Banco de Dados Relacionais e Não Relacionais

Prof. Henrique Batista da Silva

BASE e NoSQL

Alguns desafios para persistências de dados

Atomicidade (a transação é executada totalmente ou **não** é executada), **C**onsistência (sistema sempre consistente após uma operação), **I**solamento (transação não sofre interferência de outra transação concorrente), e **D**urabilidade (o que foi salvo não é mais perdido)

- Força a consistência ao final de cada transação

NoSQL: BASE

Propriedades BASE:

Basically Available – Basicamente Disponível.

Soft-State – Estado Leve

Eventually Consistent – Eventualmente Consistente.

NoSQL: BASE

Uma aplicação funciona basicamente todo o tempo (**Basicamente Disponível**);

O estado do sistema pode mudar (mesmo durante o período sem escrita – devido a consistência eventual) (**Estado Leve**); e

NoSQL: BASE

O sistema torna-se consistente em algum momento após operações de escrita (**Eventualmente Consistente**).

Problema do mundo real

Álbum de músicas

Considere uma aplicação que contém o perfil de **músicos**, **bandas**, **álbuns** e **músicas**.

Iremos resolver um problema do mundo real e veremos quais as limitações ao usar um modelo Relacional.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

Ex.: Um disco pode conter o **número de semanas que ficou em primeiro lugar** na Billboard* e o **número de músicas na primeira posição** (nem todo disco irá figurar nas primeiras posições)

* Revista norte americana sobre indústria da música

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

Cadastro dos álbuns:

além da **banda** e das **músicas**, um álbum também possui por padrão **ano de lançamento**, **ilustrador da capa**, **produtor** ou qualquer outra informação necessária.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

É necessário armazenar informações sobre o **estúdio**.

Sabe-se que muitos discos são gravados em um único estúdio, mas podem ser gravados em mais de um ou em nenhum (disco independente).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

Observe neste cenário que **muitos campos** não farão sentidos **para muitos discos**.

Seria necessário uma estrutura diferente para cada disco (usar modelo relacional?).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo Relacional

Album

Cod	nome	artista	dataLanc	estudio	produtor	semanasEmPrimeiro	numMusicasEmPrimeiro
1	The Dark Side Of The Moon	Pink Floyd	4/29/1973		Pink Floyd	1	3
2	Nevermind	Nirvana	1/11/1992	Sound City Studios, Smart Studios (Madison)		1	1
3	Independente	independente	01/01/2017				

Mesma estrutura para todos os registros.

Musica

CodMusica	Nome	CodAlbum

Observe que seria necessário também uma nova tabela chamada Estúdio, uma vez que podem haver mais de um estúdio para cada Álbum

Modelo Relacional

Album

Cod	nome	artista	dataLanc	estudio	produtor	semanasEmPrimeiro	numMusicasEmPrimeiro
1	The Dark Side Of The Moon	Pink Floyd	4/29/1973		Pink Floyd	1	3
2	Nevermind	Nirvana	1/11/1992	Sound City Studios, Smart Studios (Madison)		1	1
3	Independente	independente	01/01/2017				

Musica

CodMusica	Nome	CodAlbum

Uma desvantagem clara para este tipo de problema diz respeito a **quantidade de colunas para representar cada campo possível** (que só fará sentido para alguns registros).

Modelo Relacional (solução alternativa)

Album

Cod	**codBanda**	**nome**
1	1	Master of Puppets
2	1	...And Justice for All

Solução alternativa usando o modelo Relacional que visa evitar a inclusão de registros nulos.

