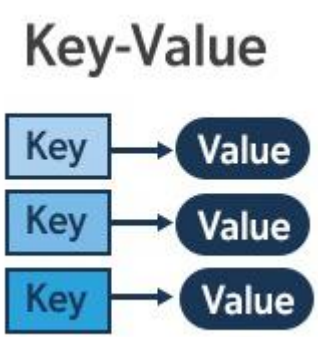
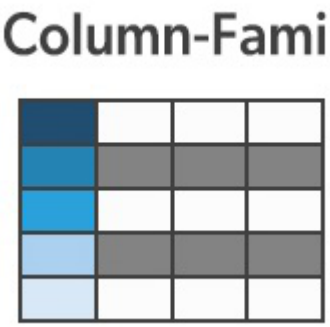
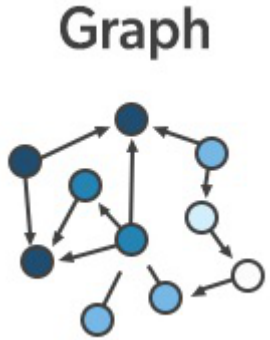
Banco de dados não relacionais







Prof. Me. Ricardo Resende de Mendonça

ricardo.mendonca@online.uscs.edu.br

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

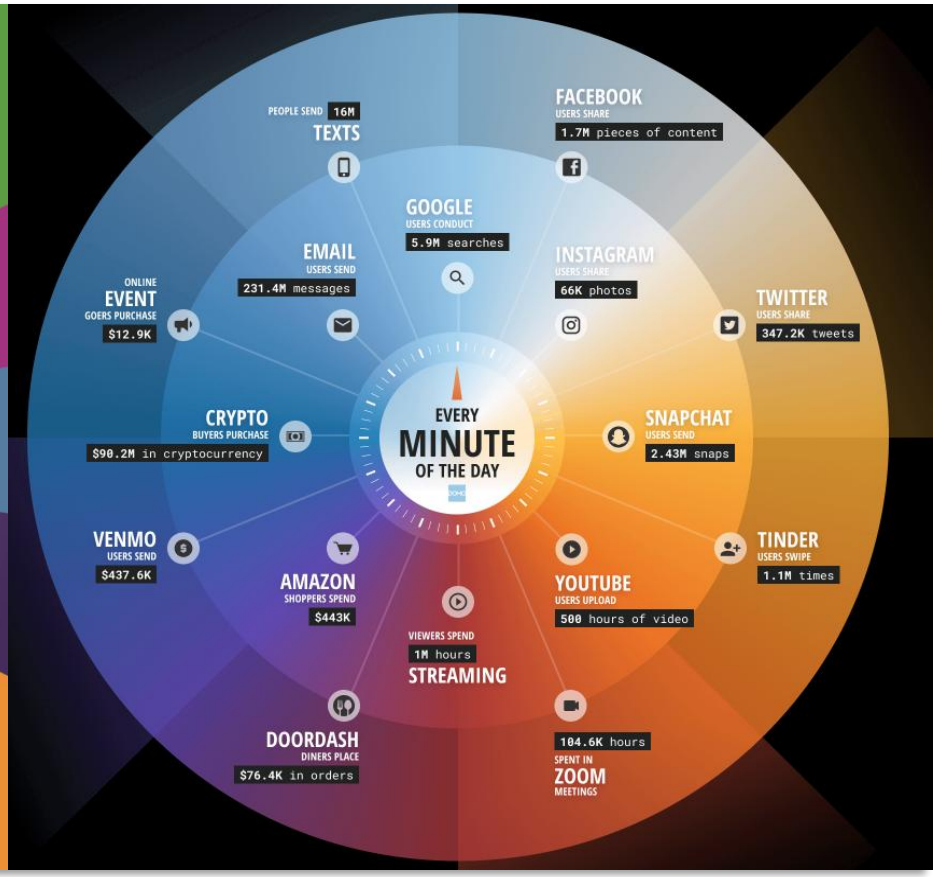
Sobre mim...

Ricardo Resende de Mendonça

- 25 anos de experiência na área de tecnologia;
- Mestre em Ciência da computação;
- Gerente de TI na empresa Commodity Supplies;
- Professor universitário há 10 anos.

