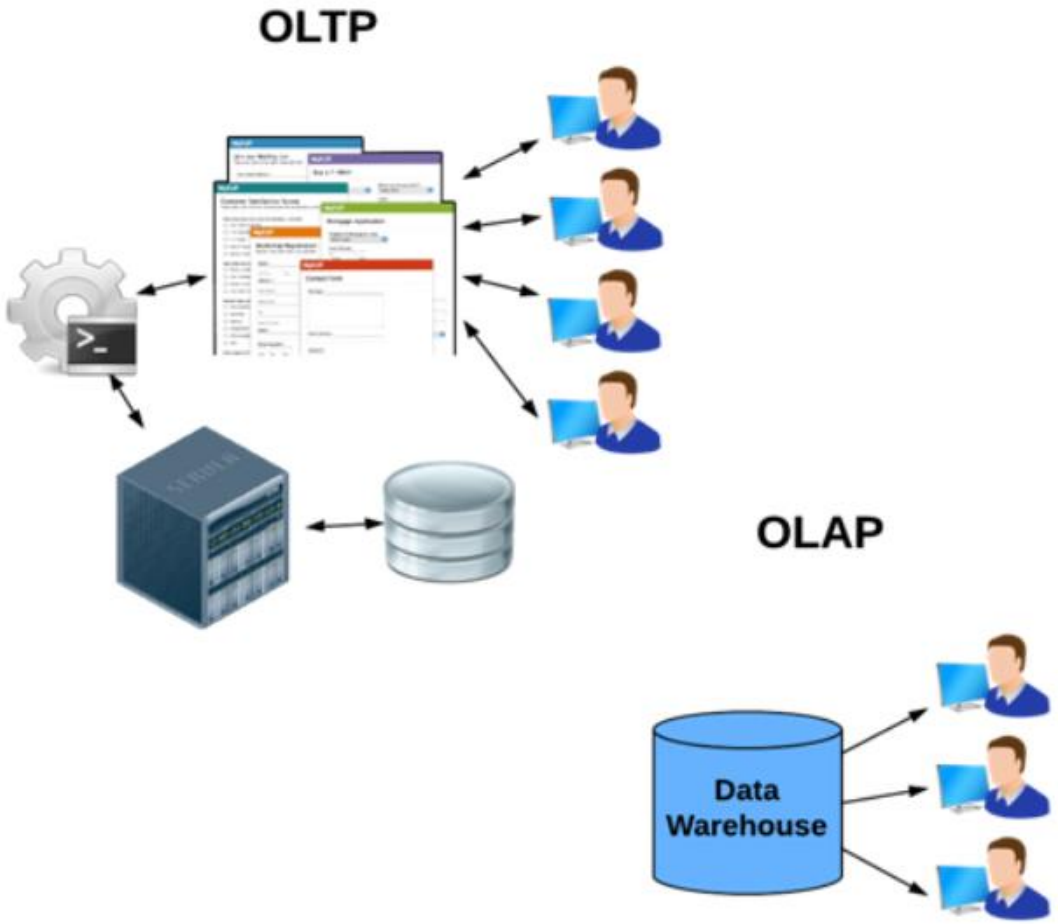
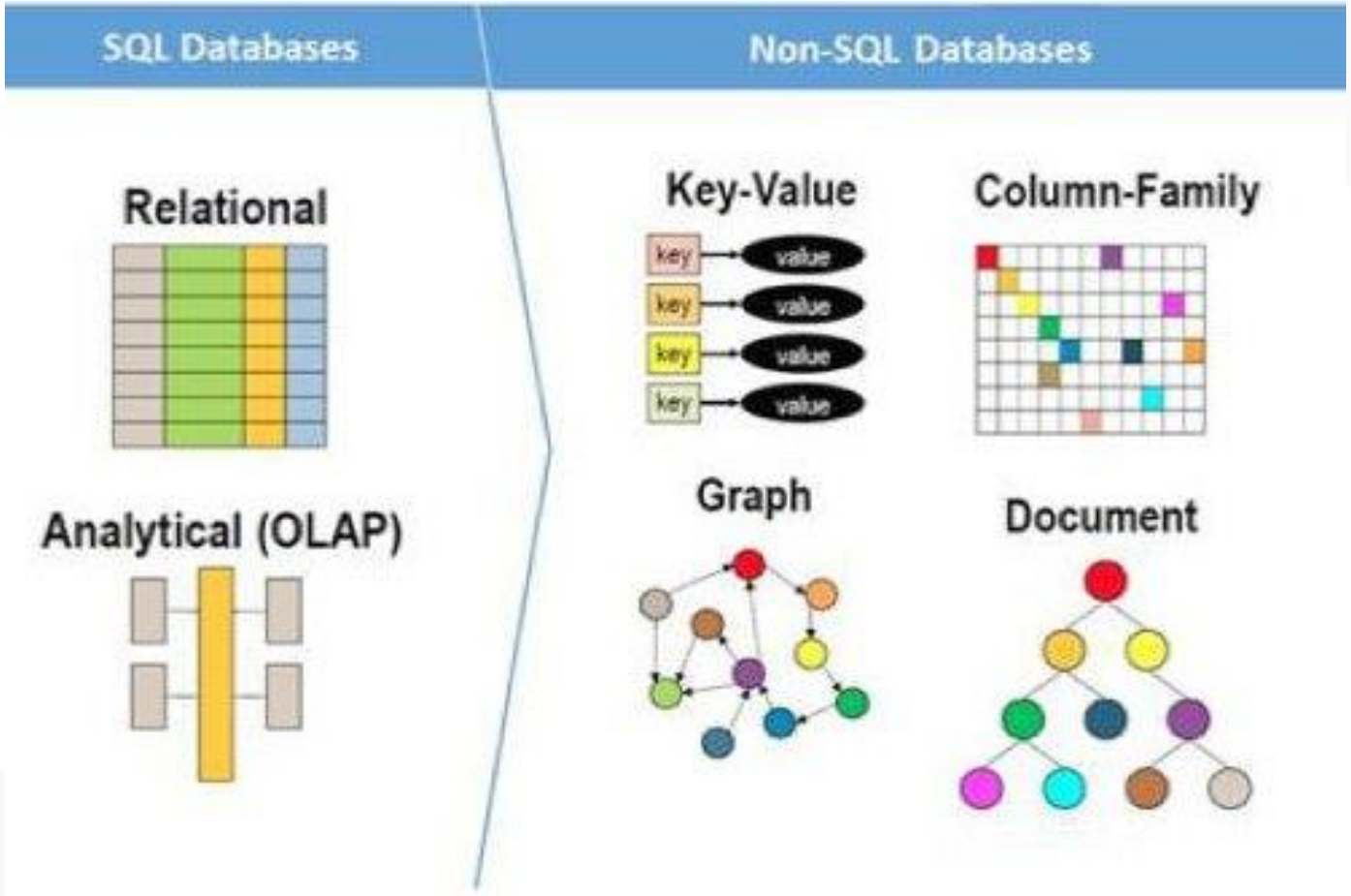
Camadas e Serviços de Consumo de Dados

Tipos de Bancos de Dados

BANCO DE DADOS RELACIONAL



BANCO DE DADOS RELACIONAL



RANKING BANCO DE DADOS

https://db-engines.com/en/ranking

O Xampp e o MySql são duas ferramentas distintas.

O MySql é um SGDB, somente o sistema para banco de dados.

O Xampp é um pacote de distribuição que contem além do MySQL o Apache, PHP, Perl, um servidor FTP e phpMyAdmin.

O MySQL contido no Xampp é o mesmo MySQL do site oficial.

No Xampp é só descompactar e usar, o MySQL esta com o usuário root sem senha.

RANKING BANCO DE DADOS

https://db-engines.com/en/ranking

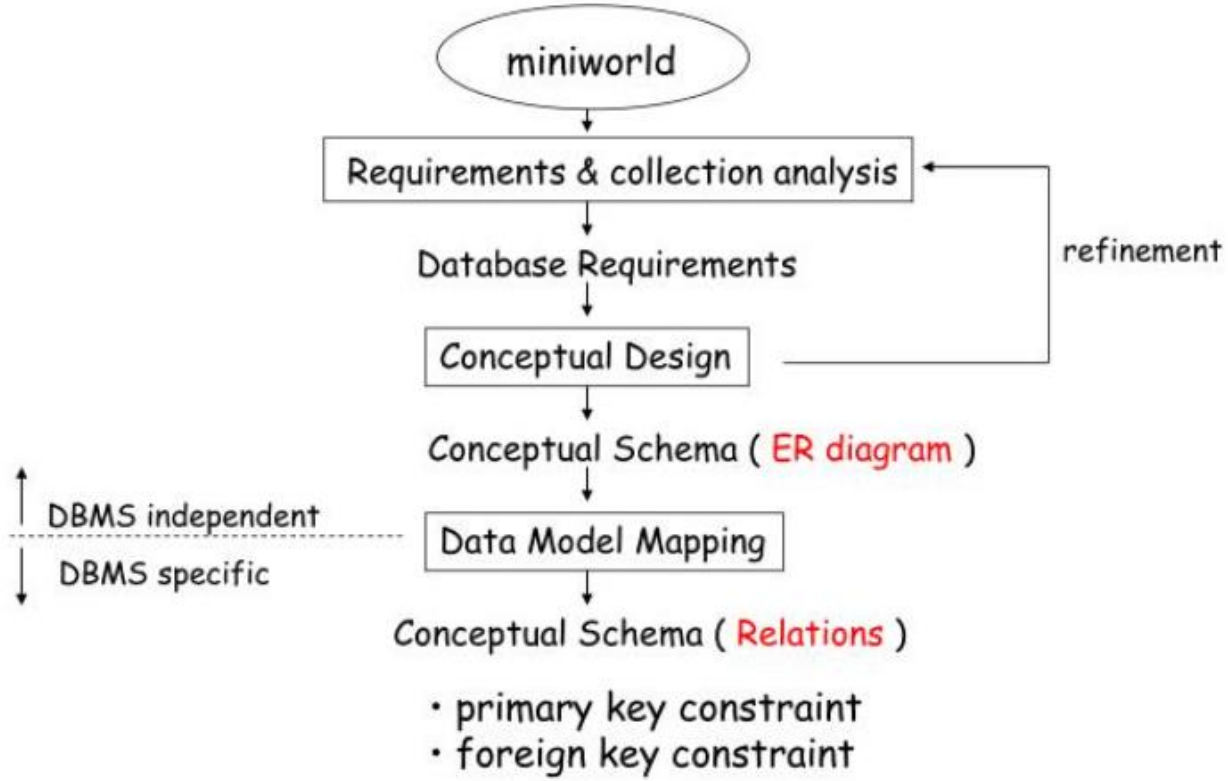
Rank			DBMS	Database Model	Score		
Aug 2020	Jul 2020	Aug 2019			Aug 2020	Jul 2020	Aug 2019
1.	1.	1.	Oracle	Relational, Multi-model	1355.16	+14.90	+15.68
2.	2.	2.	MySQL	Relational, Multi-model	1261.57	-6.93	+7.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1075.87	+16.15	-17.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	536.77	+9.76	+55.43
5.	5.	5.	MongoDB	Document, Multi-model	443.56	+0.08	+38.99
6.	6.	6.	IBM Db2	Relational, Multi-model	162.45	-0.72	-10.50
7.	↑ 8.	↑ 8.	Redis	Key-value, Multi-model	152.87	+2.83	+8.79
8.	↓ 7.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.32	+0.73	+3.23
9.	9.	↑ 11.	SQLite	Relational	126.82	-0.64	+4.10
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	119.86	+3.32	-15.47
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.84	-1.25	-5.37
12.	12.	↑ 13.	MariaDB	Relational, Multi-model	90.92	-0.21	+5.96
13.	13.	↓ 12.	Splunk	Search engine	89.91	+1.64	+4.03
14.	↑ 15.	↑ 15.	Teradata	Relational, Multi-model	76.78	+0.81	+0.14
15.	↓ 14.	↓ 14.	Hive	Relational	75.29	-1.14	-6.51

Bancos de Dados Relacionais

BANCO DE DADOS RELACIONAL

- Os dados são estruturados de acordo com o modelo relacional.
- SQL Server, Oracle, PostgreSQL, MySQL, DB2 e outros.
- Elementos básicos
  - Relações (tabelas) e registros (tuplas)
- Características fundamentais
  - ✓ Restrições de integridade
    - ✓ PK-primary key, FK-foreign key, UK-unique key, CK-check key, NN-not null
  - ✓ Normalização (formas normais)
  - ✓ Linguagem SQL (Structured Query Language)

MODELAGEM


miniworld
Requirements & collection analysis
Database Requirements
Conceptual Design
refinement
Conceptual Schema (ER diagram)
DBMS independent
Data Model Mapping
DBMS specific
Conceptual Schema (Relations)
• primary key constraint
• foreign key constraint

NORMALIZAÇÃO

Normalização é uma ferramenta usada no projeto lógico que serve para reestruturar tabelas e atributos, reduzindo assim redundâncias e permitindo o correto crescimento do banco de dados.

O processo de normalização conta com 6 formas:

- **1° Forma Normal.**
- **2° Forma Normal.**
- **3° Forma Normal.**
- **FNBC (Forma normal de Boyce e Codd):**
- **4° Forma Normal.**
- **5° Forma Normal.**

A partir da 3° Forma Normal diz-se que o banco de dados já se encontra normalizado. A FNBC, a 4FN e a 5FN são usadas para refinar ainda mais o banco.

NORMALIZAÇÃO

Cod Cliente	Nome Cliente	Tel 1	Tel 2	Endereço	Cod Produto	Nome Produto	Preço	Quantidade
1	Marcio Duarte	2098837	3298889	Rua A	1122	YYY	50	2
1	Marcio Duarte	2098837	3298889	Rua A	3344	KKK	120	1
2	Vitor da Silva	5412324	5544123	Rua B	9987	PPP	30	7
3	André Magalhães	6574565	6521787	Rua C	3344	KKK	120	5
2	Vitor da Silva	5412324	5544123	Rua B	1122	YYY	50	1

Problemas de Inserção:

Só é possível inserir um cliente se o mesmo adquirir um produto. Só é possível inserir um produto se algum cliente adquiri-lo.

