Gerenciamento de dados na nuvem

Juliane Magalhães

Computação na nuvem

A nuvem (*cloud*) é uma metáfora usada na computação para descrever a infraestrutura de comunicação representada por vários servidores web responsáveis por armazenar dados e aplicações.

Cada parte desta infraestrutura é provida como um serviço e, estes são normalmente alocados em centros de dados, utilizando hardware compartilhado para computação e armazenamento.

Computação na nuvem

Segundo o Instituto Nacional de Padrões e Tecnologia (NIST) um modelo de Computação em Nuvem deve apresentar:

- Cinco características essenciais.
- Três modelos de serviço.
- Quatro modelos de implantação.

Características essenciais

1) Self-service sob demanda: O consumidor pode adquirir recursos computacionais, como tempo de processamento no servidor ou armazenamento na rede, automaticamente quando for necessário, sem precisar de interação humana com os provedores de cada serviço.
2) Acesso à rede ampla: os recursos computacionais estão disponíveis através da rede e acessados através de plataformas heterogêneas do tipo thin ou thick client.

Características essenciais

3) Pooling de recursos: Os recursos computacionais (armazenamento, processamento, memória e largura de banda de rede) do provedor são agrupados para servir múltiplos consumidores usando um modelo multi-inquilino, dinamicamente atribuídos e ajustados de acordo com a demanda do consumidor.
    a) Multi-inquilino: empresas clientes que compartilham os mesmos recursos físicos, mas permanecem logicamente isolados.

Características essenciais

4) Elasticidade rápida: elasticidade é a capacidade de aumentar ou diminuir de forma automática os recursos computacionais demandados e provisionados para cada usuário. É a escalabilidade em duas direções: tanto cresce quanto diminui a capacidade ofertada
5) Serviço medido: sistemas em nuvem controlam e otimizam automaticamente a utilização de recursos por meio de uma capacidade de medição, ou seja, o uso dos recursos é monitorado e cobrado de acordo com o contrato de serviço .

Modelo de implantação

Existem quatro tipos de modelos de implantação para os ambientes de computação em nuvem, de acordo com o acesso e a disponibilidade:

1) **Nuvens Privadas:** a infraestrutura de nuvem é utilizada exclusivamente pela própria organização.
    a) **On-premise**: a infraestrutura de nuvem é implantada por uma organização em seu(s) *datacenter*(s).
    b) **Off-premise** (hospedada externamente): a implementação da nuvem é terceirizada para um provedor externo e os consumidores se conectam a ele através de uma rede segura.

Modelo de implantação

2) **Nuvem pública**: a infraestrutura de nuvem é disponibilizada para o usuários comuns e empresas e um provedor de serviços externo fica responsável pela proteção, hospedagem, manutenção e gerenciamento dos dados dos seus clientes.
3) **Nuvem comunidade**: fornece um ambiente em nuvem compartilhado por uma comunidade de organizações com interesses em comum. Podem ser on-premise ou off-premise, assim como nuvens privadas.

Modelo de implantação

4) **Nuvem híbrida**: sistemas críticos ou que manipulam informações confidenciais podem ser hospedados em uma rede privada enquanto outros sistemas, que não lidam com dados sigilosos, podem ser utilizados em uma rede pública.

Modelo de Serviços

1) Software como um Serviço (SaaS)
2) Plataforma como um Serviço (PaaS)
3) Infraestrutura como um Serviço (IaaS)

Modelo de Serviços - Software como um Serviço (SaaS)

O provedor de serviços oferece aplicativos a seus clientes, que são executados sobre a infraestrutura física do provedor.

- O usuário paga pelo serviço oferecido e não pelo produto.
- Não necessita adquirir ou realizar upgrade de hardware para rodar os aplicativos.
- O usuário não administra ou controla a infraestrutura subjacente.
- As atualizações de software são de responsabilidade do provedor do serviço em nuvem.
- Os aplicativos podem ser acessados por uma interface *thin client*.

Modelo de Serviços - SaaS

Exemplos de SaaS:

- Serviços do Google como o Google Drive
- Dropbox
- Netflix
- Spotify

Modelo de Serviços - Plataforma como um Serviço (PaaS)

PaaS fornece uma plataforma de software para os usuários desenvolverem e testarem suas próprias aplicações e hospedá-las na infraestrutura do próprio provedor do serviço.

- No PaaS, o usuário não administra ou controla a infraestrutura subjacente, mas tem controle sobre as configurações das aplicações hospedadas nessa infraestrutura.
- Desenvolvedores trabalham mais rapidamente.
- Fornece todas as instalações necessárias.
- Possui algumas restrições quanto ao tipo de software.

Modelo de Serviços - PaaS

Exemplos de PaaS:

- Google App Engine: é a plataforma do Google para desenvolvimento e hospedagem de aplicativos. O monitoramento do uso do aplicativo pode ser feito facilmente através de uma interface web, com relatórios e estatísticas.

Modelo de Serviços - Infraestrutura como um Serviço (IaaS)

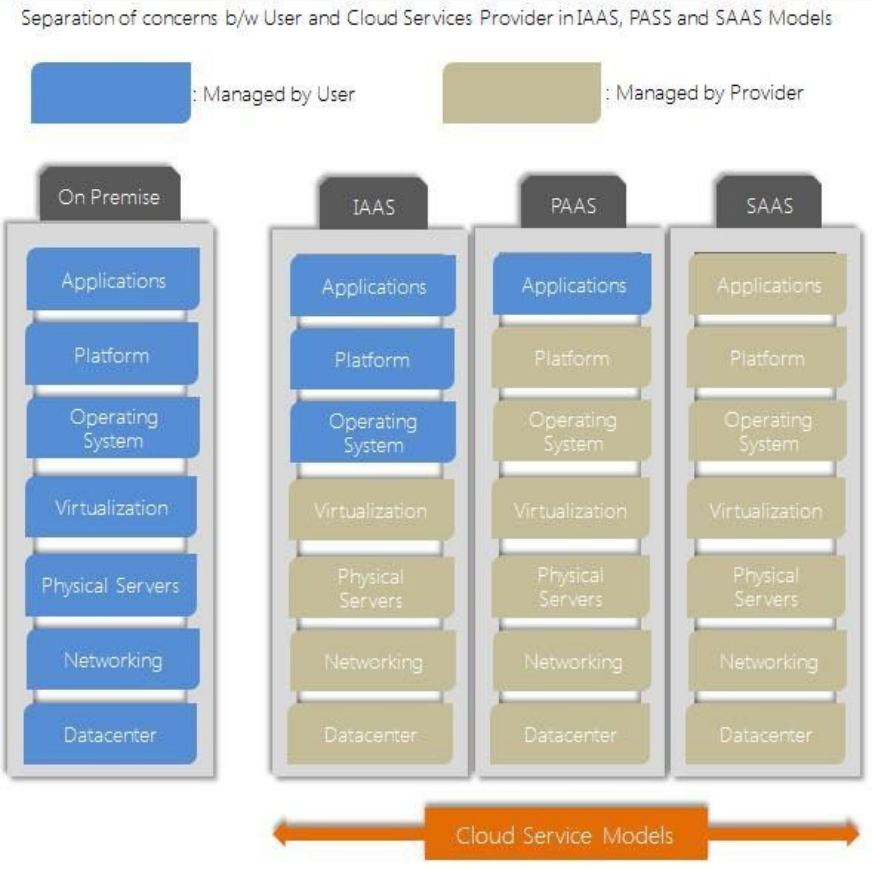
- É possível pode terceirizar os elementos de infra-estrutura, como virtualização, servidores, rede, armazenamento e recursos de computação fundamentais para instalar e executar *softwares* arbitrários, que podem incluir sistemas operacionais e aplicativos.
- O usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento e aplicativos implantados.
- A empresa contrata o serviço necessário sob demanda, sem precisar adquirir todo o equipamento.

