

Aluno: Pedro Matias

Orientador: Evandrino Gomes

Co-orientado: Cristiano Maffort

Comparativo de desenvolvimento e desempenho entre MongoDB e PostgreSQL em aplicações com mapeamento de objetos

1

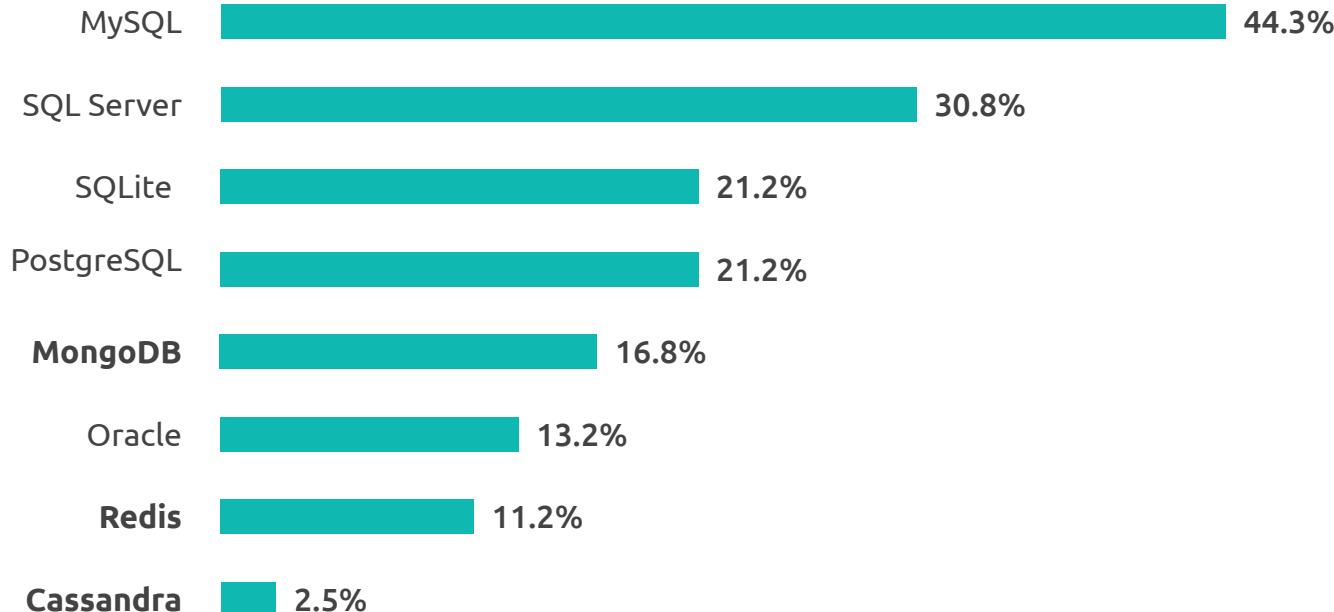
Introdução

Cenário

Crescimento no uso de bancos de dados não-relacionais como tecnologias componentes de *stacks* de softwares.

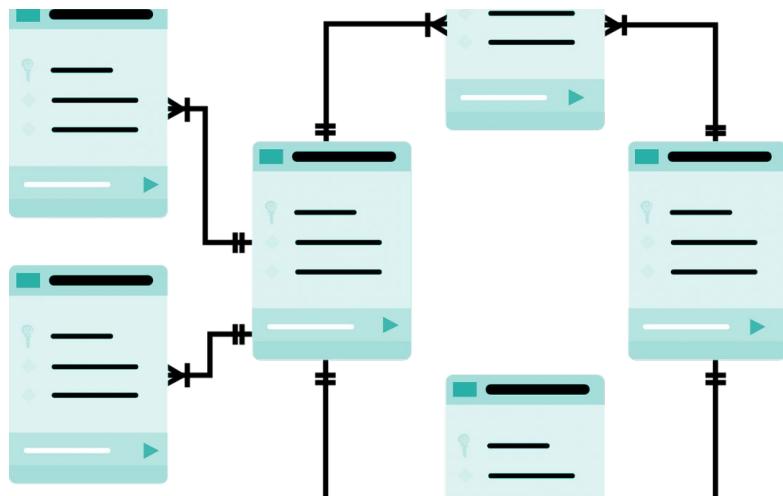


Most popular databases



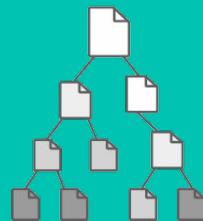
Fonte: *StackOverflow Survey (2017)*

Modelo Relacional (SQL)

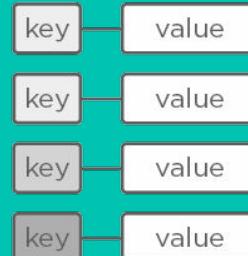


Modelo não-relacional (NoSQL)

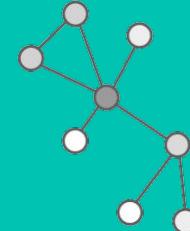
Documentos



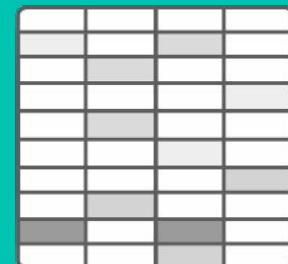
Chave-valor



Grafos



Wide-column



Pros & Contras

SQL

- **Pros**
 - Espaço de armazenamento reduzido;
 - Integridade forte por ACID (Atomicidade, Consistência, Isolamento e Durabilidade);
 - Acesso padrão aos dados via SQL;
 - Suporte de consulta mais flexível;
- **Contras**
 - Modelos de dados rígidos;
 - Dimensionamento horizontal ainda é um desafio;

NoSQL

- **Pros**
 - Escalável e altamente disponível;
 - Modelos de dados flexíveis;
 - Alto desempenho;
 - Abstrações de dados de alto nível;
- **Contras**
 - Integridade fraca com *ACID*;
 - Falta de flexibilidade nos padrões de acesso aos dados;

