Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Eduardo Henrique Geraldo

PyMongo com MongoDB, uma opção aos SGBDS relacionais

Americana, SP

2022

Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Eduardo Henrique Geraldo

PyMongo com Mongo DB, UMA OPÇÃO ATUAL AOS SGBDS RELACIONAIS

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Analise e desenvolvimento de sistemas, sob a orientação do Prof. Thiago Salhab Alves.

Área de concentração: Banco de dados.

Americana, SP

2022

Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Eduardo Henrique Geraldo

PyMongo com MongoDB, uma opção aos SGBDs Relacionais

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Curso Superior de Tecnologia em Análise e Desenvolvimento De Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana – Ralph Biasi.
Área de concentração: banco de dados

Americana, 01 de junho de 2022

Banca Examinadora:

Thiago Salhab Alves (Presidente)
Mestre
Faculdade de Tecnologia de Americana

Rodrigo Brito Battilana (Membro)
Mestre
Faculdade de Tecnologia de Americana

Clerivaldo Jose Roccia (Membro)
Mestre
Faculdade de Tecnologia de Americana

RESUMO

Hoje em dia devido a ampliação de informações geradas o usuário defronta com a problemática de como será o armazenamento eletrônico. A criação dos bancos de dados relacionais foi uma revolução, pois permitiu o armazenamento de maneira acessível e rápida, porém as limitações são sensíveis a ligeira ampliação do número de dados armazenados, os bancos de dados relacionais vêm perdendo sua autonomia no cenário das necessidades atual. Visando atender essa crescente demanda, surgiram os bancos de dados NoSQL, os quais vem conquistando cada vez mais as aplicações modernas. Este trabalho tem como objetivo demostrar a utilização do Mongo BD combinado com linguagem Python como uma alternativa aos bancos de dados relacionais. O mongo DB é um software de banco de dados orientado a documentos livres, que possui código aberto e é multiplataforma. O MongoDB é escrito em linguagem C++ e é classificado como um programa de bancos de dados NoSQL, que utiliza documentos semelhantes ao JSON. A sua utilização dispensa as estruturas dos bancos de dados relacionais e faz uso de esquemas dinâmicos, tornando dessa forma a integração de dados em determinados tipos de aplicativos mais fácil e rápida. Na sequência, este trabalho demonstrará através de um estudo como a utilização do MongoDB associado a linguagem Python pode ser a sucessão dos bancos de dados relacionais.

Palavras chave: Mongo DB, PyMongo, Bancos de dados relacionais, NoSQL, JSON.

ABSTRACT

Nowadays, due to the expansion of information generated, we are faced with the problem of how we continue with electronic storage. The creation of relational databases was a revolution, as it allowed storage in an accessible and fast way, but the limitations are sensitive to the slight increase in the number of stored data, relational databases have been losing their autonomy in the scenario of current needs. In order to meet this growing demand, NoSQL databases emerged, which are increasingly conquering modern applications. This work aims to demonstrate the use of Mongo BD combined with Python language as an alternative to relational databases. mongo DB is a free, open-source, cross-platform, document-oriented database software. MongoDB is written in C++ language and is classified as a NoSQL database program, which uses documents similar to JSON. Its use dispenses with the structures of relational databases and makes use of dynamic schemas, thus making the integration of data in certain types of applications easier and faster. Subsequently, this work will demonstrate through a study how the use of MongoDB associated with the Python language can be the succession of relational databases.

Keywords: Mongo DB, PyMongo, Relational databases, NoSQL, JSON.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A alta demanda por informações cresce de forma rápida, e necessária; desta maneira o armazenamento mais utilizado é através dos bancos de dados relacionais, de acordo Dante (2004) esta enorme evolução nos dias atuais e são cada vez menos eficientes, já que esses bancos de dados são baseados em estruturas computacionais que utilizam da troca ou da adição de processadores e adição de memória.

Dessa maneira, as grandes instituições desfrutam da possibilidade de escolha de como será o direcionamento dos processos de armazenamento e processamento de dados. De acordo com Sadalage (2013), os bancos de dados NoSQL têm ganhado destaque. Os bancos de dados NoSQL são projetados em arquiteturas distribuídas, permitindo então que eles possam processar os grandes volumes de dados que possuem, com disponibilidade e escalabilidade.

Há uma variedade de sistemas gerenciadores de bancos de dados, são os chamados de SGBD. Neste trabalho, abordaremos os SGBD não relacionais, os quais existem diferentes tipos e cada um tem as suas particularidades. Por exemplo, existem SGBDs não relacionais do tipo orientado a documentos, chave-valor, colunas e grafos. O MongoDB que será abordado nesse estudo se encaixa nos SGBDs orientados a documentos e atualmente seu uso está em constante crescimento no mercado segundo NAVATHE (2005).

Esse trabalho tem como objetivo principal analisar a possibilidade de alternativa dos bancos de dados relacionais pelo Mongo DB associado a biblioteca Python (PyMongo), como objetivos específicos conceituar o Mongo DB, a biblioteca PyMongo, demonstrar a utilização dos bancos de dados relacionais e verificar as vantagens e as desvantagens desta sequência Mongo DB com PyMongo em relação aos bancos de dados relacionais.

A metodologia utilizada para a realização desse trabalho é a pesquisa bibliográfica, que de acordo com Fachim (2010), a pesquisa bibliográfica é a mais importante no trajeto da pesquisa e que constitui o ato de ler, selecionar, fichar, organizar e compreender.

O trabalho foi estruturado em três capítulos. O primeiro capítulo conceituará Banco de Dados, o Mongo DB e a biblioteca PyMongo. O segundo capítulo demonstrará a utilização dos bancos de dados relacionais, as vantagens e desvantagens na utilização do mongo DB com PyMongo e a possibilidade de substituição dos bancos de dados relacionais pelo Mongo DB com PyMongo. Baseando-se nas informações a partir do estudo realizado, o terceiro e último capítulo será utilizado para as considerações finais sobre o trabalho que foi feito.

2. DESENVOLVIMENTO

2.1. Banco de dados

Banco de dados pode ser definido como uma organização de arquivos ou coleção organizada de informações ou dados que são armazenados de forma eletrônica em um sistema computacional. De acordo Korth (2004, p.6), "um banco de dados" é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico", ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados".

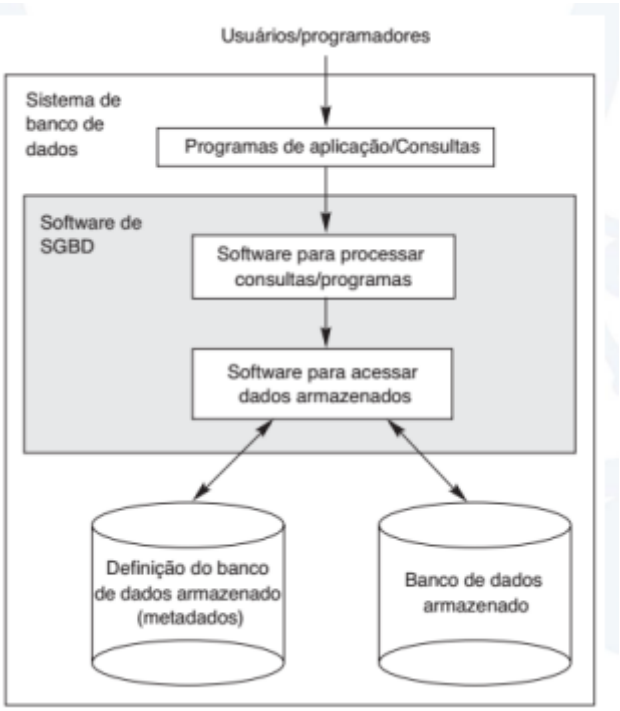
Assim, os bancos de dados podem ser definidos como dispositivos capaz de armazenar todos os tipos de dados, seguindo os conceitos Heuser (2009) podendo ser utilizados das mais variadas atividades cotidianas e necessidades, atendendo usuários de vários níveis possibilitando interações com o mundo real e com todos os colaboradores deste conteúdo. Os componentes básicos de um banco de dados são:

- O próprio banco de dados;
- Dados;

- O software gerenciador do banco de dados.

A figura 1 apresenta um esquema de como é a composição de um banco de dados.

Figura 1 – Modelo de sistema de Bancos de Dados



Fonte: Date (2004)

Uma outra definição de banco de dados que podemos citar, de acordo com Navathe (2005) é:

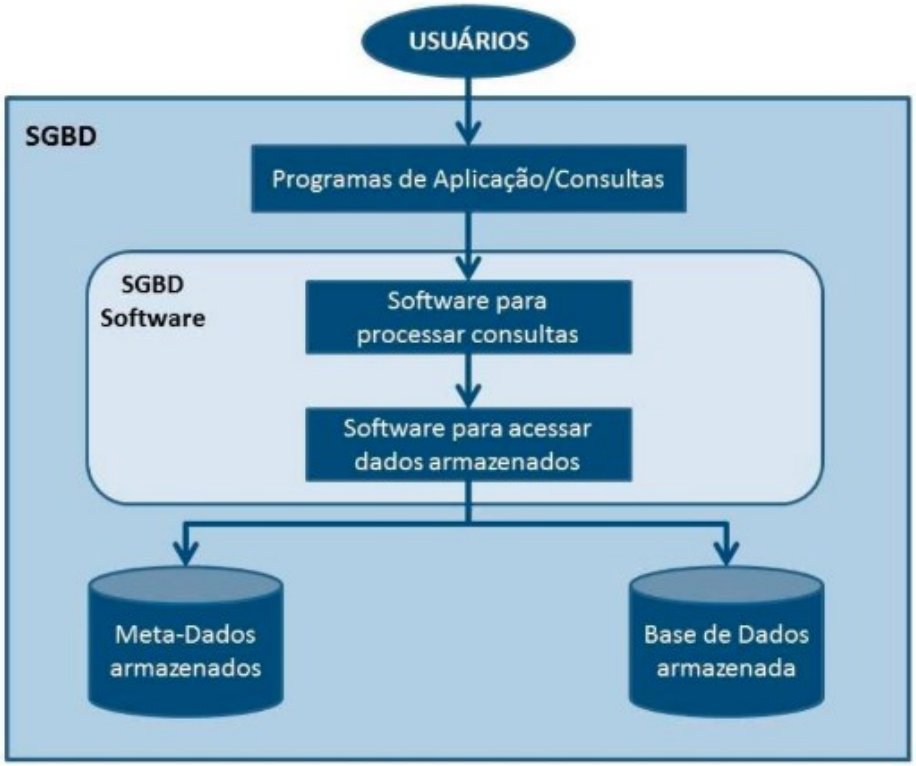
Um banco de dados representa um determinado aspecto do mundo real, chamado de minimundo ou de universo de discurso. As mudanças no minimundo são refletidas no Banco de Dados. Um banco de dados é uma coleção logicamente coerente de dados com um significado inerente. Uma variedade aleatória dos dados não pode ser chamada de bancos de dados. Um banco de dados geralmente é projetado, construído com dados para uma finalidade

especifica. Ele possui um grupo definido de usuários e possui algumas aplicações previamente concebidas nas quais os usuários estão interessados. (ELMASRI E NAVATHE, 2011, P.3).

2.1.1. Sistema Gerenciador de Banco de Dados

A gestão dos bancos de dados é feita através do Sistema Gerenciador de Banco de Dados (SGBD), que tem como principal objetivo proporcionar um ambiente eficiente para a recuperação e para o armazenamento das informações do banco de dados. (Adaptado de SILBERSCHATZ; KORTH; SUDARSHAN, 1999, P.1).

Figura 2 - Sistema Gerenciador de Bancos de Dados



Fonte: Emerson Fender (2012, p.2)

O Sistema Gerenciador de Banco de Dados (SGBD) ou em inglês *Data Base Management System* (DBMS) é o sistema de software responsável pelo

gerenciamento de um ou mais bancos de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados (Emerson Fender, 2012).

O Sistema Gerenciador de Banco de Dados disponibiliza os recursos para definição, construção, manipulação, compartilhamento, proteção e para manter o banco de dados. Portanto, ele pode ser entendido como um conjunto de programas responsáveis pelo gerenciamento do banco de dados, tendo em vista que o mesmo fornece suporte aos métodos de acesso, ou seja, as formas de armazenar e de recuperar as informações de forma apropriada. (Adaptado de NAVATHE, ELMASRI; 2005).

2.1.2. Modelo de Dados

O Modelo de Dados é um conjunto de conceitos que podem ser usados para descrever a estrutura de uma base de dados. Por estrutura de uma base de dados entende-se os tipos de dados, relacionamentos e restrições pertinentes aos dados. Então, compreendemos o modelo seria a maneira de descrever os seus dados, relacionamentos, semântica e restrições de consistência. O modelo de dados demonstra o projeto do banco de dados em sua forma física, lógica e de visão. (SILBERSCHATZ, 2006). Todos os sistemas gerenciadores de bancos de dados seguem um modelo de banco de dados para diferenciar estes dados de acordo com a sua representação.