Introdução

https://www.domo.com/learn/infographic/data-never-sleeps-11



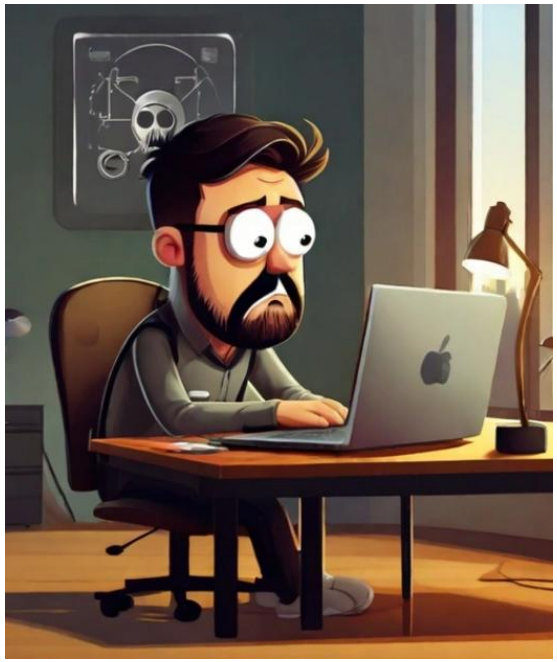
2024 **2023**

NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

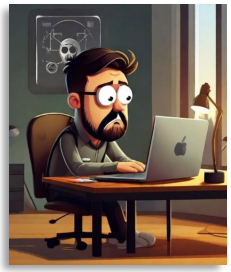
Introdução

Como processar esse grande volume de dados?



Introdução

Como processar esse grande volume de dados?



NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Introdução

Qual a diferença entre um banco de dados relacional e um não relacional?

NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Introdução

Como escolher entre um banco de dados relacional e um não relacional?

NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Introdução

NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Introdução

Aquecimento!!!

a) Cada funcionário é identificado por um número, a ficha cadastral deve permitir o registro do departamento, cargo e salário atribuídos a ele;

b) Existe uma lista de cargos contendo um número aleatório não sequencial e sua descrição;

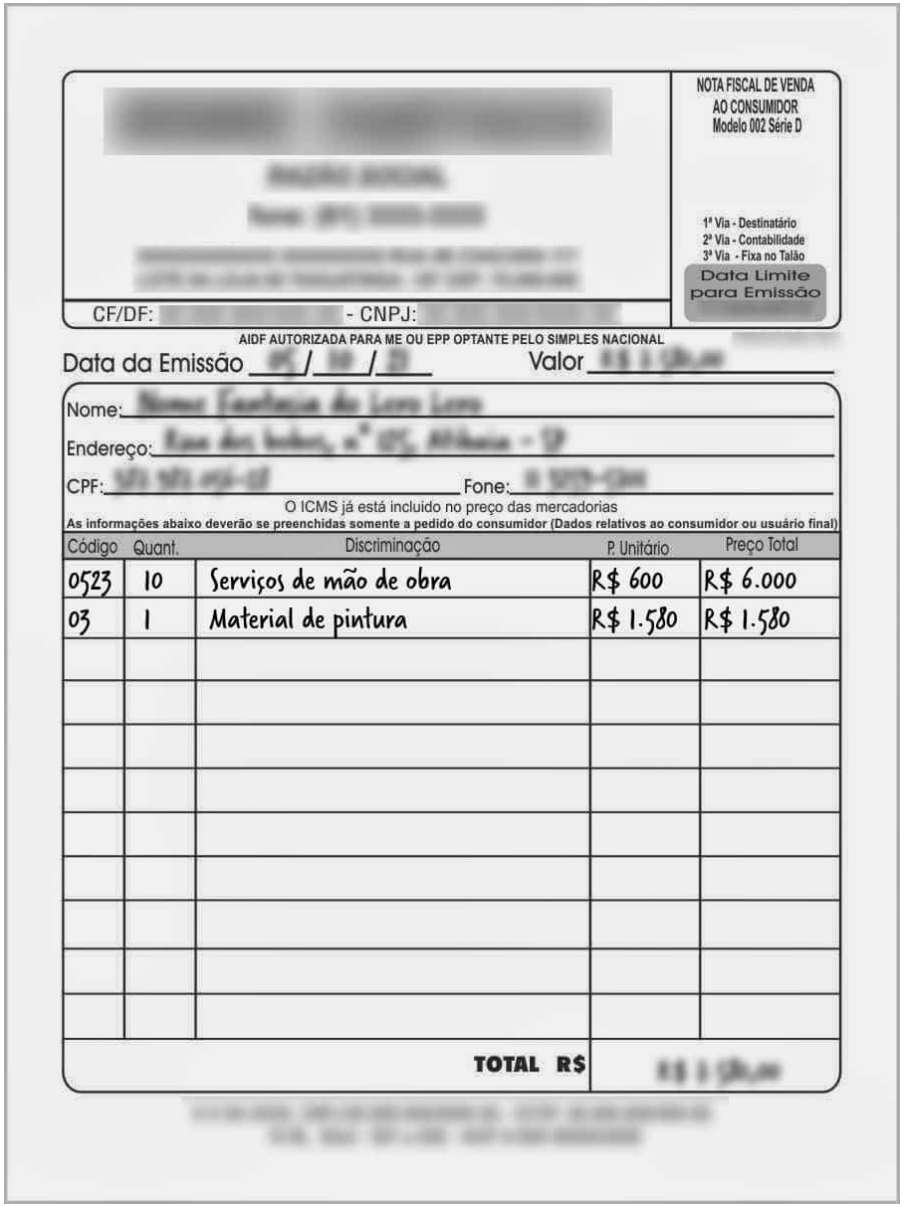
c) Existe uma lista de departamentos contendo um número aleatório não sequencial e sua descrição;