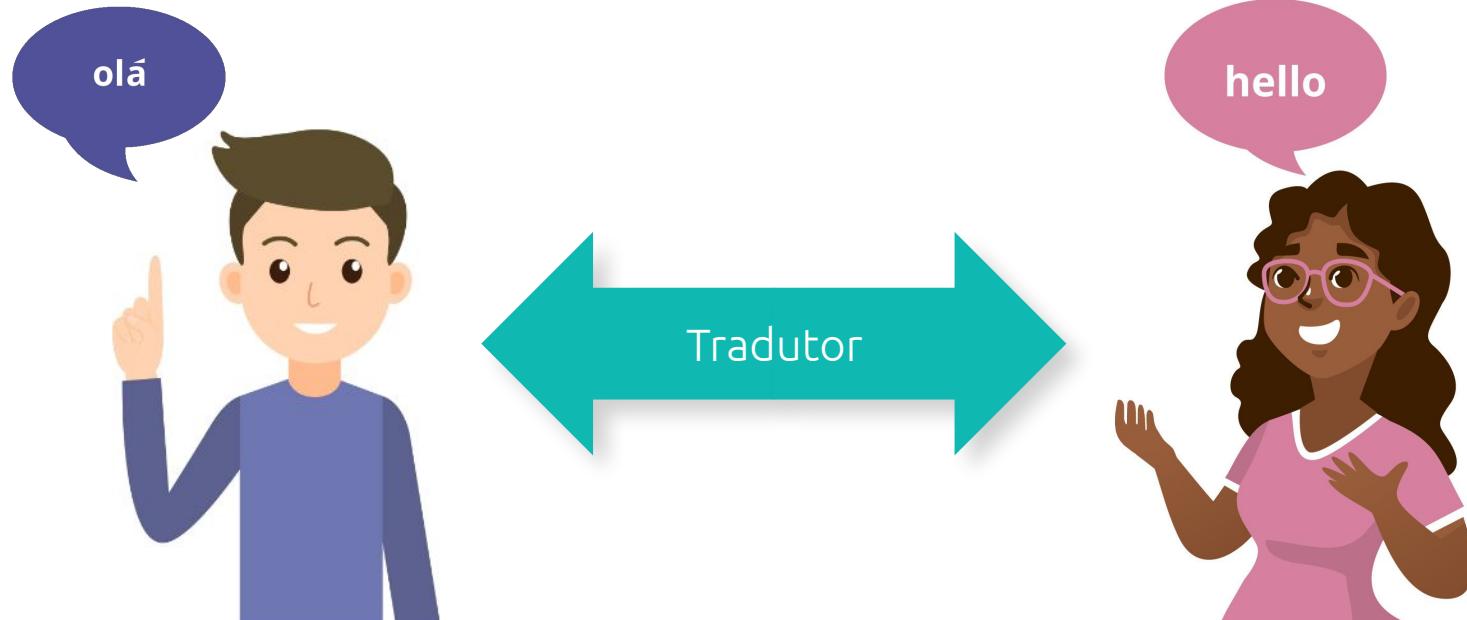
Fonte: ANDERSON e NICHOLSON, 2020



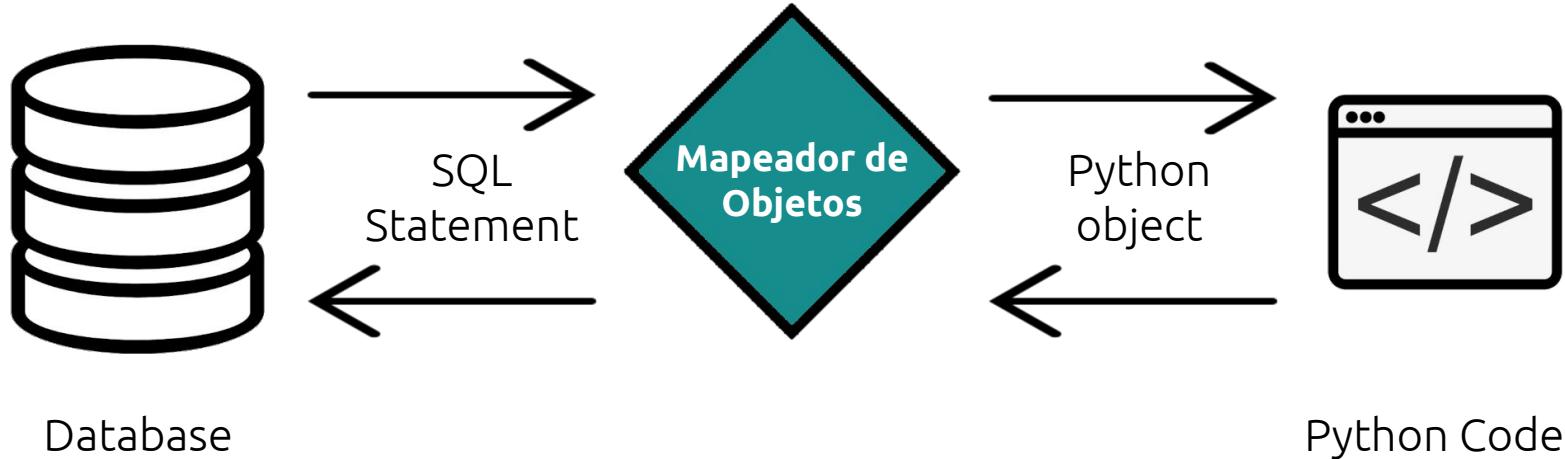
O problema do leque de opções...



As camadas de mapeamento vieram para resolver este problema.



As camadas de mapeamento vieram para resolver este problema



Object-Relational Mapping (ORM)

Object-Document Mapping (ODM)



Exemplo

```
● ● ●  
1 UPDATE Users  
2 SET firstName = 'João', lastName = 'da Silva'  
3 WHERE userId = 1
```



```
● ● ●  
1 Users.update(  
2   { firstName: "João", lastName: "da Silva" },  
3   { where: { userId: 1 } }  
4 );
```

Mapeadores



Produtividade

2

Motivação

1

Comparação

Como saber se a escolha das tecnologias foi assertiva e saber os impactos de uma opção em detrimento da outra?

+

Performance



-

Poder de processamento necessário



-

Custo



+

Lucro

2

Mercado

3

Implementação

Se os ORMs/ODMs estreitaram as diferenças entre o *SQL/NoSQL* e o paradigma de linguagem de programação a nível de sintaxe, quais as outras diferenças?