Modelo de Serviços - (IaaS)

Exemplos de IaaS:

- Amazon EC2: permite que os usuários aluguem computadores virtuais nos quais rodam suas próprias aplicações. Permite que usuário crie, lance e termine instâncias do servidor, conforme necessário, pagando por hora pelos servidores ativos, ou escalone rapidamente a capacidade para mais ou para menos, à medida que os requisitos de computação são alterados.

Separação de responsabilidades entre o usuário e o provedor



Gerenciamento de dados em nuvem

Os SGBDs em nuvem tentam minimizar os custos operacionais associados a cada usuário, reduzir a complexidade técnica, e fornecer tolerância a falhas, elasticidade, disponibilidade e escalabilidade dando uma visão de recursos quase infinitos.

Os SGBDs em nuvem devem tratar o problema de fornecimento de recursos, pois existe uma grande quantidade de usuários, múltiplos SGBDs em nuvem e grandes centros de dados.

Requisitos para gerenciamento de dados na nuvem

A definição dos requisitos é fundamental no gerenciamento de dados como um serviço:

1) Requisitos do usuário.
2) Requisitos do provedor.
3) Requisitos adicionais de nuvem pública.

Requisitos para gerenciamento de dados na nuvem - Requisitos do usuário

1) API simples com pouca configuração e administração, ou seja, sem *tuning*.
2) Alto desempenho: expresso em termos de latência e vazão, independente do tamanho da base de dados e das alterações da carga de trabalho.
3) Alta disponibilidade e confiança.
4) Acesso fácil à características avançadas: como snapshot, evolução de esquema e mineração de dados.

Requisitos para gerenciamento de dados na nuvem - Requisitos do provedor

1) Atender o SLA (Acordo de Nível de Serviço) do usuário.
2) Limitar hardware e custo de energia: o sistema deve ser eficiente na utilização dos recursos de hardware, por exemplo, fazendo ajustes dinâmicos da alocação de recursos .
3) Limitar custo de administração: ferramentas sofisticadas de análise de carga de trabalho e centralização do gerenciamento de muitos bancos de dados devem ser utilizadas.

Requisitos para gerenciamento de dados na nuvem - Requisitos de nuvem pública

1) Esquema de preço: barato, previsível e proporcional ao uso (elasticidade).
2) Garantias de segurança e privacidade.
3) Baixa latência.

Banco de dados como um serviço

Devido a necessidade de obter segurança, gerenciamento de recursos compartilhados e a extensibilidade, surgiu um novo paradigma de gerenciamento de dados onde o prestador hospeda um banco de dados e o fornece como um serviço.

DBaaS oferece a funcionalidade de um banco de dados semelhante ao que é encontrado em sistemas de gerenciamento de banco de dados relacionais (RDBMSes), como o SQL Server, MySQL e Oracle.

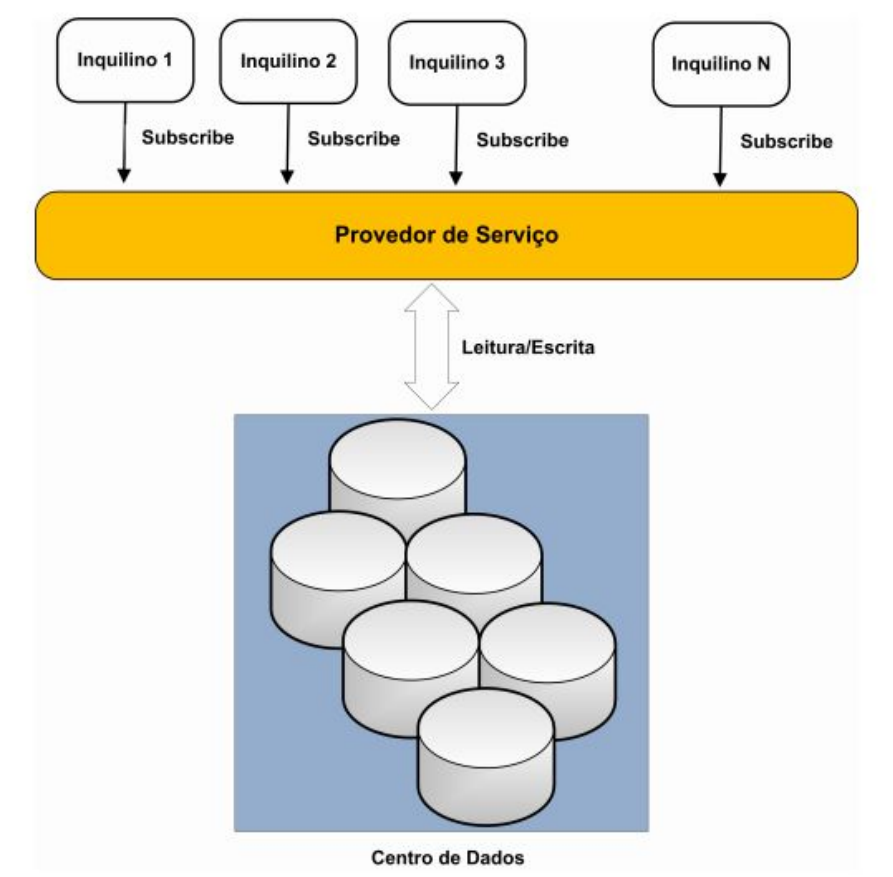
Banco de dados como um serviço - Algumas vantagens

1) Ambiente altamente escalável, flexível, disponível e rápido.
2) Não há preocupação quanto a infra-estrutura de hardware e software para fornecer o serviço de dados.
3) Fácil gerenciamento.

Banco de dados como um serviço - Algumas desvantagens

1) Falta de controle sobre os problemas de desempenho de rede, tais como falhas de latência.
2) Alguns produtos DBaaS não suportam capacidades dos RDBMS típicos, tais como compressão de dados e partições de tabela.

Banco de dados como um serviço (DaaS)



Banco de dados multi-inquilino

O termo multi-inquilino é uma estratégia para compartilhar recursos, ou seja, permite que múltiplos inquilinos (empresas/clientes) compartilhem os mesmos recursos físicos.

Um sistema de banco de dados multi-inquilino pode compartilhar máquinas, processos e tabelas.

