Banco de Dados
Não Relacional

mongoDB

Desenvolvimento de
Sistemas Multiplataformas

Introdução NoSQL

NoSQL (às vezes interpretado como Not Only SQL - Não Somente SQL) é um termo genérico para uma classe definida de banco de dados não-relacionais que rompe uma longa história de banco de dados relacionais com propriedades ACID.

Os bancos de dados que estão sob esses rótulos não podem exigir esquemas de tabela fixa e, geralmente, não suportam instruções e operações de junção SQL (JOIN). O termo NoSQL foi primeiramente utilizado em 1998 como o nome de um banco de dados relacional de código aberto que não possuía uma interface SQL.

Existem diversos tipos de bancos de dados não-relacionais, categorizados de acordo com a organização dos dados, ao contrário dos relacionais cujos dados são sempre armazenados em tabelas com linhas e colunas, sendo que existem relacionamentos entre as tabelas:

Introdução NoSQL

mongoDB

- **Orientado a Documentos**:
  Os mais populares, armazenam os dados em coleções de documentos;
- **Orientado a Grafos:**
  Armazenam os dados em grafos, com vértices e arestas;
- **Colunares:**
  Armazenam os dados agrupados pelas colunas em comum;
- **Chave-Valor**:
  Armazenam os dados no formato chave e valor, como em um array associativo, mas que também podem ter regras de conjuntos, filas, pilhas, etc;
- **Time-Series**:
  Armazenam os dados baseados em eventos temporais;
- **Multi-modelo:**
  Suportam o armazenamento dos dados em mais de um modelo (incluindo às vezes o modelo relacional) ao mesmo tempo;

Introdução NoSQL

mongoDB

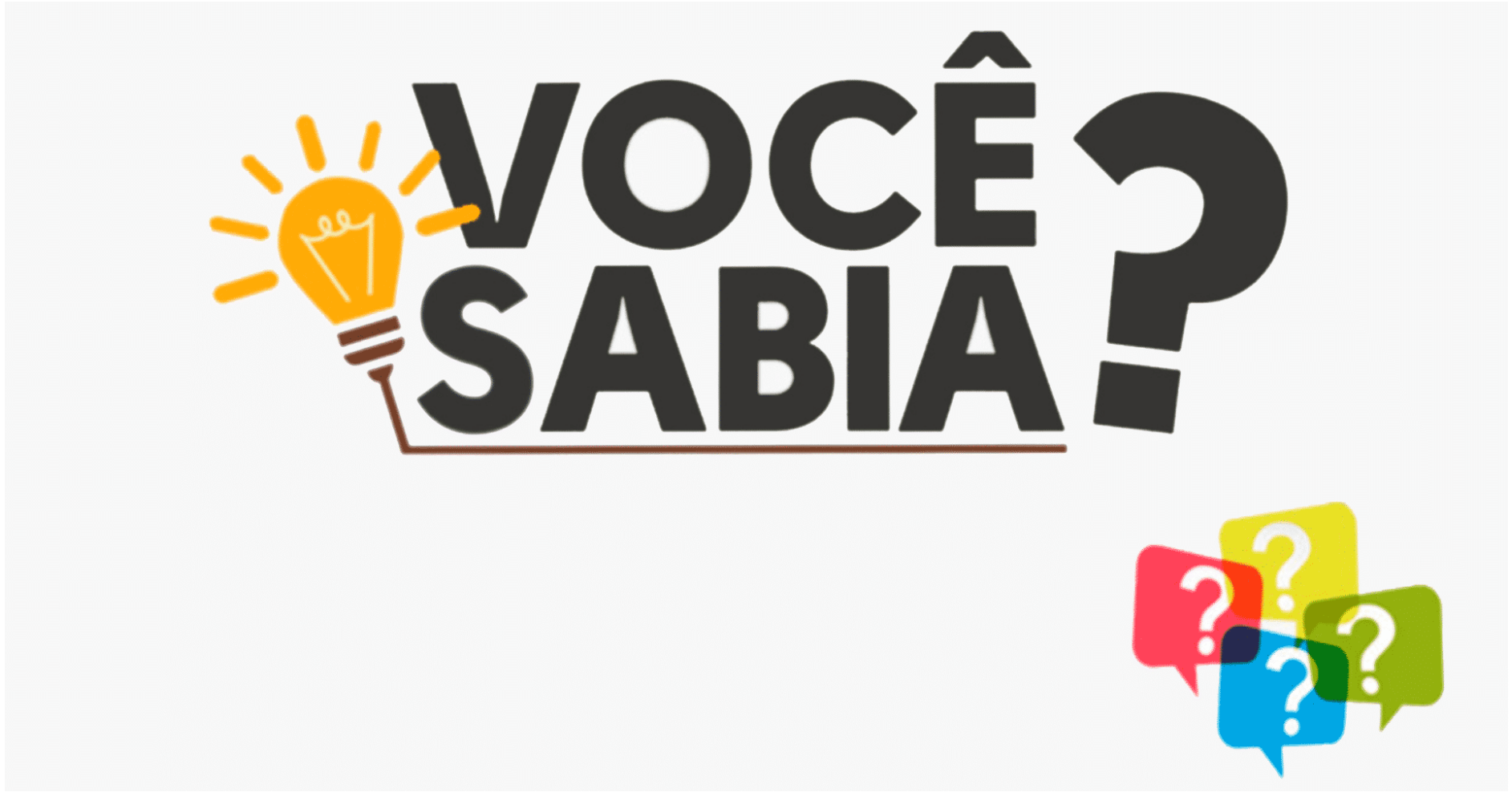
Cada modelo de armazenamento atende uma **necessidade específica da indústria de software** e **nenhum** se **propõe a substituir** de fato os bancos SQL, exceto nos cenários específicos para o qual foram projetados, sendo os bancos SQL uma solução genérica que atende de maneira razoável todos os casos.