Problemas de alteração:

Para atualizar o telefone do cliente ou o preço do produto, todos os outros registros deverão ser atualizados, ou seja, dado redundante.

Problemas de exclusão:

Se os produtos adquiridos por algum cliente forem excluídos, os dados cadastrais referente a esse cliente se perderão.

NORMALIZAÇÃO

1FN → 2FN → 3FN

- ✓ **1ª Forma Normal (1FN):** toda relação *deve ter uma chave primária* e deve-se garantir que todo atributo seja atômico. Atributos compostos devem ser separados.

Tabela (cod_cliente, nome_cliente, tel1, tel2, endereco, cod_produto, nome_produto, preco, quantidade) ✗

Cliente (cod_cliente, nome_cliente, tel1, tel2, rua, bairro, cidade, estado) ✓

Produto (cod_cliente, cod_produto, nome_produto, preco, quantidade) ✓

NORMALIZAÇÃO

1FN → 2FN → 3FN

✓ 2ª Forma Normal (2FN): toda relação deve estar na 1FN e *deve eliminar dependências funcionais parciais, ou seja, todo atributo não chave deve ser totalmente dependente da chave primária.*

Cliente (cod_cliente, nome_cliente, tel1, tel2, rua, bairro, cidade, estado) ✓ (não possui chave primária composta)

Produto (cod_cliente, cod_produto, nome_produto, preco, quantidade) ✗

cod_produto → nome_produto, preco (dependência parcial)

cod_cliente, cod_produto → quantidade (dependência total)

Resp: Produto (cod_produto, nome_produto, preco) ✓

Resp: Compra (cod_cliente, cod_produto, quantidade) ✓

NORMALIZAÇÃO

1FN → 2FN → 3FN

✓ **3ª Forma Normal (3FN):** toda relação deve estar na 2FN e devem-se eliminar dependências funcionais transitivas, ou seja, *todo atributo não chave deve ser mutuamente independente*.

Carro (placa, modelo, km_rodados, cod_fabricante, nome_fabricante) ✗

Placa, modelo → km_rodados

Placa, modelo →cod_fabricante

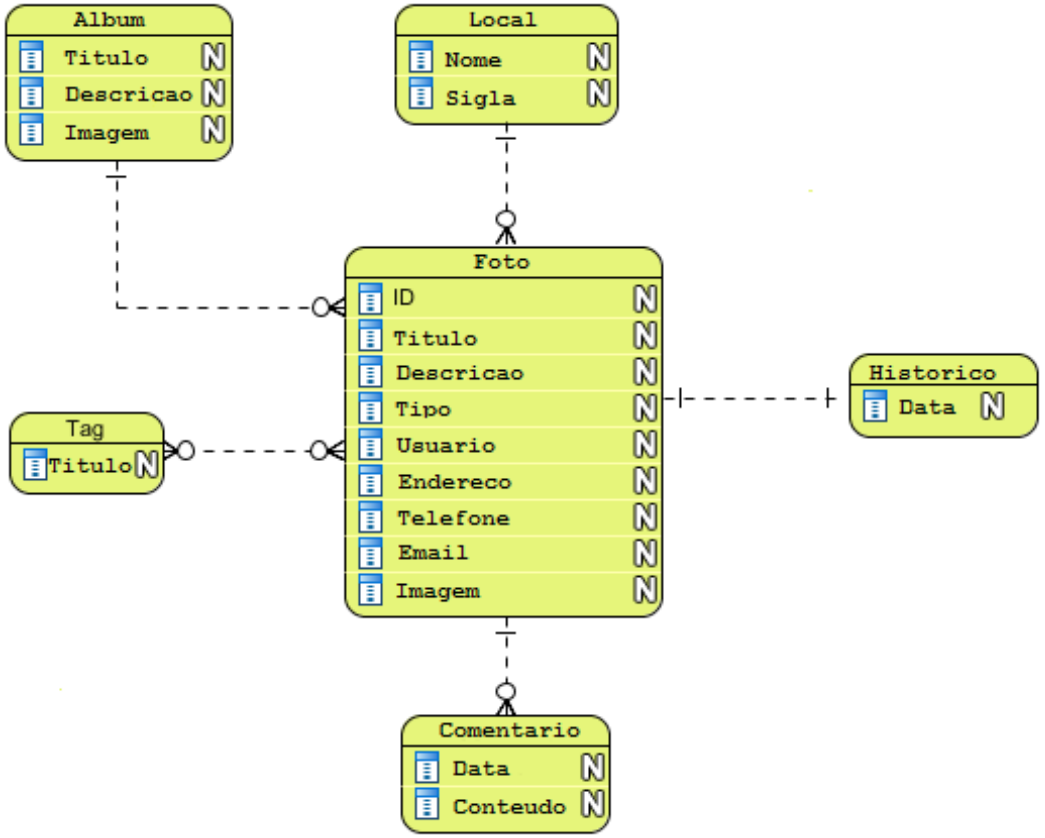
Placa, modelo → nome_fabricante

Cod_frabricante → nome_fabricante

Carro (placa, modelo, kmRodados, cod_fabricante) ✓

Fabricante (cod_fabricante, nome_fabricante) ✓

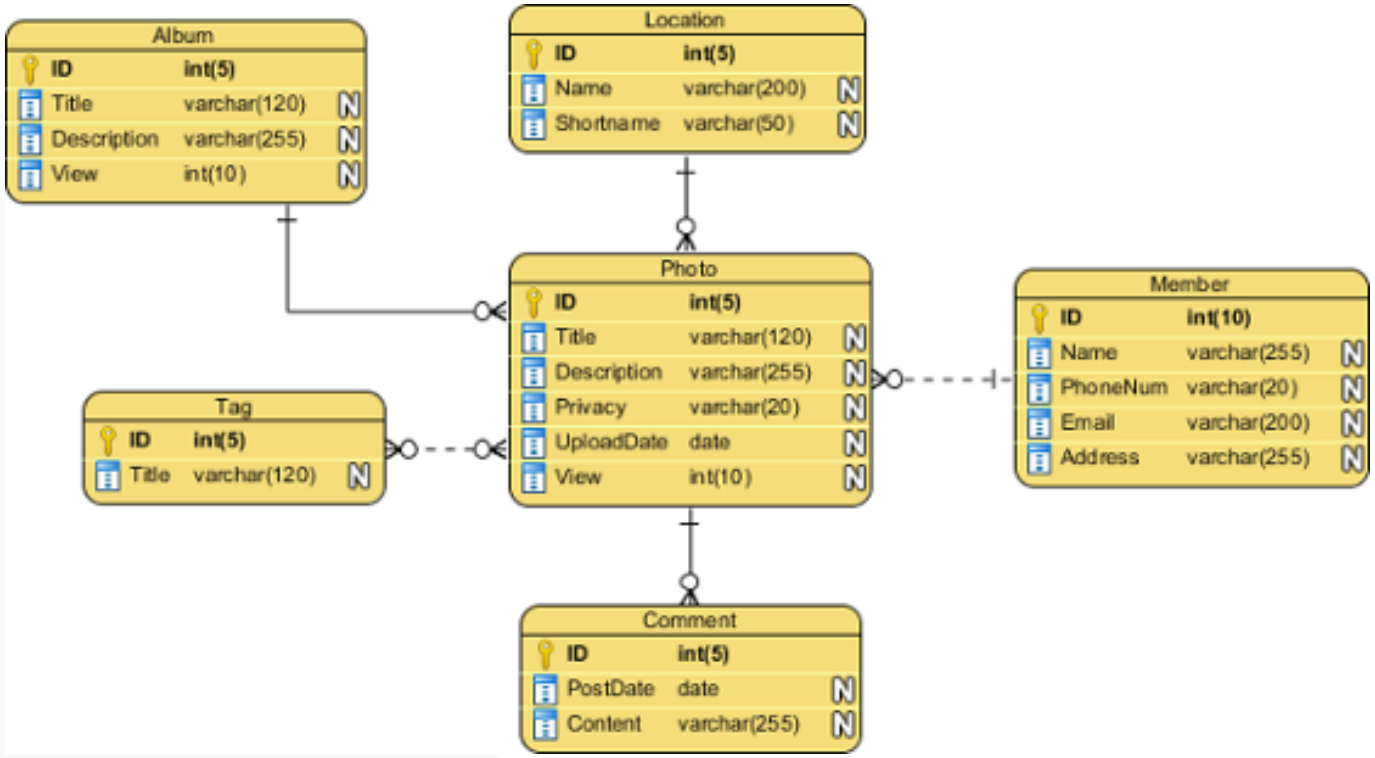
MODELO CONCEITUAL



Característica	Conceitual	Lógico	Físico
Nome de Entidade	✓	✓	
Relacionamentos de Entidade	✓	✓	
Atributos	✓	✓	
Chave Primária		✓	✓
Chave Estrangeira		✓	✓
Nome das Tabelas			✓
Nome das Colunas			✓
Tipo das Colunas			✓

Fonte: https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378_conceptual,l.html

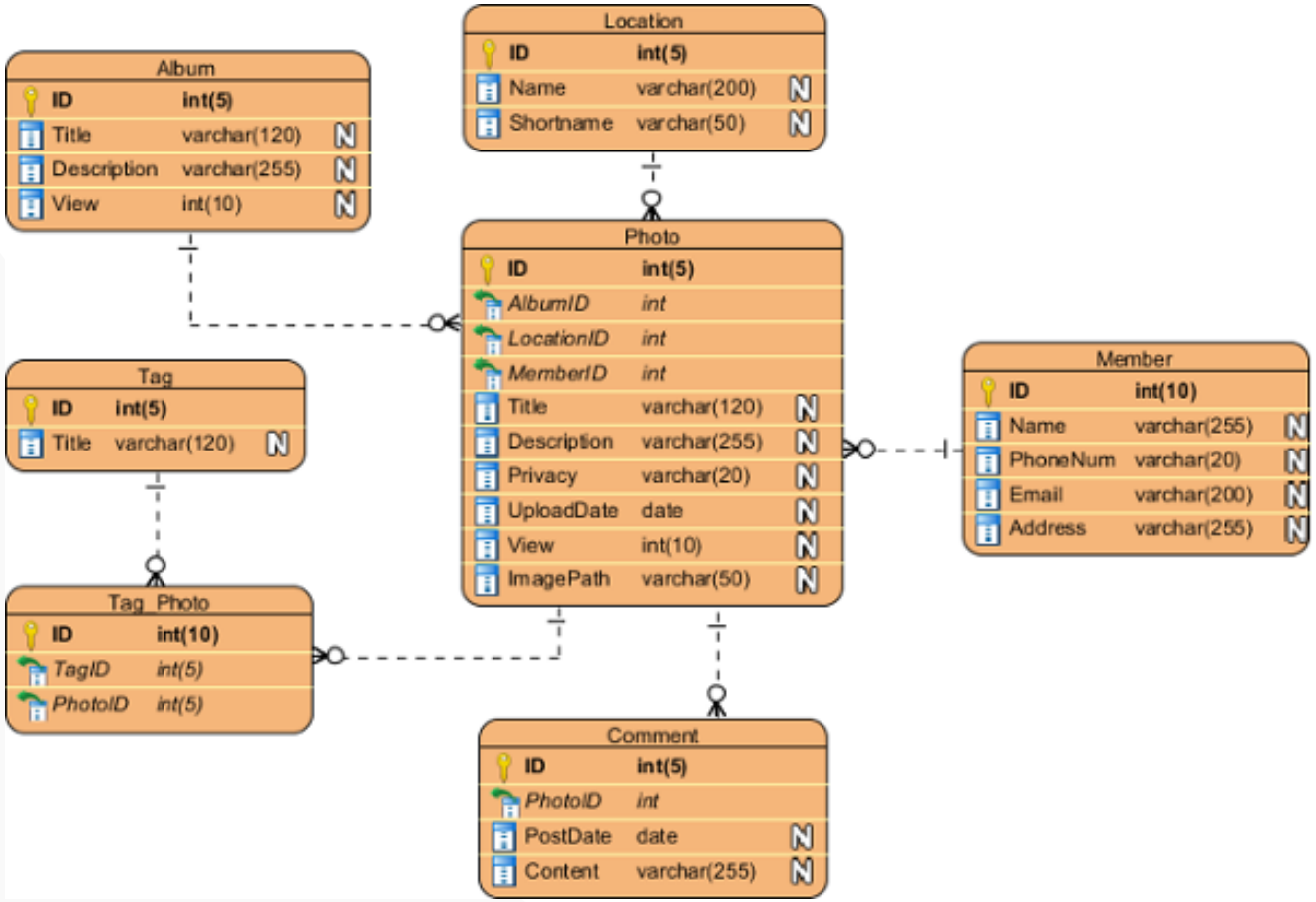
MODELO LÓGICO



Característica	Conceitual	Lógico	Físico
Nome de Entidade	✓	✓	
Relacionamentos de Entidade	✓	✓	
Atributos	✓	✓	
Chave Primária		✓	✓
Chave Estrangeira		✓	✓
Nome das Tabelas			✓
Nome das Colunas			✓
Tipo das Colunas			✓

Fonte: https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378_conceptual,l.html

MODELO FÍSICO



Característica	Conceitual	Lógico	Físico
Nome de Entidade	✓	✓	
Relacionamentos de Entidade	✓	✓	
Atributos	✓	✓	
Chave Primária		✓	✓
Chave Estrangeira		✓	✓
Nome das Tabelas			✓
Nome das Colunas			✓
Tipo das Colunas			✓