2.1.3. Modelo Relacional

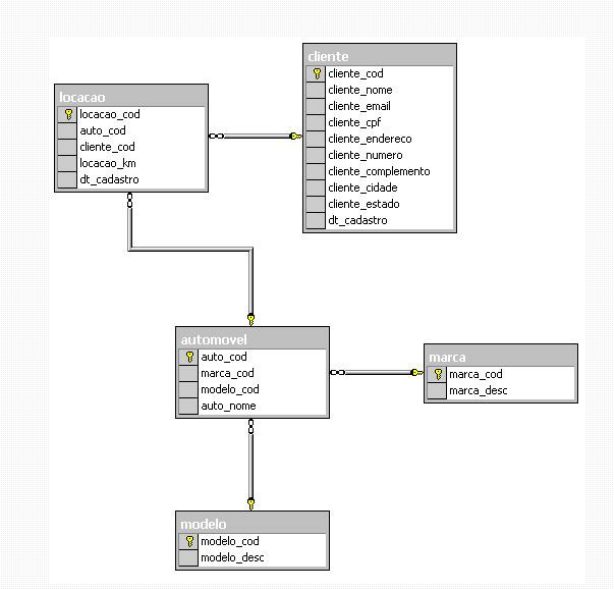
No modelo relacional as informações em uma base de dados podem ser consideradas como relações matemáticas e que podem ser representadas, de maneira uniforme, através do uso de tabelas onde as linhas representam as

ocorrências de uma entidade e as colunas representam os atributos de uma entidade do modelo conceitual (Freitas, C.M, 2015).

As relações no modelo relacional são conjuntos de dados vistos como tabelas cujas operações são baseadas na álgebra relacional (projeção, produto cartesiano, seleção, junção, união e subtração) e que manipulam conjuntos de dados ao invés de um único registro, isto é, cada operação realizada afeta um conjunto de linhas e não apenas uma única linha, ainda que algumas operações possam afetar uma única linha (conjunto com um único elemento). O modelo relacional representa os dados como um conjunto de relações que é similar a uma tabela de valores, na qual as linhas são denominadas tuplas, as colunas são denominadas de atributos e a tabela é denominada de relação. (Adaptado de ELMASRI; NAVATHE, 2005).

As tabelas são constituídas de vários elementos os quais possuem características comuns, conforme vemos na figura 3. Esses elementos são denominados de "atributos". Pode-se verificar na figura 3 o conceito de atributo, sendo que cada um destes atributos indica características que estão associadas a tabela. Os seguintes atributos estão presentes no exemplo: *id, description, data, type, unit, variable, type e symbol*. Além disso, temos o "domínio", que são os possíveis valores dos atributos, e as tuplas, que indicam as linhas existentes na relação.

Figura 3 - Modelo Relacional



Fonte: Próprio Autor (2022)

2.1.4. NoSQL

O modelo NoSQL foi utilizado pela primeira vez no ano de 1998, quando Carlo Strozzi baseando-se no modelo relacional, desenvolveu um modelo relacional sem interface SQL, criando um termo genérico para uma classe de banco de dados que rompe os paradigmas dos bancos de dados baseados nos modelos relacionais (SUISSA, 2010). É uma maneira de promover soluções de armazenamento de dados não relacionais; há diversas ferramentas que possibilitam tratamento de grandes volumes de dados.

A grande maioria dos bancos de dados NoSQL foram criados baseados nas necessidades de atenderem ao desenvolvimento de *websites*. Eles foram projetados para promoção de alternativas de armazenamento com velocidades e disponibilidade elevadas.

A figura 4 abaixo mostra algumas referências de banco de dados não relacionais:

Figura 4 - NoSQL

Fonte: Adaptado de Juan Benitez (2012, p .1)

Seguindo a classificação sugerida por Juan Benitez, os bancos de dados NoSQL são classificados conforme a sua maneira de armazenamento de dados, sendo eles:

- **Chave-valor:** armazena dados como um conjunto de pares de chave-valor em que uma chave funciona como um identificador único.

  Alguns bancos que utilizam esse padrão: DynamoDb, Couchbase, Riak, Azure Table Storage, Redis, Tokyo Cabinet, Berkeley DB.

- **Orientado a colunas:** este modelo armazena todo o seu conteúdo de maneira inversa aos bancos de dados que são orientados a linhas;

  Alguns bancos que utilizam esse padrão: Hadoop, Cassanda, Hypertable, Amazon SimpleDB.

- **Grafos:** é fundamentado na teoria dos grafos que consiste em nós, propriedades e arestas; o relacionamento, que permite que eles liguem documentos para acesso rápido.

- **Orientado a documento**: desenvolvidos para armazenar, consultar e gerenciar dados orientadas a documentos, também conhecidas como dados semiestruturados; este modelo é baseado no armazenamento de dados dentro de documentos que são semelhantes a JSON.

Alguns bancos que utilizam esse padrão: MongoDb, CouchDB, RavenDb. Uma importante característica deste modelo é armazenamento e recuperação dados semiestruturados. Em destaque o MongoDB um banco NoSQL; cujo modelo orientado a documentos permite à recuperação de documentos através de uma API ou de uma linguagem de consulta que é disponibilizada pelo banco de dados. Todos os bancos de dados atuais têm a capacidade de fornecer suporte aos documentos que estão em formato JSON (Adaptado de LEAVITT, 2010).

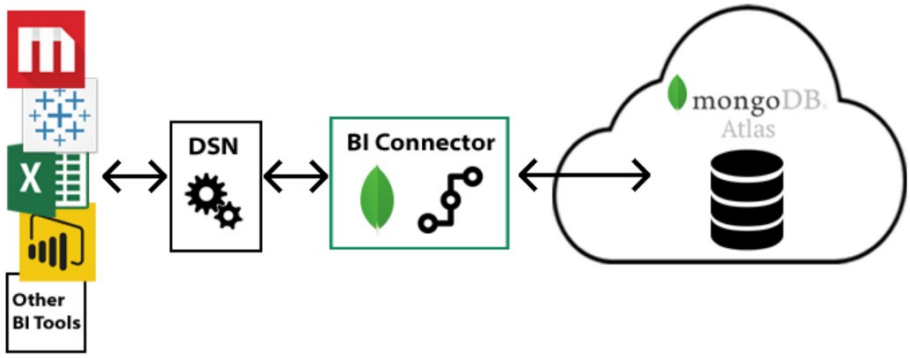
2.1.5. MONGODB

O MongoDB foi desenvolvido como um componente de serviço, pela empresa chamada de 10gen, em outubro do ano de 2007 e em 2009 passou a ser um software open source ou um document store onde não é reconhecido um esquema relacional; os documentos semelhantes ao formato JSON (Java Script Object Notation) para armazenar dados. O documento é semelhante a um registro com campos e valores (Adaptado de Juan Benitez 2012).

O nome MongoDB vem do inglês *humongous*, que significa gigantesco. O MongoDB fornece recursos para ambientes de produção como balanceamento de cargas, replicação, consultas a sistemas de arquivos e conectividade com diversos softwares como observamos na figura 5 abaixo:

A figura 5 ilustra algumas funcionalidades do MongoDB, como por exemplo a conexão com o serviço de computação em nuvem, sincronia com dispositivos móveis e a criação de funções e gatilhos para automações.

Figura 5 - MongoDB



Fonte: Adaptado segundo Anusha Dharmalingam and Igor Alekseev (2020)

Uma das principais característica do MongoDB é o armazenamento de dados quando comparado com bancos relacionais; sendo realizado em tabelas estruturas rígidas. (CUNHA, 2011).

As unidades de dados no MongoDB são os documentos tais quais comparáveis a estrutura JSON, no entanto no MongoDB são denominados BSON, sigla em inglês (Binary encoded Javascript Object Notation); devido à maneira como os dados são armazenados; é possível fazer a analogia de que o MongoDB é considerado um banco de dados orientado a documentos; sendo esta é uma das características mais marcantes nos bancos NoSQL, a forma simples como as consultas são executadas têm alta eficiência (LEAVITT, 2010).

Há possibilidade de gerenciamento ou otimização do MongoDB sendo realizado por ferramentas extras que acopladas ao MongoDB podem transformar o banco em customização; é o caso do "MongoDB Compass"; que realiza consultas, análises de dados, criação de pipelines, fornece metadados detalhados a partir de uma coleta de uma amostra.

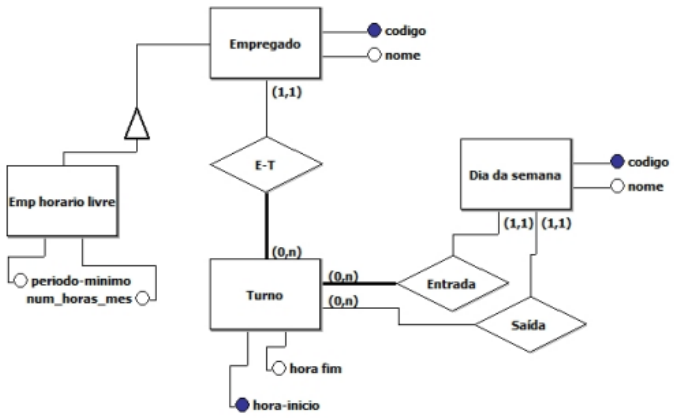
2.2. BANCO DE DADOS RELACIONAL

Visando reduzir a demanda por colaboradores executando armazenamento e organização de arquivos, a indústria ampliava esforços e investimentos em pesquisar a mecanização eficiente de processos. Por volta de 1970, Ted Codd pesquisador da IBM fazia as primeiras publicações sobre o modelo de banco de dados relacionais; neste artigo destacaria o uso de cálculo e álgebra para permitiriam armazenamento e a recuperação de grande quantidade de dados; as informações estariam armazenadas em tabelas, em torno de 1980 houve a implementação do modelo segundo MPINDA (2011).

O modelo relacional é o mais flexível e mais adequado para solucionar diversos tipos de problemas a níveis de concepção e de implementação de base de dados. A estrutura fundamental do modelo relacional é a relação e essa relação é construída por um ou mais atributos, que são os campos. Os campos têm a função de traduzir o tipo dos dados e de armazená-los LÓSCIO (2011).

A figura 6 demonstra um exemplo da estrutura de um banco de dados relacional:

Figura 6 – Modelo Banco de dados relacional



Fonte: Adaptado de Carlos A. Heuser (2009)

Pelo exemplo descrito acima observa a presença dos atributos; que desempenham papel de informar e descrever uma relação. Uma relação deve ter um atributo ou combinação de atributos que identificam cada ocorrência. Estes atributos formam a chave primária da relação. No modelo relacional as implementações das estruturas de dados são organizadas de forma que haja relações coerentes entre as tabelas e essas relações são definidas pela cardinalidade. Como é mostrado na figura 6, a cardinalidade define se uma tabela pode ter um, vários ou nenhum registro na tabela ligada a ela para cada registro na própria tabela. Para trabalhar com tabelas foram desenvolvidas algumas restrições para evitar aspectos indesejáveis no modelo relacional, tais como:

- Repetição de informações;
- Incapacidade de representação das informações e
- Perda de informações.