Para cada modelo existem expoentes no mercado de tecnologia atual, tais como:

- **Orientado a Documentos**: MongoDB, Couchbase, CouchDB, RavenDB, DynamoDB;
- **Orientado a Grafos**: Neo4j;
- **Colunares**: Cassandra, HBase;
- **Chave-Valor**: Redis;
- **Time-Series**: InfluxDB;
- **Multi-Modelo**: ArangoDB, Cosmos DB;

Introdução NoSQL

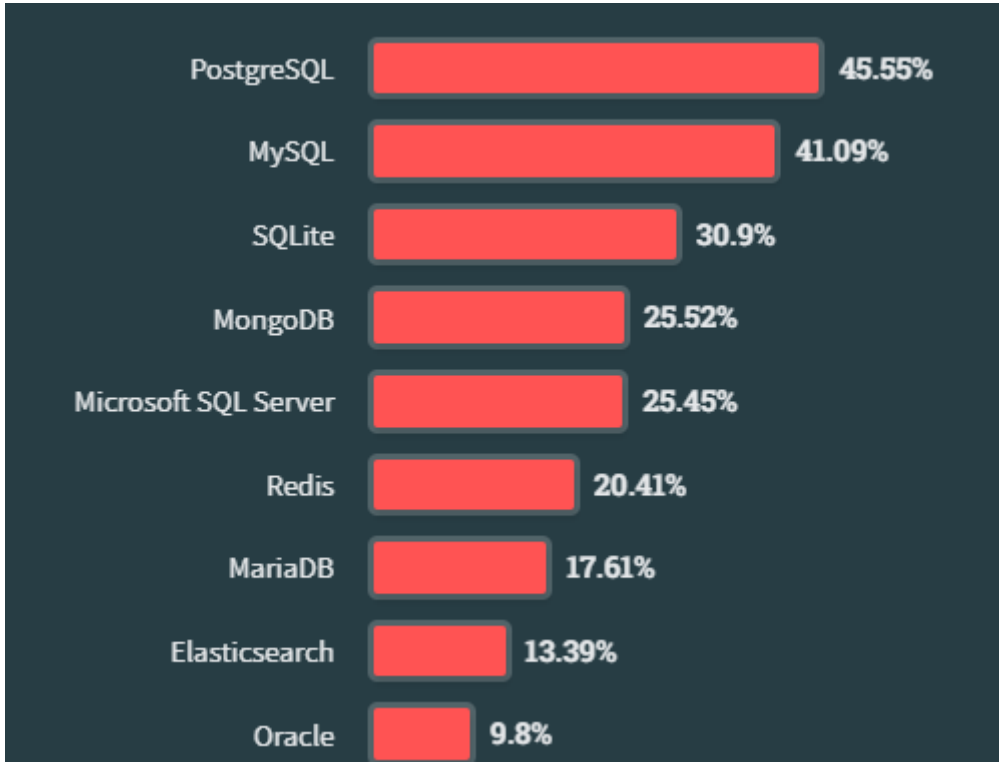
mongoDB



Introdução NoSQL

O site StackOverflow em 2023 mostra.

*"O **PostgreSQL** assumiu o primeiro lugar do **MySQL**. Desenvolvedores profissionais são mais propensos do que aqueles que aprendem a programar a usar **PostgreSQL** (**50%**) e aqueles que aprendem são mais propensos a usar **MySQL** (**54%**). **MongoDB** é usado por uma porcentagem semelhante de desenvolvedores profissionais e aqueles que aprendem a codificar e é o segundo banco de dados mais popular para aqueles que aprendem a programar (atrás do MySQL)".*



Fonte site https://survey.stackoverflow.co/2023/#technology-most-popular-technologies

Diferenças NoSQL vs SQL

NoSQL - MongoDB	SQL
Armazenam os dados em coleções de documentos, com campos e subdocumentos;	Armazenam os dados em tabelas, com linhas divididas em colunas;
Curva de aprendizagem menor;	Curva de aprendizagem maior
Não possuem relacionamentos entre as coleções e/ou documentos;	Possuem relacionamentos entre as diferentes tabelas e registros;
Foco em garantir escalabilidade, alta-disponibilidade e performance;	Foco em garantir o ACID;
Não há uso de transações e o ACID é garantido apenas a nível de documento; Transações devem ser garantidas a nível de aplicação;	Uso de transações para garantir commits e rollbacks;

Diferenças NoSQL vs SQL

NoSQL - MongoDB	SQL