d) Todo departamento ou cargo atribuído a um funcionário deve estar relacionado nas listas descritas itens b e c;

e) Deve ser possível registrar quantos números de telefones forem fornecidos pelos funcionários, o registro telefônico deve permitir o registro do ddd, número, observação e tipo de telefone que foi fornecido (Ex: Residencial, Comercial, Celular) sendo essa informação opcional.

f) Existe uma listagem destes tipos de telefone, contendo um número aleatório e não sequencial

Aquecimento!!!



NOTA FISCAL DE VENDA
AO CONSUMIDOR
Modelo 002 Série D

1ª Via - Destinatário
2ª Via - Contabilidade
3ª Via - Fixa no Talão

Data Limite para Emissão

CF/DF: - CNPJ:

AIDF AUTORIZADA PARA ME OU EPP OPTANTE PELO SIMPLES NACIONAL

Data da Emissão __/__/__ Valor

Nome:

Endereço:

CPF: Fone:

O ICMS já está incluido no preço das mercadorias

As informações abaixo deverão se preenchidas somente a pedido do consumidor (Dados relativos ao consumidor ou usuário final)

Código	Quant.	Discriminação	P. Unitário	Preço Total
0523	10	Serviços de mão de obra	R$ 600	R$ 6.000
03	1	Material de pintura	R$ 1.580	R$ 1.580

TOTAL R$

NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

BD Relacionais

- Os bancos de dados relacionais surgiram a partir do modelo relacional, desenvolvido por Codd em 1970
- São os bancos de dados mais utilizados no mercado.
- Utilizam a linguagem SQL para consulta e manipulação de dados e estruturas de dados
- Implementam as propriedades ACID para transações

Propriedades ACID

- **Atomicidade**: as transações são atômicas, executam completamente ou não executam.
- **Consistência**: as transações criam novos estados válidos, ou, em caso de falha, o banco de dados volta para o último estado válido.
- **Isolamento**: garante que uma transação em andamento não será interferida por outras transações.
- **Durabilidade**: os dados estarão disponíveis na sua forma correta mesmo se o sistema falhar ou reiniciar.

Integridade Referencial

- A integridade referencial garante a acurácia e a consistência dos dados dentro de um relacionamento entre tabelas.

- Isso é feito por meio de uma chave estrangeira, que faz referência a um valor de uma chave primária em outra tabela e a integridade referencial garantirá que esse relacionamento é íntegro (**o registro referenciado existe**)

- O BD Relacional não permite a inclusão de um registro com um valor no campo chave estrangeira que não exista na outra tabela que está relacionada.

Normalização de Dados

- É um processo que busca otimizar os bancos de dados, além de garantir a maior consistência destes, evitando a duplicação de informações e as dividindo em tabelas de acordo com o tipo de elemento que está sendo armazenado.
- É a regra de “Criar uma tabela para colocar as colunas que possuem valores que se repetem em várias registros diferentes”
- Ou seja, “Criamos uma nova tabela e colocamos chaves estrangeiras para indicar o relacionamento entre a tabela originária e a nova tabela recém criada”.
- Desse modo, as tabelas se relacionam por meio do uso de chaves estrangeiras, que garantem a consistência das informações.

BD Centralizados

- Em um banco de dados centralizado, todos os dados são armazenados em um único local.
- Este local pode ser um servidor ou um conjunto de servidores, mas eles são centralmente localizados e gerenciados.
- Há um ponto central de controle, o que facilita a administração dos dados, a implementação de políticas de segurança e a gestão de backups.
- Os usuários e aplicações acessam os dados através de uma rede, mas a fonte de dados é única.

BD Centralizados

- Simplifica o gerenciamento de dados, geralmente oferece bom desempenho para operações de leitura e escrita, e pode ser mais fácil de proteger devido ao seu ponto único de controle.
- Se o servidor central falhar, todos os usuários e aplicações perdem o acesso aos dados.
- Além disso, pode ter problemas de escalabilidade se a quantidade de dados ou o número de usuários crescer significativamente.

BD Descentralizados

- Os dados são distribuídos por vários locais ou nós.
- Cada nó pode operar de forma independente, armazenando uma cópia dos dados ou uma parte dos dados.
- Não existe um ponto único de controle.
- Cada nó pode ser gerenciado independentemente, e as decisões podem ser tomadas localmente.
- Os usuários podem acessar dados de vários locais, o que pode melhorar a redundância e a disponibilidade dos dados.