Banco de dados multi-inquilino

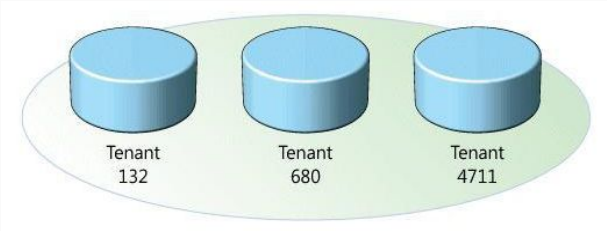
Existem três abordagens para a construção de um banco de dados multi-inquilino de acordo com o tipo de recurso compartilhado:

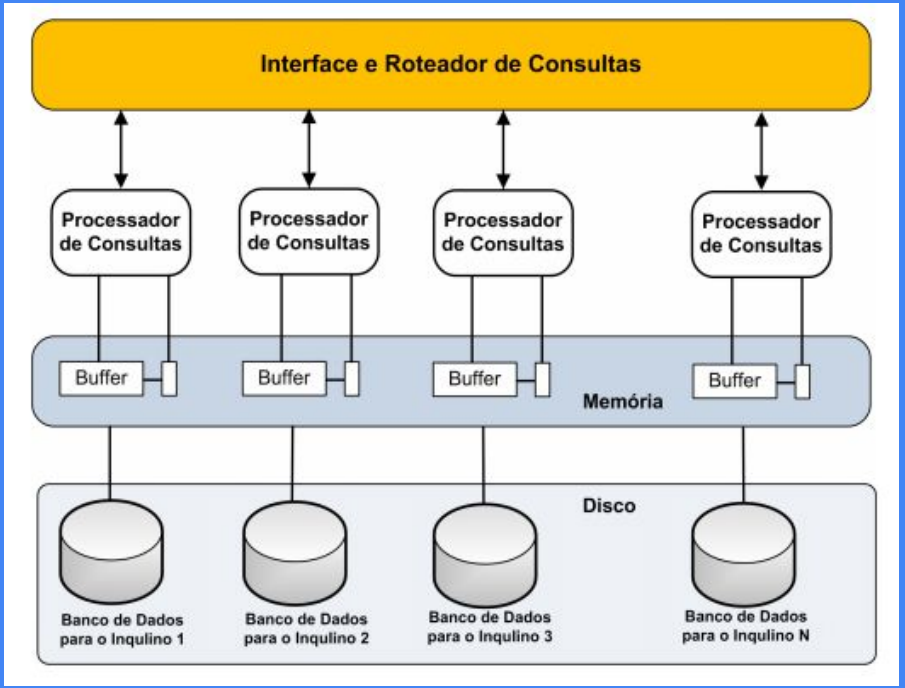
1) Banco de Dados Independentes e Instância de Banco de Dados Independente.
2) Tabelas Independentes e Instância de Banco de Dados Compartilhados.
3) Tabelas Compartilhadas e Instância de Banco de Dados Compartilhados.

Banco de Dados Independentes e Instância de Banco de Dados Independente

- Apenas o hardware é compartilhado.
- Cada inquilino tem seu próprio banco de dados que armazena os dados através da iteração com uma instância dedicada do banco de dados.

Banco de Dados Independentes e Instância de Banco de Dados Independente





Vantagens:

- Simples.
- Fácil extensão do modelo de dados.
- Isolamento dos dados, segurança e customização.

Desvantagens:

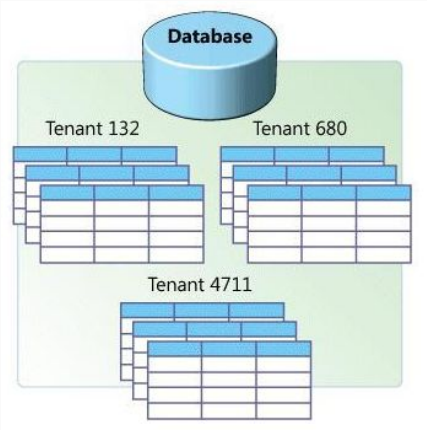
- Alto custo de manutenção.
- Alto custo de gerenciamento do equipamento e backup dos dados.
- Alto custo de hardware.

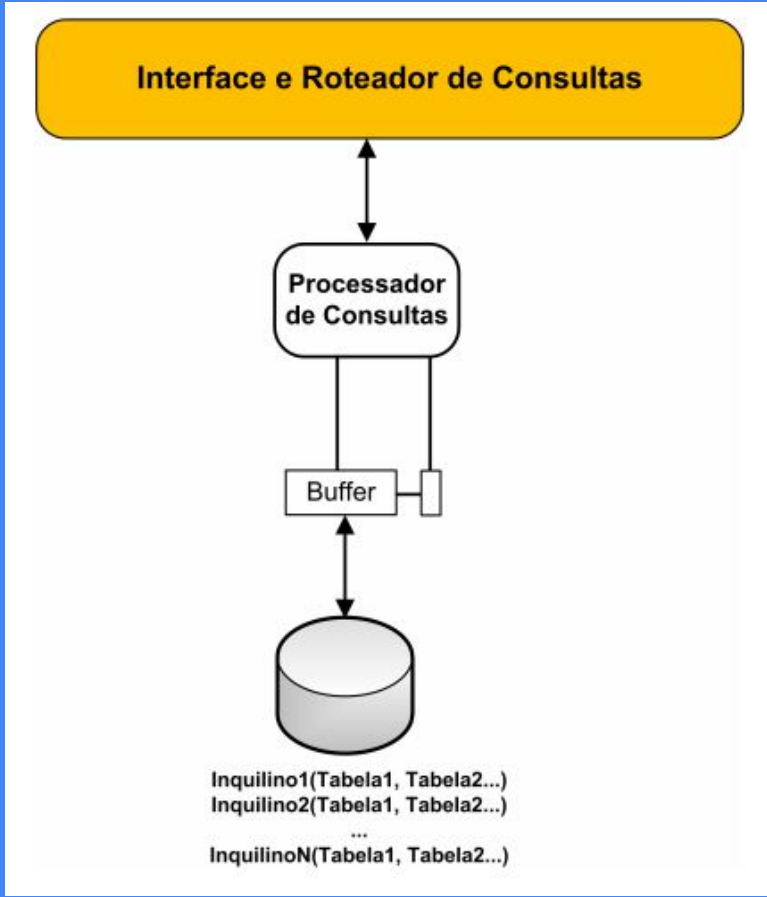
Apropriada para clientes que estejam dispostos a pagar por mais segurança e customização e que necessitam proteger suas informações necessitam proteger suas informações (gerenciamento bancário ou registros médicos).

Tabelas Independentes e Instância de Banco de Dados Compartilhados

- Os inquilinos compartilham apenas o hardware e instâncias de banco de dados.
- O provedor de serviços mantém uma grande base de dados compartilhada que serve a todos os inquilinos.
- Cada inquilino utiliza esquemas privados.
  - Um identificador para cada estrutura privada.
- Roteador de consultas é usado para reformular as consultas de acordo com o identificador.

Tabelas Independentes e Instância de Banco de Dados Compartilhados





Vantagens:

- Relativamente fácil de ser implementada.
- Fácil extensão do modelo de dados.
- Redução do custo de manutenção de gerenciamento de diferentes instâncias.
- Grau moderado de isolamento de dados e da segurança para os inquilinos.
- Oferece uma aplicação a um custo menor (acomoda mais inquilinos por servidor).

Desvantagens:

- Dados do inquilino são mais difíceis de serem restaurados no caso de uma falha.