Valores

codAlbum	**codAtributo**	**valor**
1	**1**	03/03/86
2	**1**	25/08/88
1	**2**	Sweet Silence Studio

Atributos

cod	**nome**
1	Data de lançamento
2	Estudio

Modelo Relacional (solução alternativa)

- Observe que a solução torna a manipulação do banco muito mais complexa.
- Veja o exemplo para uma simples consulta que exibe os detalhes de cada álbum.

```
SELECT atr.nome, val.valor
FROM atributos atr JOIN valores val ON val.id-atributo = atr.id
WHERE val.id-album = 1;
```

Limitações

- Problemas mais complexos podem transparecer melhor as limitações do modelo relacional. Ex.: sites de e-commerce, catálogos (Netflix, spotify), etc..

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Limitações

- O objetivo não é resolver problemas não solucionáveis com o banco relacional, e sim obter soluções mais **simples** e **práticas** (e **escaláveis**).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo de dados

Modelo de dados

Modelo de dados é o modo pelo qual

percebemos e manipulamos os dados

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Modelo de dados

Modelo de Relacional (organizado em tuplas, normalizado, e possui integridade referencial)

Tabela: Cliente	
Id	Nome
1	Marcos

Tabela: Pedido		
Id	IdCliente	IdEndEntrega
1	1	1

Tabela: ItemPedido			
Id	IdPedido	IdProduto	Preço
1	2	10	350,00

Tabela: Produto	
Id	Nome
10	Laptop

Tabela: Endereço				
Id	Logradouro	Cidade	Estado	CEP
1	Av. Sen. Salgado Filho	Natal	RN	59.056-000

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Modelo de dados agregados

Lógica por trás dos bancos NoSQL

A abordagem agregada reconhece que deseja-se trabalhar com **dados na forma de unidades** que tenham uma **estrutura mais complexa** do que um conjunto de tuplas

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Modelo de dados agregados

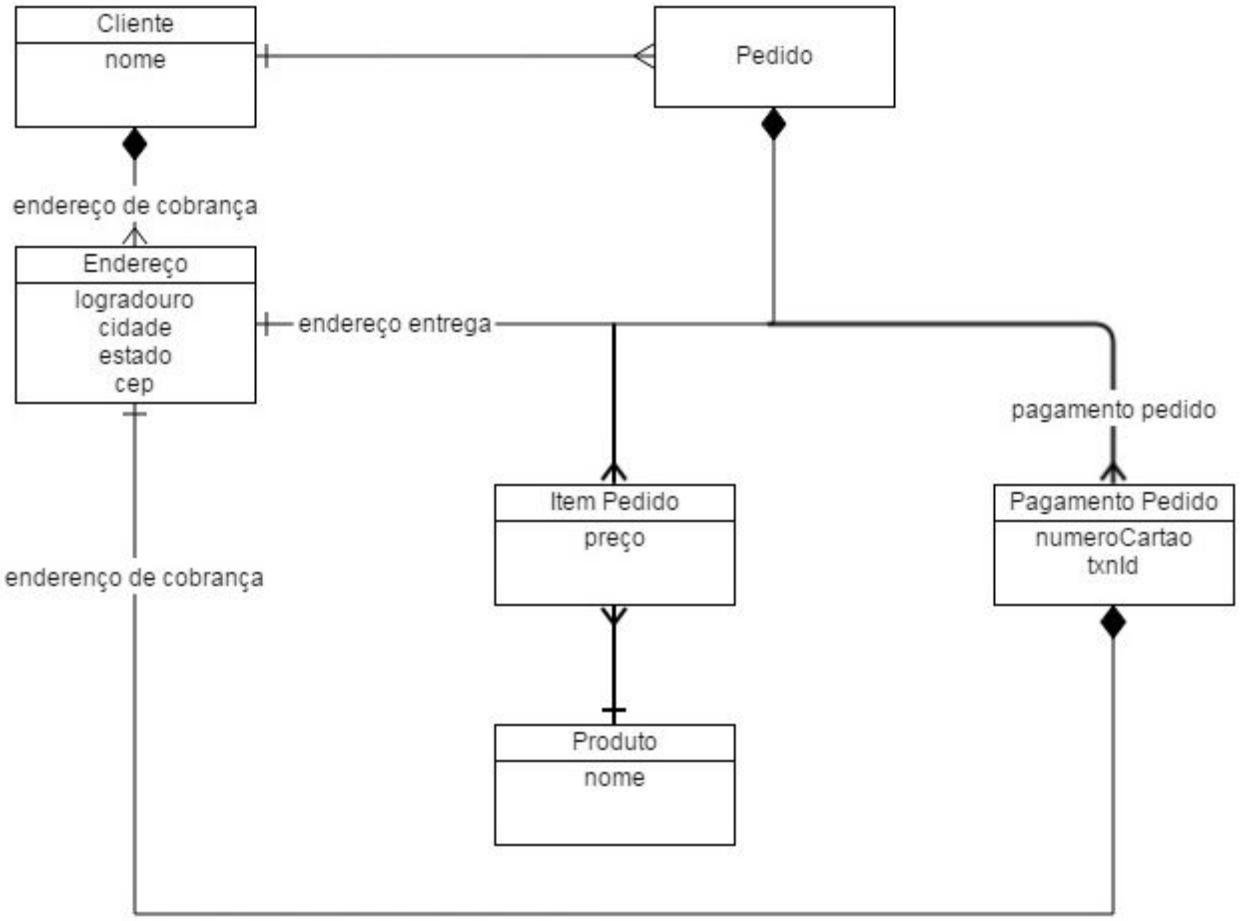
Um agregado é um conjunto de objetos relacionados