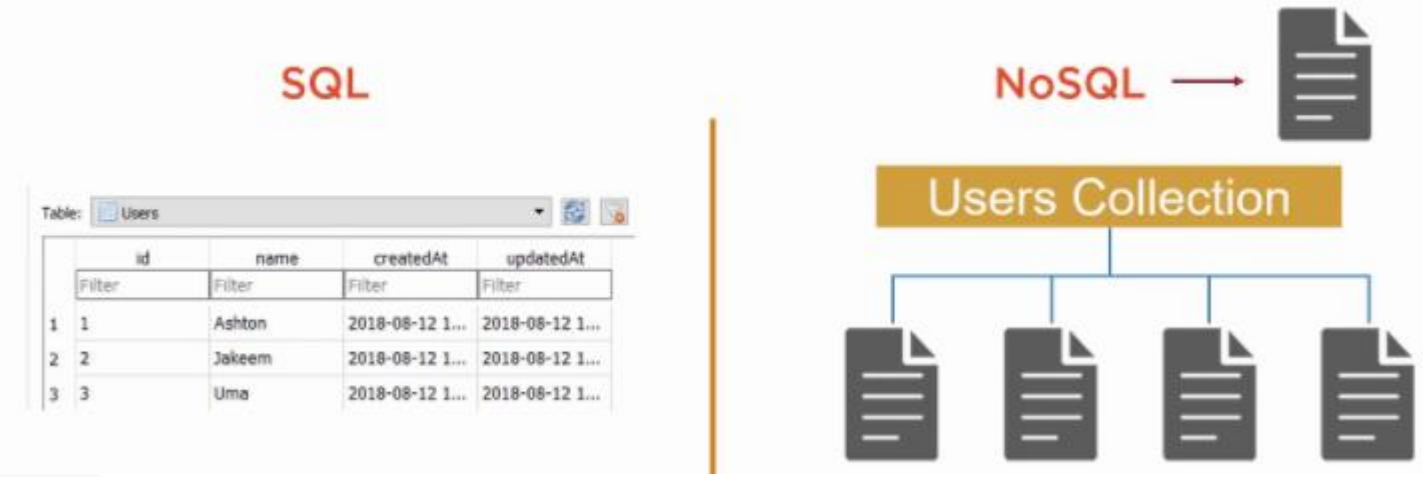
Fonte: https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378_conceptual,l.html

Bancos de Dados NoSQL

NoSQL

NoSQL é um termo genérico que define ***bancos de dados não-relacionais.***

A tecnologia NoSQL foi iniciada por companhias líderes da Internet como Google, Facebook, Amazon e LinkedIn, para superar as limitações de banco de dados relacional para aplicações web modernas.



NoSQL

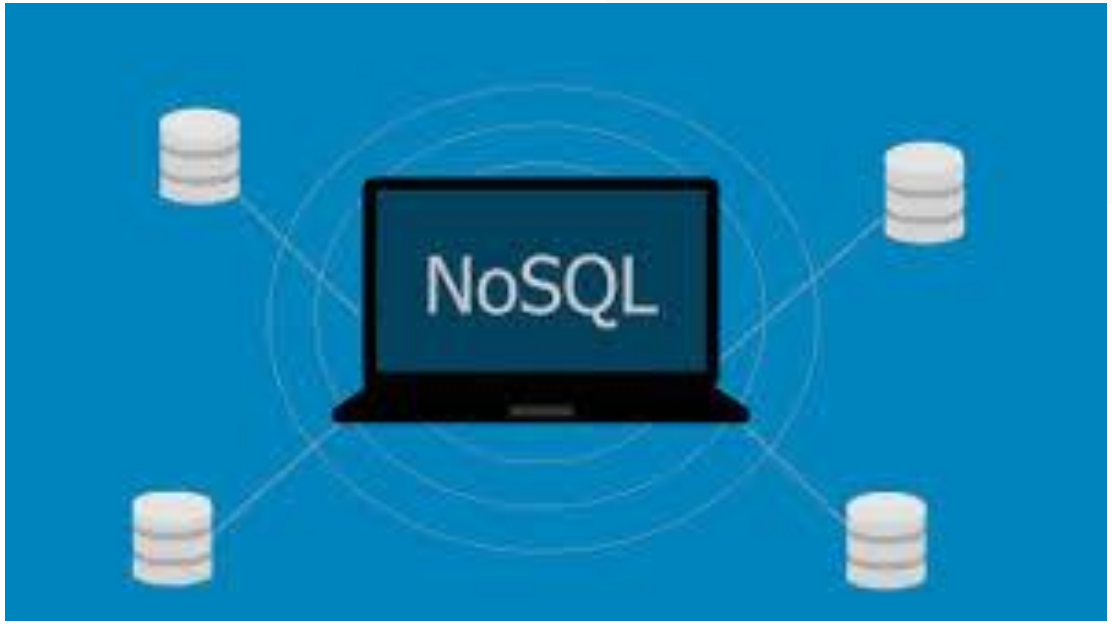
Não quer dizer que seus modelos não possuem relacionamentos e sim que ***não são orientados a tabelas***.

Not Only SQL (não apenas SQL).

Bancos de dados NoSQL são cada vez mais usados em ***Big Data e aplicações web de tempo real.***

NOSQL - CARACTERÍSTICAS

- Escalabilidade horizontal.
- Ausência de esquema ou esquema flexível.
- Suporte a replicação.
- API simples.
- Nem sempre prima pela consistência.



ALGUNS BANCOS NOSQL

Aerospike: Banco de dados NoSQL que oferece ***uma vantagem de velocidade de memória, atraindo empresas de anúncios de alta escala e aquelas que precisam de tempos de resposta em milissegundo***. Aerospike está apostando em novas categorias, incluindo jogos, e-commerce e segurança, onde a baixa latência é tudo.

Apache Cassandra: Os pontos fortes são a modelagem de dados NoSQL e ***escalabilidade linear flexível em hardware*** por conta do uso de cluster.

Amazon DynamoDB: foi desenvolvido pela Amazon para incrementar o seu e-commerce, possibilitando ter seus serviços altamente escaláveis. Inspirou o Cassandra, Riak, e outros projetos NoSQL.

ALGUNS BANCOS NOSQL

MongoDB: É o banco de dados mais popular NoSQL, com mais de sete milhões de downloads e centenas de milhares de implantações. ***Sua popularidade se deve à facilidade de desenvolvimento e manejo flexível dos dados***. Muito utilizado em aplicações de redes sociais web e móvel.

HBase: É o banco de dados que roda em cima do ***HDFS (Hadoop Distributed File System – sistema de arquivos distribuído)***, por isso dá aos usuários a capacidade única de trabalhar diretamente com os dados armazenados no Hadoop. As características incluem grande escalabilidade.

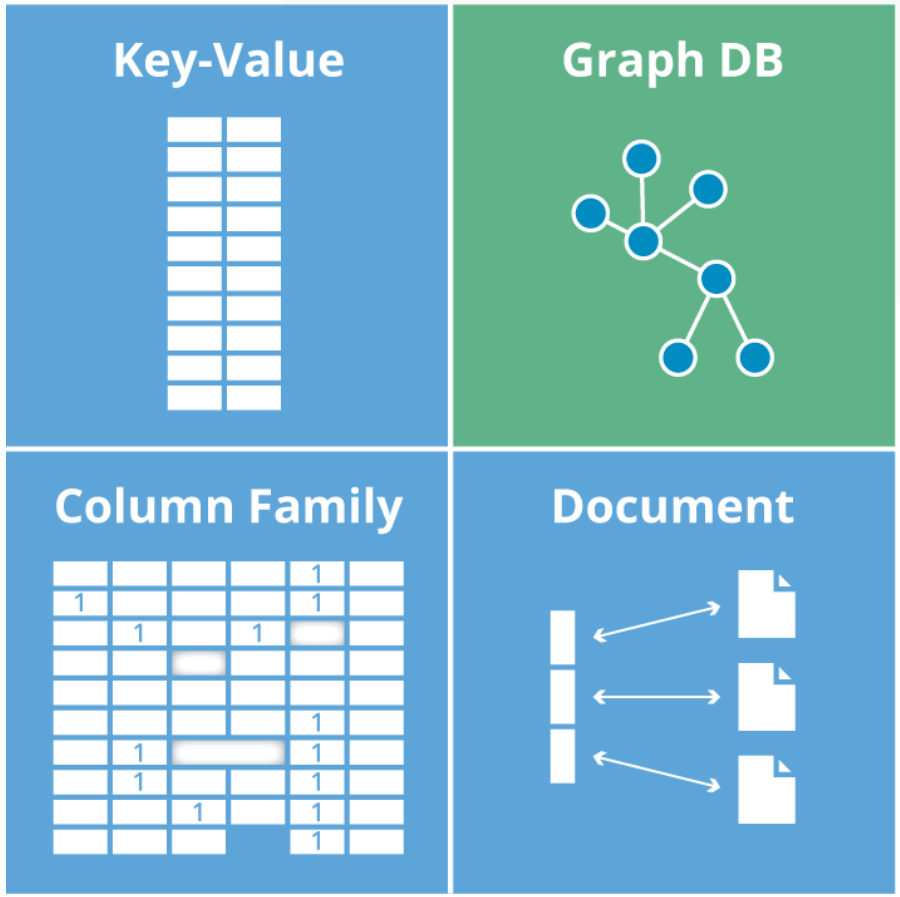
Redis: É o banco de dados NoSQL do tipo chave-valor mais conhecido. No mercado podemos encontrar diversas outras soluções que também são mecanismos de armazenamento baseado em chave-valor.

Modelos de Bancos NoSQL

MODELOS DE NOSQL

Temos quatro categorias do NoSQL:

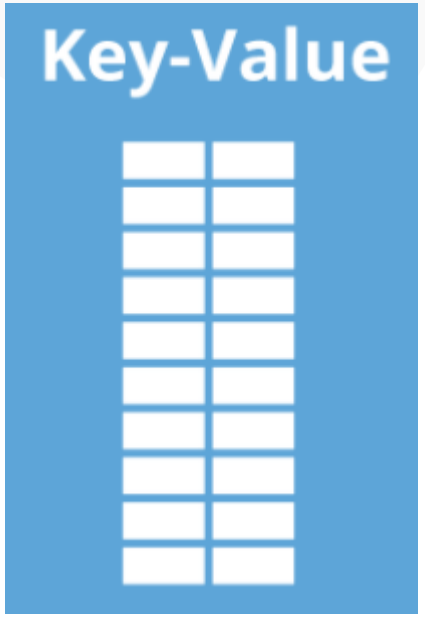
- Chave-valor (key-Value)
- Orientado a Grafos
- Orientado a Coluna (Column Family)
- Orientado a Documentos



CHAVE-VALOR (KEY-VALUE)

- Modelo mais simples.
- Permite a visualização do banco como uma grande tabela.
- Todo o banco é composto por um conjunto de chaves que estão associadas a um único valor.

Chave (Campo)	Valor (Instancia)
Nome	Hélio Rodrigues
Idade	45
Sexo	Masculino
Fone	99 99999999



ORIENTADO A GRAFOS

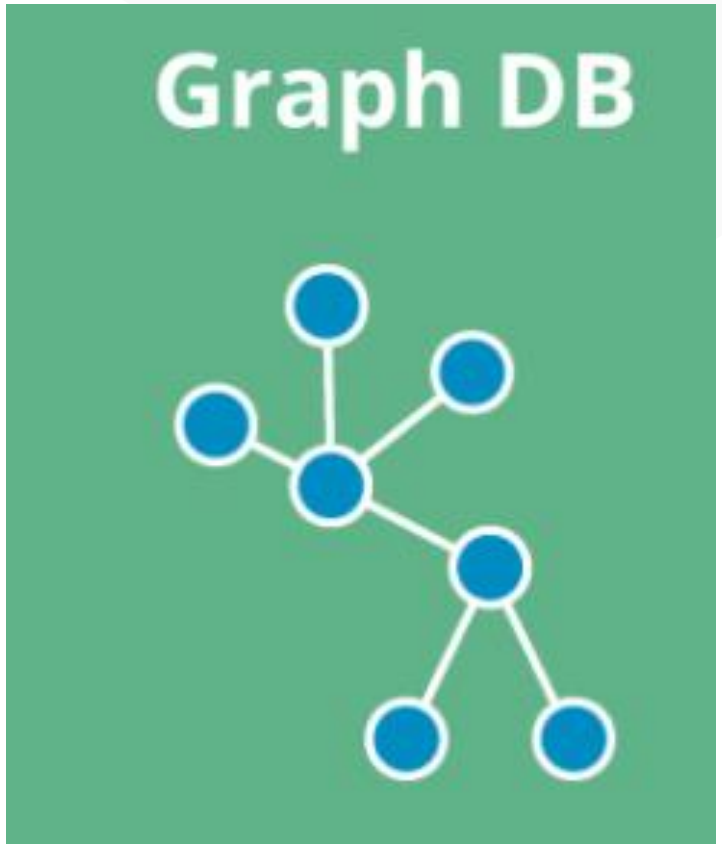
Este modelo possui três componentes básicos:

- Nós (vértices dos grafos).
- Os relacionamentos (arestas).
- As propriedades (conhecidos também como atributos).

É visto como um multigrafo rotulado e direcionado, onde cada par de nós ***pode ser conectado por mais de uma aresta.***

A utilização deste modelo é muito útil quando é necessário fazer consultas demasiadamente complexas.

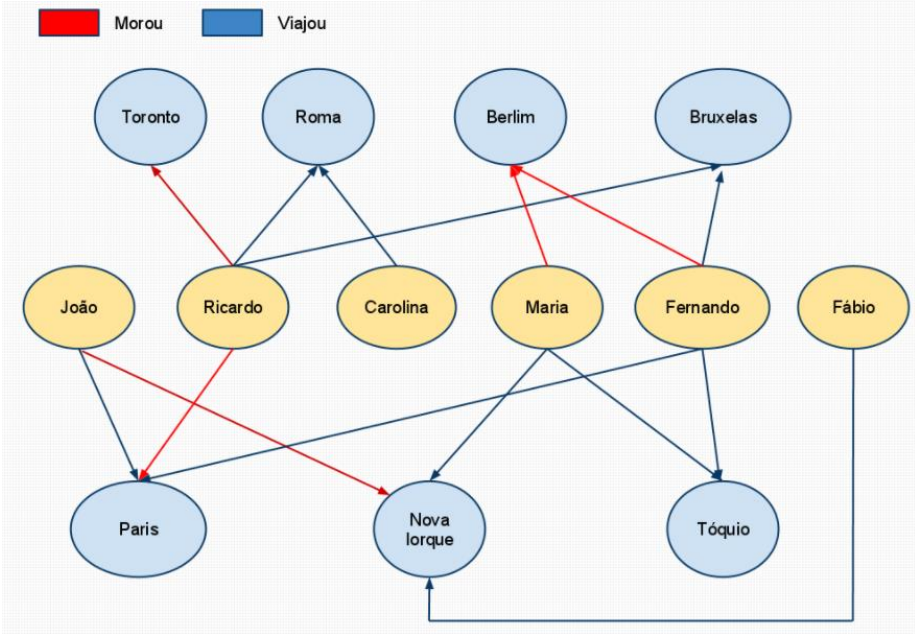
O modelo orientado a grafos possui uma alta performance, permitindo um bom desempenho nas aplicações.