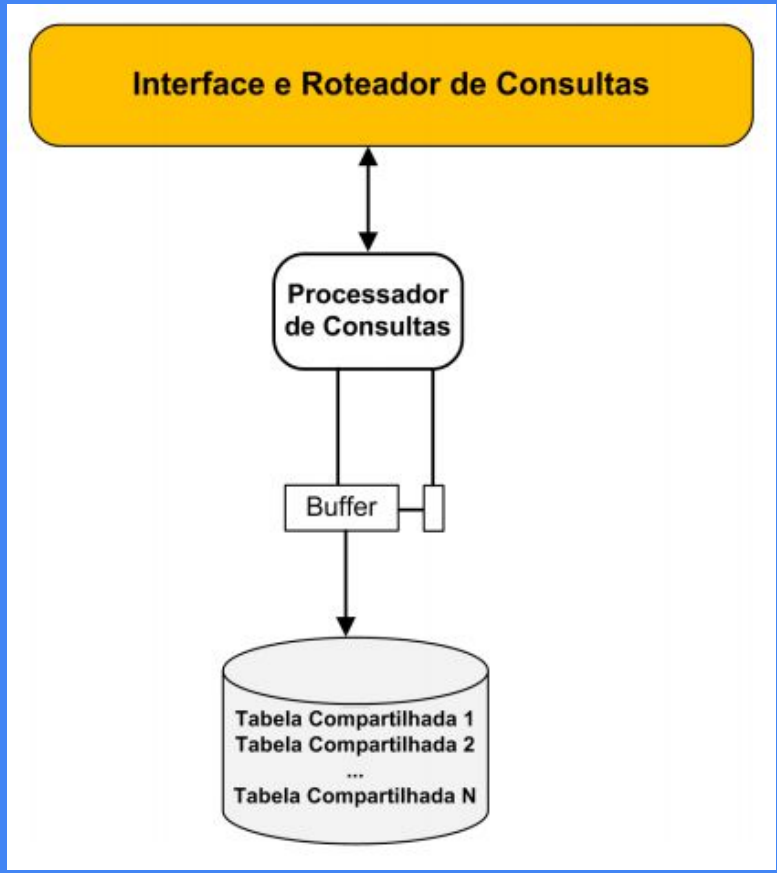
Apropriada para aplicações que usem um número pequeno de tabelas de banco de dados, na ordem de 100 tabelas por inquilino.

Tabelas Compartilhadas e Instância de Banco de Dados Compartilhados

- Os inquilinos compartilham tabelas e instâncias do banco de dados.
- Cada tabela possui um identificador para cada inquilino.
- Os atributos que não são utilizados são definidos com o valor nulo.
- Roteador de consultas é usado para reformular as consultas de acordo com o identificador.

Tabelas Compartilhadas e Instância de Banco de Dados Compartilhados

TenantID	CustName	Address
4	TenantID	ProductID	ProductNam
1	4	TenantID	Shipment	Date
6	1	4711	324965	2006-02-21
4	6	132	115468	2006-04-08
	4	680	654109	2006-03-27
		4711	324956	2006-02-23



Vantagens:

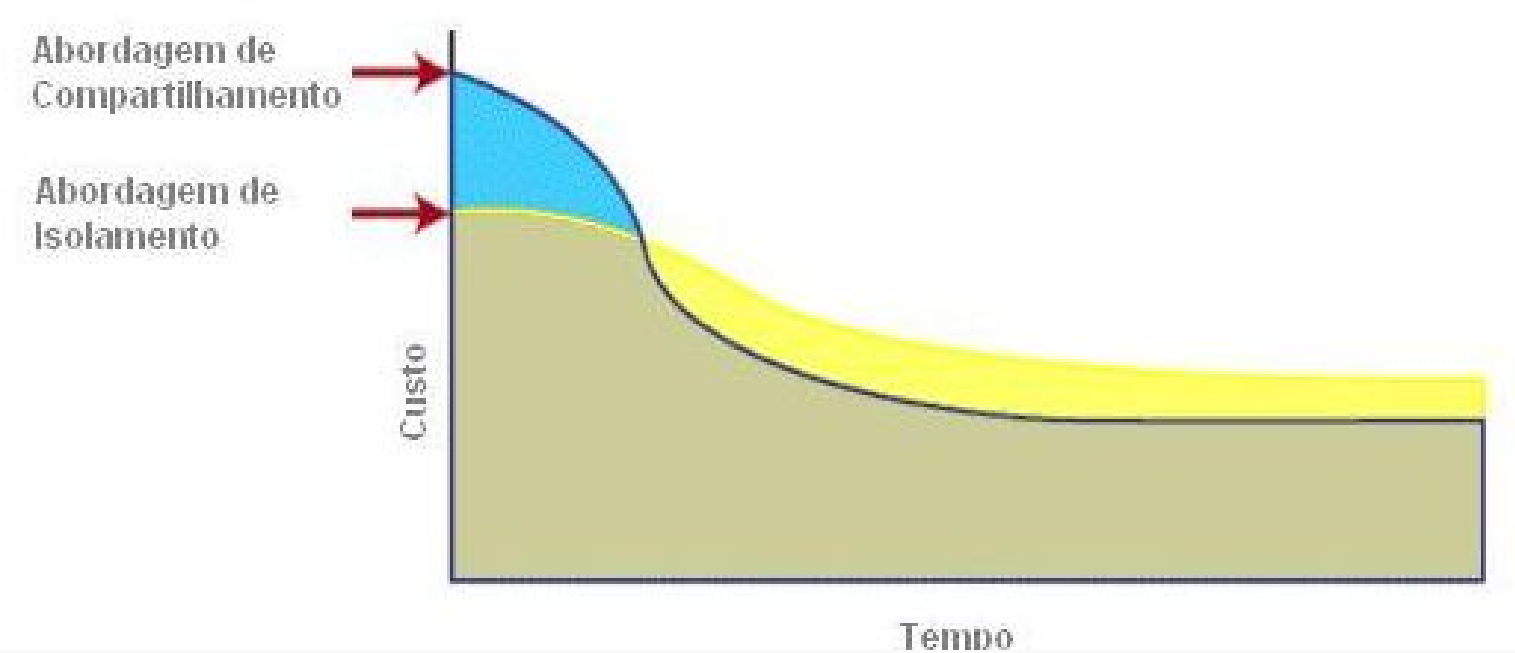
- Redução do custo de manutenção, pois o provedor de serviço mantém apenas uma instância de banco de dados simples.
- Menor custo de hardware e backup, pois permite servir o maior número de inquilinos por servidor.

Desvantagens:

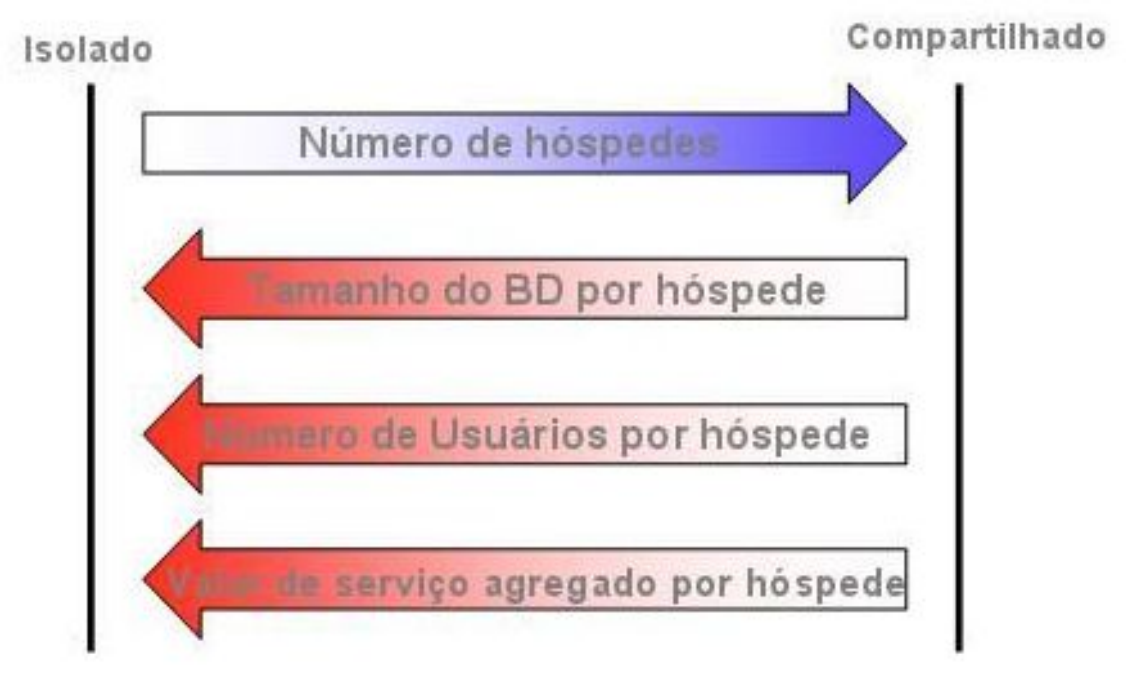
- Necessidade de indexação e otimização.
- Custo elevado na restauração de dados: linhas individuais dentro de um BD de produção precisarão ser apagadas e então re-inseridas no BD temporário.
- Esforço adicional de desenvolvimento na área de segurança

Apropriada quando se é importante que a aplicação seja capaz de servir um grande número de inquilinos com um pequeno número de servidores.