BD Descentralizados

- Oferece maior resistência a falhas, pois a falha de um nó não necessariamente afeta outros nós.
- Também pode melhorar a escalabilidade, pois novos nós podem ser adicionados à rede para distribuir a carga.
- Pode ser mais complexo de gerenciar, com desafios adicionais em termos de consistência de dados e segurança.
- Em alguns casos, pode haver um comprometimento na velocidade de acesso aos dados devido à necessidade de sincronização entre os nós.

Limitações dos BD Relacional

- Essas características regem os bancos de dados relacionais até os dias de hoje, porém, ao longo dos anos, limitações foram sendo encontradas.
- Principais limitações:
  - o fato de que esses sistemas não foram construídos para aplicações distribuídas;
  - o cruzamento de dados através de joins tende a custar caro em termos de requisitos de hardware necessários;
  - e o modelo de dados disponível é limitado, não havendo muita flexibilidade em casos em que a estrutura de dados não é completamente conhecida.
  - A dificuldade de escalar esses sistemas de forma horizontal, de modo que a escalabilidade vertical (aumento de recursos de hardware) é a opção mais simples tecnicamente, porém é a mais custosa e, em muitos casos, pode envolver a aquisição de hardware proprietário.

BD Não Relacional

- Os bancos de dados não relacionais também são chamados de NoSQL, termo criado em 1998 por Carlo Strozzi para se referir a um banco de dados relacional leve e open source que não utilizava a linguagem SQL .

- A partir daí surgiu alguns bancos de dados não relacionais, capazes de acompanhar a velocidade de crescimento da internet.
- Além de serem capazes de armazenar dados não estruturados e os processar de forma mais rápida.

BD Não Relacional

- Contudo, o termo só ganhou a conotação que tem hoje em 2009, quando Eric Evans e Johan Oskarsson o utilizaram para se referir a bancos de dados não relacionais.
- Ele pode ser interpretado de forma literal como “não SQL” (ausência de linguagem SQL) ou como “Not Only SQL” (Não Apenas SQL).
- Ou seja, os bancos de dados NoSQL são caracterizados pela ausência da linguagem SQL, apesar de alguns deles apresentarem linguagens similares.
- Esses bancos de dados geralmente são open source.
- Têm como característica a ausência de esquema, permitindo a adição de novas colunas de forma livre

BD Não Relacional

- O desafio de se armazenar e processar grandes volumes de dados com formatos diversos levou ao surgimento de bancos de dados NoSQL.
- Para conseguir atender a essas demandas, foi necessário abrir mão de características existentes em bancos de dados relacionais, a fim de dar lugar a outras características mais flexíveis.
- Isso porque lidar com grandes quantidades de dados de forma estável e permitindo que aplicações escalem a um custo baixo significa que características como a integridade garantida através de transações e a flexibilidade na criação de índices e consultas nem sempre serão possíveis.

BD Não Relacional

- Existem casos em que a redução da complexidade do banco de dados pode ser favorável, dispensando mecanismos complexos que garantiriam propriedades ACID que nem sempre são necessárias.
- A perda dessas características dá lugar à capacidade de processar grandes volumes de dados em curtos espaços de tempo.
- Esse é o caso do Google, que é capaz de processar 20 petabytes de dados armazenados no Bigtable através do MapReduce.