ORIENTADO A GRAFOS

Imagine uma aplicação que mantêm informações relativas à viagem. Uma consulta pertinente seria: “Quais cidades foram visitadas anteriormente por pessoas que foram para Nova Iorque?”

Temos diversas pessoas: João, Ricardo, Carolina, Maria, Fernando e Fábio que representam nós do grafo e estão conectadas a cidades que visitaram ou residiram.



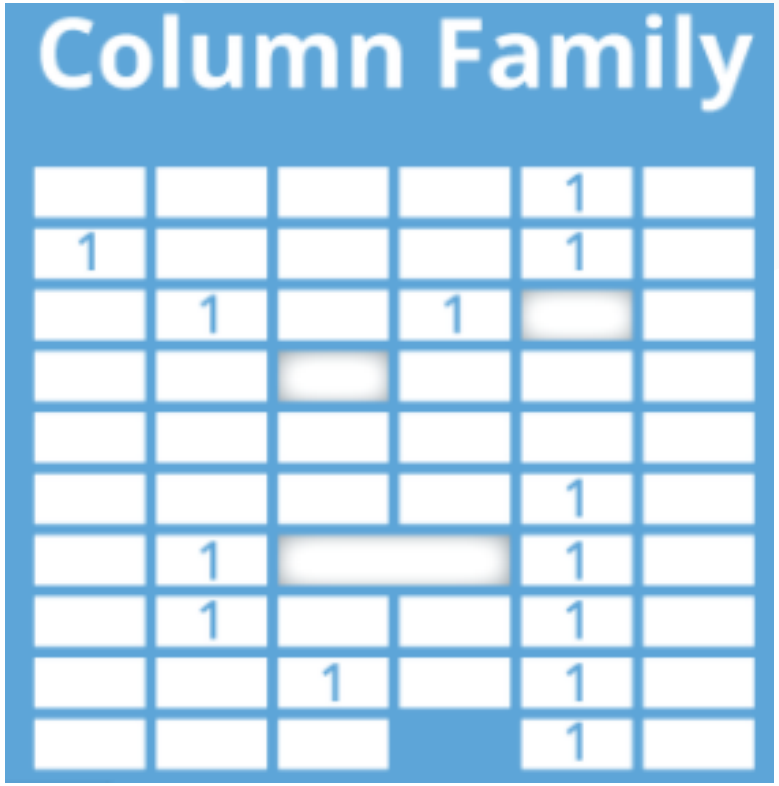
ORIENTADO A COLUNAS

Este tipo de banco de dados ***foi criado para armazenar e processar uma grande quantidade de dados distribuídos em diversas máquinas.***

Aqui existem ***as chaves, mas neste caso, elas apontam para atributos ou colunas múltiplas.***

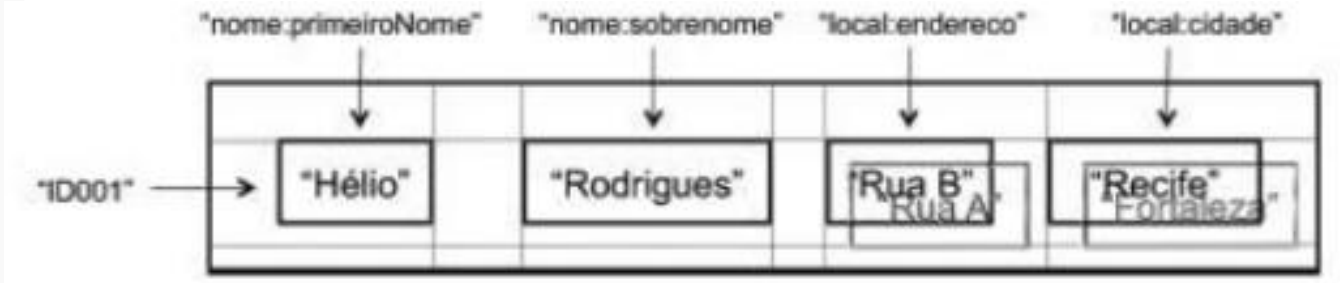
Os dados são indexados por uma tripla (coluna, linha e timestamp), ***a coluna e linha são identificadas por chaves e o timestamp permite diferenciar múltiplas versões de um mesmo dado.***

Como o próprio nome sugere, as colunas ***são organizadas por famílias de colunas.*** Demonstra maior complexidade que o de chave-valor.



ORIENTADO A COLUNAS

Relational Model	Cassandra Model
Database	Keyspace
Table	Column Family (CF)
Primary key	Row key
Column name	Column name/key
Column value	Column value

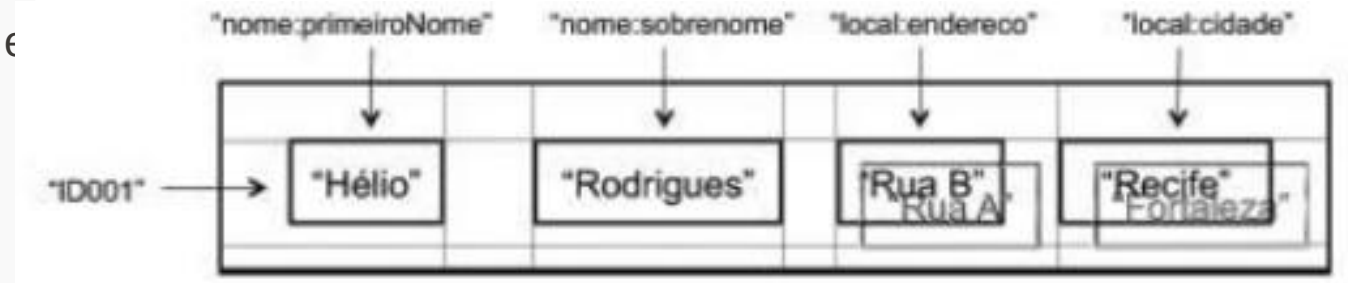


ORIENTADO A COLUNAS

Relational Model	Cassandra Model
Database	Keyspace
Table	Column Family (CF)
Primary key	Row key
Column name	Column name/key
Column value	Column value

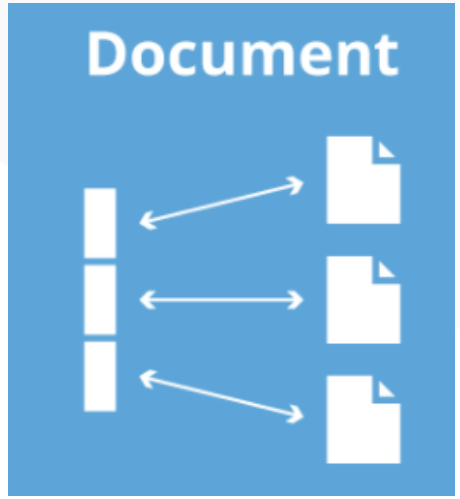
Podemos
sobrenon
“nome”.
família lo

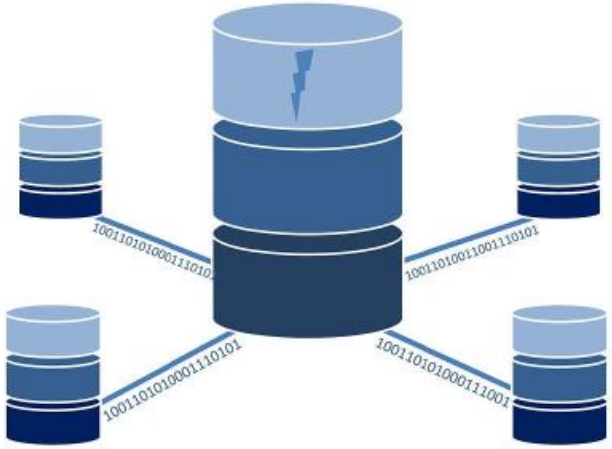
É interess
Como a
interesse seja buscar o primeiro nome da linha 001, todas as colunas serão
retornadas quando e



ORIENTADO A DOCUMENTOS

- Armazenam chave/valor.
- O valor é um documento estruturado e indexado, com metadados.
- Valor (Documento), pode ser consultado.
- JSON: JavaScript Object Notation.
  - Feito para troca de dados.
  - Mais compacto e legível que XML.





JSON

É um acrônimo de JavaScript Object Notation, e tem um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas, especificado por Douglas Crockford em 2000, que utiliza texto legível a humanos, no formato atributo-valor.

```
{
    "id":1,
    "nome":"Alexandre Gama",
    "endereco":"R. Qualquer"
}
```

MONGODB

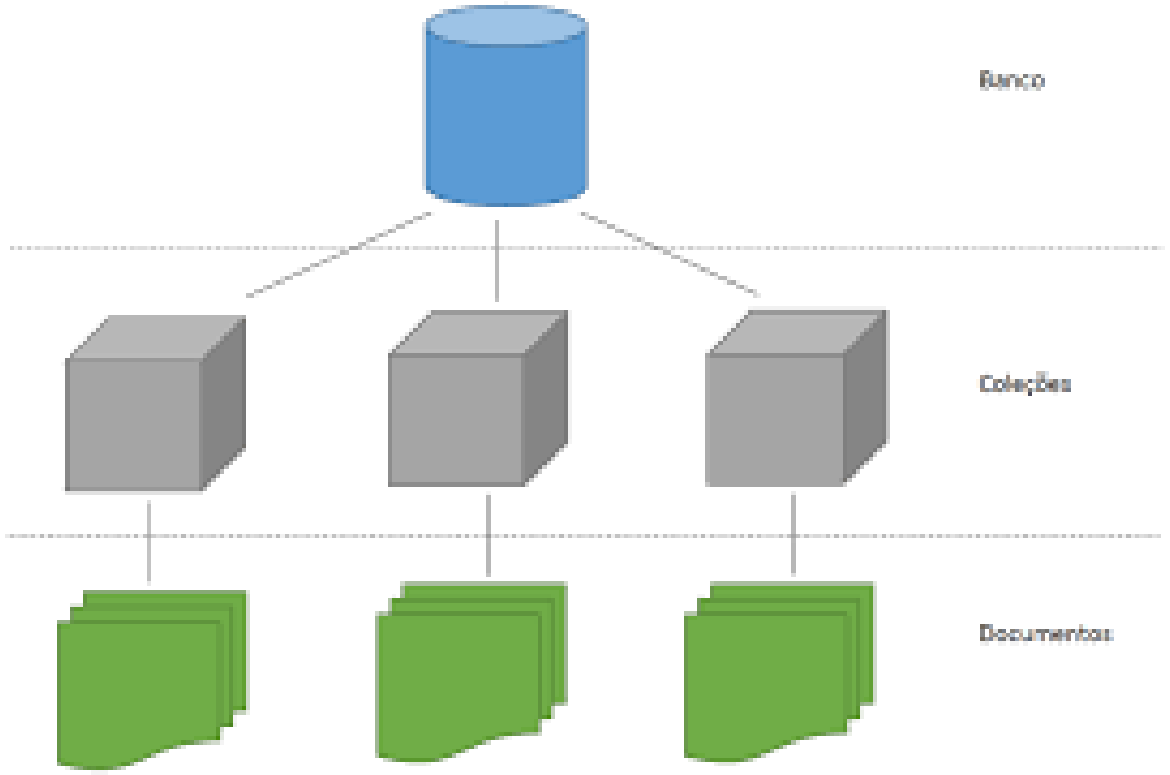
- ✓ Open source.
- ✓ Multiplataforma.
- ✓ Escalável.
- ✓ Orientado a documentos: JSON.
- ✓ Permite documentos aninhados.
- ✓ Indexados: pode-se buscar o conteúdo dos documentos.
- ✓ Não tem schema fixo.
- ✓ Não tem integridade referencial.

MONGODB

- ✓ Open source.
- ✓ Multiplataforma.
- ✓ Escalável.

Relacional	MongoDB
Banco de Dados	Banco de Dados
Tabela	Coleção
Linha	Documento
Coluna	Campo

MONGODB - DEFINIÇÕES



Bancos de Dados NoSQL - Schema

AUSÊNCIA DE ESQUEMA

Ausência de esquema (Schema-free) ou esquema flexível é uma outra característica em bancos de dados NoSQL.

