e o 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**.

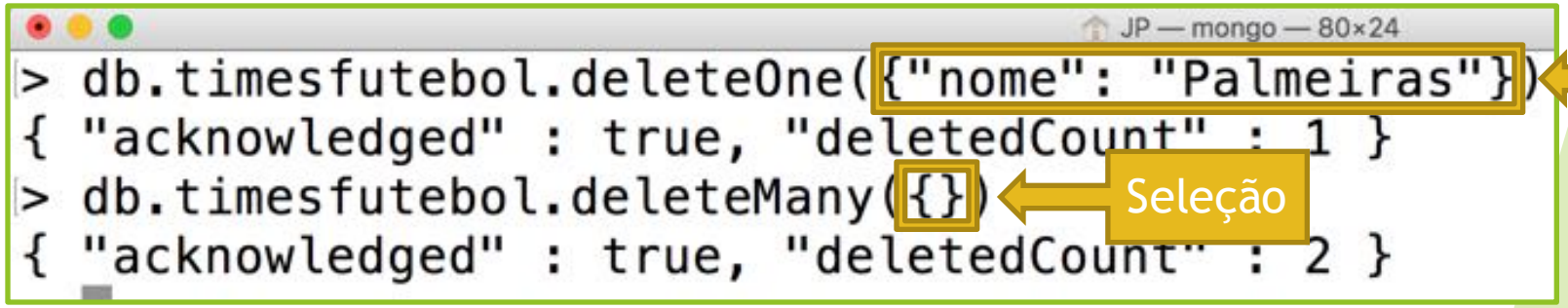
Coleções e Documentos (Delete)