Facilita a distribuição em clusters, uma vez que o agregado constitui uma unidade natural de replicação e fragmentação

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Modelo de dados agregados

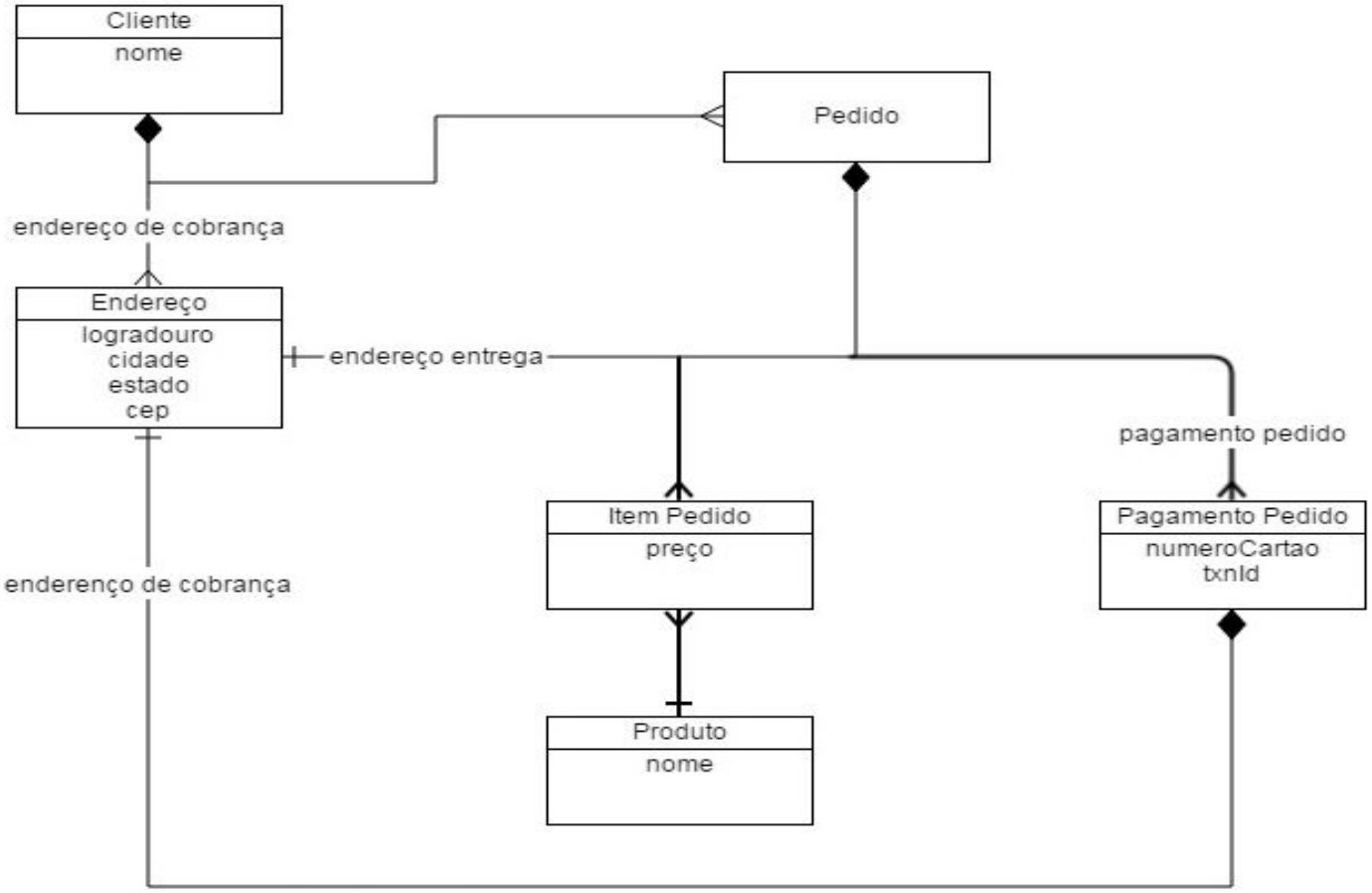
Composição: Cliente contém uma lista de endereços.




```
//em clientes
{
        "id": 1,
        "nome": "Marcos",
        "endcobranca":[{"cidade": "Chicago"}]
}

//em pedidos
{
        "id": 99,
        "idCliente": 1,
        "itensPedido": [
                {
                        "idProduto": 2,
                        "preco": 35.00,
                        "produtoNome": "Laptop"
                }
        ],
        "enderecoEntrega": [{"cidade": "Natal"}],
        "pagamentoPedido": [
                {
                        "numCartao": "1000-1000-1000-1000",
                        "endCobranca": [{"cidade": "Natal"}],
                }
        ]
}
```

Modelo de dados agregados



Poderia também colocar todos os pedidos de clientes no agregado do cliente

Modelo de dados agregados

Consequências:

(I) Relacionais **não possuem** conceito de agregados no modelo de dados. (II) Necessidade de conhecer previamente **como** e **o quê** deseja-se saber sobre os dados. (III) Conhecimento da estrutura agregada ajuda a armazenar e distribuir os dados

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Iniciando com o MongoDB

Banco de dados de documento

O MongoDB é um projeto Open Source (Linux, Mac e Windows)

A linguagem utilizada para manipulação dos dados é JavaScript.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Banco de dados de documento

Para instalação do MongoDB para Windows siga o tutorial (também disponível em outras plataformas):

https://docs.mongodb.com/manual/installation/

Criando um documento

No MongoDB precisamos criar uma instância para adicionar os documentos. Com o MongoDB instalado o comando **`show dbs`** mostra quais são os banco já criados.

Comando para criar o primeiro banco: **`use nome_do_banco`** (se não existir, ele irá criar)

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Criando um documento

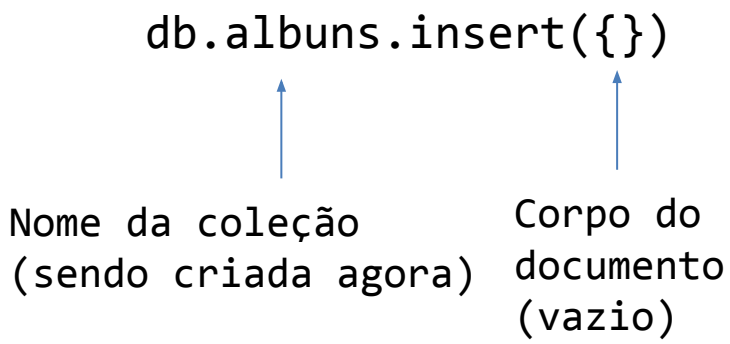
Após criar um novo banco de dados, precisamos criar uma nova coleção (equivalente a uma tabela no SGBDR).