Uso de comandos JavaScript para manipular o banco;	Uso da linguagem de consulta SQL para manipular o banco;
Cada campo de um documento pode armazenar múltiplos valores ou até mesmo outro documento;	Cada coluna de uma linha pode armazenar apenas um dado;
Não possuem "FKs" nem "JOINs";	Possuem chaves-estrangeiras e JOINs;
Schema variável conforme uso da aplicação;	Schema pré-definido e rígido;
Foco em acesso rápido de dados	Foco em não-repetição de dados
Maior consumo de disco (normalmente)	Menor consumo de disco;
Maior consumo de memória (normalmente)	Menor consumo de memoria

Quando usar MongoDB?

MongoDB foi criada com Big Data em mente.

Ele suporta tanto escalonamento horizontal quanto vertical usando replica sets (instâncias espelhadas) e sharding (dados distribuídos), tornando-o uma opção muito interessante para grandes volumes de dados, especialmente os desestruturados.

Dados desestruturados são um problema para a imensa maioria dos bancos de dados relacionais, mas não tanto para o MongoDB. Quando o seu schema é variável, é livre, usar MongoDB vem muito bem a calhar.

Os documentos BSON (JSON binário) do Mongo são schemaless e aceitam quase qualquer coisa que você quiser armazenar, sendo um mecanismo de persistência perfeito para uso com tecnologias que trabalham com JSON nativamente, como JavaScript (e consequentemente Node.js).

Quando usar MongoDB?

Os **use cases** listados e definidos pelo time de engenharia do **MongoDB** são os cenários em que ele melhor desempenha a sua função, o que corrobora com a máxima que "***MongoDB não substitui os bancos relacionais em todos cenários***, ***mas sim em alguns cenários específicos***", que, segundo o time do **MongoDB**, são:

Quando usar MongoDB?