Basicamente ***é a ausência parcial ou total de esquema que define a estrutura de dados.***

É justamente essa ausência de esquema que ***facilita uma alta escalabilidade*** e alta disponibilidade, ***mas em contrapartida não há a garantia de integridade dos dados, fato este que não ocorre no Modelo Relacional.***

AUSÊNCIA DE ESQUEMA

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987

Document 1

```
{
  "id": "1",
  "name": "John Smith",
  "isActive": true,
  "dob": "1964-30-08"
}
```

Document 2

```
{
  "id": "2",
  "fullName": "Sarah Jones",
  "isActive": false,
  "dob": "2002-02-18"
}
```

Document 3

```
{
  "id": "3",
  "fullName":
  {
    "first": "Adam",
    "last": "Stark"
  },
  "isActive": true,
  "dob": "2015-04-19"
}
```

Bancos de Dados NewSQL

NEWSQL

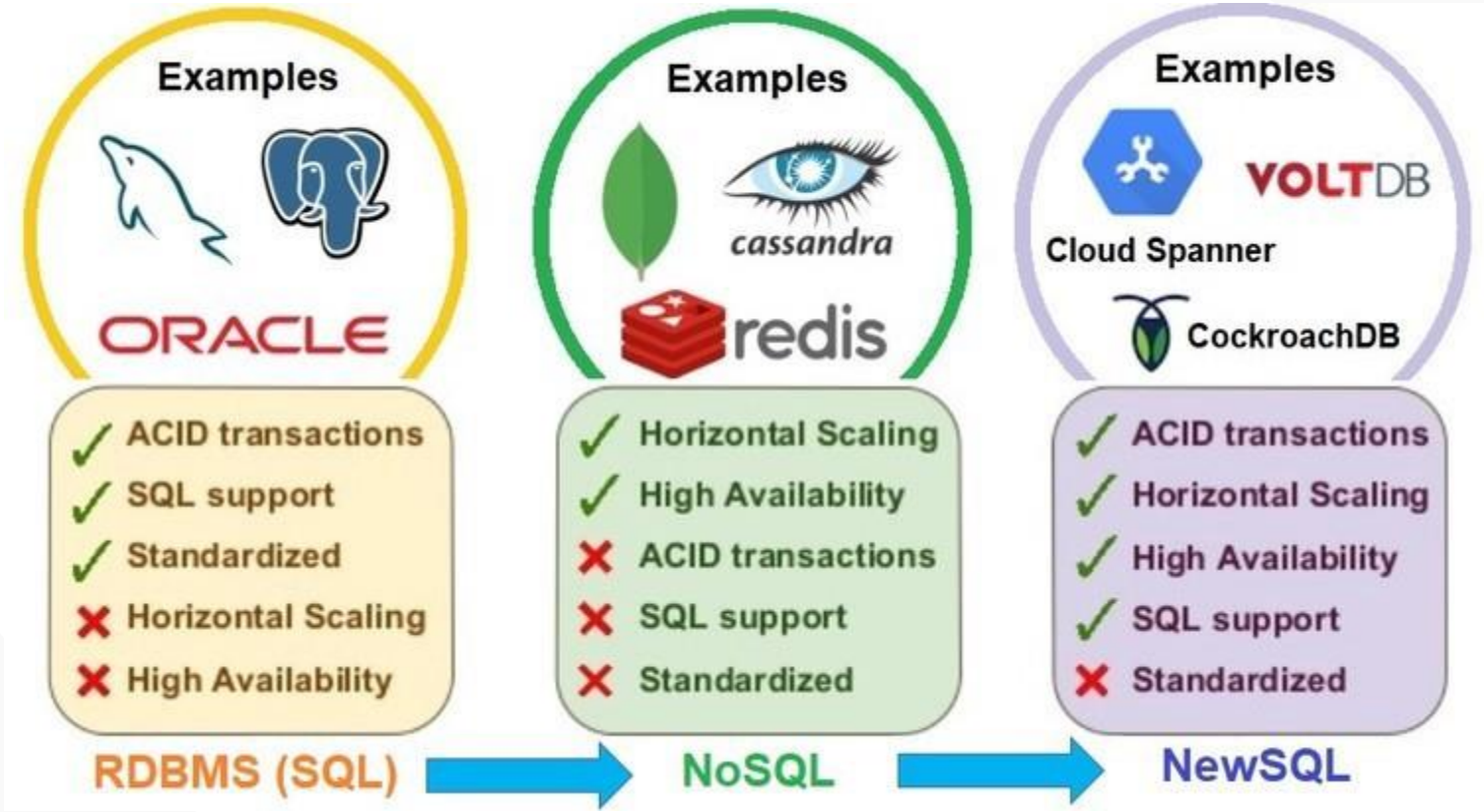
Pode ser definido como uma classe de SGBDs relacionais modernos que buscam fornecer o mesmo desempenho escalonável do NoSQL para cargas de trabalho OLTP e, simultaneamente, ***garantir a conformidade ACID para transações como no RDBMS.***

Basicamente esses sistemas desejam alcançar a escalabilidade do NoSQL sem ter que descartar o modelo relacional com SQL e suporte a transações do DBMS legado.

SQL, NOSQL OU NEWSQL

	Old SQL	NoSQL	NewSQL
Relational	Yes	No	Yes
SQL	Yes	No	Yes
ACID transactions	Yes	No	Yes
Horizontal scalability	No	Yes	Yes
Performance / big volume	No	Yes	Yes
Schema-less	No	Yes	No

SQL, NOSQL OU NEWSQL



ALGUNS BANCOS NEWSQL

MemSQL: Como o próprio nome sugere, ***é operado em memória***, e é um sistema de banco de dados de alta escala por sua combinação de desempenho e compatibilidade com o SQL transacional e ACID na memória, adicionando uma interface relacional em uma camada de dados in-memory.

VoltDB: Projetado por vários pesquisadores de sistema de banco de dados bem conhecidos, esse banco ***oferece a velocidade e a alta escalabilidade dos bancos de dados NoSQL, mas com garantias ACID***, e sua latência em milissegundo e ***integração com Hadoop.***

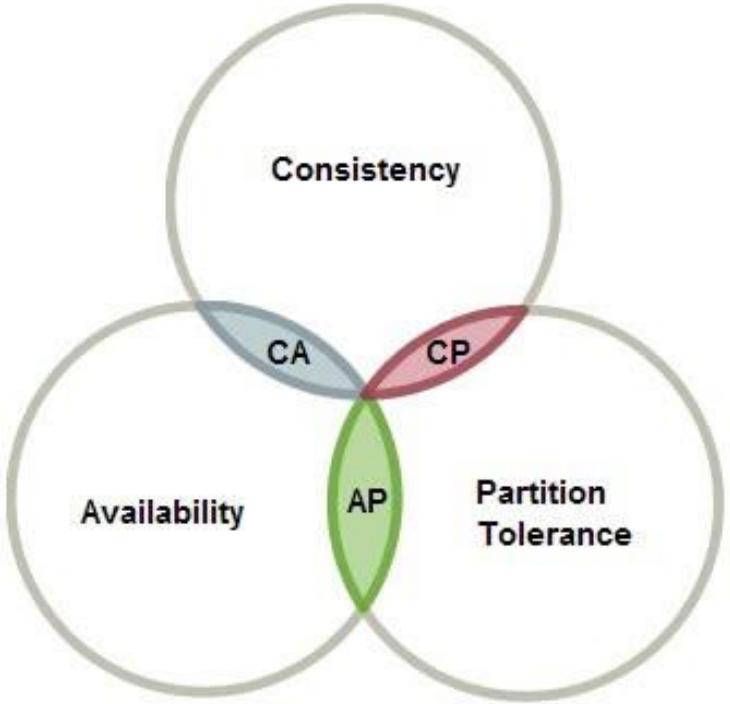
SQLFire: Servidor de ***banco de dados NewSQL da VMware***, desenvolvido para escalar em plataformas nas nuvens e tomar as vantagens de infraestrutura virtualizadas.

MariaDB: foi ***desenvolvido pelo criador do MySQL*** e é totalmente compatível com o MySQL. Também pode interagir com os bancos de dados NoSQL, como Cassandra e LevelDB.

Teorema CAP

TEOREMA CAP

De acordo com o teorema **CAP**, ***um sistema distribuído de bancos de dados somente pode operar com dois desses comportamentos ao mesmo tempo***, mas ***jamais com os três simultaneamente***.

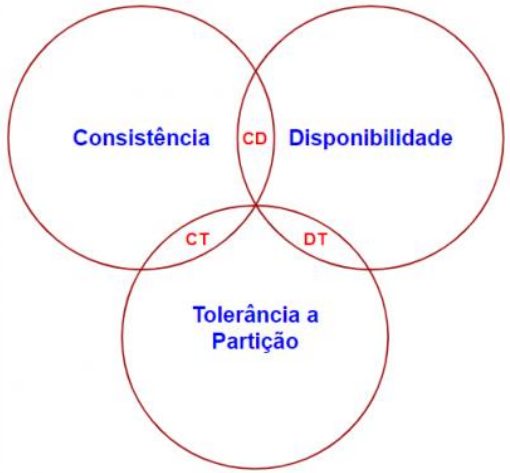


CONSISTÊNCIA EVENTUAL

É um conceito interessante derivado do teorema CAP.

O sistema prioriza as escritas de dados (armazenamento), sendo o sincronismo entre os nós do servidor realizado em um momento posterior – o que causa um pequeno intervalo de tempo no qual o sistema como um todo é inconsistente.

Para isso, ***são implementadas as propriedades Disponibilidade e Tolerância a Partição.***



CONSISTÊNCIA EVENTUAL

Exemplos de sistemas de bancos de dados que implementam a consistência eventual são o ***MongoDB, Cassandra e RavenDB (bancos NoSQL),*** entre outros.

Em Bancos Relacionais, é muito comum implementar as propriedades ***Consistência*** e ***Disponibilidade***. Como exemplos, citamos os SGBDRs ***Oracle, MySQL, PostgreSQL, SQL Server*** e outros.

Ao criar um banco de dados distribuído é importante ***ter em mente o teorema CAP***.

Você terá de decidir se o banco será consistente ou disponível, pois bancos de dados distribuídos são sempre tolerantes a partição.

Escalabilidade

Escalabilidade

- ***À medida em que o volume de dados cresce***, aumenta-se a necessidade de escalabilidade e melhoria do desempenho.
- Podemos escalar ***verticalmente*** (adicionar CPU, memória e disco) ou podemos escalar ***horizontalmente*** (adicionar mais nós).

Sistemas de Arquivos Distribuídos

Sistemas de Arquivos

- Foram originalmente desenvolvidos como um ***recurso do S.O*** que fornece uma interface de programação conveniente para armazenamento em disco.
- ***São responsáveis pela organização, armazenamento, recuperação, atribuição de nomes, compartilhamento e proteção de arquivos.***
- Projetados para ***armazenar e gerenciar um grande número de arquivos***, com recursos para criação atribuição de nomes e exclusão de arquivos.

Sistemas de Arquivos

- Diretório é um arquivo de tipo especial;
- Fornece um mapeamento dos nomes textuais para identificadores internos;
- Podem incluir nomes de outros diretórios.

Tamanho do Arquivo
Horário de Criação
Horário de Acesso (Leitura)
Horário de Modificação (Escrita)
Horário de Alteração de Atributo
Contagem de Referência
Proprietário
Tipo de Arquivo
Lista de Controle de Acesso

Sistemas de Arquivos Distribuídos (DFS ou SAD)

Um sistema de arquivos distribuídos ***permite aos programas armazenarem e acessarem arquivos remotos exatamente como se fossem locais***, possibilitando que os usuários acessem arquivos a partir de qualquer computador em uma rede.” (COULOURIS, et. al, p. 284).

- **Objetivo**: permitir que os programas armazenem e acessem arquivos remotos exatamente como se fossem locais.
- **Permitem que vários processos** compartilhem dados por longos períodos, de modo seguro e confiável.
- **O desempenho e segurança** no acesso aos arquivos armazenados em um servidor devem ser compatíveis aos arquivos armazenados em discos locais.

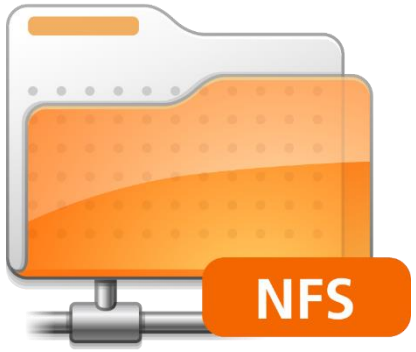
Requisitos de um Sistemas de Arquivos Distribuídos

- Transparência.
- Atualização concorrente de arquivos.
- Replicação de arquivos.
- Heterogeneidade.
- Tolerância a falha.
- Consistência.
- Segurança.
- Eficiência.

Arquitetura do SAD

- Modelo abstrato de arquitetura que serve para o ***Network File System (NFS) e Andrew File Systen (AFS).***

- Divisão de responsabilidades entre três módulos.
  - Cliente.
  - Serviço de arquivos planos.
  - Serviço de diretórios.



- Design aberto
  - Diferentes módulos cliente podem ser utilizados para implementar diferentes interfaces.
  - Simulação de operações de arquivos de diferentes S.O.
  - Otimização de performance para diferentes configurações de hardware de clientes e servidores.

Sistemas de Arquivos Distribuídos (DFS ou SAD)

Funções de um Sistema de Arquivos Distribuído:

- ***Armazenar e compartilhar programas e dados***
  - Funções idênticas às de um sistema centralizado (local).
- ***Ênfase na disponibilidade, confiabilidade e segurança.***
- ***Desempenho***
  - Questão importante porque acessos remotos podem ser significativamente mais lentos que os locais.
  - Não se pretende em geral que o SAD seja mais rápido que um SA local, mas sim que a degradação seja aceitável.

Sistemas de Arquivos Distribuídos (DFS ou SAD)

Sistemas de arquivo distribuídos ***devem ser vistos pelos clientes como um sistema de arquivo local.***

A ***transparência*** é muito importante para seu bom funcionamento.