Para criar uma nova coleção, basta adicionar um documento vazio utilizando o comando `insert`.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Criando um documento

Iremos chamar a função `insert` a partir do nosso objeto base db e do nome da coleção, veja:



Use o comando "`show collections`" para visualizar as coleções criadas

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Criando um documento

Da mesma forma que utilizamos o `insert` para inserir um novo documento, usamos o comando `find` para retornar documentos na coleção.

Retorna o _id do documento, equivalente a chave primária de uma linha no SGBDR (mas é gerenciado internamente pelo mongoDB)

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

{“_id” : ObjectId (“5921f80828b4776a45fd6e64”)}
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Trabalhando com documentos

Criando um documento

No MongoDB um documento é criado utilizando JavaScript. Portanto, o documento deve ser um objeto JavaScript, ou um JSON.

```
db.albuns.insert({"nome" : "The Dark Side of the Moon ",
"data" : new Date(1973, 3, 29)})
```

Chave

Valores (strings, date, etc) – no tipo Date, o mês começa do zero.

Utilizamos { e } para criar um documento e definimos as chaves e seus respectivos valores

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Criando um documento

Criando alguns documentos:

```
db.albuns.insert({"nome" : "Master of Puppets", "dataLancamento" : new
Date(1986, 2, 3), "duracao" : 3286})
```

```
db.albuns.insert({"nome" : "...And Justice for All", "dataLancamento" :
new Date(1988, 7, 25), "duracao" : 3929})
```

```
db.albuns.insert({"nome" : "Among the Living", "produtor" :
"Eddie Kramer"})
```

Criando um documento

Criando alguns documentos:

```
db.albuns.insert({"nome" : "Nevermind", "artista" : "Nirvana",
"estudioGravacao" : ["Sound City Studios", "Smart Studios (Madison)"],
"dataLancamento" : new Date(1992, 0, 11)})
```

```
db.albuns.insert({"nome" : "Reign in Blood", "dataLancamento" : new
Date(1986, 9, 7), "artista" : "Larry Carroll", "duracao" : 1738})
```

Criando um documento

Criando alguns documentos:

```
db.albuns.insert({"nome" : "Seventh Son of a Seventh Son", "artista" :
"Iron Maiden", "produtor" : "Martin Birch", "estudioGravacao" :
"Musicland Studios", "dataLancamento" : new Date(1988, 3, 11)})
```

Buscando um documento

Após a inserção dos documentos no banco, vamos realizar consultas sobre estes documentos.

Vimos que podemos usar o comando "`find`" para encontrar um documento passando como argumento {} (criteria).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Buscando um documento

Mas se desejar apenas listar todos os documentos inseridos, utilize: **`db.album.find()`** ou **`db.album.find().pretty()`** para exibição de forma estruturada.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Buscando um documento

Para buscar documentos por campos específicos, podemos fazer da seguinte forma:

```
db.albuns.find({"nome" : "Seventh Son of a Seventh Son"})
```

Equivalente ao SGBDR:

```
SELECT *
FROM albuns
WHERE nome = "Seventh Son of a Seventh Son"
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Buscando um documento

Para buscar documentos por campos específicos, podemos fazer da seguinte forma (para exibição de forma estruturada):

```
db.albuns.find({"nome" : "Seventh Son of a Seventh Son"}).pretty()
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Buscando um documento

Este comando irá retornar uma lista de documentos que satisfazem a condição.