3

Objetivo

Objetivo

Analisar e comparar as diferenças entre bancos de dados SQL (*PostgreSQL*) e NoSQL (*MongoDB*), no que diz respeito a performance, produtividade, consistência, eficiência e usabilidade, empregados em uma aplicação próxima do mundo real com utilização de mapeadores de objetos.

4

Trabalhos relacionados

Trabalhos relacionados



1. *A performance comparison of document-oriented NoSQL databases* (YISHAN e MANOHARAN, 2017)



2. *Testing Spatial Data Deliverance in SQL and NoSQL Database Using Node.js Full Stack Web App* (LAKSONO, 2018)



3. Monitoramento de métricas de código-fonte em projetos de software livre (MEIRELLES, 2013)

5

Projeto e Desenvolvimento

Qual aplicação utilizar para a comparação?

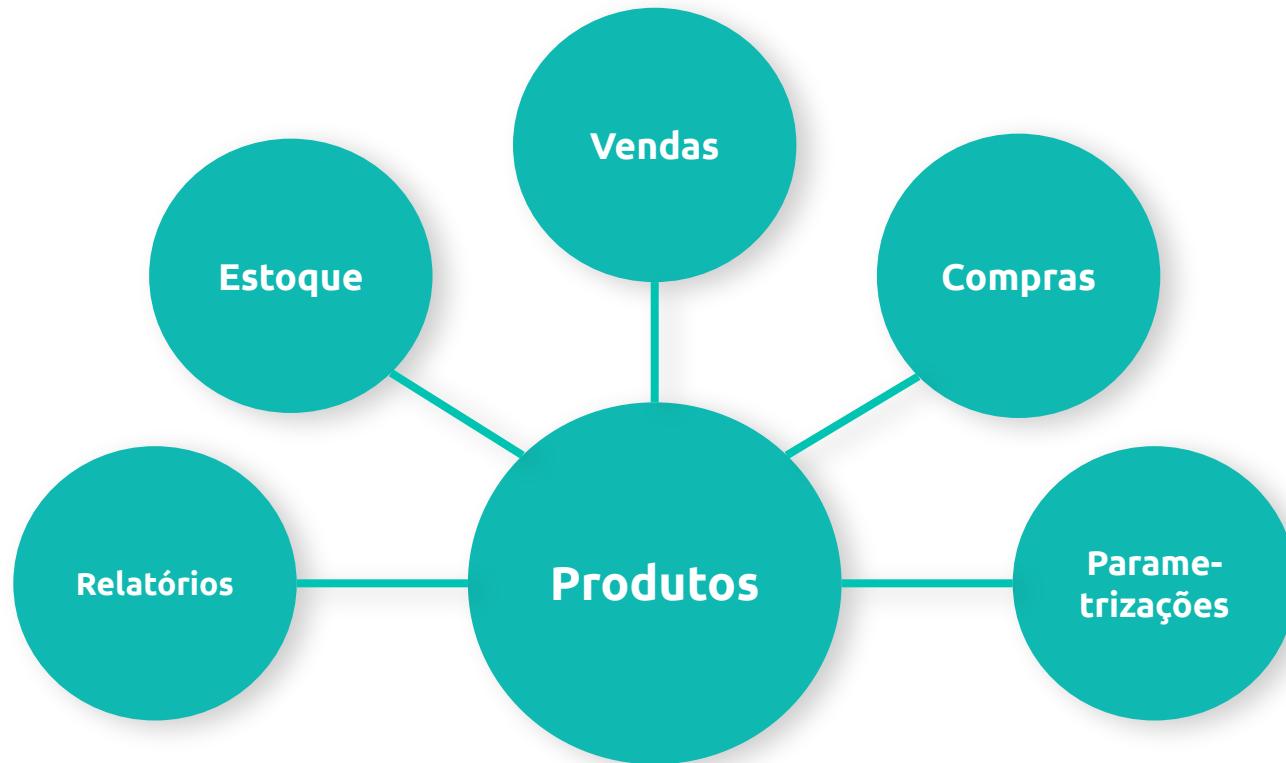
**Um software clássico
de computação**

ERP (WEB)