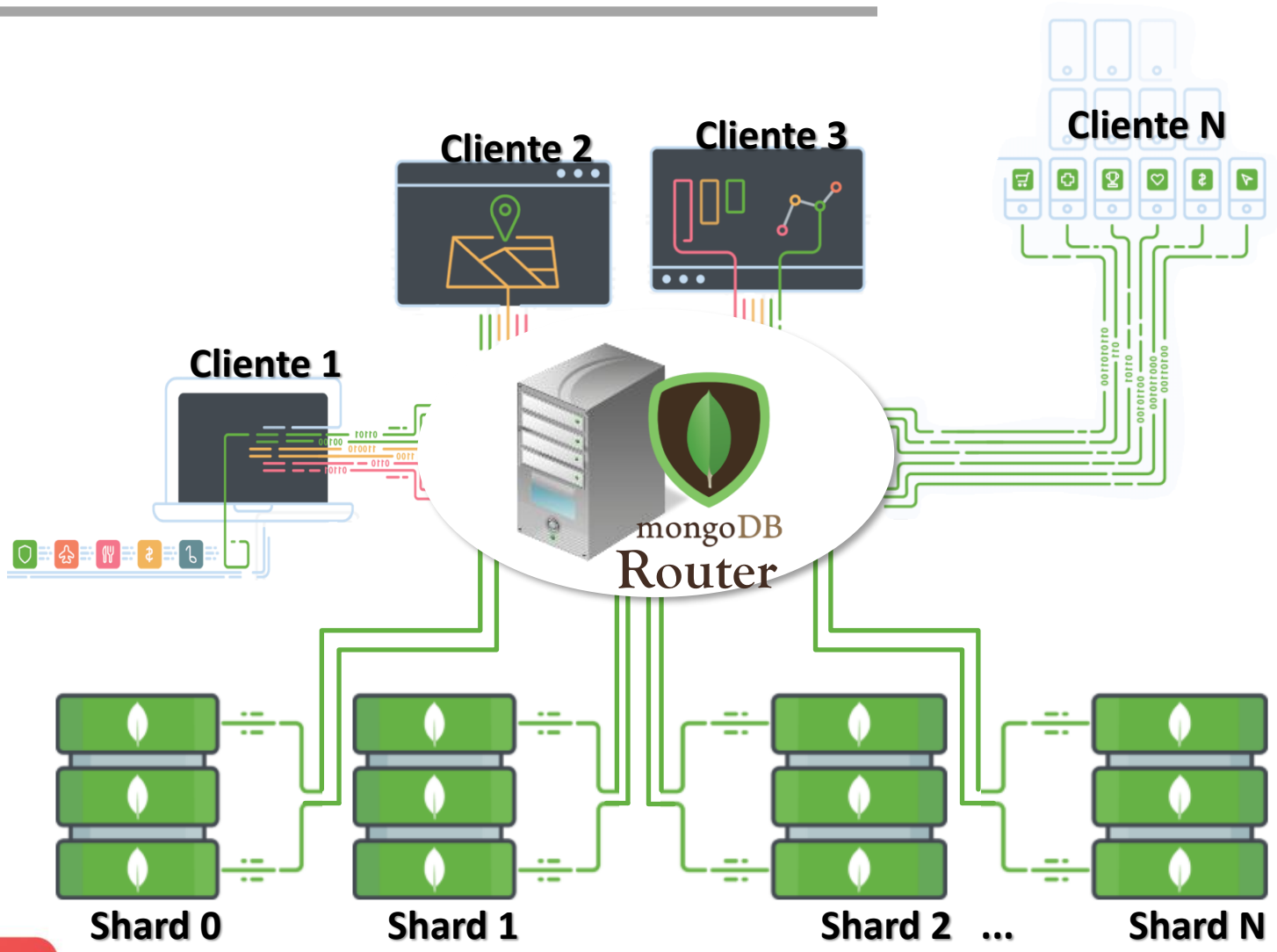
BD Não Relacional

- Outra vantagem dessa classe de bancos de dados é que eles foram criados para escalar,
- Ou seja, eles são capazes de responder a mais requisições e comportar volumes maiores de dados por meio da adição de um hardware ou da adição de novas máquinas em um cluster.
- Dessa forma, essas soluções geralmente escalam horizontalmente com maior facilidade e com um hardware padrão (sharding).

Sharding

- Sharding é o processo de dividir um banco de dados maior em partes menores, cada uma armazenada em um servidor separado.
- Cada "shard" é um banco de dados independente, e coletivamente, eles compõem o banco de dados inteiro.
- Os dados são divididos horizontalmente ou verticalmente. A divisão horizontal, mais comum, significa dividir as linhas de uma tabela entre diferentes shards, enquanto a divisão vertical divide as colunas.

Sharding


Cliente 2
Cliente 3
Cliente N
Cliente 1
mongoDB
Router
Shard 0
Shard 1
Shard 2
...
Shard N

Sharding

- Geralmente, a distribuição dos dados é feita com base em uma chave de shard, que determina como os dados são atribuídos a cada shard.
- Essa chave pode ser um ID de usuário, um intervalo de tempo, uma região geográfica, ou qualquer outro critério relevante.
- Essa técnica permite que cada computador seja responsável por uma parcela de todos os dados, aumentando, assim, a capacidade do banco de dados de responder a requisições e de armazenar dados.

Teorema CAP

- O Teorema CAP, também conhecido como Princípio de Brewer, é um conceito fundamental na teoria dos sistemas distribuídos e bancos de dados.
- Foi formulado por Eric Brewer em 2000 e posteriormente formalizado por Seth Gilbert e Nancy Lynch da MIT em 2002.
- O teorema afirma que é impossível para um sistema de banco de dados distribuído garantir simultaneamente mais de duas das seguintes três propriedades:

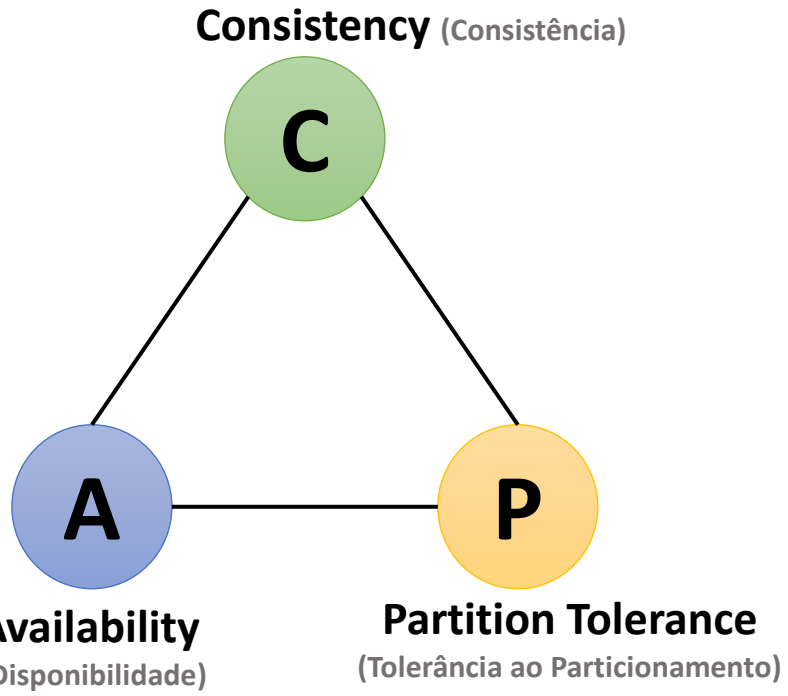
Teorema CAP

Dr. Eric Brewer

2000

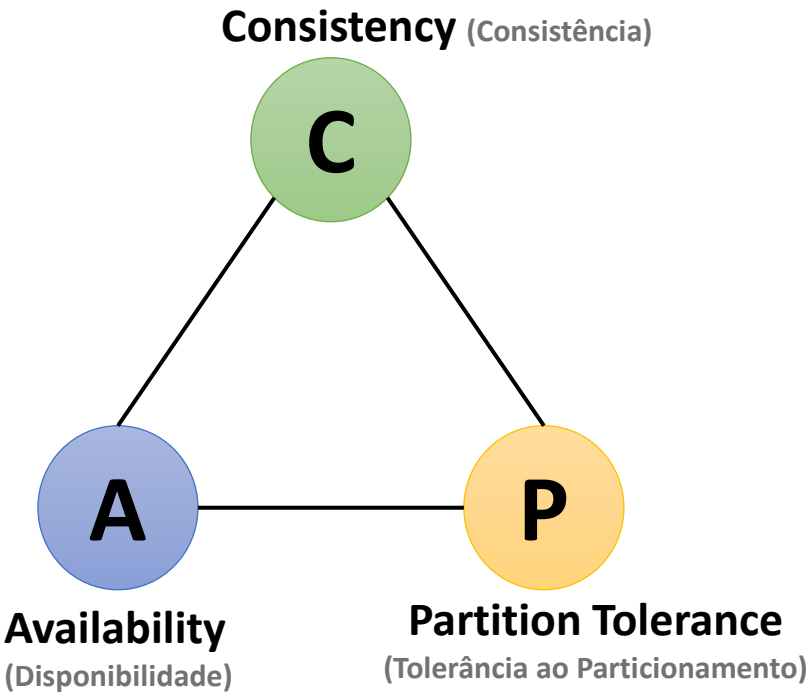
2002

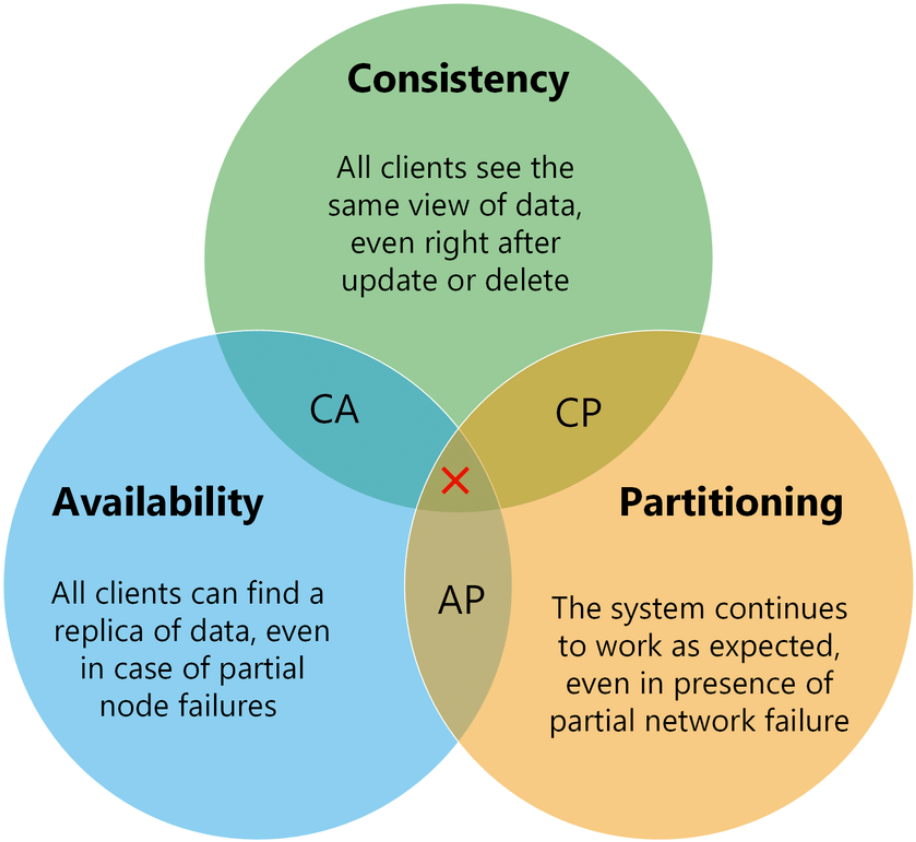
Seth Gilbert e Nancy Lynch



Seth Gilbert and Nancy Lynch. 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33, 2 (June 2002), 51–59. DOI: **https://doi.org/10.1145/564585.564601**

Teorema CAP





NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Teorema CAP

Modelo	O que prioriza?	Exemplo de banco	Problema
CP (Consistência + Partição Tolerante)	Dados sempre atualizados, mesmo que precise aguardar	**MongoDB (Replica Set), PostgreSQL (replicação síncrona)**	Pode atrasar operações ou bloquear o sistema em caso de falha na rede
AP (Disponibilidade + Partição Tolerante)	Sistema sempre funcionando, mas pode ter dados temporariamente desatualizados	**Cassandra, DynamoDB**	Usuários podem ver dados inconsistentes temporariamente
CA (Consistência + Disponibilidade)	Dados sempre atualizados e disponíveis enquanto o servidor estiver ativo	**PostgreSQL/MySQL em um único servidor**	Se o servidor cair, todo o sistema para

Teorema CAP

1. Consistency (C):

- Consistência
- Indica se e quando um sistema está em um estado consistente após a execução de uma operação.
- Todos os nós veem os mesmos dados ao mesmo tempo.

- Em outras palavras, após uma gravação, todas as leituras retornarão o valor dessa gravação mais recente.

Teorema CAP

2. Availability (A):

- Disponibilidade
- O sistema deve permanecer disponível mesmo após a falha de nós ou a indisponibilidade de algum hardware ou software.

- Garante que cada solicitação recebe uma resposta sobre se foi bem-sucedida ou não.
- Isso não significa que cada transação será concluída com sucesso, mas que cada transação receberá uma resposta.