Considerações Econômicas



Considerações sobre inquilinos



Características do Gerenciamento de Dados em Nuvem

- Armazenamento e processamento de consultas
- Transações
- Escalabilidade
- Desempenho
- Tolerância à falhas e distribuição de dados

Armazenamento e processamento de consultas

Dentre as diversas abordagens para gerenciar dados, podemos destacar:

- Novos sistemas de arquivos: Google File System e o Hadoop File System.
- *Frameworks*: MapReduce.
- Propostas para o armazenamento e processamento de dados: Distributed Hash Table (DHT), colunares, orientado a documento e grafo.

Armazenamento e processamento de consultas - Google File System

Google File System (GFS) é um sistema de arquivos distribuídos desenvolvido pelo Google. O GFS foi projetado usando grandes clusters de servidores e otimizado para ser executado em centros de dados e fornecer acesso eficiente e confiável dos dados, baixa latência e tolerância a falhas individuais de servidores.

Armazenamento e processamento de consultas - Hadoop File System

O Hadoop File System (HDFS) foi inspirado pelo GFS e é um sistema de arquivos distribuídos de código livre que armazena grandes arquivos em vários servidores e obtém a confiabilidade por meio da replicação de dados.

Armazenamento e processamento de consultas - *Framework* MapReduce

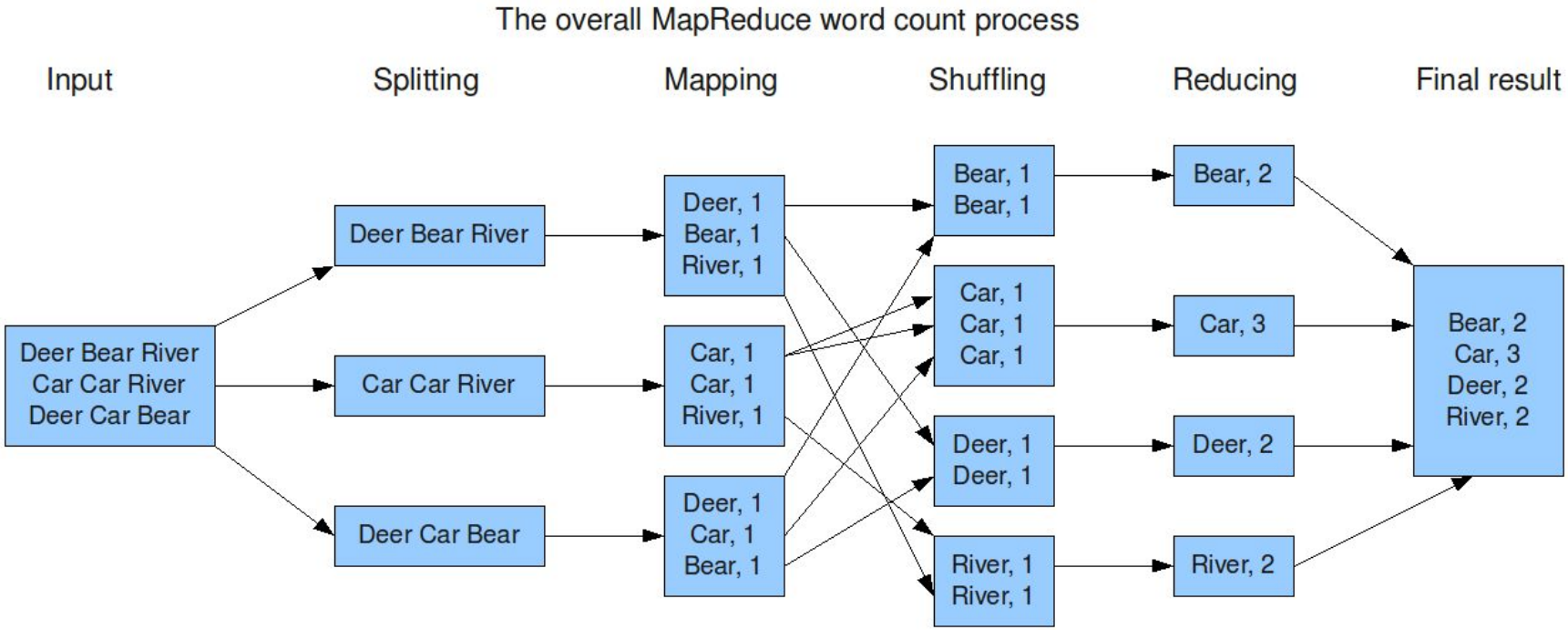
- O *framework* MapReduce foi introduzido pelo Google e permite o processamento paralelo de grandes conjuntos de dados em clusters de servidores.
- MapReduce processa os dados na forma de pares chave-valor.
- Existem algumas implementações de código livre, dentre as quais se destaca o Hadoop MapReduce.

MapReduce

O modelo MapReduce é executado em 6 passos usando duas funções: Map e Reduce:

1. Entrada: O leitor lê os dados de entrada de um sistema de arquivos distribuído.
2. Separação: processo de tokenização.
3. Mapeamento: é uma função definida pelo usuário, que recebe uma porção da entrada e emite um conjunto de tuplas intermediárias no formato chave-valor.
4. Misturar e Ordenar: os pares de chave-valor são combinados.
5. Redução: os pares de chave-valor são agregados pela quantidade.
6. Resultado final: Ao final, todas as entradas (palavras) podem ser mostradas em uma tabela, contendo o índice (a própria palavra) e o número de vezes que essa palavra apareceu.