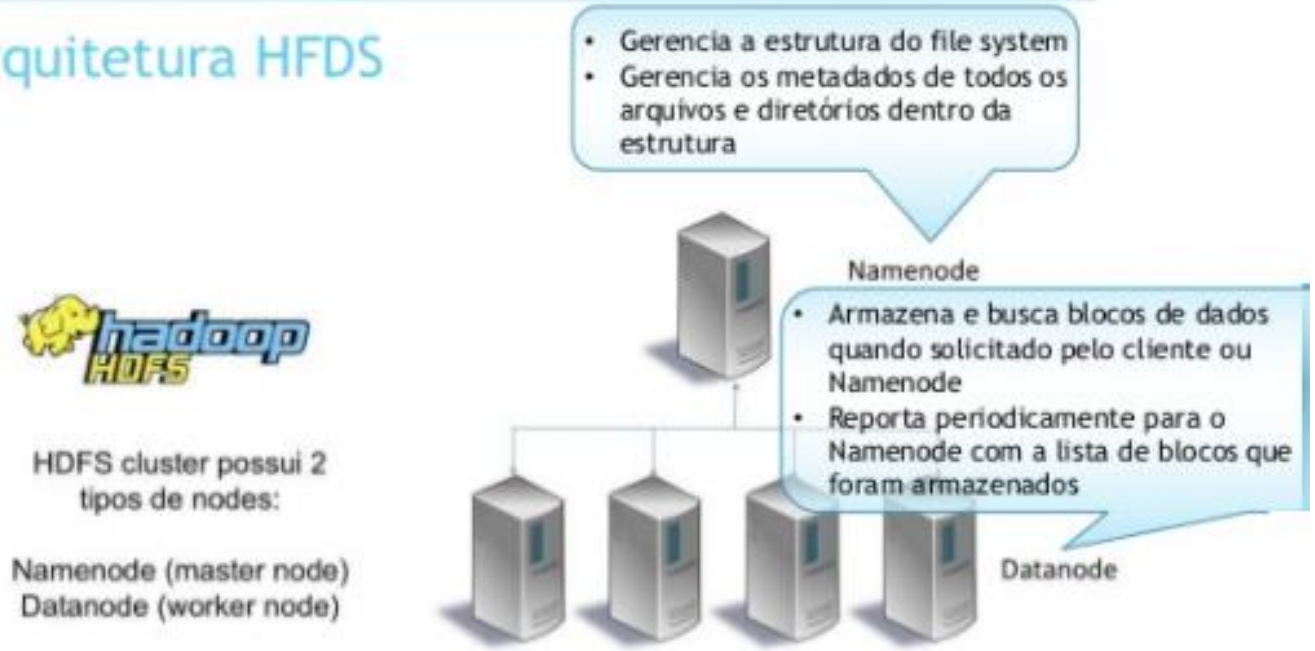
É necessário ***um bom controle de concorrência*** no acesso.

Cache é importante.

Apache Hadoop

Hadoop

Arquitetura HFDS



Apache Hadoop

Hadoop é uma ***plataforma de software de código aberto para o armazenamento e processamento distribuído de grandes conjuntos de dados, utilizando clusters de computadores com hardware commodity.***

Os serviços do Hadoop fornecem armazenamento , processamento, acesso, governança, segurança e operações de Dados.

Benefícios do Apache Hadoop

Algumas das razões para se usar Hadoop é a sua "capacidade de armazenar, gerenciar e analisar grandes quantidades de dados estruturados e não estruturados de forma rápida, confiável, flexível e de baixo custo.

- **Escalabilidade e desempenho** – distribuídos tratamento de dados local para cada nó em um cluster Hadoop permite armazenar, gerenciar, processar e analisar dados em escala petabyte.
- **Confiabilidade** – clusters de computação de grande porte são propensos a falhas de nós individuais no cluster. Hadoop é fundamentalmente resistente – quando um nó falha de processamento é redirecionado para os nós restantes no cluster e os dados são automaticamente re-replicado em preparação para falhas de nó futuras.

Benefícios do Apache Hadoop

- **Flexibilidade** – ao contrário de sistemas de gerenciamento de banco de dados relacionais tradicionais, você não tem que esquemas estruturados criados antes de armazenar dados. ***Você pode armazenar dados em qualquer formato, incluindo formatos semi-estruturados ou não estruturados***, e em seguida, analisar e aplicar esquema para os dados quando ler.
- **Baixo custo** – ao contrário de software proprietário, ***o Hadoop é open source*** e é executado em hardware commodity de baixo custo.

HDFS

O HDFS (Hadoop Distributed File System) é um sistema de arquivos distribuído, ***projetado para armazenar arquivos muito grandes***, com padrão de acesso aos dados streaming , utilizando clusters de servidores facilmente encontrados no mercado e de baixo ou médio custo.

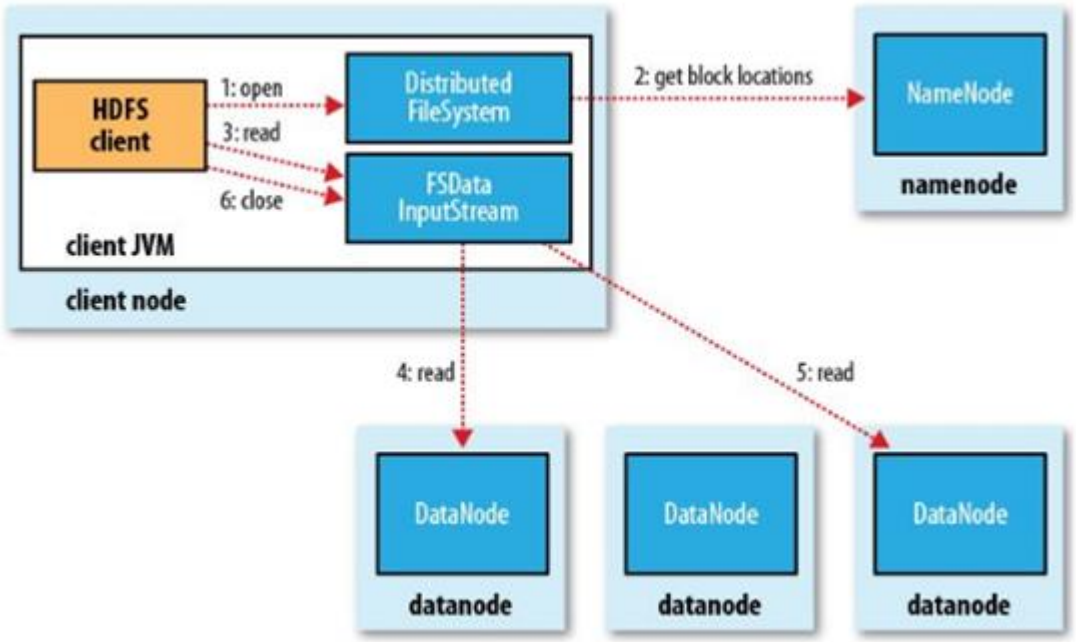
Não deve ser utilizado para aplicações que precisem de acesso rápido a um determinado registro e sim para aplicações nas quais é necessário ler uma quantidade muito grande de dados.

Outra questão que deve ser observada é que ***não deve ser utilizado para ler muitos arquivos pequenos***, tendo em vista o overhead de memória envolvido.

Recursos do HDFS

- O HDFS tem 2 tipos de Nós : **Master** (ou Namenode) e **Worker** (ou Datanode).
  - O **Master** armazena informações da distribuição de arquivos e metadados.
  - Já o **Worker** armazena os dados propriamente ditos. Logo o Master precisa sempre estar disponível. Para garantir a disponibilidade podemos ter um backup (similar ao Cold Failover) ou termos um Master Secundário em um outro servidor. Nesta segunda opção, em caso de falha do primário, o secundário pode assumir o controle muito rapidamente.
- Tal como um sistema Unix, é possível utilizar o HDFS via linha de comando.

HDFS


HDFS client
1: open
3: read
6: close
Distributed FileSystem
FSData InputStream
client JVM
client node
2: get block locations
NameNode
namenode
4: read
5: read
DataNode
datanode
DataNode
datanode
DataNode
datanode

Apache Hadoop

- Hadoop é uma plataforma de software em Java de computação distribuída voltada para clusters e processamento de grandes volumes de dados, com atenção a tolerância a falhas.

- Foi inspirada no MapReduce (um modelo de programação paralela para processamento largamente distribuído de grandes volumes de dados proposto primeiramente pela empresa Google) e no GoogleFS (Google File System é um sistema de arquivos distribuído proprietário desenvolvido pelo Google para fornecer acesso eficiente e confiável aos dados usando grandes clusters de hardware comum).
- É disponibilizado pela Amazon e IBM em suas plataformas.

Módulos do Framework do Apache Hadoop

- **Hadoop Common** - Contém as bibliotecas e arquivos comuns e necessários para todos os módulos Hadoop.
- **Hadoop Distributed File System (HDFS)** - Sistema de arquivos distribuído que armazena dados em máquinas dentro do cluster, sob demanda, permitindo uma largura de banda muito grande em todo o cluster.
- **Hadoop Yarn** - Trata-se de uma plataforma de gerenciamento de recursos responsável pelo gerenciamento dos recursos computacionais em cluster, assim como pelo agendamento dos recursos.
- **Hadoop MapReduce** - Modelo de programação para processamento em larga escala.

Computação em Nuvem

Computação em Nuvem

Nós temos três principais fornecedores de computação em nuvem, ***AWS, Microsoft Azure e Google Cloud,*** que possuem pontos fortes e fracos que os tornam ideais para diferentes cenários.

Arquiteturas Monolíticas

Com as arquiteturas monolíticas, todos os processos são altamente acoplados e executam como um único serviço.

Isso significa que se um processo do aplicativo apresentar um pico de demanda, toda a arquitetura deverá ser escalada. A complexidade da adição ou do aprimoramento de recursos de aplicativos monolíticos aumenta com o crescimento da base de código. Essa complexidade limita a experimentação e dificulta a implementação de novas ideias.

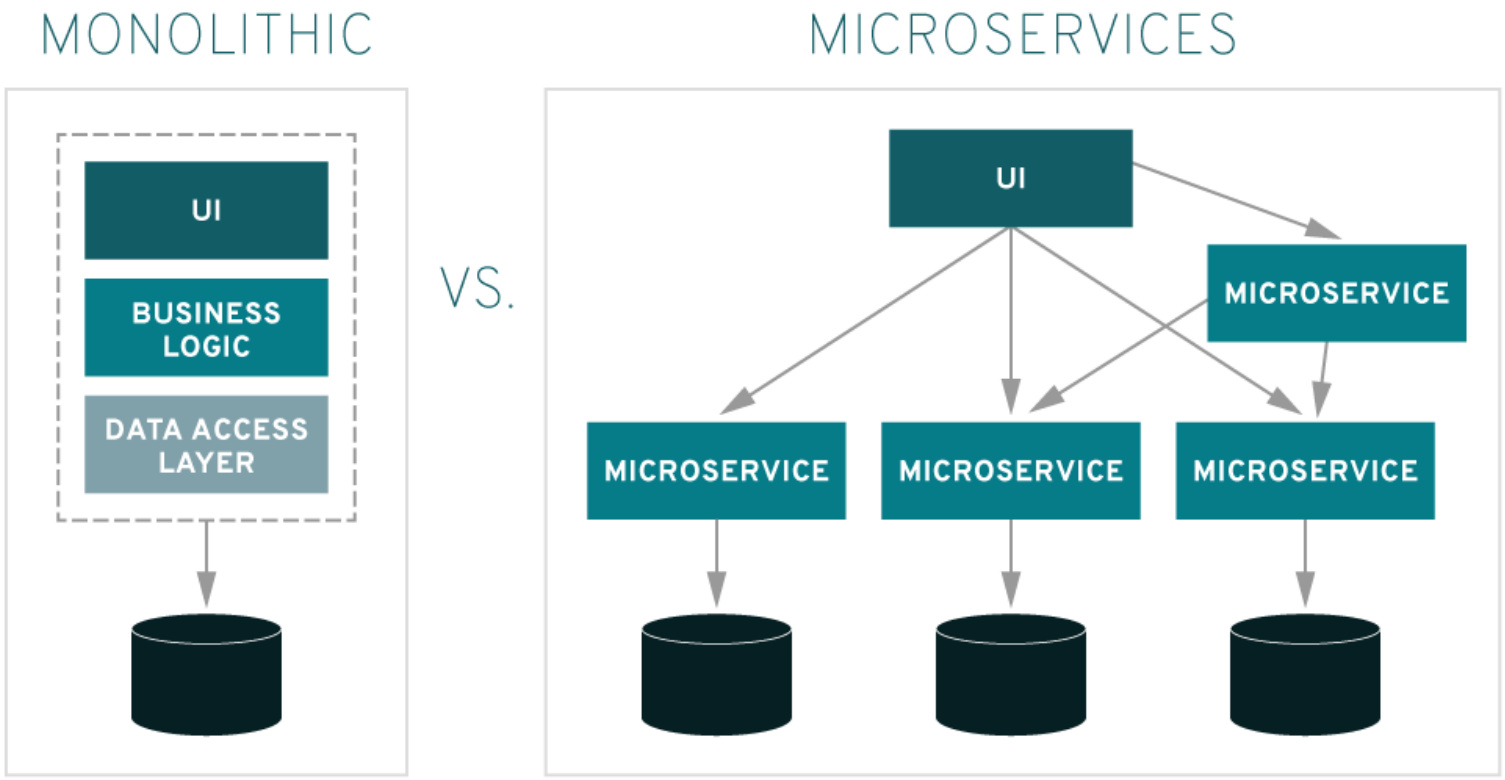
As arquiteturas monolíticas aumentam o risco de disponibilidade de aplicativos, pois muitos processos dependentes e altamente acoplados aumentam o impacto da falha de um único processo.

Arquitetura de Microserviços

Com uma arquitetura de microserviços, um aplicativo é criado como componentes independentes que executam cada processo do aplicativo como um serviço.

Esses serviços se comunicam por meio de uma interface bem definida usando APIs leves. Os serviços são criados para recursos empresariais e cada serviço realiza uma única função. Como são executados de forma independente, cada serviço pode ser atualizado, implantado e escalado para atender a demanda de funções específicas de um aplicativo.