Teorema CAP

3. **Partition tolerance (P):**

- Tolerância a falhas de partição
- O sistema continua a operar apesar de um número arbitrário de mensagens serem perdidas ou atrasadas pela rede entre nós.

- Em outras palavras, o sistema continua a funcionar mesmo que haja "partições" (falhas de comunicação) na rede que impedem que alguns nós comuniquem-se entre si.

Teorema CAP

Segundo o teorema CAP, um sistema distribuído só pode ter duas das três propriedades descritas, gerando, assim, três combinações possíveis de sistemas, descritas a seguir.

1. **CA**:

- São sistemas que garantem a consistência dos dados e possuem alta disponibilidade, mas não suportam falhas de partição.
- Por exemplo: Vertica, Greenplum.

Teorema CAP

2. CP:

- São sistemas que garantem a consistência dos dados e são tolerantes a falhas de partição, porém são bancos de dados que não têm boa disponibilidade.
- Esses sistemas possuem as propriedades ACID: Google BigTable, Hypertable, HBase, MongoDB.

Teorema CAP

3. AP:

- São sistemas que estão sempre disponíveis, mas podem apresentar inconsistências nos dados.
- Esses sistemas possuem as propriedades BASE: Voldemort, Tokyo Cabinet, CouchDB, Riak

Consistência eventual

- Com o advento de sistemas largamente distribuídos, garantias fortes de consistência passaram a ser deixadas de lado, dando espaço para modelos mais fracos, sendo a consistência eventual o modelo mais notável.
- A consistência eventual fornece algumas garantias: caso não haja mais nenhuma atualização em determinado dado, todas as leituras para esse dado, eventualmente, retornarão o mesmo valor.
- Esse modelo não descarta a possibilidade de o cliente ler um dado inconsistente, mas garante que, em algum momento do futuro, eventualmente o dado estará consistente.