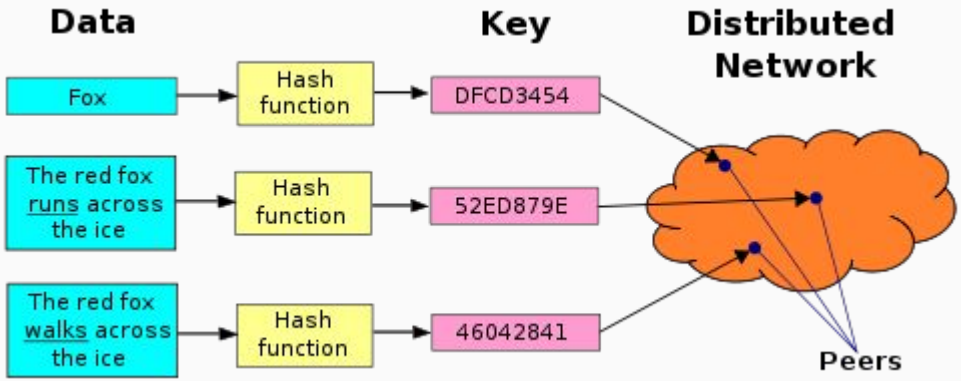
MapReduce

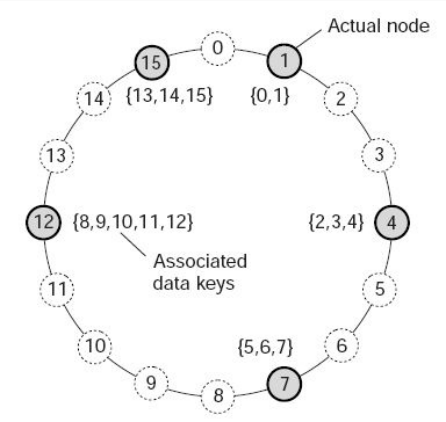

The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Armazenamento e processamento de consultas - *Tabela Hash distribuída*

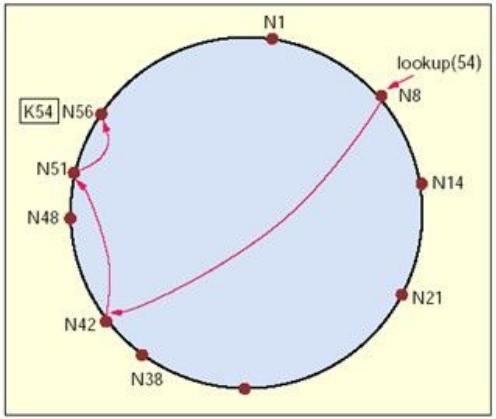
- Tabelas hash mapeiam chave em valores.
- Estruturas de chave-valor são armazenados de forma descentralizada, onde cada um destes nós do sistema possui parte dos dados.
- Uma DHT provê uma função de *lookup(k)* que busca o nó responsável pela chave k, podendo recuperar ou modificar o valor associado a essa dada chave.
- As APIs básicas utilizam operações tais como get(key) e put(key, value) para acessar os dados.

Tabela Hash distribuída





- Dados distribuídos em múltiplos nós.
- Nós mantêm informações sobre os seus vizinhos através de tabelas de roteamento.
- Operação lookup(k): ao receber uma requisição para a chave k, a consulta é repassada para o nó apropriado, que contém o valor associado a chave.



Armazenamento e processamento de consultas - Colunares

- Os dados são organizados em tabelas e estas possuem diversas colunas.
- Cada coluna armazena um valor, acessado por meio de uma chave.

Id	Nome	Salário
1	Pedro	5.000
2	Marcos	2.000

Linha → 1, Pedro, 5.000; 2, Marcos, 2.000;

Coluna → 1, 2; Pedro, Marcos; 5.000, 2.000;

Armazenamento e processamento de consultas - Documento

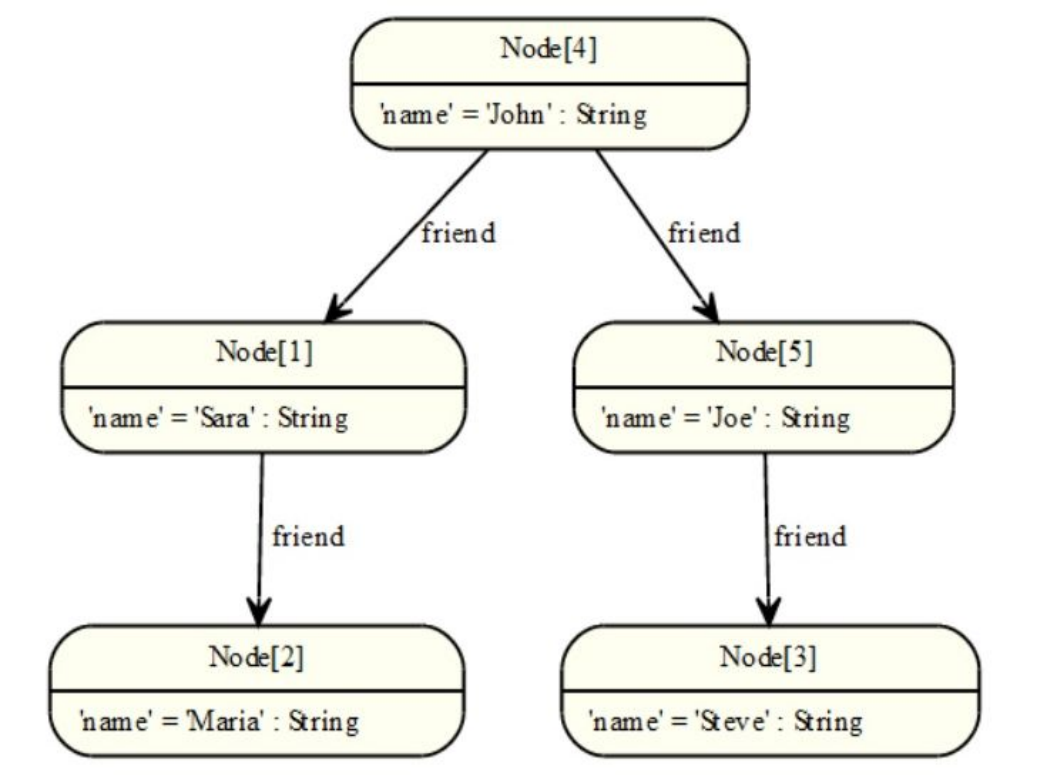
- Os dados são armazenados, consultados e apresentados na forma de uma estrutura de documento de texto.
- Essa abordagem utiliza diversos formatos, como o XML e o JSON.

Armazenamento e processamento de consultas - Grafos

Na abordagem baseada em grafo, os dados são gerenciados da seguinte forma:

- Vértice: cada nó do grafo representa uma instância de um objeto com identificador único.
- Arestas: os relacionamentos entre dois nós são representados por arestas, sendo que estes nós possuem uma direção e um tipo de relacionamento.
- Atributo: são pares de strings chave-valor do objeto que podem existir tanto em um nó quanto em um relacionamento.

Abordagem baseada em grafos



Transações

Para obter controle do processamento de dados no ambiente de computação em nuvem, é definido o uso de transações distribuídas, que tem a responsabilidade de garantir as propriedades ACID ou variações.

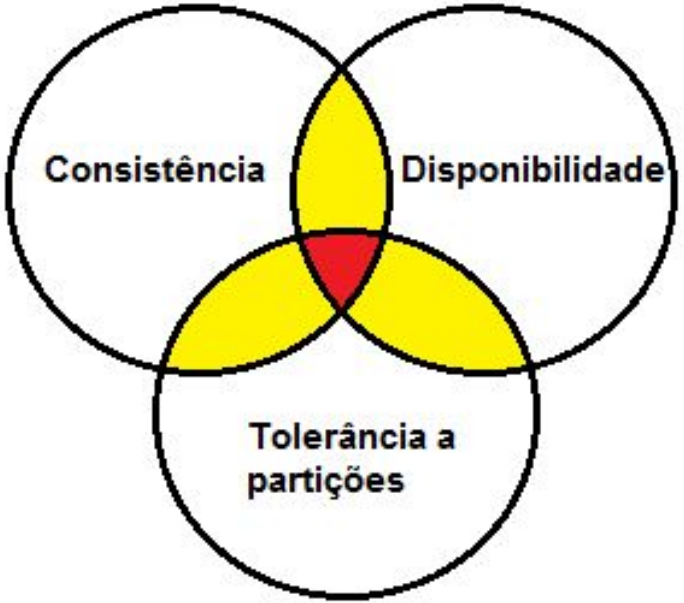
Transações - ACID

- **A**tomicidade: na transação ou se faz tudo, ou nada, sem meio termo.
- **C**onsistência: garante que o banco de dados antes da transação esteja consistente e, que após a transação o banco permaneça consistente.
- **I**solamento: objetiva garantir que nenhuma transação seja interferida por outra até que ela seja completada.
- **D**urabilidade: garante que a informação gravada no banco de dados dure de forma imutável até que alguma outra transação de atualização, ou exclusão afete-a.