SQL



Seleção

MONGO



Seleção

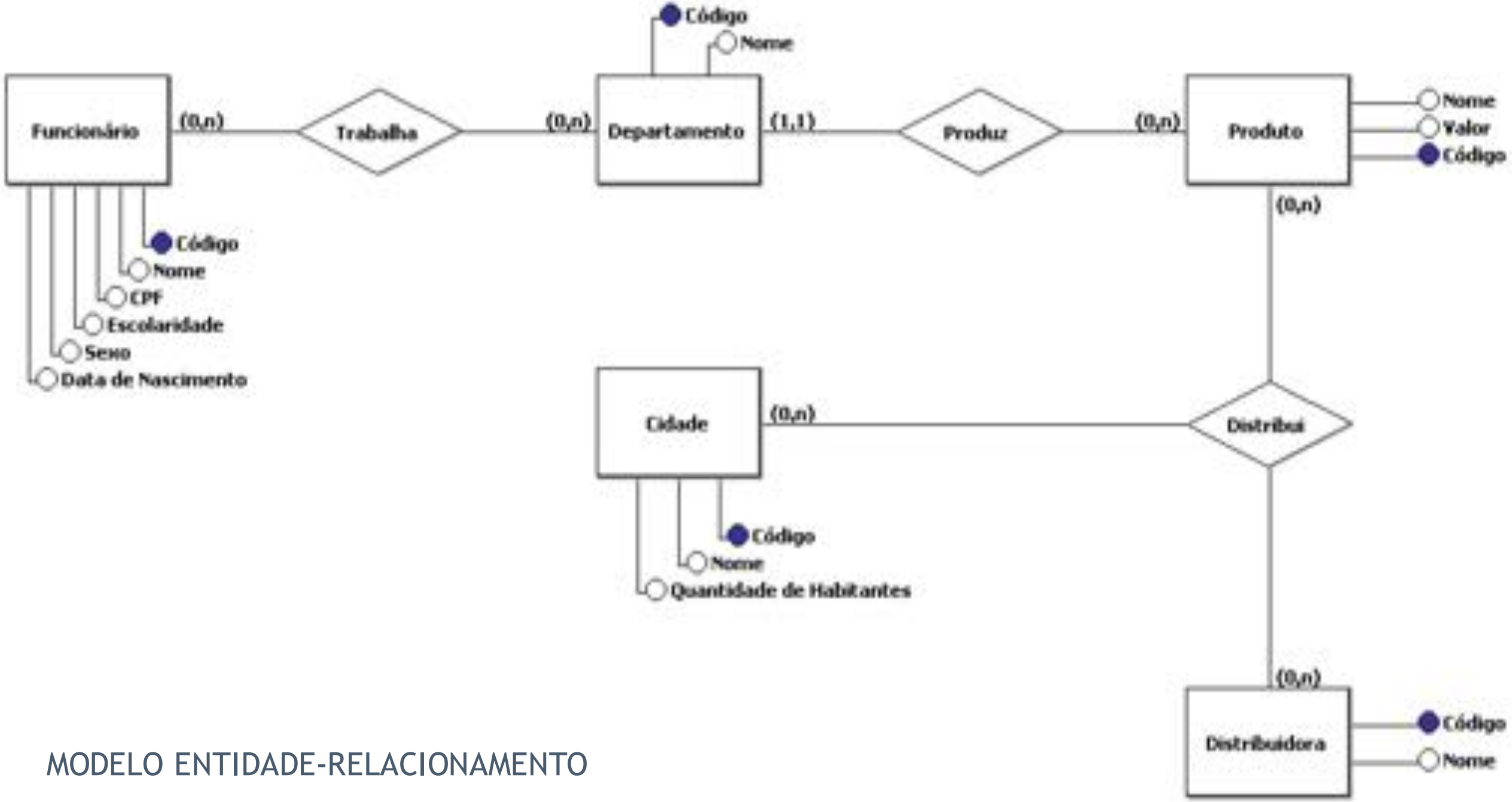
Seleção

Roteiro

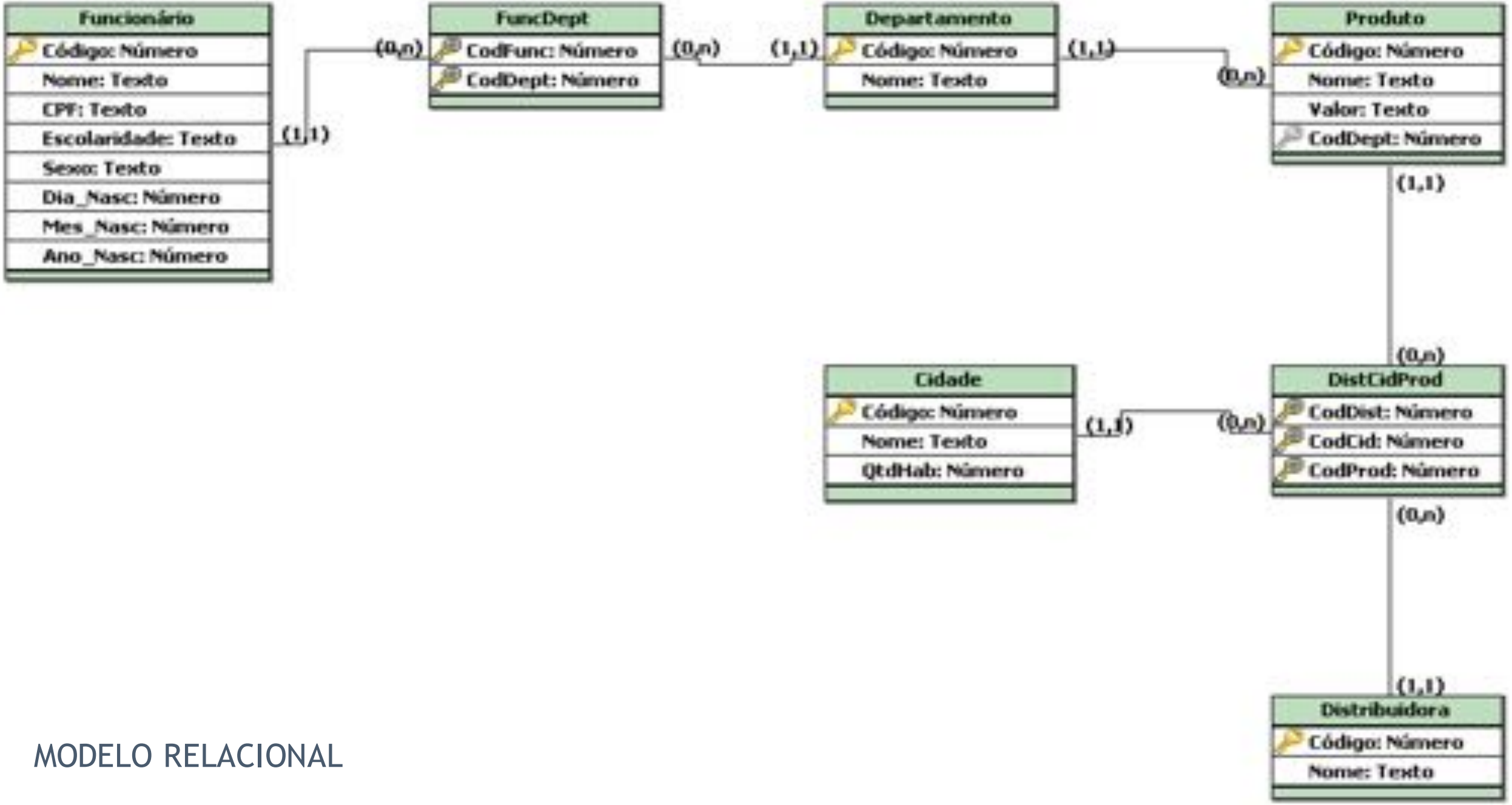
1. O MongoDB e sua participação no mercado;
2. Conceitos iniciais: execução, coleções, documentos e comandos básicos;
3. Relacionamentos: conversão de um MER para um modelo de documentos;
4. Métodos avançados de recuperação de informações no MongoDB.