- ✓ **Catalog**:
  Como catálogo de produtos (em ecommerces) e serviços;
- ✓ **Content Management**:
  Como em sistemas dos tipos ECM (gerenciadores de conteúdo corporativo) e CMS (blogs, portais, etc);
- ✓ **Internet of Things**:
  Como em sistemas com sensores real-time, como na indústria 4.0, em wearable computing (smartwatches e smart glasses, por exemplo);
- ✓ **Mobile**:
  Como em aplicações móveis de baixa latência e alta escala, como os grandes players que temos em nossos smartphones;
- ✓ **Personalization**:
  Como em sistemas com schema de dados personalizável para entregar conteúdo relevante aos usuários, muito usados no marketing digital;
- ✓ **Real-Time Analytics**:
  Como em sistemas de estatísticas real-time (analytics);
- ✓ **Single View**:
  Como em sistemas de dashboards, gerenciais e outros que agregam informações de diferentes fontes em uma base só;

Quando não usar MongoDB?

Nem tudo são flores e o **MongoDB** , ele **não resolve** todos os tipos de problemas de persistência existentes.

Você não deve utilizar **MongoDB** quando relacionamentos entre diversas entidades são importantes para o seu sistema.

Se for ter de usar muitas "chaves estrangeiras" e "JOINs", você está usando do jeito errado, ou, ao menos, não do jeito mais indicado.

Além disso, diversas entidades de pagamento (como bandeiras de cartão de crédito) não homologam sistemas cujos dados financeiros dos clientes não estejam em bancos de dados relacionais tradicionais.

Obviamente isso não impede completamente o uso de MongoDB em sistemas financeiros, mas o restringe apenas a certas partes (como dados públicos).

MongoDB o Queridinho

Algumas características do MongoDB têm levado a uma adoção sem precedentes deste banco de dados no mercado mundial de desenvolvimento de software. Algumas delas são:

- **MongoDB armazena os dados em formato JSON/BSON.**
  - Por ser orientado a documentos JSON (armazenados em modo binário, apelidado de JSON), muitas aplicações podem modelar informações de modo muito mais natural, pois os dados podem ser aninhados em hierarquias complexas e continuar a ser indexáveis e fáceis de buscar, igual ao que já é feito em JavaScript.

- **MongoDB foi criada com Big Data em mente**.
  - Ele suporta tanto escalonamento horizontal quanto vertical usando replica sets (instâncias espelhadas) e sharding (dados distribuídos), tornando-o uma opção muito interessante para grandes volumes de dados, especialmente os desestruturados.

- **MongoDB é muito leve e é multiplataforma**.
  - Isso permite que você consiga rodar seus projetos em servidores com o SO que quiser, diminuindo bastante seu TCO (principalmente se estava usando Oracle antes e/ou pagava licenças de Windows). A economia com servidores de alguns projetos meus chegou a 80% com a mudança de SQL Server para MongoDB.

MongoDB o Queridinho

- **Aceitação da tecnologia no ambiente corporativo**.
  - Hoje MongoDB é adotado por muitas empresas, incluindo grandes nomes como Gov.UK, ebay, McAfee, Adobe, Intuit, Citigroup, ADP, HSBC, Forbes, Verizon e Telefonica Digital.

- **Ferramentas**.
  - Ferramentas modernas de gerenciamento e modelagem específicas para bancos MongoDB como Compass, Robo3T (antigo Robomongo), Studio3T (antigo MongoChef), MongoView e muito mais.

- **Schemaless**.
  - Dados desestruturados são um problema para a imensa maioria dos bancos de dados relacionais, mas não tanto para o MongoDB. Quando o seu schema é variável, é livre, usar MongoDB vem muito bem a calhar. Os documentos BSON (JSON binário) do Mongo são schemaless e aceitam quase qualquer coisa que você quiser armazenar, sendo um mecanismo de persistência perfeito para uso com tecnologias que trabalham com JSON nativamente, como JavaScript (e consequentemente Node.js).