Se desejar retornar apenas um documento (o primeiro que satisfaz a condição ou `null` se nenhum documento for encontrado), utilize `findone()`

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Buscando um documento

Para buscar documentos usando parte de uma string (“like”):

```
db.albuns.find({"nome" : /of/})
```

Equivalente ao SGBDR:

```
SELECT *
FROM albuns
WHERE nome LIKE '%of%'
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Operações de Update e Delete

Excluindo um documento

As mesma condições usada para filtrar um documento no MongoDB em uma consulta podem ser utilizadas para excluir um documento de uma coleção. Ao invés de usar o comando `find`, utilizamos o comando `remove`.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Excluindo um documento

Exemplo: vamos deletar o álbum “The Dark Side of the Moon”.

```
db.albuns.remove({"nome": "The Dark Side of the Moon"})
```

Equivalente em SQL

```
DELETE
FROM albuns
WHERE nome = " The Dark Side of the Moon "
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Excluindo um documento

Se desejar remover todos os documentos, utilize o comando remove sem nenhum critério:

```
db.albuns.remove({})
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Alterando um documento

Podemos utilizar o comando update para atualizar um documento na coleção:

```
db.albuns.update({"nome" : "Among the Living"}, {$set : {"duracao" : 3013}})
```

Agora busque pelo documento e verifique a alteração

```
db.albuns.find({"nome" : "Among the Living"})
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Realizando consultas mais complexas

Consultas mais complexas

Podemos realizar consultas mais complexas utilizando o MondoDB. Por exemplo, como encontrar um (ou mais) álbum(ns) cuja duração seja menor que um valor especificado na consulta.

Em um SGBDR, teríamos:


```
SELECT *
FROM albuns
WHERE duracao < 1800
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

Há vários operadores de comparação:

Operador	Descrição
$gt	maior que o valor específico na query.
$gte	maior ou igual ao valor específico na query.
$in	quaisquer valores que existem em um array específico em uma query
$lt	valores que são menores que o valor específico na query.
$lte	valores que são menores ou iguais que o valor específico na query
$ne	todos os valores que não são iguais ao valor específico na query.
$nin	valores que não existem em um array específico da query.

Tabela adaptada de : Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

Por exemplo, como encontrar um (ou mais) álbum(ns) cuja duração seja menor que um valor especificado na consulta.

```
db.albuns.find({"duracao" : {"$lt" : 1800}}).pretty()
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

```
{"nome" : "Reign in Blood",
"dataLancamento" : new Date(1986, 9, 7),
"artista" : "Larry Carroll",
"duracao" : 1738}
```

→ `"duracao" : 1738`

```
{"nome" : "Master of Puppets",
"dataLancamento" : new Date(1986, 2, 3),
"duracao" : 3286}
```

→ `"duracao" : 3286`

```
{"nome" : "...And Justice for All",
"dataLancamento" : new Date(1988, 7, 25),
"duracao" : 3929}
```

→ `"duracao" : 3929`

`duracao < 1800`

```
{"nome" : "Among the Living",
"produtor" : "Eddie Kramer"}
```

↑ Não possui duração

Resultado

```
{"nome" : "Reign in Blood",
"dataLancamento" : new Date(1986, 9, 7),
"artista" : "Larry Carroll",
"duracao" : 1738}
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

Outro exemplo, como encontrar um (ou mais) álbum(ns) cuja duração seja 1738 ou 3286.

```
db.albuns.find({"duracao" : {"$in" : [1738,3286]}}).pretty()
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

No MongoDB temos vários operadores lógicos:

Operador	Descrição
$and	Retorna documentos com ambas as condições verdadeiras
$nor	Retorna documentos com ambas as condições falsas
$not	Inverte o resultado de uma condição
$or	Retorna documentos com um das condições verdadeiras.