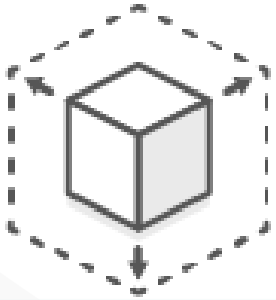
Arquitetura



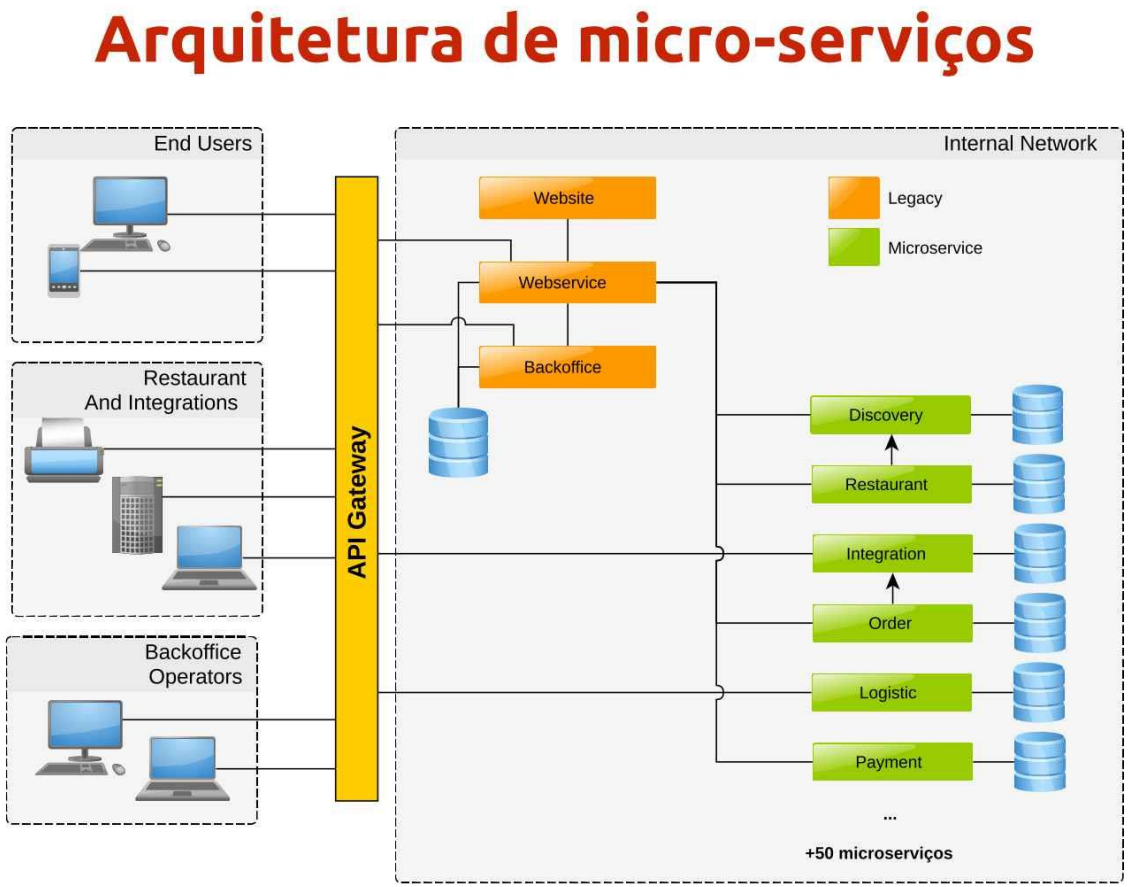
Arquitetura de Microserviços

Escalabilidade flexível

Os microserviços permitem que cada serviço seja escalado de forma independente para atender à demanda do recurso de aplicativo oferecido por esse serviço. Isso permite que as equipes dimensionem corretamente as necessidades de infraestrutura, meçam com precisão o custo de um recurso e mantenham a disponibilidade quando um serviço experimenta um pico de demanda.



Arquitetura

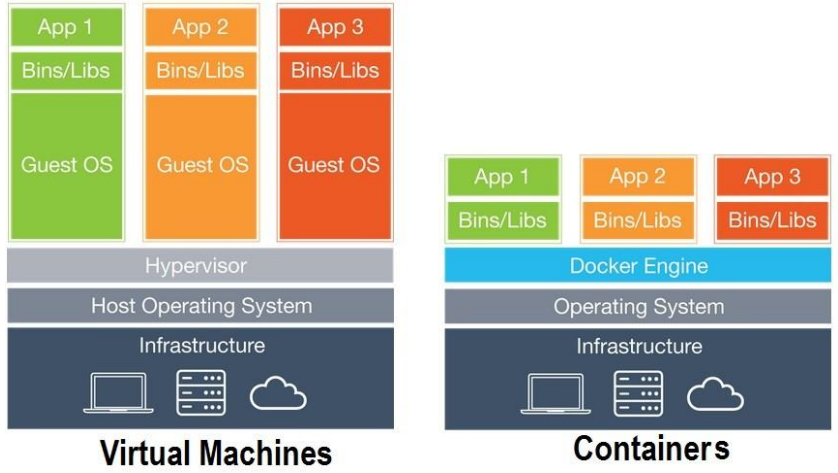


Containers

Container

Em vez de usar um sistema operacional para cada estrutura, como na virtualização, os Containers são blocos de espaços divididos pelo Docker em um servidor, o que possibilita a implementação de estruturas de Microserviços que compartilham o mesmo sistema operacional. Porém, de forma limitada (conforme a demanda por capacidade).

O fato de os Containers não terem seus próprios sistemas operacionais, permite que eles consumam menos recursos e, com isso, sejam mais leves.



Ferramentas de Orquestração de Containers

As ferramentas de orquestração de containers são aplicações em nuvem que permitem fazer o gerenciamento de múltiplos contêineres.

Seus principais objetivos são:

- Cuidar do ciclo de vida dos containers de forma autônoma, subindo e distribuindo, conforme nossas especificações ou demandas;
- Gerenciar volumes e rede, que podem ser local ou no cloud provider de sua preferência.

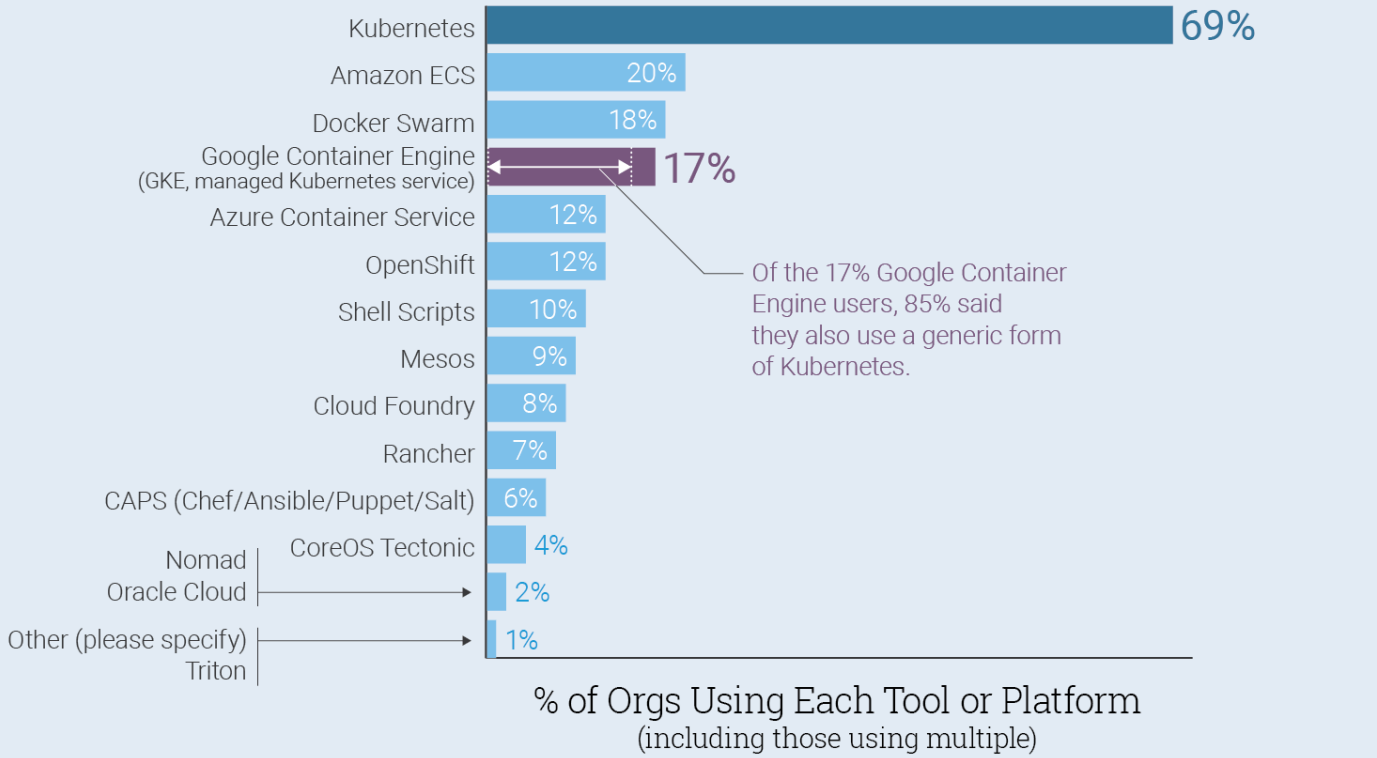
Orquestradores de Containers

O Kubernetes, ECS e o Docker são as principais plataformas de gerenciamento de contêineres.

Dessa forma, essa é uma ferramenta para viabilizar a utilização de Containers e Microserviços em servidores com mais facilidade, pois permite empacotar os aplicativos para que possam ser movimentados facilmente.

O Docker permite, por exemplo, que uma biblioteca possa ser instalada em diferentes Containers sem que haja qualquer interdependência entre eles. Essa característica tem o objetivo de facilitar o gerenciamento de códigos e aplicativos.

Orquestradores de Containers



Arquiteturas com Alta Disponibilidade (HA)

Alta Disponibilidade

- Alta disponibilidade: Refere-se a um conjunto de tecnologias que minimiza as interrupções de TI proporcionando a continuidade dos negócios de serviços de TI por meio de componentes redundantes e tolerantes a falhas ou protegidos contra failover no mesmo data center.

Sistemas Distribuídos

Os sistemas distribuídos são recursos computacionais compartilhados em uma rede permitindo um aumento no desempenho, tolerância a falhas e escalabilidade do sistema.

A distinção fundamental é que em um sistema distribuído, uma coleção de computadores independentes mostra-se aos usuários como sendo um sistema único.

Sistemas de Arquivos Distribuídos

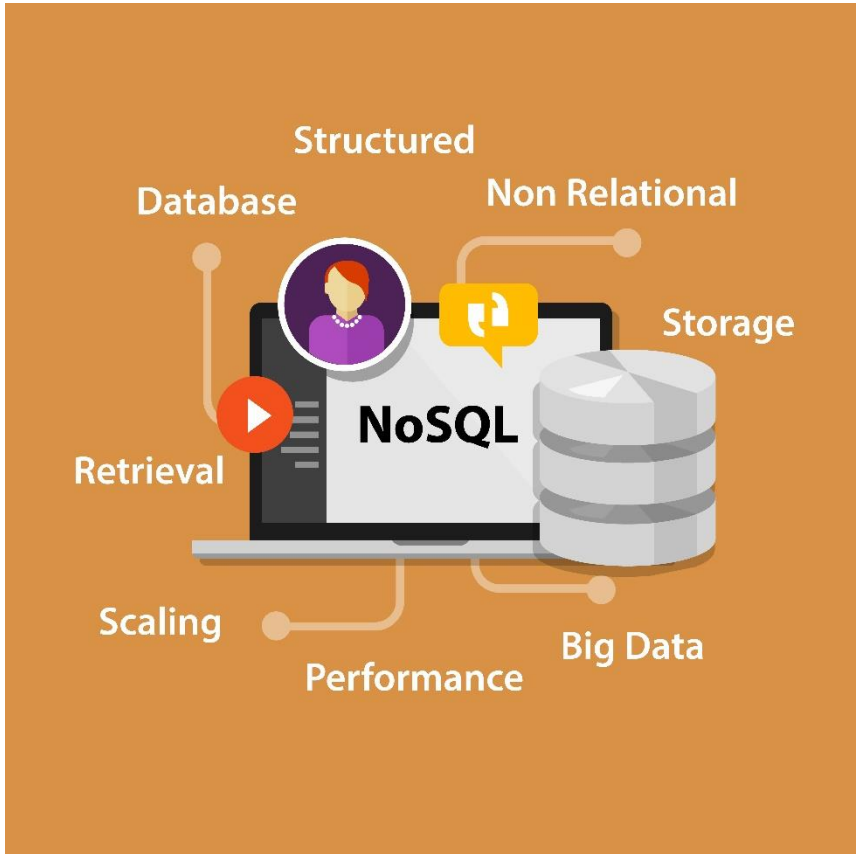
Sistema de arquivo distribuído (SAD) permite os programas armazenar e acessar arquivos remotos exatamente como se fosse um acesso local, permitindo o acesso a partir de qualquer computador em uma

rede.

A **replicação** é o segredo da eficácia dos sistemas distribuídos, pois ela é a chave para prover melhor desempenho, alta disponibilidade e tolerância a falhas.

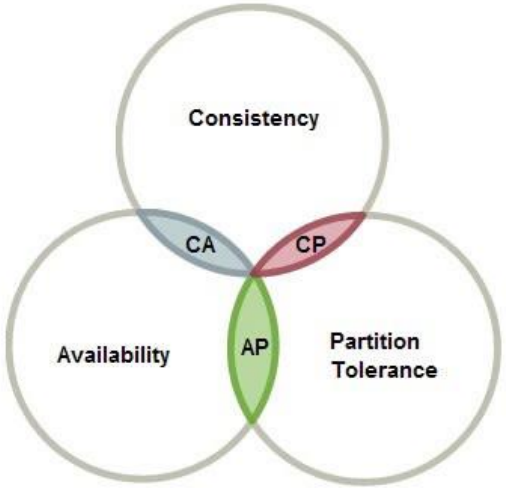
Banco de Dados NoSQL

- Escalabilidade horizontal.
- Ausência de esquema ou esquema flexível.
- Suporte à replicação.
- API simples.
- Nem sempre prima pela consistência.



Teorema CAP

- De acordo com o teorema **CAP**, ***um sistema distribuído de bancos de dados somente pode operar com dois desses comportamentos ao mesmo tempo***, mas ***jamais com os três simultaneamente***.