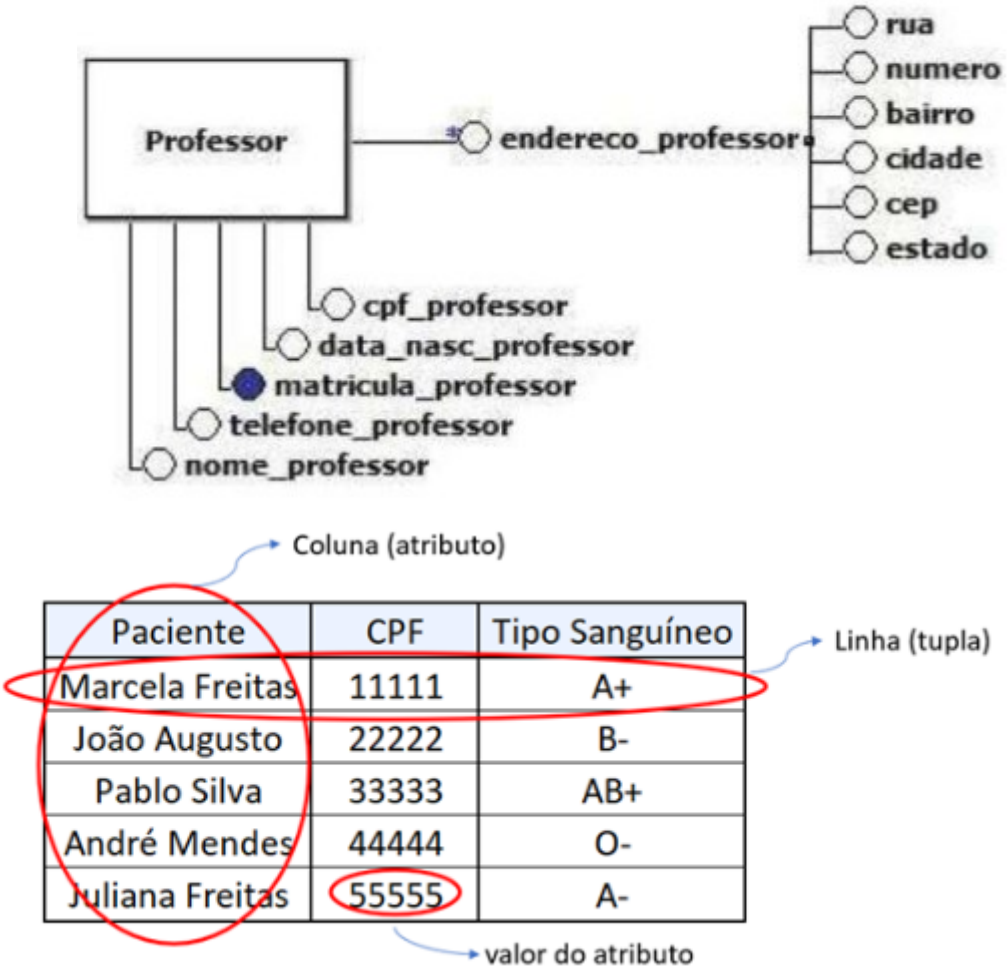
Estas restrições são:

- Integridade referencial;
- Chaves;
- Integridade de junção destas relações.

Seguindo a adaptação de Elmasri e Navathe (2005), o modelo relacional representa o banco de dados como uma coleção das relações, assim cada uma dessas relações se assemelha a uma tabela de valores.

Portanto, pode-se definir um modelo de dados como sendo uma combinação de um conjunto de estruturas de dados; da tabela ou relação; conjunto de operadores que são responsáveis pela manipulação do banco de dados e regras de integridade que são definidas no esquema do banco de dados. Essas regras são aplicadas nas instâncias dos bancos de dados.

Figura 7 – Modelo Banco de dados com representação de atributos



Paciente	CPF	Tipo Sanguíneo
Marcela Freitas	11111	A+
João Augusto	22222	B-
Pablo Silva	33333	AB+
André Mendes	44444	O-
Juliana Freitas	55555	A-

Fonte: Autoria própria (2022)

No modelo relacional, cada linha é denominada de tupla, os títulos das colunas são os atributos e a tabela é chamada de relação. Os tipos de dados descrevem os tipos de valores que podem aparecer em cada uma das colunas e são denominados de domínio.

2.2.1. Chaves Primárias e Estrangeira

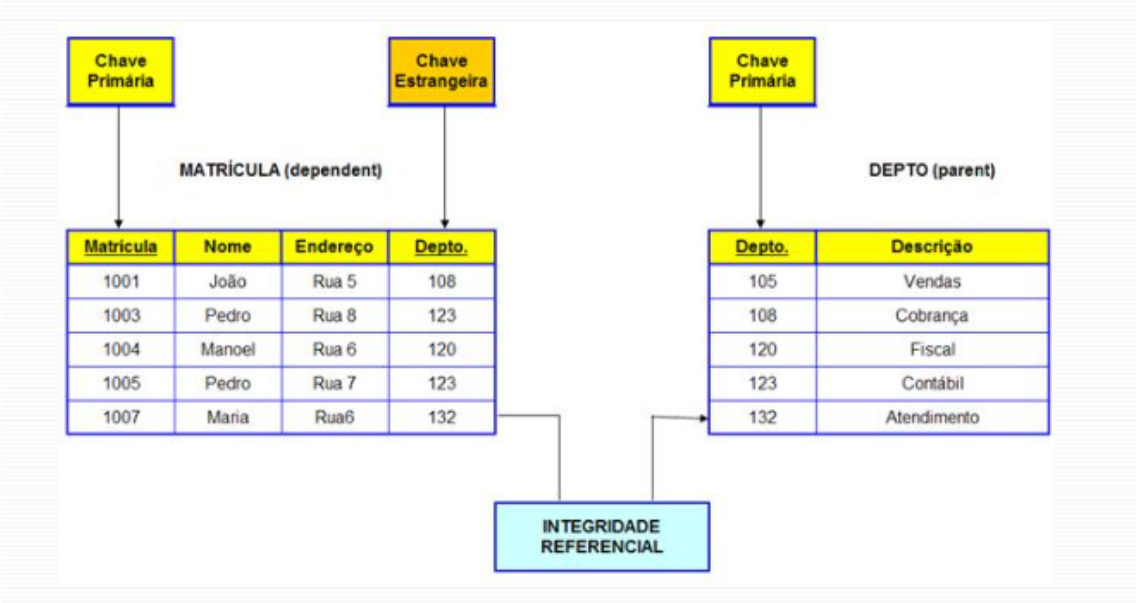
Nos bancos de dados relacionais os dados são armazenados em diversas tabelas que conectam dados operacionais dos sistemas diariamente em uma organização. A quantidade e o tipo de dados armazenados; produzem a personalização dos bancos de dados; os quais determinam seus domínios e aplicações. A variedade de possibilidades de utilização de tipos de chaves acaba por gerarem restrições de integridade e tipo de identidade (Adaptado de SUISSA, 2010).

Para seleção das restrições de integridade e o estabelecimento de relacionamentos entre as tabelas, podem ser utilizados campos que são identificados como chaves e são denominadas de chaves primárias e estrangeiras. As chaves primárias correspondem a uma ou a várias colunas que não possuem valores duplicados dentro de uma tabela. Já a chave estrangeira é aquela correspondente a uma ou a diversas colunas em que seus valores estejam identificados como chaves primárias de outra tabela. A chave estrangeira pode ser denominada como o mecanismo que define todos os relacionamentos de um banco de dados relacional. (Adaptado de HEUSER, 2001).

A chave primária é escolhida a partir de um conjunto de chaves candidatas possíveis para uma determinada entidade. Da mesma maneira que as chaves candidatas representam uma entidade, as chaves primárias representam um valor único e não nulo. DATE (2003).

As chaves estrangeiras são um conjunto de atributos de uma relação que possuem valores que devem ser correspondentes a valores de uma determinada chave candidata de outra relação. As chaves estrangeiras são consideradas ferramentas poderosas para auxílio na manutenção da integridade do banco de dados. A figura 8 abaixo demonstra um modelo de banco de dados com as respectivas chaves.

Figura 8 – Representação de chaves no banco de dados



Fonte: (Adaptado de DORNELLES Carlos Alberto, 2019)

2.2.2. Limitações do Modelo Relacional

O mercado de desenvolvimento de sistemas impulsiona a crescente demanda de novas aplicações; possibilitando surgirem novas características bem como novas aplicações. Neste contexto; atenderem a necessidade de criar uma representação dos tipos de dados, suporte para transações longas, suporte ao processamento de conhecimentos com tratamento de restrições e de gatilhos, contando com um aumento no grau de inter-relacionamentos entre os dados. CARDOSO (2003).

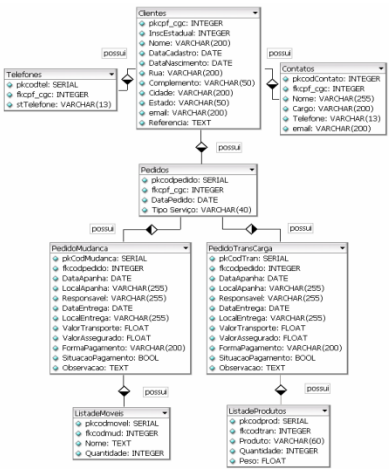
A simplicidade de funcionamento dos bancos de dados relacionais é ocasionada devido às poucas associações entre as relações, no entanto elas apresentam deficiências quando suas aplicações são mais complexas e necessitam ser projetadas. As principais limitações referentes à ausência de tipos que são definidos pelos usuários são: Guimarães (2003).

- Dificuldade de representação de objetos do mundo real em suas aplicações;
- Incompatibilidade com linguagens de programação;
- Divergências entre os objetos da linguagem e as estruturas das entidades do modelo relacional;
- Suporte a grandes bancos de dados;
- Suporte a transações longas; e
- Suporte a processamento de conhecimentos.

Os bancos de dados orientados a objetos foram propostos para fazer frente às necessidades de aplicações mais complexas, ou seja, que compõem muitos atributos e uma grande massa de dados. O método de orientação a objetos oferece muita flexibilidade para lidar com exigências sem limites, sendo uma solução para esses problemas.

A figura 9 demonstra um exemplo de sistema baseado no modelo relacional com oito entidades. A complexidade do banco é ocasionada pelo número de entidades, assim a aplicação de um banco NoSQL faz mais sentido.

Figura *9* - Modelo Relacional



Fonte: (Adaptado de DORNELLES Carlos Alberto, 2019)

2.3. MONGODB X MODELO RELACIONAL

Com a crescente geração e armazenamento de dados possibilitou a conquista dos bancos de dados relacional; embora representasse uma enorme conquista, à estrutura tecnológica impossibilita algumas aplicações de dinamismo e desempenho (Adaptado segundo MPINDA, A.; BUNGAMA, P.; MASCHIETTO, 2015).

O raciocínio implica no: "Quanto mais dados, maior será a demanda de espaço de memória e servidor." Assim a grande diferença de um banco relacional quando comparado com um NoSQL, exemplo do Mongo DB, é a vantagem de gerenciar grande quantidade de dados com fácil acesso e disponibilidade. (Adaptado segundo FREITAS, C. M, 2015).

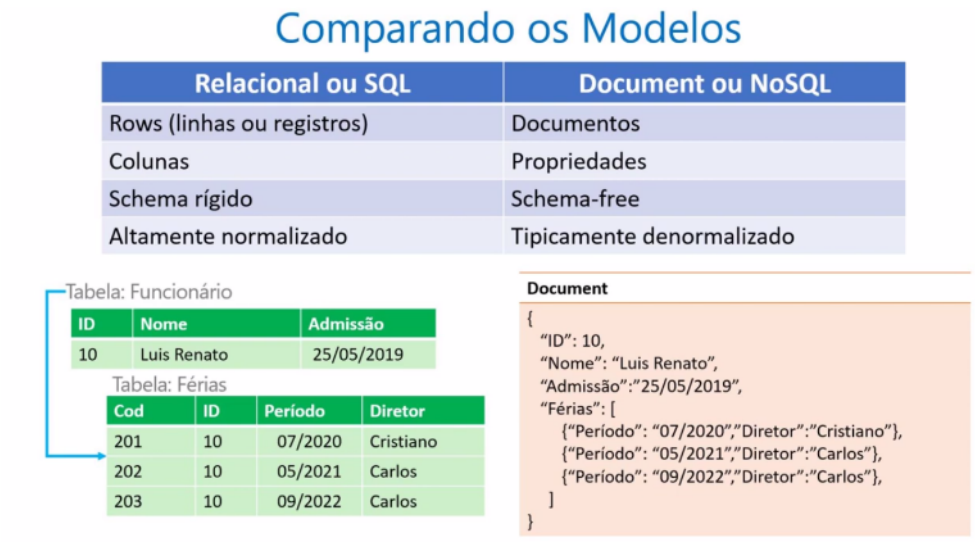
Com a tecnologia NoSQL a organização de dados possibilita maior flexibilidade de acesso as informações; permitindo acessibilidade e manipulação dos dados sem regras específicas. É notado que cada um dos SGBD possui suas características particulares de organização das informações o que implicará todo conhecimento para acessar e manipular tais informações desejadas. (Adaptado segundo MARTINS FILHO, M. A. P, 2015).

A tabela abaixo demonstra uma análise comparativa entre as classes dos bancos de dados.

A principal referência da utilização dos modelos SQL é a linguagem de consulta a qual executa diferentes comandos como criar, alterar, gerenciar e consultar dados armazenados em tabelas, sequência de linhas, mantendo consistência das informações guardadas isto acessado através da linguagem SQL; por estes conceitos estes bancos seguem a modelagem relacional.

Os bancos NOSQL foram idealizados como sinônimo de performance suprindo assim pontos ineficientes dos bancos relacionais; onde se aplica o conceito de schema; uma chave de valor é que é utilizada para recuperar valores ou conjunto de itens.