Tabela adaptada de : Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

Sintaxe da consulta com operadores lógicos:

```
{operador : [expressão 1, expressão 2, expressão n]}.
```


Critérios para a consulta

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

Exemplo: retornar todos os discos lançados em 1986.

```
db.albuns.find(
{$and : [{"dataLancamento" : {$gte : new Date(1986, 0, 1)}},
         {"dataLancamento" : {$lt : new Date(1987, 0, 1)}}]}
).pretty()
```

O valor deve ser maior na primeira cláusula e menor na segunda

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Consultas mais complexas

Exemplo: retornar todos os discos lançados em 1986.

```
db.albuns.find(
{$and : [{"dataLancamento" : {$gte : new Date(1986, 0, 1)}},
         {"dataLancamento" : {$lt : new Date(1987, 0, 1)}}]}
).pretty()
```

Equivalente em SQL (SGBDR)

```
SELECT *
FROM albuns
WHERE dataLancamento >= '1986-01-01' AND dataLancamento < '1987-01-01'
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Desafio 1: Realizando consultas

Consultas mais complexas

Exercício: retorne os álbuns que foram lançados antes de 1990.

Consultas mais complexas

Exercício: retorne os álbuns que foram lançados antes de 1990.

Solução:

```
db.albuns.find({"dataLancamento" : {"$lt" : new Date(1990,0,1)}}).pretty()
```

Consultas mais complexas

Exercício: retorne os álbuns que **não** foram lançados após 1980.

Consultas mais complexas

Exercício: retorne os álbuns que **não** foram lançados após 1980.

Solução:

```
db.albuns.find(
{"dataLancamento" : {$not : {$gte : new Date(1979, 11, 31)}}}
).pretty()
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Documentos aninhados no MongoDB

Documentos aninhados

Em muitas situações, pode ser mais interessante (por questões de performance por exemplo) aninhar um documento dentro do outro, como por exemplo:

```
{"nome" : "Master of Puppets",
"dataLancamento" : new Date(1986, 2, 3),
"duracao" : 3286
"artista" : {"nome" : "Metallica"}}
```

Ao invés de criar a coleção artistas, embutindo suas informações dentro do documento do álbum.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Documentos aninhados

Incialmente pode parecer estranho ter que repetir a mesma informação do artista dentro de vários documentos.

Entretanto, isto ajudará na performance da consulta (a mesma ficará mais simples).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Documentos aninhados

Vamos adicionar dois novos álbuns na coleção:

```
db.albuns.insert(
{"nome" : "Somewhere Far Beyond",
"dataLancamento" : new Date(1992, 5, 30),
"duracao" : 3328,
"artista" : {"nome" : "Blind Guardian"}});

db.albuns.insert(
{"nome" : "Imaginations from the Other Side",
"dataLancamento" : new Date(1995, 3, 4),
"duracao" : 2958,
"artista" : {"nome" : "Blind Guardian"}});
```

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Documentos aninhados

É possível realizar buscas pelos atributos dos subdocumentos:

```
db.albuns.find({"artista" : {"nome" : "Blind Guardian"}}).pretty();
```

Esta consulta retornar todos os álbuns do “Blind Guardian”, que é um atributo do subdocumento de album

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Documentos aninhados

Resolvemos um problema de performance duplicando informações dos artistas (que possuem apenas dois campos).

No caso de artistas possuírem muitos campos, uma solução parcial seria duplicar apenas o *nome* e o *_id* de cada artista e manter as demais informações em coleções separadas.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Desafio 2 - Dataset Airbnb

Realizando Consultas

Quando usar e não usar

Situações apropriadas para o uso

Registro de eventos: Diversos aplicativos tem necessidades de salvar logs de eventos. O banco de dados de documento pode atuar como um repositório central de eventos. Ideal principalmente quando há mudanças constantes no tipo de dados obtido pelo evento.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Situações apropriadas para o uso

Sistema de gerenciamento de conteúdo Web: por identificar o padrão JSON, é muito apropriado para aplicativos de publicações de websites, trabalhando com comentário de usuário, perfis e documentos visualizados.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Situações apropriadas para o uso

Comércio eletrônico: Muito último para armazenar informações de produtos que possuem diferentes características.