Arquiteturas com Alta Disponibilidade (HA) – Exemplo MongoDB

Escalabilidade

- ***À medida em que o volume de dados cresce***, aumenta-se a necessidade de escalabilidade e melhoria do desempenho.
- Podemos escalar ***verticalmente*** (adicionar CPU, memória e disco) ou podemos escalar ***horizontalmente*** (adicionar mais nós).

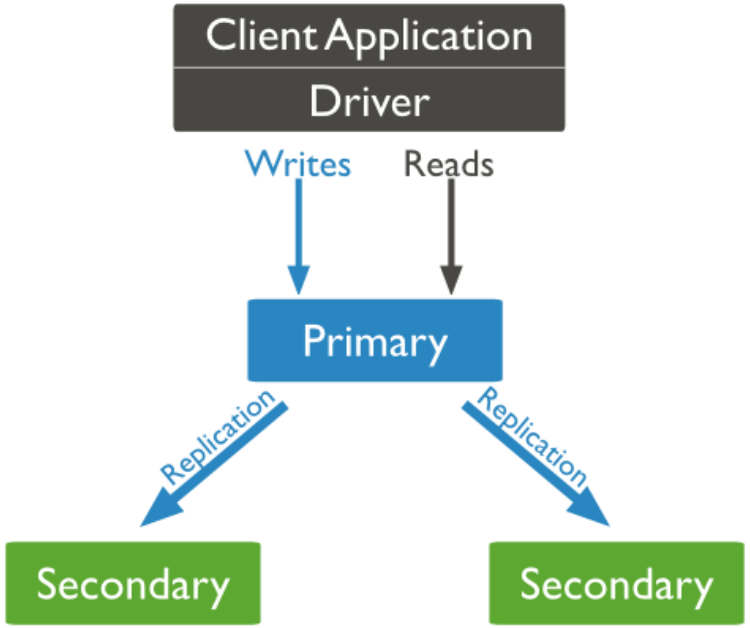
Sistemas de Arquivos Distribuídos

Na busca de sistemas mais confiáveis, alguns meios foram desenvolvidos para oferecer mais confiança aos sistemas, entre eles está a **tolerância a falhas**. Dessa forma, é essencial para o sistema de arquivos distribuídos que **continuem a funcionar diante de falhas que aconteçam em servidores.**

A **replicação** é o método utilizado para **aumentar a disponibilidade de um serviço de arquivos**, onde os arquivos são armazenados em dois ou mais servidores e caso um deles não esteja disponível, outro servidor poderá fornecer os serviços solicitados.

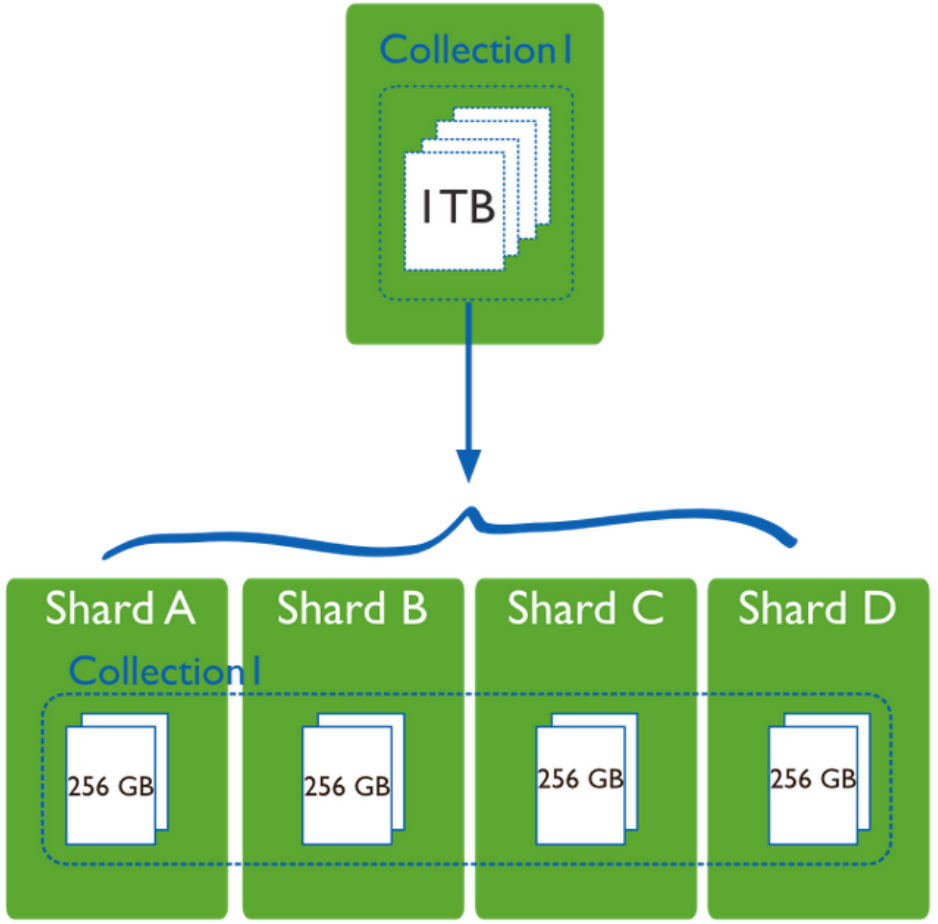
Arquitetura Mongodb

- Um conjunto de réplicas no MongoDB é um grupo de instâncias mongod que mantém o mesmo conjunto de dados. Ele contém vários nós de suporte de dados e, opcionalmente, um nó de árbitro.
- O nó primário recebe todas as operações de leitura e gravação e registra tudo nos seus conjuntos de dados e logs.
- Os secundários replicam o log do primário e aplicam as operações a seus conjuntos de dados de forma assíncrona.
- Tendo os conjuntos de dados secundários que refletem o conjunto de dados do primário, o conjunto de réplicas pode continuar a funcionar, apesar da falha de um ou mais membros.
- Se o nó primário não estiver disponível, um secundário elegível fará uma eleição para se eleger como o novo nó primário.

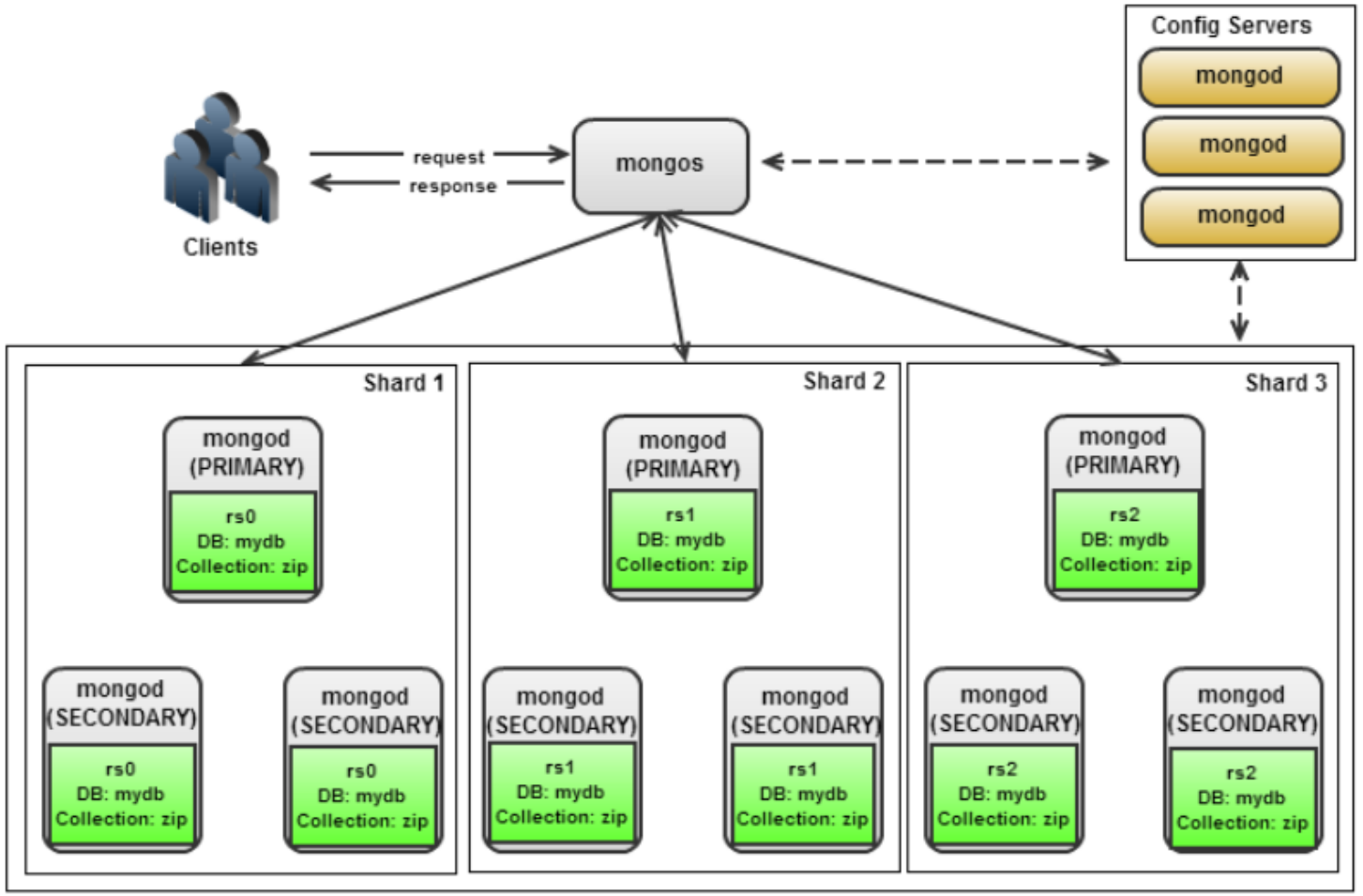


Sharding

- Um cluster (agrupamento) no mongodb é um agrupamento fragmentado:
  - Escale leituras e gravações ao longo de vários nós.
  - Cada nó não lida com todos os dados, portanto, você pode separar os dados ao longo de todos os nós do fragmento.



Arquitetura usando Sharding e ReplicaSet.



Blocos Funcionais em uma Arquitetura de Dados

Tipos de Sistemas Gerenciadores de Dados

- **Data Warehouses:** um Data Warehouse agrega dados de diferentes fontes de dados relacionais em uma empresa em um repositório único, central e consistente.
- **Data Marts:** um Data Mart é uma versão focada de um Data Warehouse que contém um subconjunto menor de dados importantes e necessários para uma única equipe ou um grupo seleto de usuários dentro de uma organização, como o departamento de RH.
- **Data Lakes:** enquanto os Data Warehouses armazenam dados processados, um Data Lake armazena dados brutos, normalmente petabytes deles. Um Data Lake pode armazenar dados estruturados e não estruturados, o que o torna exclusivo de outros repositórios de dados.

Tipos de Arquitetura de Dados

- **Data Fabrics:** é uma arquitetura que se concentra na automação da integração de dados, engenharia de dados e governança em uma cadeia de valor de dados entre provedores de dados e consumidores de dados. Uma malha de dados é baseada na noção de “metadados ativos” que usa gráfico de conhecimento, semântica, mineração de dados e tecnologia de aprendizado de máquina (ML) para descobrir padrões em vários tipos de metadados (por exemplo, logs do sistema, social, etc.). Em seguida, aplica esse insight para automatizar e orquestrar a cadeia de valor dos dados. Por exemplo, ele pode permitir que um consumidor de dados encontre um produto de dados e, em seguida, tenha esse produto de dados provisionado para ele automaticamente.

Tipos de Arquitetura de Dados

- **Data Meshes:** uma malha de dados é uma arquitetura de dados descentralizada que organiza os dados por domínio de negócios. Usando uma malha de dados, a organização precisa parar de pensar nos dados como um subproduto de um processo e começar a pensar neles como um produto por si só. Os produtores de dados atuam como proprietários de produtos de dados. Como especialistas no assunto, os produtores de dados podem usar sua compreensão dos principais consumidores dos dados para projetar APIs para eles. Essas APIs também podem ser acessadas de outras partes da organização, fornecendo acesso mais amplo aos dados gerenciados.

Benefícios de Arquitetura de Dados

- **Reduzindo a redundância:** pode haver campos de dados sobrepostos em diferentes fontes, resultando em risco de inconsistência, imprecisões de dados e oportunidades perdidas de integração de dados. Uma boa arquitetura de dados pode padronizar como os dados são armazenados e potencialmente reduzir a duplicação, permitindo análises holísticas e de melhor qualidade.
- **Melhorando a qualidade dos dados:** Arquiteturas de dados bem projetadas podem resolver alguns dos desafios de Data Lakes mal gerenciados, também conhecidos como “pântanos de dados”. Um pântano de dados (DATA SWAMP) carece de qualidade de dados apropriada e práticas de governança de dados para fornecer aprendizados perspicazes.

Benefícios de Arquitetura de Dados

- **Habilitando a integração:** os dados geralmente ficam isolados, como resultado de limitações técnicas no armazenamento de dados e barreiras organizacionais dentro da empresa. As arquiteturas de dados de hoje devem ter como objetivo facilitar a integração de dados entre domínios, para que diferentes geografias e funções de negócios tenham acesso aos dados uns dos outros.
- **Gerenciamento do ciclo de vida dos dados:** uma arquitetura de dados moderna pode abordar como os dados são gerenciados ao longo do tempo.

Características de uma Arquitetura Moderna de Dados

- **Cloud-native e cloud-enabled,** para que a arquitetura de dados possa se beneficiar do dimensionamento elástico e da alta disponibilidade da nuvem.
- **Pipelines de dados robustos, escaláveis e portáteis**, que combinam fluxos de trabalho inteligentes, análises cognitivas e integração em tempo real em uma única estrutura.
- **Integração de dados perfeita,** usando interfaces de API padrão para conectar-se a aplicativos legados.
- **Habilitação de dados em tempo real,** incluindo validação, classificação, gerenciamento e governança.
- **Desacoplado e extensível**, portanto, não há dependências entre os serviços e os padrões abertos permitem a interoperabilidade.
- **Otimizado** para equilibrar custo e simplicidade.


PUC Minas
Virtual