Figura *10* – Comparando modelos SQL e NoSQL



Fonte: (Adaptado de MARTINS FILHO, M. A. P. SQL X NoSQL, 2015)

Tabela 1 - Modelo Relacional X NoSQL

	Modelo Relacional	NoSQL
Escalabilidade	Possível, mas complexo. Devido à natureza estruturada do modelo, a adição de forma dinâmica e transparente de novos nós no grid não é realizada de modo natural.	Uma das principais vantagens desse modelo. Por não possuir nenhum tipo de esquema pré-definido, o modelo possui maior flexibilidade o que favorece a inclusão transparente de outros elementos
Consistência	Ponto mais forte do modelo relacional. As regras de consistência presentes propiciam um maior grau de rigor quanto à consistência das informações.	Realizada de modo eventual no modelo: só garante que, se nenhuma atualização for realizada sobre o item de dados, todos os acessos a esse item devolverão o último valor atualizado.
Disponibilidade	Dada a dificuldade de se conseguir trabalhar de forma eficiente com a distribuição dos dados, esse modelo pode não suportar a demanda muito grande de informações do banco.	Outro fator fundamental do sucesso desse modelo. O alto grau de distribuição dos dados propicia que um maior número de solicitações aos dados seja atendido por parte do sistema e que o sistema fique menos tempo não disponível.

Fonte: (Adaptado de Brito, 2019)

Tabela 2 - Modelo Relacional X NoSQL e atribuições

Característica	RDBMS	NoSQL
Cumprimento ACID (dados, integridade de transações)	Sim	Não
OLAP / OLTP	Sim	Não
A análise dos dados (agregados, transformar, etc.)	Sim	Não
Rigidez do esquema (mapeamento rigoroso da modelo)	Sim	Não
Flexibilidade formato de dados	Não	Sim
A computação distribuída	Sim	Sim
Escala para cima (vertical) / Dimensionar (horizontal)	Sim	Sim
Desempenho com crescimento de dados	Rápido	Rápido
Sobrecarga de desempenho	Enorme	Moderado
Popularidade / Suporte comunidade	Enorme	Crescente

Fonte: (Adaptado de Brito, 2019)

A prevalência de bancos de dados como MongoDB em algumas situações não é interpretado como maneira de substituição aos modelos relacionais; e sim representarem atendimento as exigências necessárias de usuários e sistemas; visto que possibilita uma maneira mais amigável e flexível com carregamento de dados e manuseios de grandes volumes de informações. (Adaptação de Almeida e Brito, 2012).

2.3.1. Propriedades do NoSQL

Uma das principais característica dos bancos de dados NoSQL, seria no diferirem aos sistemas de banco de dados relacionais; adequando a

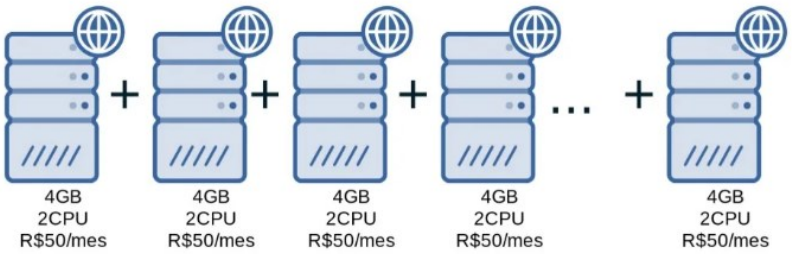
possibilidade de armazenamento de grandes volumes de dados não estruturados ou semiestruturados. Os NoSQL apresentam algumas particularidades como manuseio de API's; a flexibilidade em manusear grande volume de dados sem regras rígidas, possibilitando melhores capacidades de processamentos, tornando mais escaláveis (Adaptado segundo Freitas, 2015).

- **Escalabilidade:** de acordo com Delgado (2011, p.30), os bancos de dados NoSQL tem como principal característica a capacidade de oferecer a uma aplicação uma estrutura de dados flexíveis que possibilitam que seus dados sejam manipulados livres de regras e que seus bancos de dados sejam distribuídos em um maior nível do que seria possível se fosse seguida uma estrutura relacional ou que fosse pouco flexível.

A escalabilidade de um banco de dados é determinada pela forma da divisão do banco de dados, tendo em vista que para cada petição existe um limite de dados que é suportado. Para demonstrar isso, a figura 10 abaixo ilustra uma certa quantidade de servidores que são utilizados para uma única aplicação, solução que é conhecida como escalonamento horizontal e tem sido a mais utilizada nas camadas de aplicações, principalmente quando se trata de aplicações web.

Figura *11* – Representação de escalonamento Horizontal

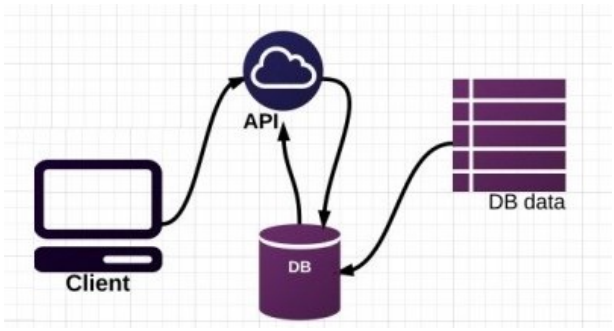
É compreendido que com o escalonamento horizontal o processamento de equipamentos conectados serão ampliados isto aplica a maior poder de execução de tarefas e mais servidores são adicionados para suprirem carga de atividade.



Fonte: Adaptação segundo Delgado (2011)

- **API simples de acesso aos dados:** o NoSQL tem como objetivo prover uma maneira eficiente de acesso aos dados para oferecer alta disponibilidade e escalabilidade, ou seja, o foco não está no fato de como os dados são armazenados e sim como estes poderão ser recuperados de uma maneira eficiente. Para isso é necessário que uma API seja desenvolvida para facilitar o acesso a estas informações, permitindo desta forma que qualquer aplicação possa utilizar os dados do banco de forma mais rápida e eficiente. A figura 12 mostra a movimentação entre uma aplicação e um banco de dados, tendo uma API como intermediário.

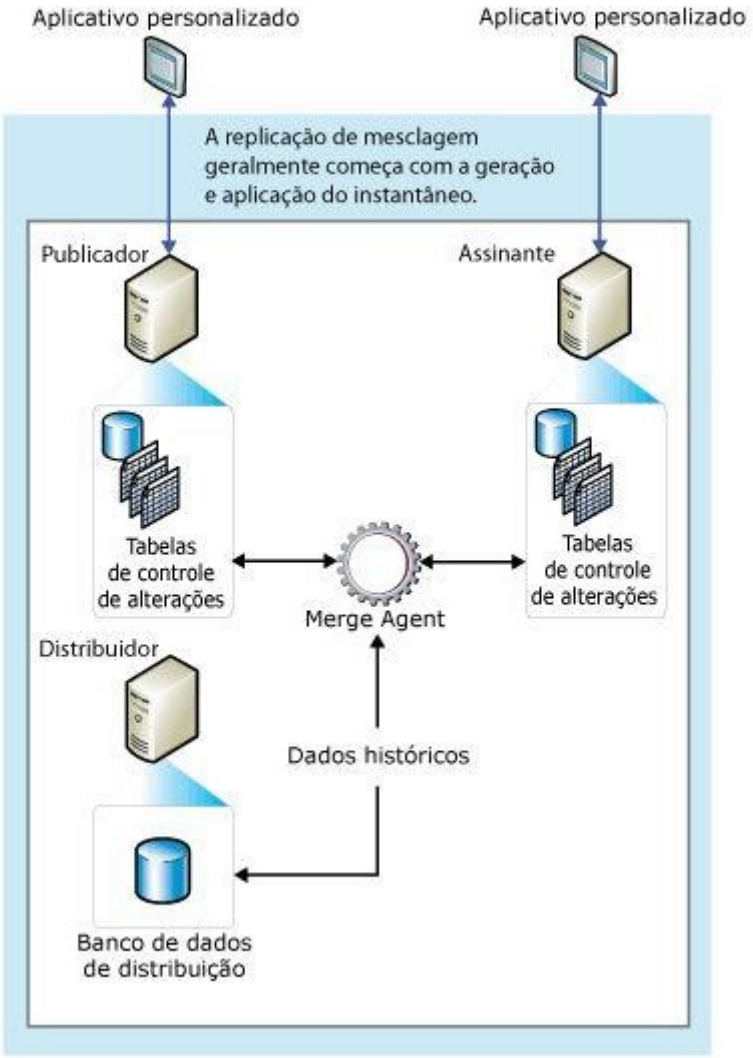
Figura *12* – Representação de escalonamento Horizontal



Fonte: Adaptação segundo Ivan de Souza (2016)

- **Ausência de esquemas ou esquemas flexíveis:** os bancos de dados NoSQL possuem a característica de ausência completa ou quase completa do esquema que define a estrutura de dados modelados. Assim essa ausência de esquema é uma facilidade tanto para a escalabilidade quanto para contribuir para aumentar a disponibilidade do sistema. Em contrapartida, não existem garantias da integridade dos dados, fato que acontece nos bancos de dados relacionais, devido a sua rígida estrutura.
- **Suporte nativo a replicação:** a replicação é uma outra forma de prover a escalabilidade, pois ela consiste no armazenamento de um mesmo conjunto de dados em vários servidores distintos de uma forma nativa, diminuindo assim o tempo gasto para recuperação das informações. A figura 12 abaixo mostra um exemplo de como seria a replicação dos dados de uma empresa.

Figura *13* – Sistemas de Replicação de Dados



Fonte: Adaptada segundo Delgado (2001, p.35)

A utilização de replicação permite processos de leitura de maneira potencializada; visto que a operação da leitura de dados não passa a ser realizada de forma concentrada em um único servidor; mas sim distribuída entre todo o sistema.

- **Taxonomia do modelo de dados NoSQL:** a taxonomia descreve os tipos de bancos de dados NoSQL que são baseados em chave-valor, orientados a documentos, banco de dados de família, de coluna e banco de dados de grafos. Para Sadalage e Fowler (2013, p.41) o modelo de dados na maioria das vezes serve para falar o modelo através do qual o

gerenciador do banco de dados organiza estes dados, que é denominado de metamodelo. Através do surgimento da tecnologia NoSQL, cada uma das soluções possui modelos diferentes, que são divididos em quatro categorias. São elas: chave-valor, documento, família de colunas e grafos. As três primeiras possuem características comuns em seu modelo de dados, denominada de orientação agregada, que é um conjunto de objetos relacionados que são tratados como uma unidade.

2.4. MongoDB utilizado como alternativa ao Banco de Dados Relacional

A utilização de Bancos de Dados relacionais nem sempre é a melhor opção para os projetos, já que eles possuem uma grande complexidade e por serem de difícil escalabilidade, esta forma de armazenamento de dados é ineficiente em algumas situações (Adaptado segundo HEUSER, 2006).

Quando existe um grande volume de dados ou quando existe a necessidade de uma baixa latência no acesso ou na extração das informações, torna-se mais vantajosa a utilização dos bancos de dados NoSQL (que significa "*Not Only* SQL"). Os bancos de dados não relacionais fogem do modelo tradicional de tabelas para realizar a implementação de soluções que são mais eficientes na manipulação de grandes massas de dados, e portanto, deveriam ser preferidos nessas ocasiões. (Adaptado segundo MARTINS FILHO, M. A. P., 2015).

De acordo com uma pesquisa realizada pelo site DB-Engines o MongoDB é o primeiro NoSQL mais utilizado no mundo e é o quinto mais utilizado no ranking geral, ficando atrás somente dos SQLs mais consolidados, como o Oracle e MySQL. Ele é um banco de dados gratuito, de código aberto e que adota o modelo baseado em documentos e armazena os dados em arquivos denominados de BSON, que é como se fosse uma extensão do JSON. Ele

oferece formas para o escalonamento, para replicação e para a realização de backups dos dados. O MongoDB disponibiliza também suporte a queries dinâmicas nos documentos utilizando uma linguagem poderosa. Outro destaque que o MongoDB apresenta é a conectividade com as linguagens de programação. Como o Python, por exemplo, basta instalar a biblioteca PyMongo que então será possível utilizar os comandos do MongoDB dentro da linguagem Python (Adaptado segundo Weinberger, Claudius 2018).