Enterprise Resource Planning



Núcleo do sistema

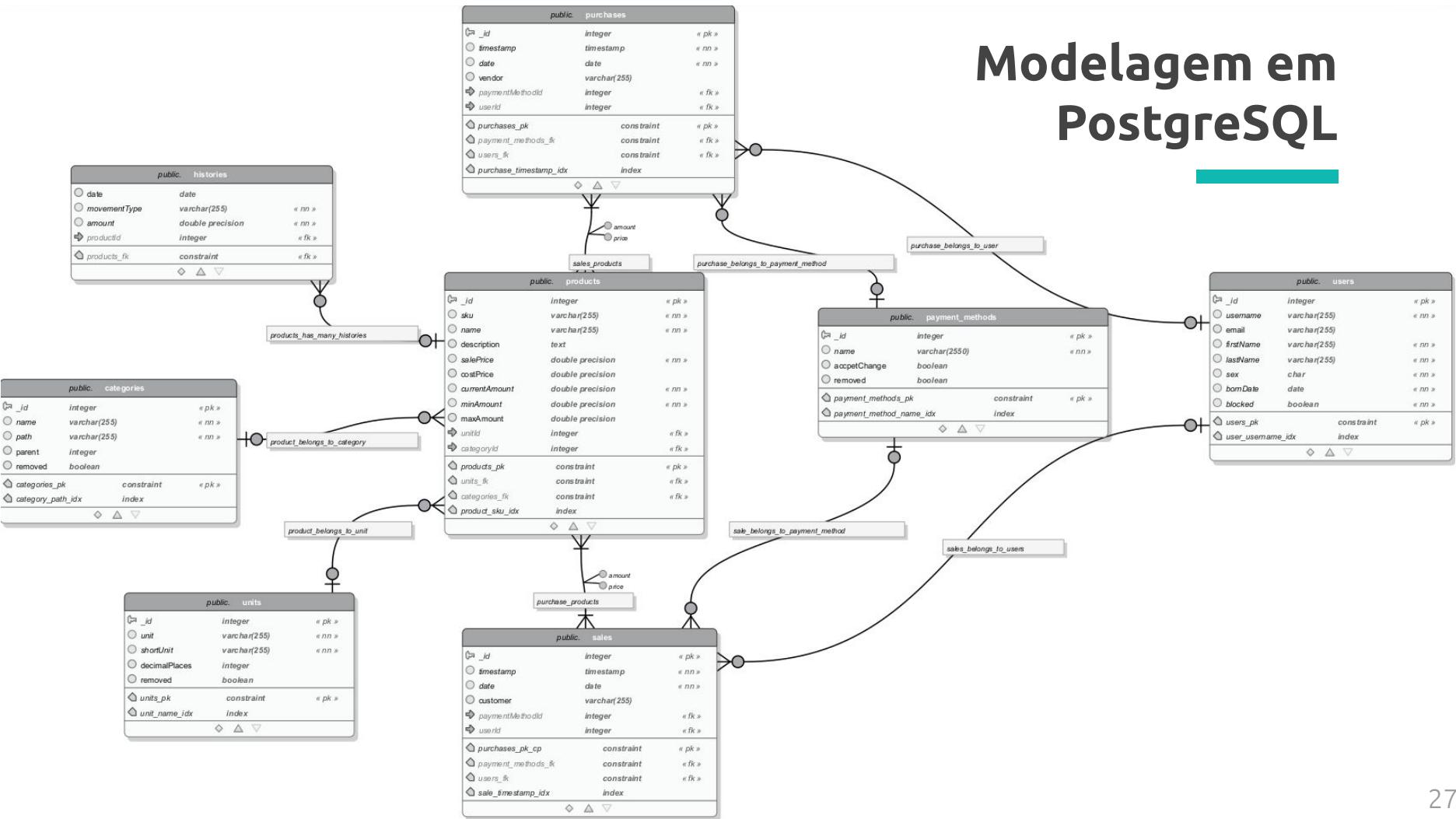


Principais funcionalidades

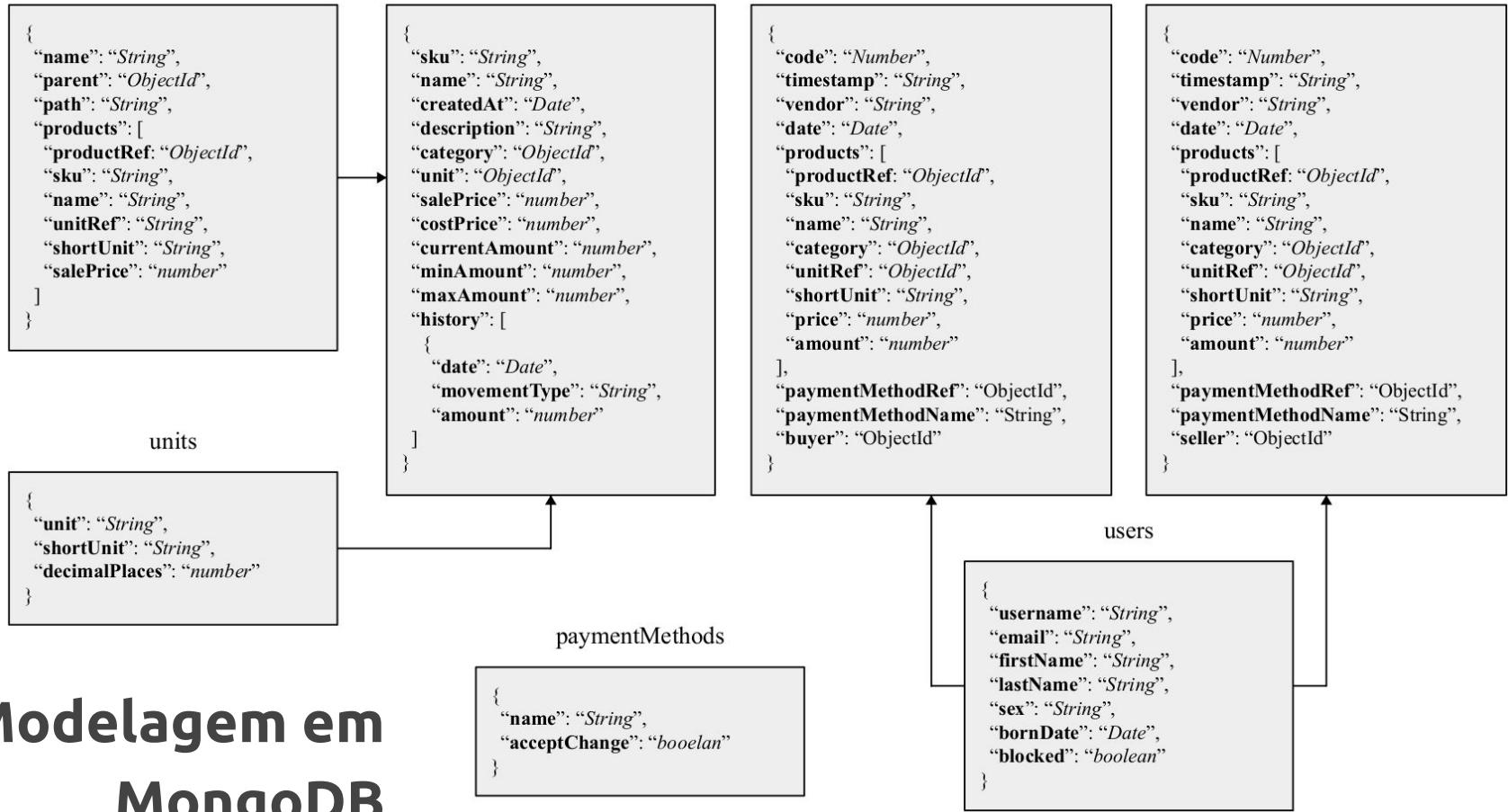
(em ordem de possível frequência de utilização)

1. Ponto de Vendas;
2. Compras e vendas por buscas;
3. Emissão de relatórios;
4. Consulta e atualização de estoque;
5. Consultas gerais;
6. Demais edições e remoções;