Análise de dados: fácil para armazenar visualizações de páginas e visitantes em tempo real.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Situações para *não* usar

Transações complexas: Operações atômicas (ou a transação será totalmente executada ou não será) em múltiplos documentos podem não ser ideias para este tipo de banco.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Para saber mais

Consulte o site do desenvolvedor:

The MongoDB 3.0 Manual

https://docs.mongodb.com/v3.0/#getting-started

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Links para saber mais:

Para conhecer mais sobre tipos de bancos NoSQL:

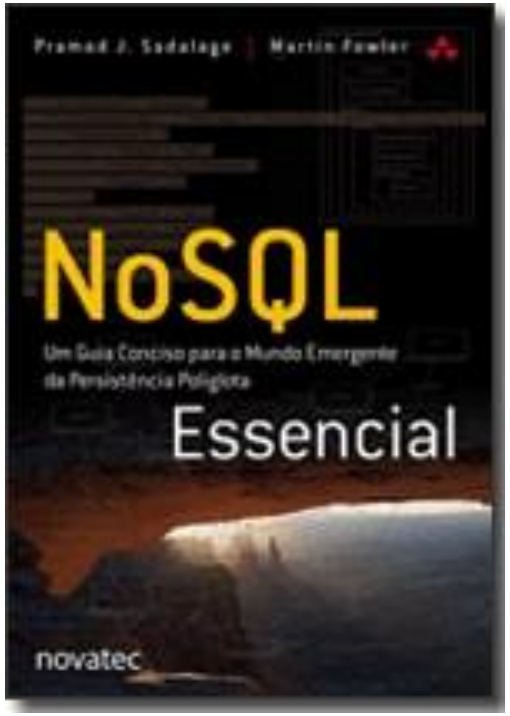
http://nosql-database.org/

Para saber mais: Consulte o site dos desenvolvedores

The MongoDB 3.0 Manual

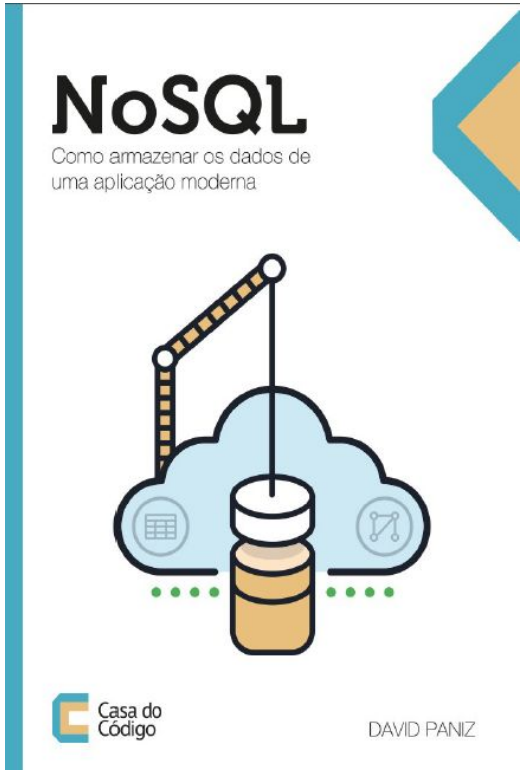
https://docs.mongodb.com/manual/

Principais Referências



Pramod J.; Sadalage, Martin Fowler. **NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota**. Novatec Editora, 2013.

Principais Referências



Paniz, David. NoSQL: **Como armazenar os dados de uma aplicação moderna**. Casa do Código, 2017.

Principais Referências

Boaglio, Fernando. MongoDB: **Construa novas aplicações com novas tecnologias**. Casa do Código, 2017.