2.4.1. Funcionamento do MongoDB

O MongoDB guarda os dados de uma determinada entidade em um arquivo que é formatado em uma sintaxe JSON-like. Ele é autossuficiente e contém todas as informações relacionadas a uma entidade, evitando dessa forma a necessidade de "*JOINS"* entre as tabelas para fazer as buscas no banco. (Adaptado segundo FOWLER, M., 2005)

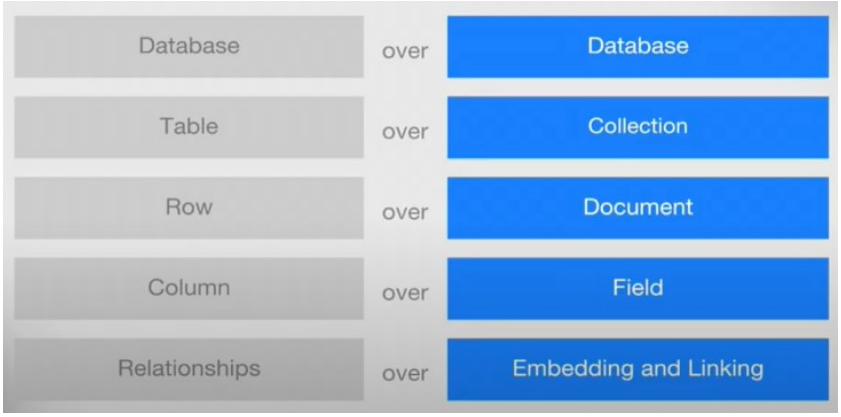
O MongoDB possuí um método denominado de Sharding. Esse método possibilita o armazenamento de dados ao longo de múltiplas máquinas, permitindo dessa forma que haja uma escalabilidade horizontal. No MongoDB todas as coleções estão contidas em um banco, assim ele é denominado como sendo um banco de dados schemaless, ou seja, ele não força nenhuma estrutura definida para os documentos, assim uma coleção pode possuir diversos documentos diferentes e cada um conter campos distintos entre si (Adaptado segundo Martins, 2015).

2.4.2. Casos de uso para utilização do MongoDB em substituição ao BD Relacional

O MongoDB utilizado em substituição ao Banco de Dados Relacional é uma excelente ferramenta para quem trabalha com BigData, tendo em vista que ele permite guardar qualquer tipo de dado. Desta forma os arquivos binários, tais como vídeos e imagens, podem ser armazenados em documentos. Essa é uma

excelente forma de guardar os dados que não são estruturados para serem manipulados ou analisados (Martins Filho, M. A. P., 2015).

Figura 14 – Estrutura de banco MongoDB



Fonte: Adaptada Mongodb.com (2020)

Seguindo o modelo descrito e adaptador por (Mpinda, A.; Lewis, T.G; 2015), o conjunto MongoDB é capaz de fazer queries mais velozes e desta forma fornecer informações que são solicitadas em baixa latência, ele é recomendado para aplicações tanto mobile como web com requisições simultâneas e que possuem a necessidade de apresentar os dados que foram salvos em diversas instancias. Dessa maneira ele permite que as informações de uma determinada página sejam reunidas em um único documento e não em várias tabelas como no banco de dados relacional.

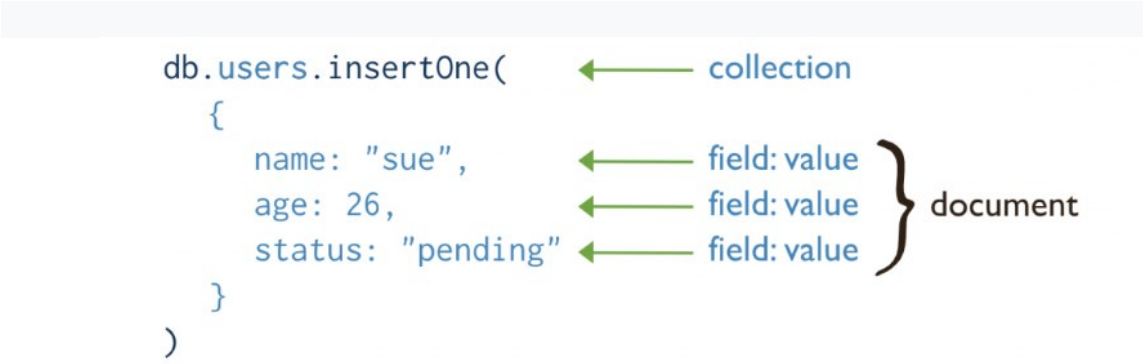
O MongoDB garante a alta disponibilidade através da replicação. O servidor MongoDB pode realizar copias duplicadas em mais de dois servidores de seus bancos de dados. Essas copias são conhecidas como um conjunto de réplicas que são organizados através de um processo de eleição onde os membros da réplica realizam a votação de qual servidor se torna o principal e quais atuam como secundários.

2.4.3 Exemplo prático do MongoDB x MySQL

Como foi visto anteriormente, o MongoDB permite a criação de documentos que não precisam necessariamente ter uma estrutura definida como no MySQL, o que permite que eles possam ter dados diferentes, como por exemplo, na figura 15 vemos que nem todas as entradas possuem o campo notas.

O MongoDB utiliza valores dos nomes ou campos nome-valor pares ou campos, que então formam documentos, que formam coleções.

Figura 15 – Estrutura de declarações no banco MongoDB



Fonte: Adaptada Mongodb.com (2020)

Abaixo o destaque fica com a aplicação de conceitos atribuídos a criação de um banco com o MongoDB.

Figura 16 – Criação do banco de alunos no MongoDB

Alunos

	nome String	_id ObjectId	data_nascimento String	curso Object	habilidades Mixed	notas String
1	"Felipe"	ObjectId('61e087455525fe7e350…	"1994/02/26"	{} 1 fields	[] 3 elements	"[10.0, 9.0, 4.5]"
2	"Julio"	ObjectId('61e08c3d5525fe7e350…	"1972/09/30"	{} 1 fields	[] 1 elements	No field
3	"Alberto"	ObjectId('61e09035525fe7e350…	"1972/10/30"	{} 1 fields	[] 1 elements	No field
4	"Daniela"	ObjectId('61e090b95525fe7e350…	"1995/09/30"	{} 1 fields	[] 1 elements	No field
5	"Jonas"	ObjectId('6294bf51f93ae1867b4…	"1990/05/12"	{} 1 fields	[] 3 elements	"[10.0, 10.0, 8.7, 9.2]"
6	"Adriana"	ObjectId('6294c330f93ae1867b4…	"1999/12/01"	{} 1 fields	[] 2 elements	"[8.0, 9.0]"
7	"Marshall"	ObjectId('6294c432f93ae1867b4…	"1991/03/30"	{} 1 fields	[] 3 elements	"[10.0, 10.0, 10.0, 5.0]"
8	"Bruna"	ObjectId('6294fb09f93ae1867b4…	"1998/07/15"	{} 1 fields	[] 2 elements	No field
9	"Emerson"	ObjectId('6294fbb2f93ae1867b4…	"1998/11/05"	{} 1 fields	""	"[6.0, 9.0, 7.0, 5.0]"
10	"Vanessa"	ObjectId('6294fd18f93ae1867b4…	"1998-02-10"	{} 1 fields	[] 1 elements	No field

Fonte: Autoria própria (2022)

Para esse mesmo banco de dados, porém feito no MySQL, ilustração na figura seguinte onde todas as pessoas possuem o atributo "notas", entretanto alguns deles estão em branco.

Figura 17 – Tabela Pessoas no MySQL

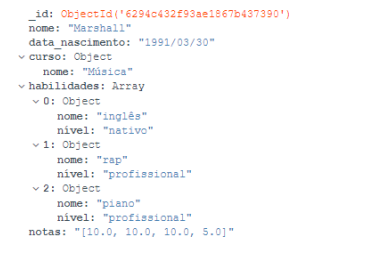
id_pessoa	nome	data_nascimento	curso	notas
1	Felipe	1994-02-26	Sistemas da Informação	[10, 9, 4.5]
2	Daniela	1995-09-30	Moda	
3	Jonas	1990-05-12	Engenharia Aeronautica	[10, 10, 8.7, 9.2]
4	Adriana	1999-12-01	Advocacia	[8, 9]
5	Marshall	1991-03-30	Música	[10, 10, 10, 5]
6	Bruna	1998-07-15	Jogos Digitais	
7	Emerson	1998-11-05	Enfermagem	[6, 9, 7, 5]
8	Vanessa	1998-02-10	Biologia	
9	Alberto	1972-10-30	Engenharia Química	
10	Julio	1972-09-30	Medicina	

Fonte: Autoria própria (2022)

Outra diferença em destaque observado é que no MongoDB a base de dados foi feita com o campo "curso" e "habilidades" que estão definidos como *arrays*, ou seja, eles têm um conjunto de dados dentro deles. Isso não é possível de ser feito no banco de dados relacional de uma maneira direta, para representar um *array* no MySQL temos que criar mais de uma tabela, relacionando com a tabela principal (que no caso do exemplo desse trabalho é a tabela Pessoas) através de uma chave estrangeira e posteriormente executar uma query com o comando join para juntar as tabelas.

No exemplo criado para esse trabalho, ilustrado na figura seguinte onde o campo "habilidades" possui dois dados, o nome e o nível.

Figura 18 – Exemplo de documento no MongoDB



Fonte: Autoria própria (2022)

Para conseguir replicar isso no MySQL, além da tabela Pessoas, foram criadas mais uma tabela denominada Habilidades.

Figura 19 – Tabela Habilidades no MySQL

id_habilidade	habilidade	nivel	id_pessoa
1	inglês	avançado	1
2	taekwondo	básico	1
3	guitarra	profissional	1
4	alemão	básico	2
5	alemão	avançado	3
6	natação	avançado	3
7	piano	profissional	3
8	inglês	fluente	4
9	yoga	avançado	4
10	inglês	nativo	5
11	rap	profissional	5
12	piano	profissional	5
13	design digital	avançado	6
14	javascript	profissional	6
15	botânica	profissional	8
16	inglês	intermediário	9
17	inglês	avançado	10

Fonte: Autoria própria (2022)

A importância notada foi preciso entrar com uma linha de dados para cada habilidade de cada pessoa, contendo o nome da habilidade, o nível e a chave estrangeira "id_pessoa" para podermos relacionar as habilidades às pessoas. Em seguida, para conseguirmos um resultado que mostre todos os dados como na figura 17, devemos executar o seguinte comando de busca no MySQL:

"select Pessoas.id_pessoa, nome, data_nascimento, curso, notas, habilidade, nivel from Pessoas left join Habilidades on Habilidades.id_pessoa = Pessoas.id_pessoa;"

Após a execução dessa busca, que irá "juntar" as tabelas, o sequenciamento será obter o seguinte resultado:

Figura 20 – Resultado do join no MySQL

id_pessoa	nome	data_nascimento	curso	notas	habilidade	nivel
1	Felipe	1994-02-26	Sistemas da Informação	[10, 9, 4.5]	inglês	avançado
1	Felipe	1994-02-26	Sistemas da Informação	[10, 9, 4.5]	taekwondo	básico
1	Felipe	1994-02-26	Sistemas da Informação	[10, 9, 4.5]	guitarra	profissional
2	Daniela	1995-09-30	Moda		alemão	básico
3	Jonas	1990-05-12	Engenharia Aeronautica	[10, 10, 8.7, 9.2]	alemão	avançado
3	Jonas	1990-05-12	Engenharia Aeronautica	[10, 10, 8.7, 9.2]	natação	avançado
3	Jonas	1990-05-12	Engenharia Aeronautica	[10, 10, 8.7, 9.2]	piano	profissional
4	Adriana	1999-12-01	Advocacia	[8, 9]	inglês	fluente
4	Adriana	1999-12-01	Advocacia	[8, 9]	yoga	avançado
5	Marshall	1991-03-30	Música	[10, 10, 10, 5]	inglês	nativo
5	Marshall	1991-03-30	Música	[10, 10, 10, 5]	rap	profissional
5	Marshall	1991-03-30	Música	[10, 10, 10, 5]	piano	profissional
6	Bruna	1998-07-15	Jogos Digitais		design digital	avançado
6	Bruna	1998-07-15	Jogos Digitais		javascript	profissional
7	Emerson	1998-11-05	Enfermagem	[6, 9, 7, 5]	NULL	NULL
8	Vanessa	1998-02-10	Biologia		botânica	profissional
9	Alberto	1972-10-30	Engenharia Química		inglês	intermediário
10	Julio	1972-09-30	Medicina		inglês	avançado