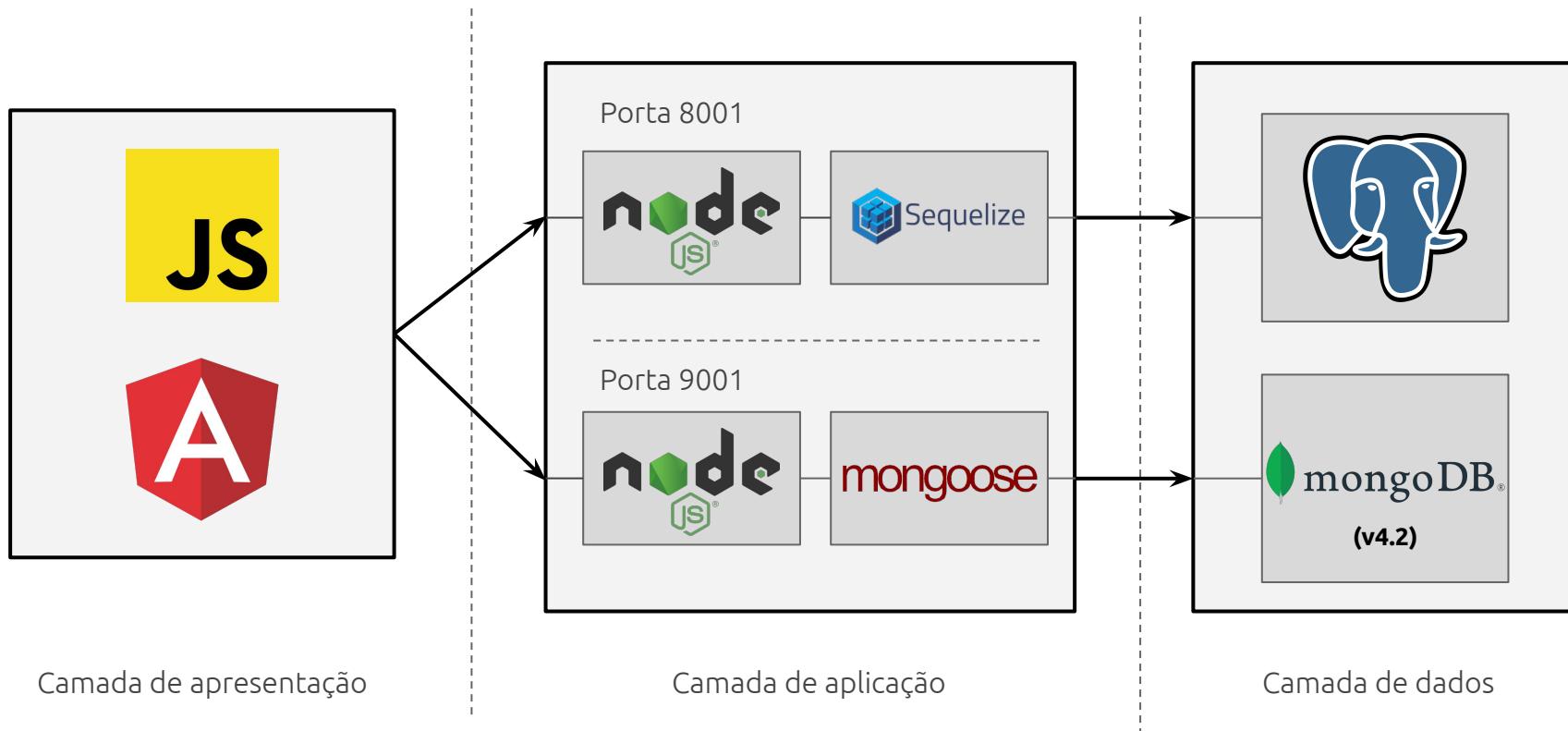
Modelagem em PostgreSQL



Modelagem em MongoDB



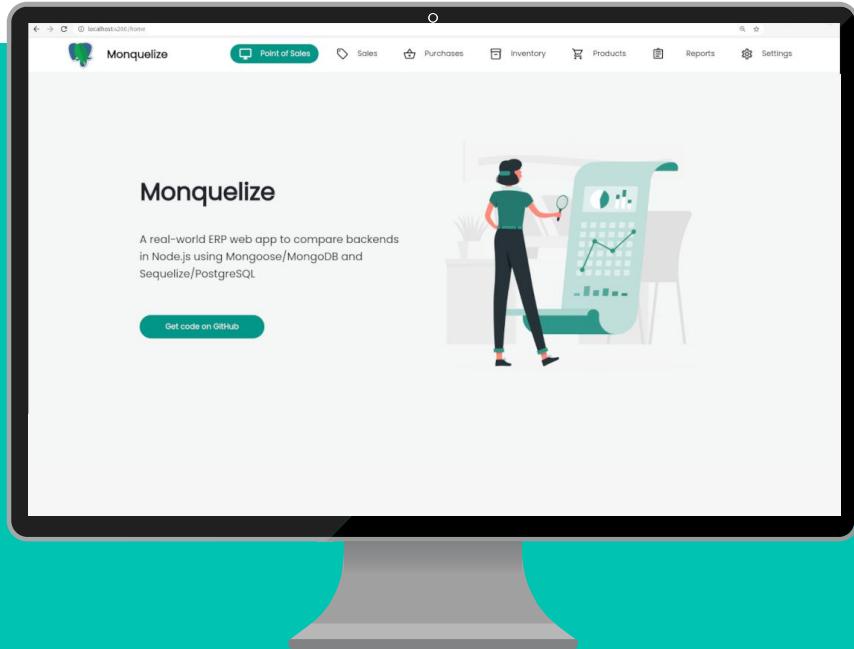
Arquitetura do Aplicação



Camada de apresentação

Camada de aplicação

Camada de dados



Monquelize

Um *real-world ERP web app* para comparar back-ends em Node.js e bancos de dados utilizando **Mongoose/MongoDB** e **Sequelize/PostgreSQL**



Cola branca/Marca A/P	▼ 1 ▲	\$4.50
Cola branca/Marca C/M	▼ 2 ▲	\$9.00

[Back](#) Cadernos Caixas Lapis Cor Caixas Tinta Acril. Canetas Colas Lapis Grafite IA Lapis Grafite IB Lapis Grafite IC Lapiseiras 0.5
Lapiseiras 0.7 Lapiseiras 0.9 Mochilas Tesouras