Comandos MongoDB

- **c:\mongo\bin> mongosh**
  - Este utilitário é o cliente de MongoDB (que se conecta no servidor para fazer consultas).
    - Após a conexão funcionar, se você olhar no prompt onde o servidor do Mongo está rodando, verá que uma conexão foi estabelecida e um sinal de "**>**" no console 'mongo' indicará que você já pode digitar os seus comandos e queries para enviar à essa conexão.

```
Microsoft Windows [versão 10.0.22631.3155]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\wagne>mongosh
Current Mongosh Log ID: 65e50071334e07bb97397fac
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.1.1
Using MongoDB:          7.0.4
Using Mongosh:          2.1.1
mongosh 2.1.5 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-02-22T11:27:54.083-03:00: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

Enterprise test>
```

- Ao contrário dos bancos relacionais, no MongoDB você não precisa construir a estrutura do seu banco previamente antes de sair utilizando ele. Tudo é criado conforme você for usando, o que não impede, é claro, que você planeje um pouco o que pretende fazer com o Mongo.

Comandos MongoDB

O comando abaixo no terminal cliente mostra os bancos existentes nesse servidor:

- **show databases ou show dbs**

```
Enterprise test> show databases
admin       40.00 KiB
config      72.00 KiB
escola      72.00 KiB
local       72.00 KiB
meubanco   184.00 KiB
test        40.00 KiB
Enterprise test>
```

```
Enterprise test> show dbs
admin       40.00 KiB
config      72.00 KiB
escola      72.00 KiB
local       72.00 KiB
meubanco   184.00 KiB
test        40.00 KiB
```

- Se é sua primeira execução ele deve listar as bases admin e local.
  - Não usaremos nenhuma delas.
    - Agora digite o seguinte comando para "use" o banco de dados "workshop" (*um banco que você sabe que não existe ainda*):

Comandos MongoDB

- **use workshop**
  - O terminal vai lhe avisar que o contexto da variável "**db**" mudou para o banco workshop, que nem mesmo existe ainda (mas não se preocupe com isso!).

```
Enterprise test> use workshop
switched to db workshop
Enterprise workshop>
```

  - Essa variável "**db**" representa agora o banco workshop e podemos verificar quais coleções existem atualmente neste banco usando o comando abaixo:

- **show collections**
  - Isso também não deve listar nada, mas não se importe com isso também, é porque ainda não salvamos nada em nosso banco.

```
Enterprise workshop> show collections

Enterprise workshop>
```

Comandos MongoDB

Assim como fazemos com objetos JavaScript que queremos chamar funções, usaremos a variável '**db**' para listar os documentos de uma coleção de customers (clientes) da seguinte forma:

- **db.customers.find()**
  - **find** é a função para fazer consultas no MongoDB e, quando usada sem parâmetros (os parâmetros vão entre os parênteses), retorna todos os documentos da coleção. Obviamente não listará nada pois não inserimos nenhum documento ainda, o que vamos fazer agora com a **função insertOne**:

```
Enterprise workshop> db.clientes.find()

Enterprise workshop>
```

Comandos MongoDB

A **função insertOne()** espera um documento JSON por parâmetro com as informações que queremos inserir, sendo que além dessas informações o MongoDB vai inserir um campo _id automático como chave primária desta coleção.

```
Enterprise workshop> db.clientes.insertOne({nome:"Wagner Toth", idade:43})
{
  acknowledged: true,
  insertedId: ObjectId('65e505f0334e07bb97397fad')
}
```

A **função insert()** ela reproduz o mesmo efeito porem é uma **função depreciada**, por este motivo ao inserirmos um dado na coleção, traz uma mensagem de alerta.

```
Enterprise workshop> db.cliente.insert({nome:"João da Silva", idade:34})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65e50614334e07bb97397fae') }
}
```

Comandos MongoDB

mongoDB

Para verificarmos se deu tudo certo utilizamos a função **find()**.

```
Enterprise workshop> db.clientes.find()
[
  {
    _id: ObjectId('65e505f0334e07bb97397fad'),
    nome: 'Wagner Toth',
    idade: 43
  },
  {
    _id: ObjectId('65e508c0334e07bb97397faf'),
    nome: 'João da Silva',
    idade: 34
  }
]
Enterprise workshop>
```

Obrigado!!!!