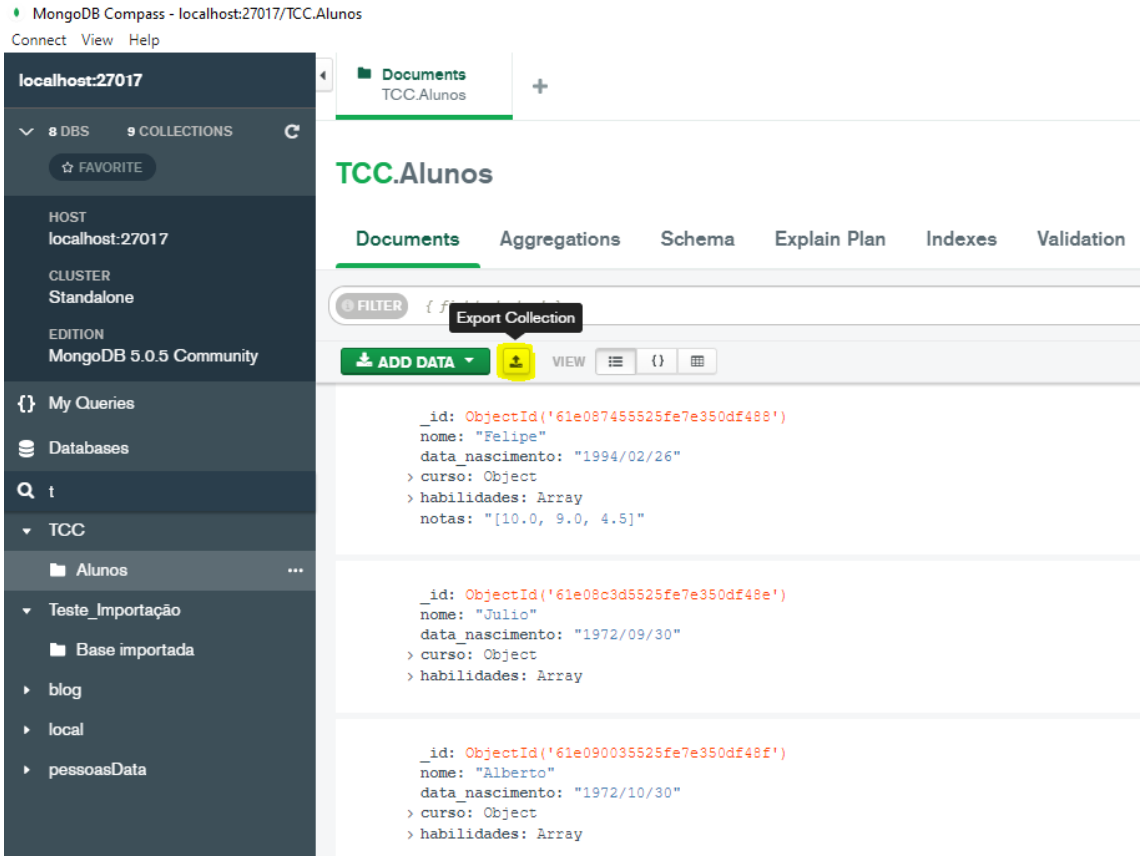
Fonte: Autoria própria (2022)

Vale ressaltar que a obtenção dos dados desejado, é notável que muitas informações se repetem, como por exemplo o nome, data de nascimento, curso e as notas das pessoas, algo que pode tornar o entendimento da tabela mais difícil e pesado.

Além disso, como foi dito mais cedo nesse trabalho, a utilização da interface gráfica do Mongo, o MongoDB Compass, traz uma facilidade para salvar a base de dados, replicá-la, fazer análises rápidas do tipo e quantidade de dados presente na base de dados e também é possível criar validações. As validações são uma funcionalidade que pode limitar os dados que podem ser inseridos na base de dados, tornando-a mais estruturada e próxima de um banco de dados relacional.

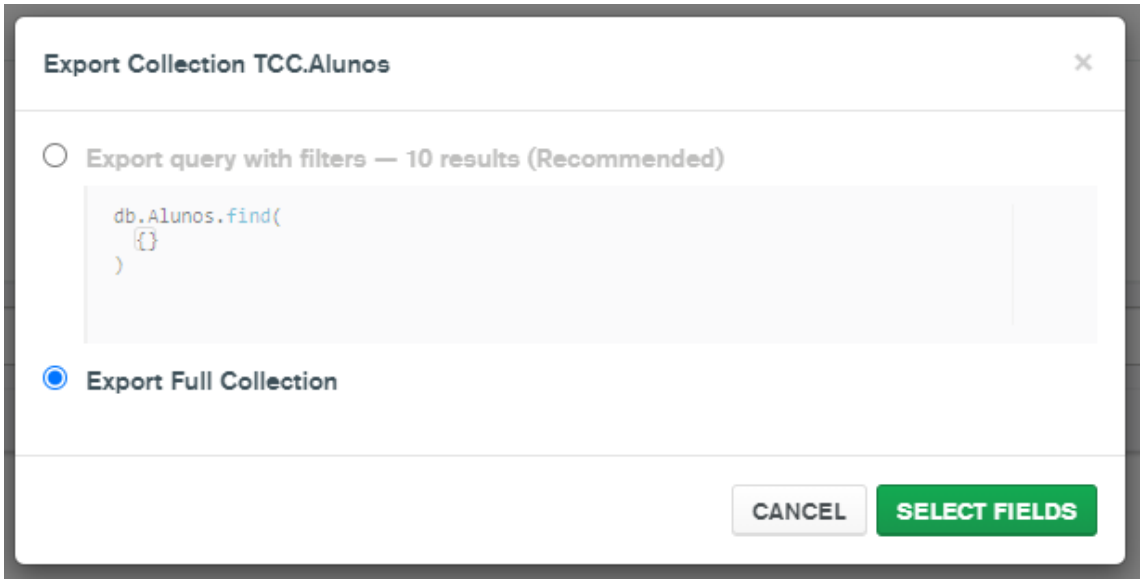
A sequência das figuras 18 a 25 demonstram no passo a passo de como fazer para salvar a base de dados do MongoDB em um único documento JSON que pode ser exportado e reutilizado em outra base de dados.

Figura 21 – Botão Export Collection



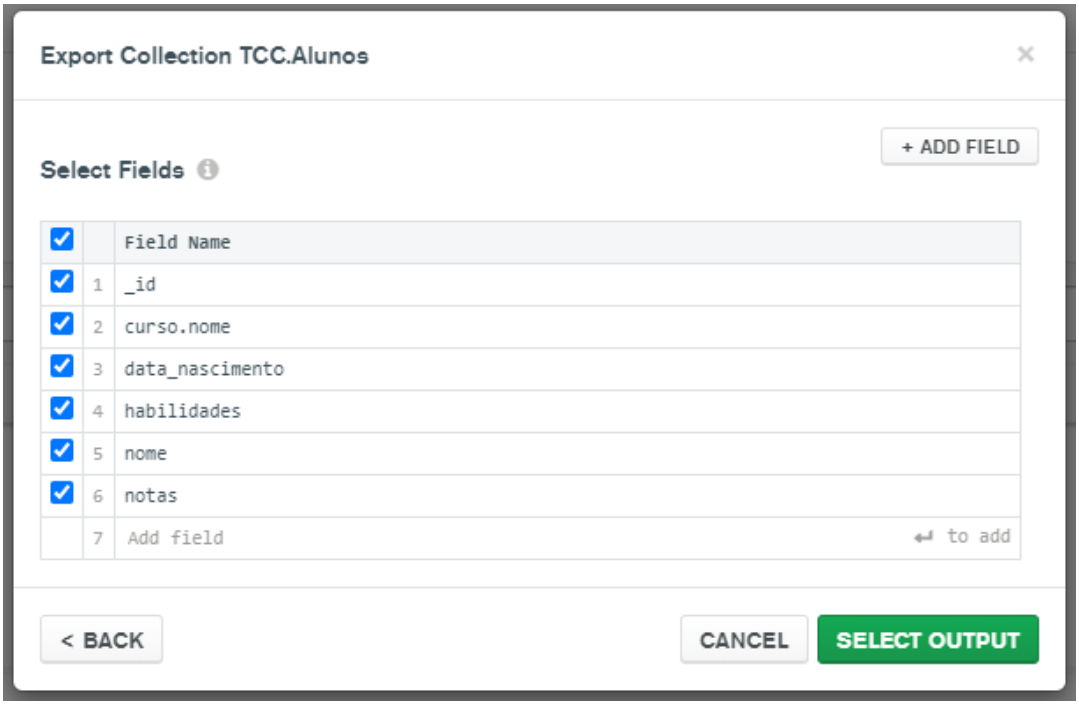
Fonte – Autoria Própria (2022)

Figura 22 – Escolha do conteúdo



Fonte – Autoria Própria (2022)

Figura 23 – Escolha dos campos

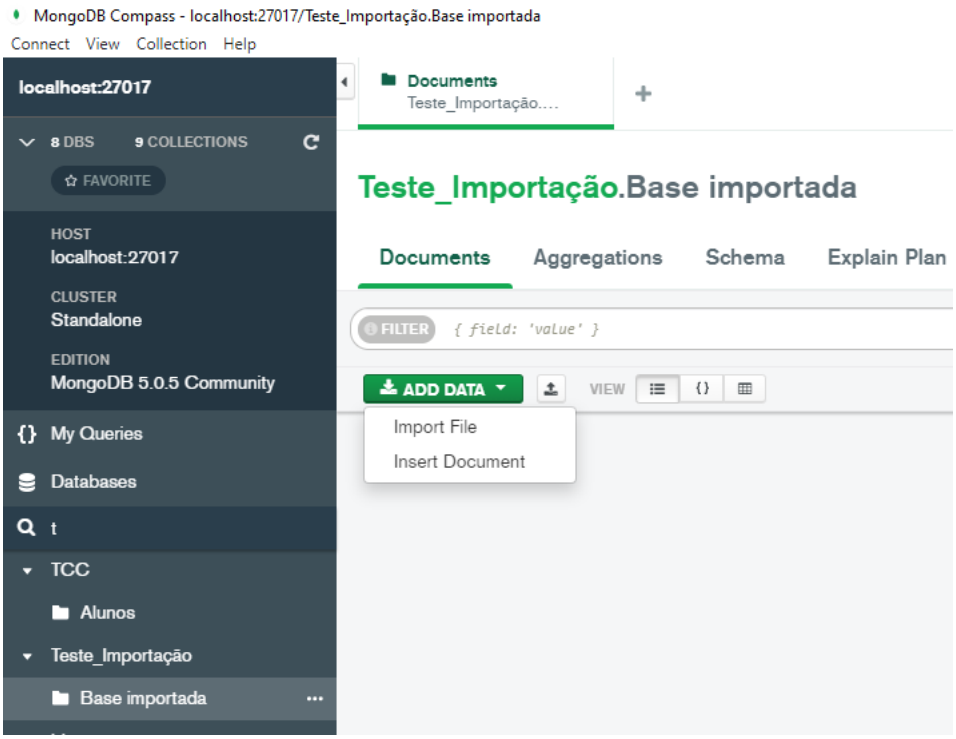


Fonte – Autoria Própria (2022)

Figura 24 – Escolha do tipo de arquivo e diretório

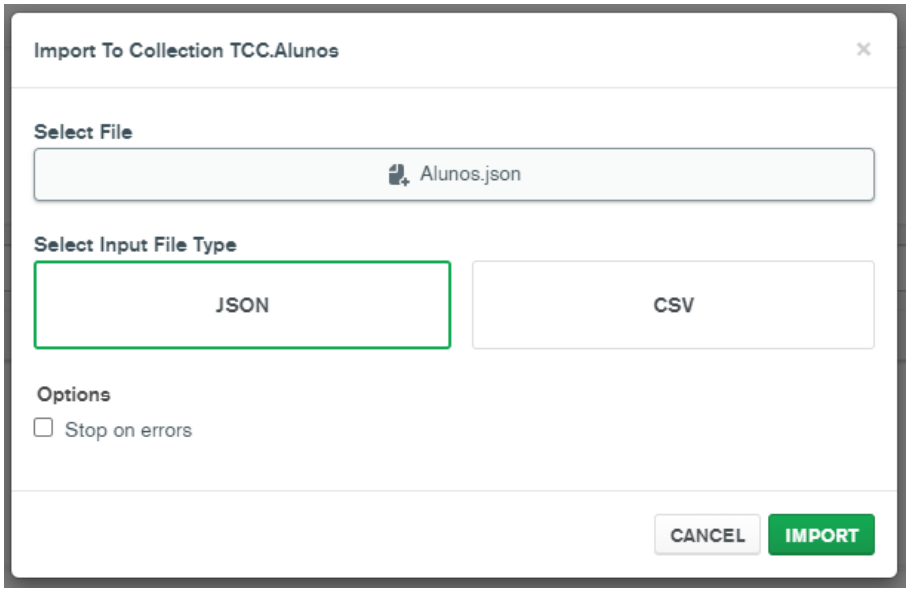
Fonte – Autoria Própria (2022)