Cola branca/Marca A/P \$ 4.5	Cola branca/Marca A/M \$ 4.5	Cola branca/Marca A/G \$ 4.5	Cola branca/Marca B/P \$ 4.5	Cola branca/Marca B/M \$ 4.5	Cola branca/Marca B/G \$ 4.5	Cola branca/Marca C/P \$ 4.5
Cola branca/Marca C/M \$ 4.5	Cola branca/Marca C/G \$ 4.5	Cola branca/Marca D/P \$ 4.5	Cola branca/Marca D/M \$ 4.5	Cola branca/Marca D/G \$ 4.5	Cola branca/Marca E/P \$ 4.5	Cola branca/Marca E/M \$ 4.5
Cola branca/Marca E/G \$ 4.5	Cola branca/Marca F/P \$ 4.5	Cola branca/Marca F/M \$ 4.5	Cola branca/Marca F/G \$ 4.5	Cola branca/Marca G/P \$ 4.5	Cola branca/Marca G/M \$ 4.5	Cola branca/Marca G/G \$ 4.5
Cola branca/Marca H/P \$ 4.5	Cola branca/Marca H/M \$ 4.5	Cola branca/Marca H/G \$ 4.5	Cola branca/Marca I/P \$ 4.5	Cola branca/Marca I/M \$ 4.5	Cola branca/Marca I/G \$ 4.5	Cola branca/Marca J/P \$ 4.5

\$13.50

Cancel

Payment



New sale



(with Sequelize)

✓ 184.772857 ms



(with Mongoose)

✓ 226.199227 ms

New sale

\$ 19.5

Customer

Date *

3/24/2021



Product

Amount

Value

Subtotal

(1778D64D751) Caderno/Branco/Marca A/P

1

15

\$ 15



(1778D7C17D6) Cola branca/Marca A/M

1

4.5

\$ 4.5



Add product

Total: \$ 19.5

Payment method *

Débito

Seller



Reports >

Sales reports

Sales by date

Products by category

Advanced report



PostgreSQL

(with Sequelize)

✓ 7.050614 ms



mongoDB

(with Mongoose)

✓ 8.122076 ms

Sales by date

Filter by
Month

< January - 2021 >





Sales by date

Products by category

Advanced report



(with Sequelize)

✓ 5.453004 ms



(with Mongoose)

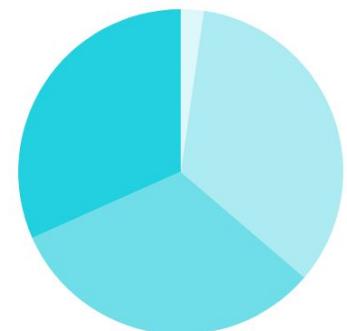
✓ 11.090317 ms

Products by category

Filter by
Month

March - 2021

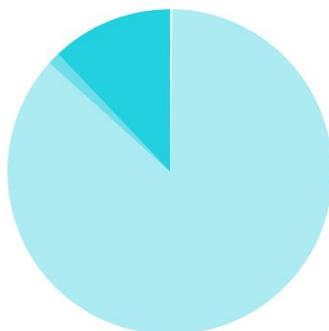
Amount



Categories

Colas Mochilas Canetas Cadernos

Total



Categories

Colas Mochilas Canetas Cadernos



Products

Time to count



PostgreSQL

(with Sequelize)

✓ 30.881758 ms



mongoDB

(with Mongoose)

✓ 2.29346 ms

Time to query



PostgreSQL

(with Sequelize)

✓ 7.934394 ms



mongoDB

(with Mongoose)

✓ 12.928174 ms

+ Add product

↻ Refresh

SKU	Name	Category	Unit	Sale price
I778D79A296	Mochila/Branco/Marca A/P	Mochilas	un	\$ 99.9
I778D79A297	Mochila/Branco/Marca A/M	Mochilas	un	\$ 99.9
I778D79A298	Mochila/Branco/Marca A/G	Mochilas	un	\$ 99.9
I778D79A299	Mochila/Branco/Marca B/P	Mochilas	un	\$ 99.9



Inventory



PostgreSQL

(with Sequelize)

✓ 13.200501 ms



mongoDB®

(with Mongoose)

✓ 17.768066 ms



Search

Caderno/Azul/Marca A

⟳ Refresh

SKU	Name	Category	Unit	Current Amount
I778D64D7CB	Caderno/Azul/Marca A/G	Cadernos	un	999996
I778D64D7CA	Caderno/Azul/Marca A/M	Cadernos	un	999970
I778D64D7C9	Caderno/Azul/Marca A/P	Cadernos	un	999987



Inventory >

Product inventory - Caderno/Azul/Marca A/G (1778D64D7CB)



(with Sequelize)

✓ 13.408814 ms



(with Mongoose)

✓ 7.54985 ms

↻ Refresh



999996

Current amount



-

Minimum amount



-

Maximum amount



\$14,999,940.00

Inventory value

History

+ Add inventory movement

↑ Inventory adjustment

Date	Movement type	Amount
24/03/2021 - 19:56	Input	1000000
24/03/2021 - 19:57	Output	1
24/03/2021 - 19:57	Output	2
24/03/2021 - 19:59	Output	1



Settings

Users

Categories

Units

Payment methods

6

Análise de resultados

1. Modelagem

- A modelagem em SQL é muito mais direta e simples.
- A modelagem do MongoDB requereu muito mais revisões quando comparada com PostgreSQL.
- Mudanças na modelagem do MongoDB impactaram mais mudanças nas implementações das regras de negócio.