Teorema CAP

O teorema CAP (Consistency, Availability, Partition Tolerance) é considerado por todos os sistemas em nuvem um ponto fundamental na construção de sistemas distribuídos.

Este teorema mostra que os sistemas distribuídos podem suportar apenas duas dessas três propriedades simultaneamente.



Teorema CAP

- Consistência: característica que descreve como e se o estado de um sistema fica consistente após uma operação, e que uma vez escrito um registo, este fica disponível para ser utilizado imediatamente.
- Disponibilidade: falhas em nós não impedem os demais nós de continuar a operar.
- Tolerância a partições: no caso de uma falha na rede, a base de dados ainda deve ser capaz de fornecer operações de leitura/escrita mesmo estando particionada.

Consistência

Formas de consistência:

- Consistência forte: garante que após uma atualização ser concluída, qualquer acesso subsequente fornecerá o valor atualizado.
- Consistência fraca: o sistema não garante que os acessos subsequentes irão retornar o valor atualizado.
- Consistência eventual: é uma forma específica de consistência fraca, na qual o sistema garante que, se nenhuma atualização for feita ao dado, todos os acessos irão devolver o último valor atualizado.

BASE

Um sistema fortemente consistente são sistemas que implementam as características ACID.

Em decorrência do teorema CAP, no outro extremo, surgiram outras abordagens que utilizam diferentes formas de consistência para o gerenciamento de dados em nuvem, como a abordagem Basically Available, Soft state, Eventually consistent (BASE), que é caracterizada pelo sistema ser:

- Basicamente disponível, pois este parece estar em funcionamento todo o tempo.
- Em estado leve, já que o sistema não precisa estar sempre consistente.
- Eventualmente consistente, que define que o sistema torna-se consistente em um determinado momento.

Escalabilidade

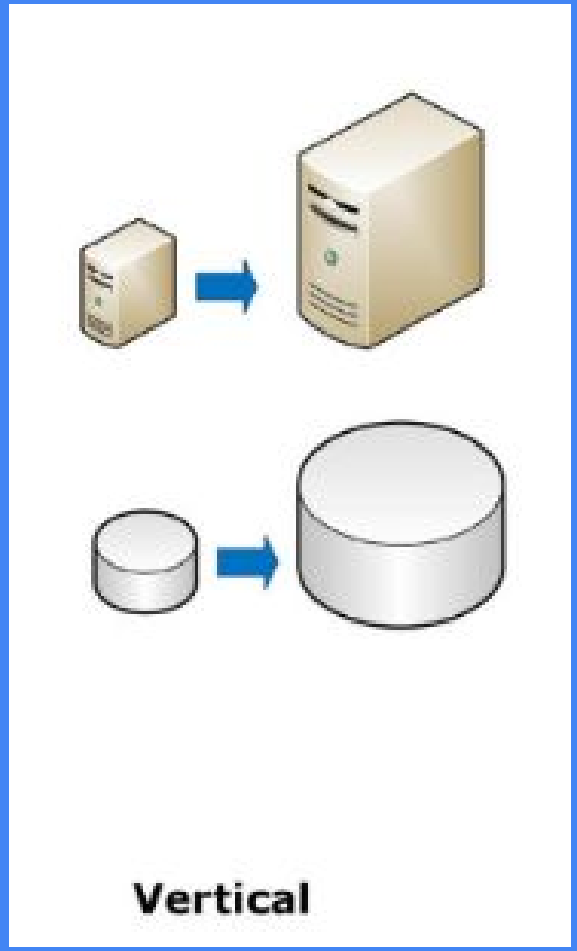
A escalabilidade pode ser definida como a possibilidade de ampliar ou reduzir recursos tecnológicos de acordo com a demanda.

A nuvem é composta por milhares máquinas que necessitam prover escalabilidade transparente para os usuários.

É possível identificar duas dimensões de escalabilidade: vertical e horizontal.

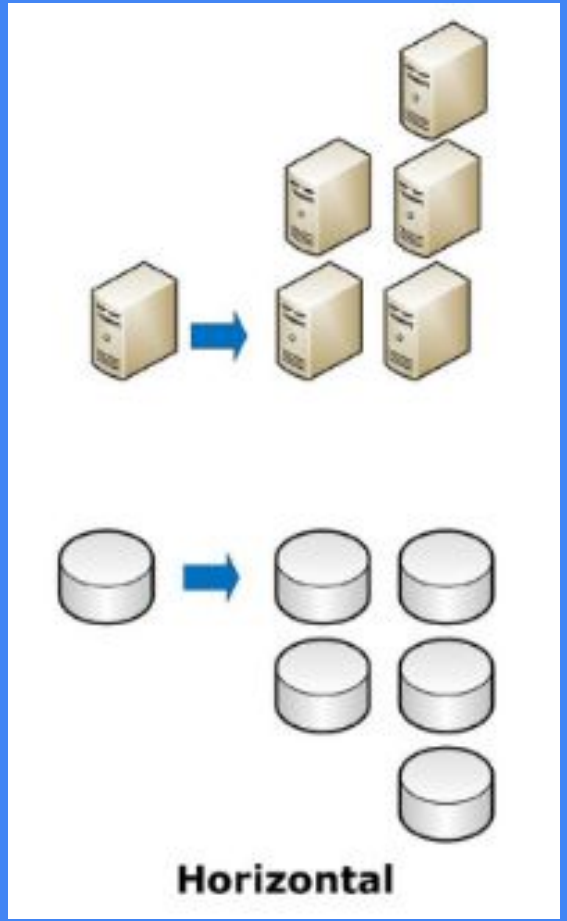
Escalabilidade Vertical

- Melhora-se a capacidade do hardware
- Isto funciona bem para os dados, mas tem várias limitações tais como a aquisição constante de hardware de maior capacidade, aumentando os custos.



Escalabilidade Horizontal

- Consiste em adicionar mais máquinas à solução atual, distribuindo requisições e dados por vários SGBDs.
- Ocorre a fragmentação dos dados.
- Oferece maior flexibilidade, mas necessita de um planejamento específico



Desempenho

Diversos fatores que influenciam no desempenho:

- Diferentes tecnologias.
- Heterogeneidade do hardware utilizado pelas máquinas.
- Falhas no hardware.
- Disputa por recursos não virtualizados.
- Carga de trabalho dividida igualmente entre as máquinas com configurações diferentes.

Tolerância a Falhas e Distribuição de Dados

- Soluções em nuvem tem que ser tolerante a falhas.
- Para tratar as falhas, as soluções em nuvem utilizam técnicas que auxiliam a distribuição dos dados:
  - Fragmentação
  - Replicação

Distribuição de Dados - Fragmentação

Por meio da fragmentação dos dados é possível melhorar a disponibilidade e distribuir a carga, tanto para operações de escrita quanto de leitura.

Se apenas uma máquina falhar, os dados pertencentes a essa máquina são afetados, mas não o armazenamento de dados como um todo.