Figura 25 – Importando uma base de dados



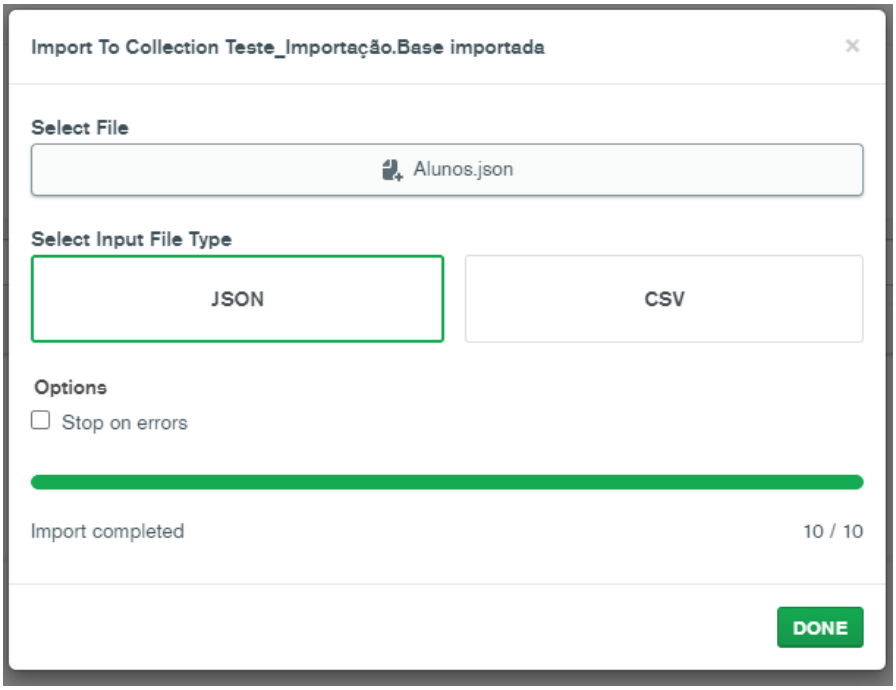
Fonte – Autoria Própria (2022)

Figura 26 – Escolhendo o arquivo



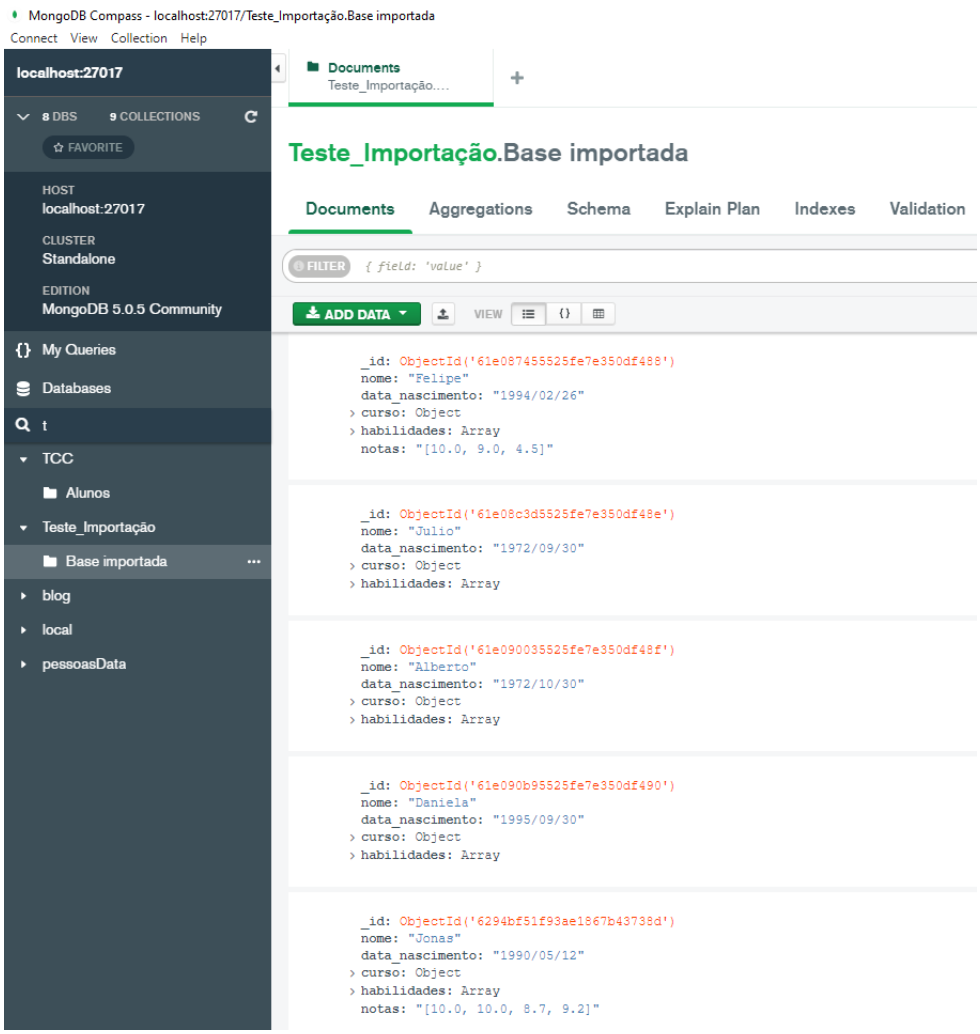
Fonte – Autoria Própria (2022)

Figura 27 – Importação completa



Fonte – Autoria Própria (2022)

Figura 28 – Base importada



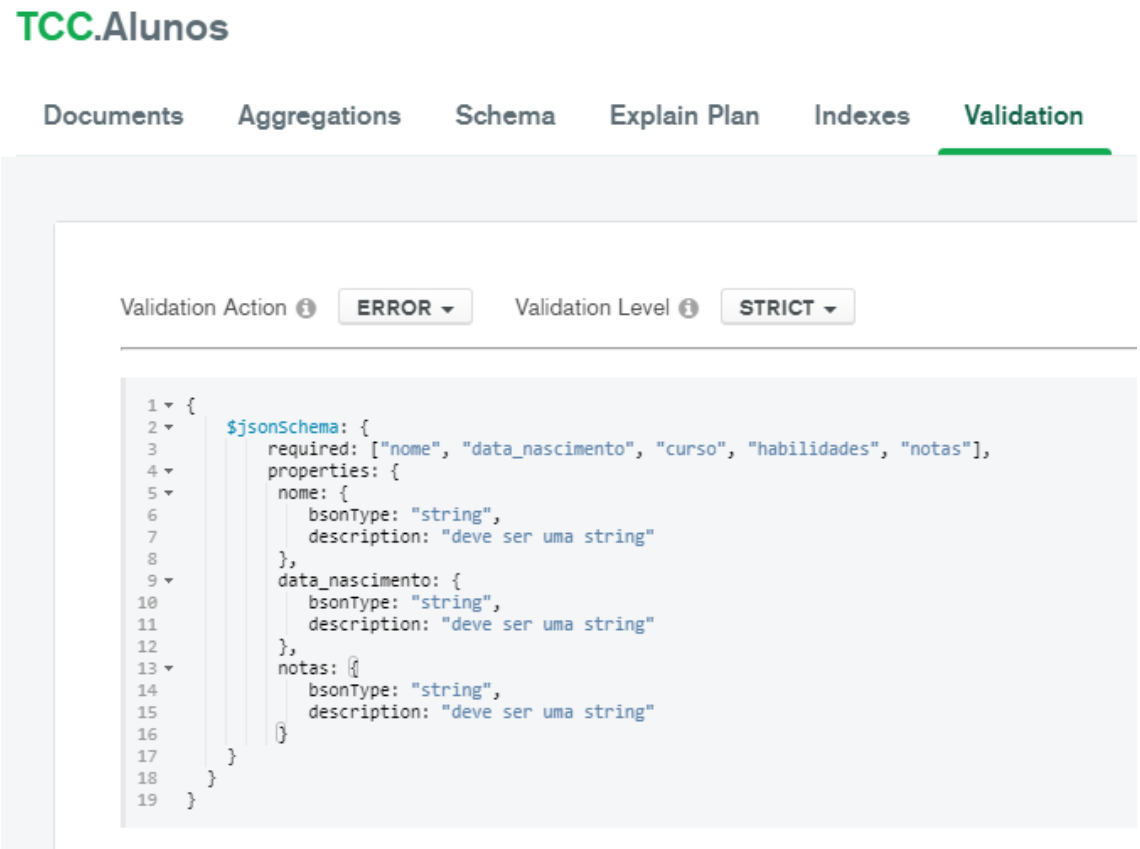
Fonte – Autoria Própria (2022)

Para exportar a base de dados, é necessário clicar no botão export collection que irá abrir uma nova janela na qual deve escolher quais documentos queremos exportar através de uma busca ou exportar toda a base. Em seguida escolher quais os campos dos documentos escolhidos vamos exportar, o que nos leva a tela seguinte, na qual deve-se escolher o tipo do documento (JSON ou CSV) e o local no qual deseja salvar o arquivo. Então basta clicar no botão "export" e a base de dados estará salva em um arquivo no formato escolhido.

Feito isso, para importar a base que foi salva em um outro banco, deve-se selecionar o banco desejado no menu esquerdo e depois clicar no botão "Add Data" e selecionar "Import File". Em seguida basta escolher o arquivo que se deseja importar e clicar no botão "Import". Caso não haja nenhum erro, a importação estará completa.

Utilizando a base de dados de alunos que foi mostrada na figura 16, pode ser criada uma validação para que futuras adições a base de dados sejam feitas igualmente aos dados já existentes. Um exemplo de validação pode ser visto na figura 29 a seguir:

Figura 29 – Exemplo de validação

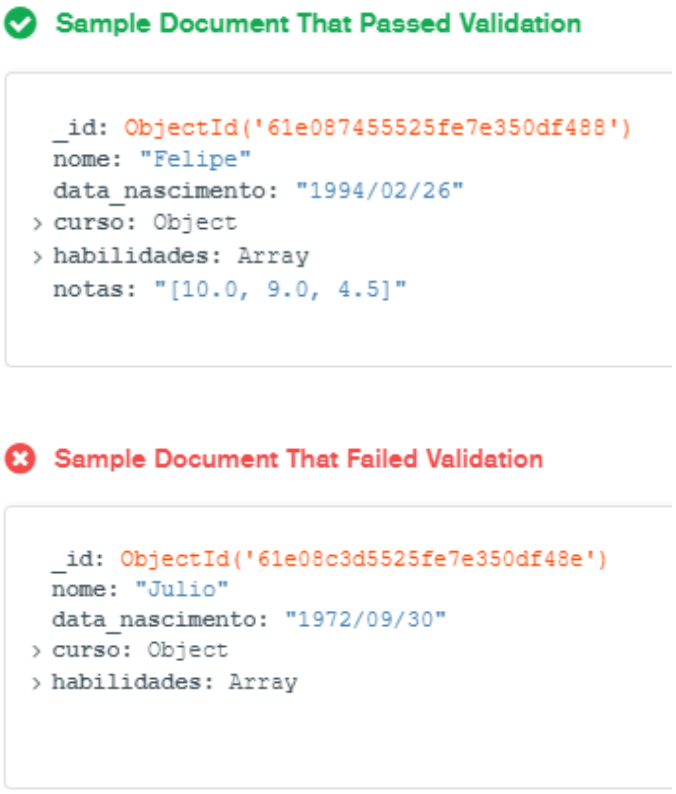


Fonte – Autoria Própria (2022)

Nesse exemplo, foi necessário que todos os documentos que forem inseridos na base de dados contenham os campos nome, data_nascimento,

curso, habilidades e notas. Além disso o conteúdo dos campos nome, data_nascimento e notas devem ser do tipo string. Ao escrever o código de validação, o Compass já mostra dentro da sua base de dados um exemplo de um documento que passou pela validação e um que não passou, como mostra a figura 27 a seguir:

Figura 30 – Documento validado X não validado



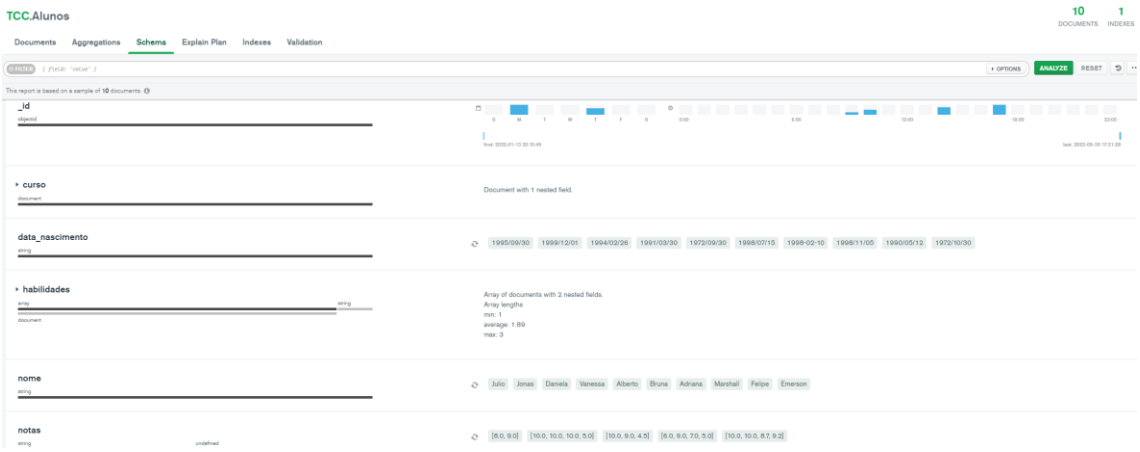
Fonte – Autoria Própria (2022)

Nesse caso, observa-se que um dos documentos da base de dados em questão não foi validado pois não apresenta o campo notas.