O ORM (*Sequelize*) auxiliou bastante na criação das entidades

Definir as tabelas como objetos *JavaScript* (JSON) auxiliaram muito a implementação da modelagem.

```
14 const Product = sequelize.define('product',  
15   {  
16     _id: {  
17       type: DataTypes.INTEGER,  
18       autoIncrement: true,  
19       primaryKey: true,  
20     },  
21     sku: {  
22       type: DataTypes.STRING,  
23     },  
24     name: {  
25       type: DataTypes.STRING,  
26       allowNull: false,  
27     },  
28     description: {  
29       type: DataTypes.STRING,  
30     },  
31     salePrice: {  
32       type: DataTypes.FLOAT,  
33       allowNull: false,  
34     },  
35     costPrice: {  
36       type: DataTypes.FLOAT,  
37     },  
38     currentAmount: {  
39       type: DataTypes.FLOAT,  
40       allowNull: false,  
41     },  
42     minAmount: {  
43       type: DataTypes.FLOAT,  
44       allowNull: false,  
45       defaultValue: 0,  
46     },  
47     maxAmount: {  
48       type: DataTypes.FLOAT,  
49     },  
50     removed: {  
51       type: DataTypes.BOOLEAN,  
52       allowNull: false,  
53       defaultValue: false,  
54     },  
55   },  
56   {  
57     indexes: [  
58       {  
59         unique: true,  
60         fields: ['sku'],  
61       },  
62     ],  
63   }  
64 );
```

O MongoDB possui menos entidades



PostgreSQL

```
models
├── category.model.js
├── history.model.js
├── index.js
├── payment-method.model.js
├── product.model.js
├── purchase-product.model.js
├── purchase.model.js
├── sale-product.model.js
├── sale.model.js
└── unit.model.js
└── user.model.js
```



```
models
├── category.model.js
├── index.js
├── payment-method.model.js
├── product.model.js
├── purchase.model.js
├── sale.model.js
├── sequence.model.js
└── unit.model.js
└── user.model.js
```

2. Implementação

A edição de produtos no Postgres/Sequelize

```
 1  'use strict';
 2
 3  const { Product } = require('../models');
 4
 5  module.exports = async function editProduct(productId, product) {
 6    let [updatedProduct] = await Product.update(product, {
 7      where: { _id: productId },
 8      returning: true,
 9      plain: true,
10    });
11    updatedProduct = updatedProduct.dataValues;
12
13    return updatedProduct;
14  };
15
16
```

```
1  'use strict';
2
3  const { productModel } = require('../models');
4  const { editProductInSales } = require('../sale');
5  const { editProductInPurchases } = require('../purchase');
6  const {
7    editProduct: editProductInCategory,
8    addProduct: addProductInCategory,
9    removeProduct: removeProductInCategory,
10   } = require('../category');
11
12 module.exports = async function editProduct(productId, productData, session) {
13   const productDoc = await productModel.retrieve(productId, session);
14
15   const productObj = productDoc.toObject();
16
17   const updatedProductDoc = await productDoc.edit(productData);
18
19   await updatedProductDoc
20     .populate([
21       { path: 'category', select: 'name' },
22       { path: 'unit', select: ['unit', 'shortUnit'] },
23     ])
24     .execPopulate();
25
26   if (hasToUpdateInSalesOrPurchases(productObj, updatedProductDoc)) {
27     await editProductInPurchases(updatedProductDoc._id, updatedProductDoc, session);
28     await editProductInSales(updatedProductDoc._id, updatedProductDoc, session);
29   }
30
31   await updateProductInCategory(productObj, updatedProductDoc, session);
32
33   return updatedProductDoc;
34 };
35
36 function hasToUpdateInSalesOrPurchases(oldProduct, newProduct) {
37   const relevantFields = ['sku', 'name', 'category', 'unit'];
38
39   return relevantFields.some((field) => oldProduct[field] !== newProduct[field]);
40 }
41
42 async function updateProductInCategory(oldProduct, newProduct, session) {
43   if (oldProduct.category.equals(newProduct.category._id)) {
44     await editProductInCategory(oldProduct.category, newProduct, session);
45   } else {
46     await addProductInCategory(newProduct.category.id, newProduct, session);
47     await removeProductInCategory(oldProduct.category, oldProduct._id, session);
48   }
49 }
```

A mesma operação no MongoDB/Mongoose

A implementação do sistema em MongoDB/Mongoose foi consideravelmente mais complexa.

- Muitas operações no MongoDB utilizam transactions, que são tratadas manualmente pelo programador.
- Ter dados replicados para melhorar a eficiência pode ser arriscado se não houverem ferramentas extras para contornar este problema.