Consistência eventual

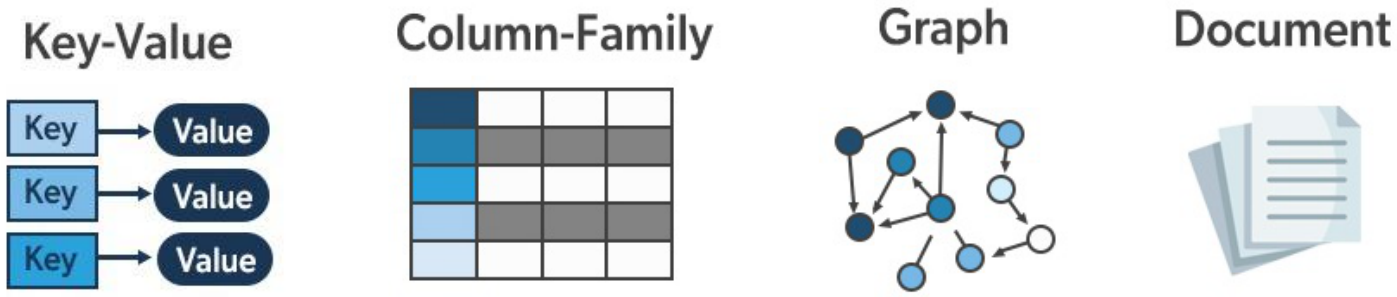
- Uma das vantagens da consistência eventual é a possibilidade de criar sistemas de larga escala capazes de continuar operando mesmo na presença de falhas.
- Ou seja, eles estão sempre disponíveis, mesmo que isso signifique que uma parcela dos usuários verá dados desatualizados por um determinado período de tempo.
- Os sistemas eventualmente consistentes geralmente propagam atualizações de forma assíncrona, garantindo que, em algum momento, todos os nós terão a mesma cópia do dado atualizada.

Consistência eventual

- A propagação de atualizações, que podem ocorrer de forma concorrente para um mesmo dado, implica o uso de mecanismos para garantir que os nós de um sistema consigam saber qual é a atualização mais nova que deve ser aplicada.
- Um desses mecanismos é o uso de um valor de relógio atrelado a cada escrita.

Introdução

Tipos de banco de dados NOSQL



NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Tipos de BD Não Relacionais

- São agrupados nessa categoria por terem algumas semelhanças com relação ao problema que se propuseram a resolver:
  - lidar com grandes volumes de dados através de escalabilidade horizontal.
  - ser open source em sua maioria
  - lidar com tipos de dados diversos.
  - muitas vezes sem um esquema de dados definido.

Introdução

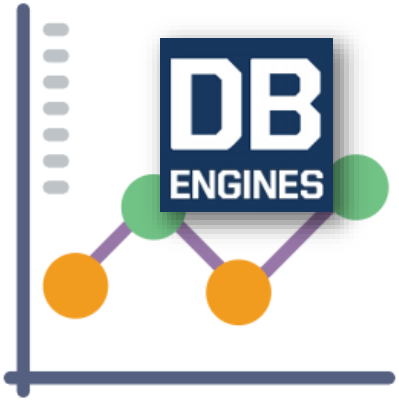
Tipos de banco de dados NOSQL

mongoDB ---> 05º

redis ---> 06º

neo4j ---> 22º

cassandra ---> 12º



https://db-engines.com

NoSQL

USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Tipos de BD Não Relacionais

- Os bancos de dados não relacionais, ou NoSQL, podem ser categorizados em quatro tipos principais, cada um com suas características e casos de uso específicos.
- Vamos explorar cada um deles:
    1. Bancos de Dados Baseados em Documentos
    2. Bancos de Dados de Chave-Valor
    3. Bancos de Dados Colunares
    4. Bancos de Dados de Grafos

BD Baseados em Documentos

- Armazenam dados em documentos (geralmente em formatos como JSON, BSON).
- Os documentos contêm pares chave-valor e podem conter estruturas aninhadas.
- Exemplos: MongoDB, CouchDB.
- Adequados para aplicações que lidam com dados semi-estruturados, como sistemas de gerenciamento de conteúdo, catálogos de produtos e aplicações que necessitam de flexibilidade no esquema de dados.

BD de Chave-Valor

- Armazenam dados como um conjunto de pares de chave-valor.
- Cada chave é única e mapeia diretamente para um valor.
- Exemplos: Redis, DynamoDB.
- Ideal para cenários que exigem armazenamento e recuperação rápida de dados, como :
  - cache de sessões,
  - configurações de aplicativos
  - e sistemas de recomendação.

BD Colunares

- Organizam dados por colunas em vez de linhas.
- Cada coluna pode ser armazenada separadamente, o que otimiza o desempenho de leitura e escrita para determinados tipos de consultas.
- Exemplos: Cassandra, HBase.
- Eficientes para análises de grandes volumes de dados e para aplicações que requerem grande escalabilidade, como o processamento de big data e armazenamento de séries temporais.

BD de Grafos

- Projetados para armazenar e manipular relações entre os dados.
- Eles usam nós (para representar entidades) e arestas (para representar relações) com propriedades em ambos.
- Exemplos: Neo4j, ArangoDB.
- Excelentes para aplicações que necessitam de análise complexa de relações, como redes sociais, sistemas de recomendação e detecção de fraudes.

Dúvidas?