Dessa maneira, com poucas linhas de código é atingido a garantia de uma estrutura a nossa base de dados bem similar aos bancos de dados relacionais.

Outra funcionalidade do Compass é a análise que ele apresenta das *collections* presentes no seu banco de dados. A figura 31 mostra a página de análise do Compass.

Figura 31 – Exemplo de análise do Compass



Fonte – Autoria Própria (2022)

Essa analise traz todos os campos presentes na base de dados tanto quanto o tipo de variável e no lado direito também pode ser considerada a amostragem de algumas entradas de dados dos campos correspondentes. Além disso, na parte superior direita o Compass mostra alguns gráficos de barras que mostram em quais dias o banco foi utilizado e em quais horários. Essa funcionalidade é bem interessante e é algo que não é encontrado nos bancos de dados tradicionais.

2.4.4. Aplicação da biblioteca python PyMongo em banco MongoDB

O PyMongo contém ferramentas que possibilitam interagir com o banco de dados MongoDB do Python. O pacote bson é uma implementação do formato BSON para Python. O pacote pymongo é um driver Python nativo para MongoDB. O pacote gridfs é uma implementação do gridfs sobre o pymongo.

A instalação da biblioteca PyMongo feita para MongoDB é simples e acessível.

Basta seguir com o comando mostrado na figura 32 para instalação da biblioteca.

Figura 32 – Importação de bibliotecas no ambiente Colab

```
pip install pymongo
```

Fonte – Autoria Própria (2022)

Com o PyMongo instalado, é necessário estabelecermos a conexão com o banco MongoDB, e na sequência criar a conexão, é possível utilizar o localhost, ou uma URL do seu banco, cujo formato segue:

Figura 33 – Conexão com o banco de dados

```
URL =
"mongodb://[username:password@]host1[:port1][,host2[:port2],.
..[,hostN[:portN]]][/[database][?options]]"
```

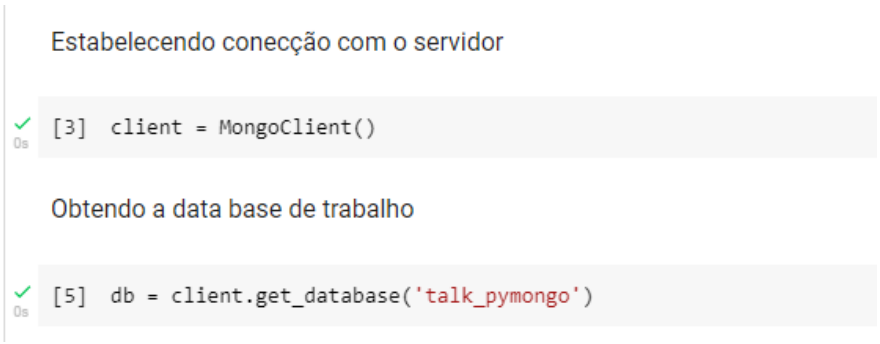
Fonte – Autoria Própria (2022)

Aplicamos o sequenciamento abaixo para obter o banco através das bibliotecas python relacionadas ao banco MongoDB contextualizados anteriormente.

Figura 34 – Instalação da biblioteca no ambiente Colab (google)

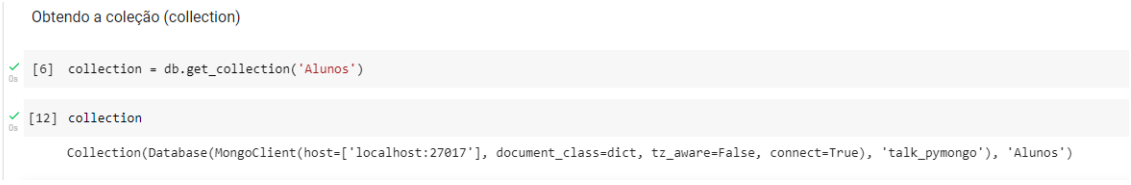
Fonte – Autoria Própria (2022)

Figura 35 – Comunicação com servidor



Fonte – Autoria Própria (2022)

Figura 36 – Obtendo coleção de dados criada



Fonte – Autoria Própria (2022)

Figura 37 – Criação de novos registros

Criando novos registros na collect

```
[ ]  data= {
         'id_pessoa': 15
         'nome': 'Eduardo',
         'data_nascimento': 1982-02-01,
         'curso':"Sistemas da Informação",
         'notas':[10,8,7],
         'habilidade': "comunicação",
         'nivel':"avançado"
       }
     collection.insert(data)
```

Fonte – Autoria Própria (2022)

Figura 38 – Criação de registros sequenciais na coleção

Criando vários registros sequênciais

```
data= [
    {
    'id_pessoa': 15
    'nome': 'Eduardo',
    'data_nascimento': 1982-02-01,
    'curso':"Sistemas da Informação",
    'notas':[10,8,7],
    'habilidade': "comunicação",
    'nivel':"avançado"
    {
    'id_pessoa': 16
    'nome': 'Bernardo',
    'data_nascimento': 2012-03-04,
    'curso':"Colegial",
    'notas':[8,8,8],
    'habilidade': "Educação Física",
    'nivel':"avançado"
    }

]
collection.insert(data)
```

Fonte – Autoria Própria (2022)

Figura 39 – Representação de linhas de códigos a partir da coleção criada

Aplicando alguns comando de buscas de informações:

```
[ ]  for Alunos in collection.find():
       print(Alunos.get('id'))
```

```
[ ]  for Alunos in collection.find({'nome:Eduardo'}):
       print(Alunos.get('id'))
```

```
[ ]  for Alunos in collection.find({'curso':"Sistemas da informação"}):
       print(Alunos.get('id'))
```

Fonte – Autoria Própria (2022)

Figura 40– Comando de atualizações na coleção

Aplicando alguns comandos de atualizações:

```
[ ]  collection.update(
         {'nome':'Eduardo Henrique Geraldo'},
         {'notas':[7,6,9]}
     )
```

Fonte – Autoria Própria (2022)

Figura 41 – Removendo elementos da coleção criada

Removendo alguns elementos da coleção

```
[ ]  from os import remove
     collection.remove({'aluno':'Eduardo'})
```

Fonte – Autoria Própria (2022)

3. CONCLUSÃO

A utilização do MongoDB combinados com bibliotecas python demonstram ser uma crescente e importante alternativa como possibilidade de substituição de Banco de Dados Relacional pois possibilita diversas vantagens tendo em vista que este modelo não relacional é um banco de dados orientado a documentos, livre de estruturas, possui código aberto, e é multiplataforma podendo ser conduzido através de linguagens C++ e Python.

A conecção PyMongo com MongoDB cria um banco de dados NoSQL, que permite a utilização de documentos que são semelhantes ao JSON como formato de armazenamento de dados e suas atribuições e características, além do que permitem aplicações de modelamento de informações de maneira natural, uma vez que seus dados são alinhados em hierarquias complexas que podem continuar a serem indexadas e tornam fáceis busca por informações.

Com todas as atribuições e características da conectividade PyMongo com o MongoDB pavimentam à área de ciência de dados e Big Data, pois estes segmentos trabalham com imensidade de dados e, portanto, se beneficiam da velocidade de busca e eficiência do sistema e processos.

Podemos assumir que o objetivo de construção de uma possível alternativa aos bancos relacionais tradicionais com vantagens de manuseios e compreensão de processos e eficiência foram atingidos; porém ressaltamos que vantagens e desvantagens serão intrínsecas para cada contexto e cenário; uma vez que cada projeto a ser desenvolvido terá uma validação de banco a ser seguida e modelos tradicionais devem também serem ponderados.

4. REFERÊNCIAS BIBLIOGRAFICAS

BSONSPEC. Binary bson**. Disponível em: http://bsonspec.org. Acesso em: 01 fev. 2015.**

CLAUDIUS. NoSQL Performance Benchmark 2018**. 2018. Disponível em: https://www.arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangod.. Acesso em: 01 maio 2018.**

CONCEITUANDO BANCO DE DADOS E SGBD. **Sao Paulo: Addison Wesley, 2012.**

DATE. Introdução a Sistemas de Bancos de Dados 8ED**. 8. ed. Rio de Janeiro: Elsevier, 2003.**

DB-ENGINES. RANKING DOS SGBDS MAIS UTILIZADOS**. Disponível em: https://db-engines.com/en/ranking. Acesso em: 15 fev. 2000.**

FOWLER, M. UML Essencial: um breve guia para linguagem padrão. 3. ed. Porto Alegre: Bookman, 2005

FREITAS, C. M. Mapeamentos Conceituais entre Modelo Relacional e Estruturas NoSQL: um estudo de caso com documentos. **2015. 105 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2015.**

GUIMARÃES, CÉLIO (2003) **Fundamentos de Bancos de Dados:** Modelagem, Projeto e Linguagem SQL. Editora UNICAMP, 1a.

HEUSER, Carlos Alberto. Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS**. Rio Grande do Sul: Bookman Editora, 2009.**

JSON.ORG. Introducing JSON**. Disponível em: JSON. Introducing JSON. Disponível em http://www.json.org. Acesso em: 01 fev. 2015.**

LOPES, Ronan. PYTHON COM MONGODB (NOSQL) – GETTING STARTED. **Disponível em: https://ronanlopes.me/python-com-mongodb-nosql-getting-started/#:~:text=Novamente%2C%20para%20estabelecer%20uma%20conex%C3%A3o%20com%20o%20MongoDB,2%20from%20pymongo%20import%20MongoClient%20client%20%3D%20MongoClient%28%22mongodb%3A%2F%2Flocalhost%3A27017%2F%22%29. Acesso em: 28 mar. 2020.**

LÓSCIO, B. F.; OLIVEIRA, H. R. d.; PONTES, J. C. d. S. Nosql no desenvolvimento de aplicações web colaborativas**. VIII Simpósio Brasileiro de Sistemas Colaborativos, v. 10, n. 1, p. 11,2011.**

MARTINS FILHO. Análise de desempenho do uso do MongoDB em relação ao uso do PostgreSQL**. 2015. 53 f. Tese (Doutorado) - Curso de Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Pernambuco, 2015.**

MAURER, W. D.; LEWIS, T. G. Hash table methods. ACM Computing Surveys (CSUR), v. 7, n. 1, p. 5-19, 1975

MONGODB. MongoDB**. Disponível em: https://www.mongodb.com. Acesso em: 01 fev. 2014**

MPINDA, A.; BUNGAMA, P.; MASCHIETTO, L. From relational database to column-oriented NoSQL database: migration process. International Journal of Engineering Research & Technology, v. 4, n. 5, p. 399-403, 2015.

NAVATHE, Elmasri &. Sistemas de Banco de Dados**. Passo Fundo: Person Addison Wesley, 2005**

NOSQL: UM GUIA CONCISO PARA O MUNDO EMERGENTE DA PERSISTêNCIA POLIGLOTA. **Sao Paulo: Novatec, 01 jun. 2013.**

PERERA. Considerações sobre o Banco de Dados Apache Cassandra**. Disponível em: http://www.ibm.com/developerworks/br/library/os-apache-cassandra. Acesso em: 15 maio 2014.**

PROJETO DE BANCO DE DADOS. **Porto Alegre: Sagra Luzzatto, 01 maio 2006**

SILBERSCHATZ, ABRAHAM; KORTH, HENRY F.; SUDARSHAN, S. **Sistema De Banco De Dados.** 3. ed. São Paulo: Makron Books, 1999.

SISTEMAS DE BANCO DE DADOS. **Sao Paulo: Addison Wesley, 2011.** FACHIN, ODÍLIA. **Fundamentos de Metodologia**. 3. ed. São Paulo: Saraiva, 2001

SUISSA. Introdução ao NoSQL**. Disponível em: http://www.nosqlbr.com.br/introducao-ao-nosql.html. Acesso em: 01 fev. 2010.**