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-Documentos).



MODELO ENTIDADE-RELACIONAMENTO



MODELO RELACIONAL

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/EMBUTIDOS)

DEPARTAMENTO


```
{
    "_id" : 0,
    "nome" : "DEPT0",
    "funcionarios" : [
                        {
                            "_id" : 0,
                            "nome" : "riv roca",
                            "sexo" : "F",
                            "escolaridade" : "MC",
                            "cpf" : "462.726.237-04",
                            "datanasc" : {"dia" : 13,"mes" : 11,"ano" : 1985}
                        },
                        {
                            "_id" : 25000,
                            "nome" : "cupi mafudaj",
                            "sexo" : "F",
                            "escolaridade" : "MC",
                            "cpf" : "965.261.845-05",
                            "datanasc" : {"dia" : 14,"mes" : 7,"ano" : 1955}
                        },
                        {
                            "_id" : 50000,
                            "nome" : "bimo pitibe",
                            "sexo" : "M",
                            "escolaridade" : "MC",
                            "cpf" : "807.201.566-70",
                            "datanasc" : {"dia" : 27,"mes" : 1,"ano" : 1968}
                        },
                        {
                            "_id" : 75000,
                            "nome" : "lob citame",
                            "sexo" : "F",
                            "escolaridade" : "MI",
                            "cpf" : "180.733.432-58",
                            "datanasc" : {"dia" : 8,"mes" : 6,"ano" : 1965}
                        }
                    ]
}
```


DISTRIBUIDORA


```
{
    "_id" : 0,
    "nome" : "DIST0",
    "produto_cidade" : [
                        [
                            {
                                "_id" : 0,
                                "departamento_id" : 0,
                                "nome" : "PRODUTO0",
                                "valor" : 75.48667144773539
                            },
                            {
                                "_id" : 0,
                                "nome" : "CIDADE0",
                                "qtdhab" : 1231843
                            }
                        ],
                        [
                            {
                                "_id" : 50000,
                                "departamento_id" : 0,
                                "nome" : "PRODUTO50000",
                                "valor" : 46.70875607259885
                            },
                            {
                                "_id" : 0,
                                "nome" : "CIDADE0",
                                "qtdhab" : 1231843
                            }
                        ]
                    ]
}
```

Roteiro

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.

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(VALOR) FROM PRODUTO GROUP BY CODDEPT HAVING SUM(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(VALOR) FROM PRODUTO GROUP BY CODDEPT HAVING SUM(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}}}])
```