As técnicas de armazenamento com DHT e colunas facilitam a fragmentação dos dados.

Distribuição de Dados - Replicação

Tem o objetivo de criar e iniciar várias cópias idênticas de um mesmo dado e mantê-los em diversos SGBDs.

- Reduz a quantidade total de armazenamento e, portanto, os custos associados.
- Reduz o tráfego na rede
- Melhora a disponibilidade para operações de leitura, pois se um nó estiver inacessível, os dados podem ser consultados no nó que contiver a réplica.

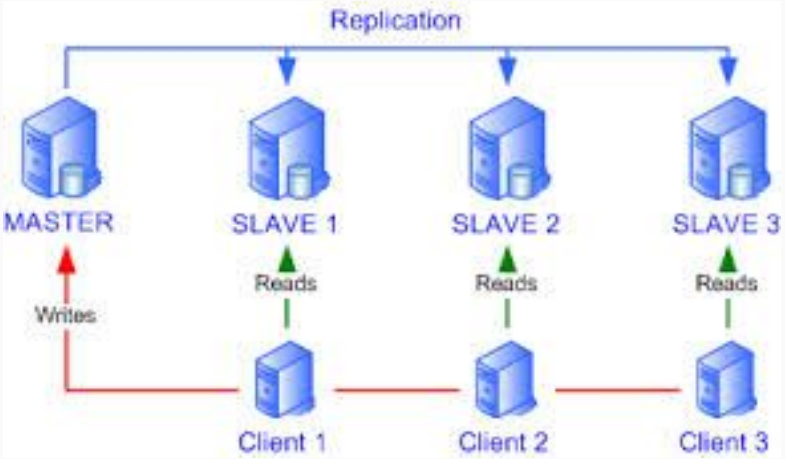
Existem dois modelo de replicação:

- Replicação mestre/escravo.
- Replicação multimestre.

Replicação Mestre/Escravo

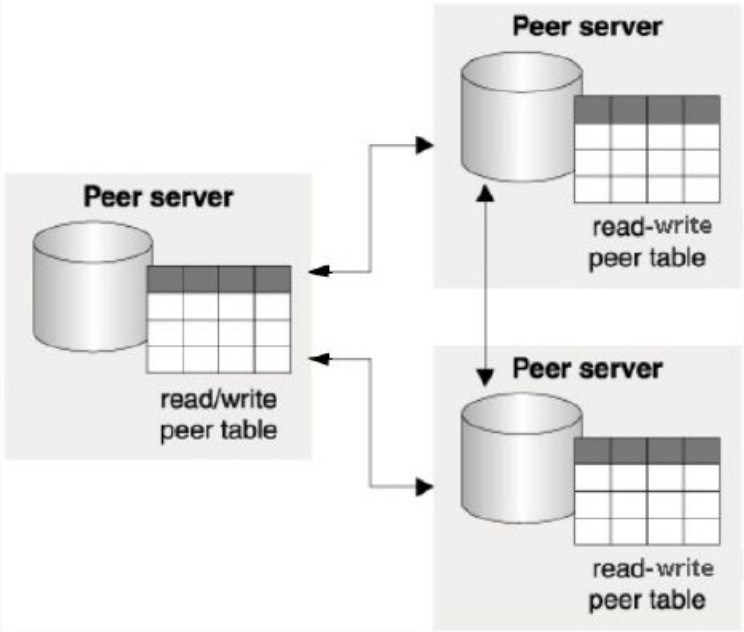
Apenas o nó mestre recebe atualizações do cliente e pode realizar escritas, enquanto os escravos realizam apenas leitura.

Um problema desta abordagem é que qualquer falha em um nó mestre impede que sejam realizadas atualizações



Replicação Multimestre

Vários nós gerenciam os objetos replicados do banco de dados, de forma que leituras e gravações podem ser realizadas em qualquer uma deles. Uma falha em qualquer nó não impede o correto funcionamento dos demais



Propagação de dados

Existem duas estratégias de propagação de dados:

- Forma síncrona
- Forma assíncrona

O tipo de estratégia escolhida pode determinar o nível de consistência, confiabilidade e desempenho do sistema.

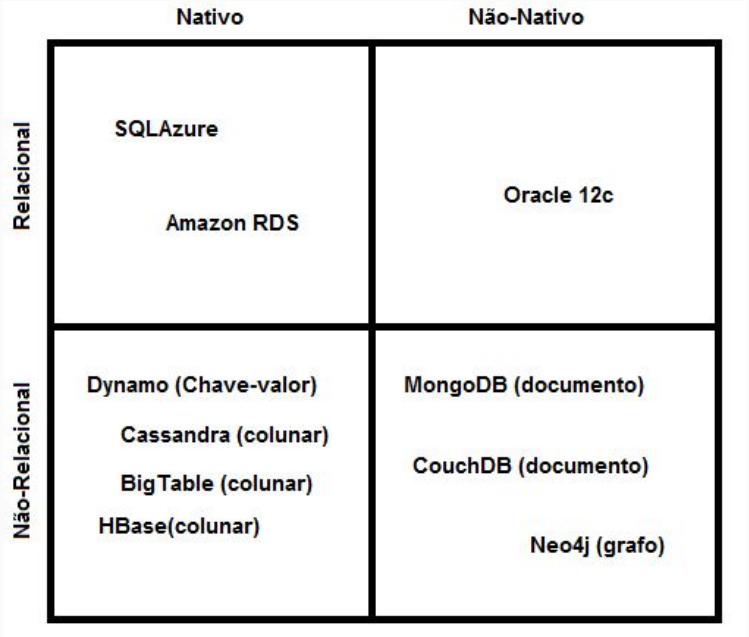
Propagação de dados - Síncrona

Se alguma cópia do banco é alterada, essa alteração será imediatamente aplicada a todos os outros bancos dentro da transação. Essa técnica garante consistência na transação entre servidores, porém há perda da performance do sistema, pois a transação só retornará para o usuário que executou a ação, quando as demais bases estiverem atualizadas. No entanto, essa técnica possui alto custo associado, pois exige um meio de transmissão de dados de alta velocidade com padrão de qualidade superior ao modelo de replicação assíncrona.

Propagação de dados - Assíncrona

Se um banco de dados é alterado, a alteração será propagada e aplicada nos outros SGBDs dentro de uma transação separada, ou seja, a replicação é concluída e depois replicada. Esta ação poderá ocorrer em segundos, minutos, horas ou até dias depois, dependendo da configuração pré-estabelecida. Essa técnica não garante consistência na transação entre servidores, pois as informações nas máquinas envolvidas não estarão sempre atualizadas, até que o processo de replicação seja iniciado.

Sistemas de Gerenciamento de Dados em Nuvem



Sistemas de Gerenciamento de Dados em Nuvem - Relacionais *versus* NoSQL

Bancos de dados relacionais têm grande facilidade a distribuição vertical de servidores, ou seja, quanto mais dados, mais memória e mais disco um servidor precisa.

O NoSQL tem grande facilidade na distribuição horizontal, ou seja, mais dados, mais servidores, não necessariamente de alta performance.

Sistemas de Gerenciamento de Dados em Nuvem - Relacionais *versus* NoSQL

Os SGBDs relacionais atendem principalmente sistemas financeiros e corporativos, como a Previdência Social e Caixa Econômica Federal, que necessitam de alta consistência de dados.

Enquanto os SGBDs não relacionais são muito utilizados por empresas, como o Twitter, Facebook, Netflix e aplicativos da Google, que são empresas que devem processar um grande volume de dados de forma rápida.

Dúvidas?