3. Consistência

4. Performance

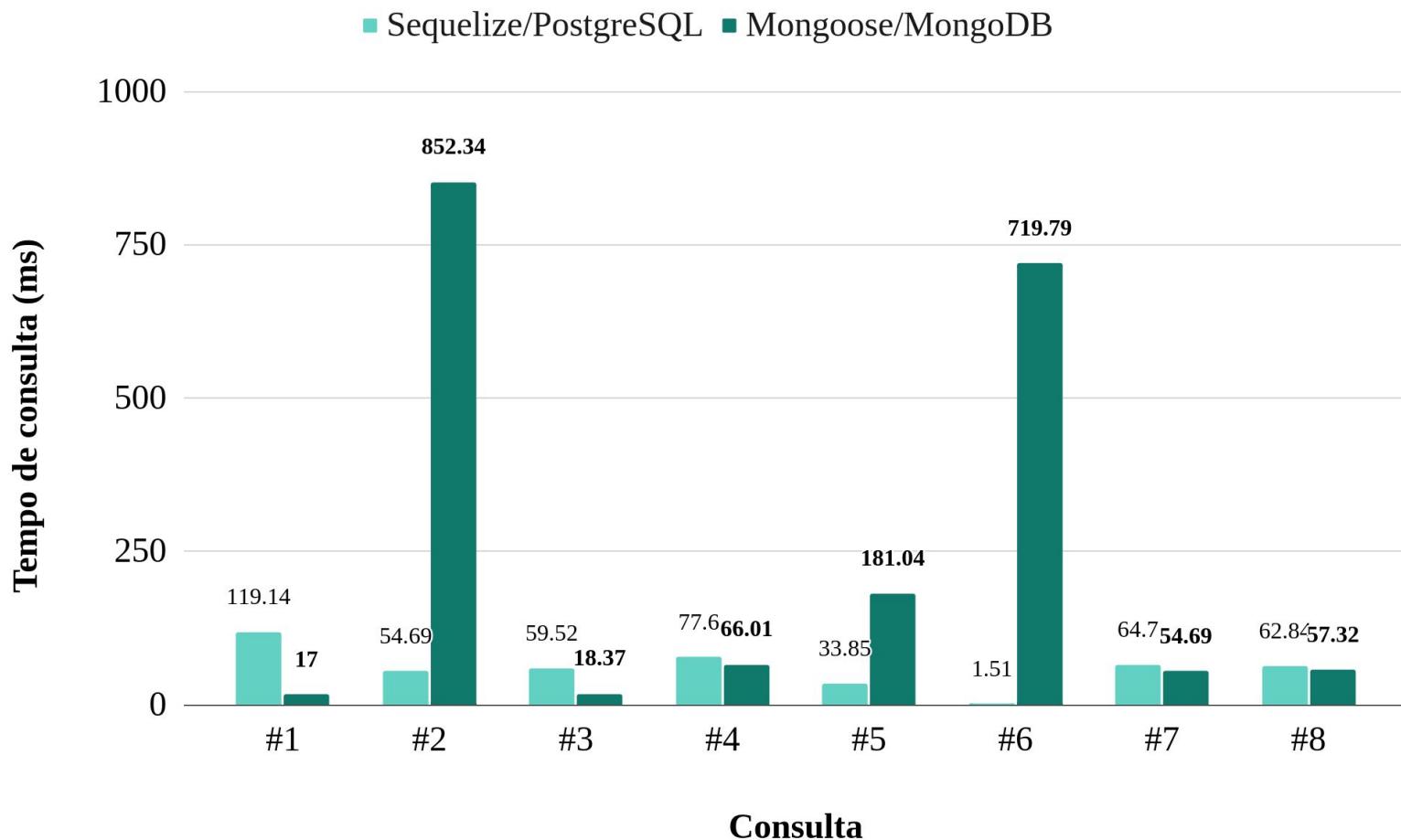
Carga de dados

- 12 usuários, que atuam no sistema como vendedores e compradores;
- 14 categorias de produtos;
- 2 unidades de medida;
- 6 formas de pagamento;
- 3.189 produtos, pertencentes as categorias cadastradas;
- 345.983 vendas, distribuídas ao longo de 12 meses (média de 100 vendas diárias);

Consultas realizadas nos testes

Nota: Cada consulta foi repetida 10 vezes em ambiente controlado, e o valor final do tempo corresponde à média entre os valores

1. Consulta de produtos por categoria, para realização de vendas na frente de caixa;
2. Consulta de vendas por mês;
3. Consulta de produtos, utilizando paginação *server-side*;
4. Inserção de novas vendas;
5. Edição de produtos;
6. Remoção de produto;
7. Relatório de vendas por mês, retornando somatório do total vendido e quantidade de vendas;
8. Relatório de vendas por mês distribuídas por categorias de produtos, retornando total vendido e quantidade de vendas por categoria



O *MongoDB* foi pouco melhor quando era esperado que realmente o fosse e muito pior quando também era previsto tal desempenho.

5. Usabilidade

Somente haverá impactos para o usuário final se o volume de dados trabalhados for grande.

Por mais que os tempos relativos às performance de consultas sejam bastante diferentes, o usuário final só irá perceber essa diferença quando as consultas exigirem mais dados retornados.

Conclusão e contribuições

Conclusão

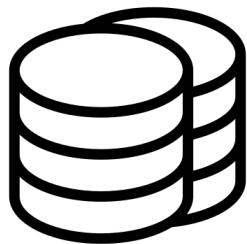
A partir da conclusão do projeto, foi possível perceber e mensurar qualitativamente as diferenças entre as camadas de persistência.

Se for comparado o esforço e custo de implementação *versus* o ganho de performance do MongoDB, os resultados não foram satisfatórios.

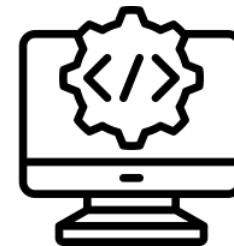
As duas abordagens (relacional e não relacional) podem ser utilizadas em conjunto para extrair o melhor dos dois mundos.

Para novas aplicações, os objetivos do projeto podem impactar diretamente nas escolhas das tecnologias.

Contribuições

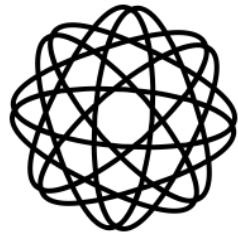


Bancos de dados

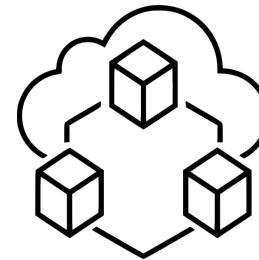


Engenharia de Software (Arquitetura)

Trabalhos futuros



Incrementar complexidade de relação entre as entidade



Incrementar o volume de dados para testes

Referencias

- ◆ Anderson, B. and Nicholson, B. (2020). Sql vs. nosql databases: What's the difference?
- ◆ Date, C. J. (2004). *Introdução a sistemas de bancos de dados*. Editora Campus, 8 edition.
- ◆ Diogo, M., Cabral, B., and Bernardino, J. (2019). Consistency models of nosql databases. *Future Internet*, 11(2).
- ◆ Elmasri, R. and Navathe, S. B. (2011). *Sistemas de Bancos de Dados*. Addison-Wesley, 7 edition.
- ◆ Fonseca, E. (2019). O que é orm?
- ◆ Jatana, N., Puri, S., Ahuja, M., Kathuria, I., and Gosain, D. (2012). A survey and comparison of relational and non-relational database. *International Journal of Engineering Research Technology (IJERT)*, 6:6.
- ◆ Kumar, K. and Azad, S. K. (2017). Database normalization design pattern. In *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, pages 318–322.
- ◆ Laksono, D. (2018). Testing spatial data deliverance in sql and nosql database using nodejs fullstack web app.
- ◆ Lemos, M. F., de Oliveira Leandro César Ruela, P. C., da Silva Santos, M., and Silveira, T. C. (2013). Aplicabilidade da arquitetura mvc em uma aplicação web (webapps). *Revista Eletrônica Científica de Ciência da Computação*, 8(1).
- ◆ Meirelles, P. R. M. (2013). Monitoramento de métricas de código-fonte em projetos de software livre.
- ◆ Wu, X. (2019). Metadata-based image collecting and databasing for sharing and analysis.
- ◆ Yishan, L. and Manoharan, S. (2017). A performance comparison of sql and nosql databases.

Perguntas