**Aula01 - Introdução aos Bancos de Dados Não Relacionais**

**1. O que são bancos de dados não relacionais (NoSQL)**

Bancos de dados NoSQL são sistemas de gerenciamento de banco de dados que não seguem o modelo relacional tradicional. Eles são projetados para lidar com grandes volumes de dados, alta velocidade de leitura e escrita, e estruturas de dados flexíveis.

**2. Diferenças entre bancos de dados relacionais e não relacionais**

* **Modelo de Dados**: Relacionais usam tabelas com linhas e colunas; NoSQL usa diversos modelos (chave-valor, documentos, colunas, grafos).
  + **Modelo Relacional**

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

* + **Modelo Chave-Valor**

**Chave-Valor** Neste modelo, os dados são armazenados como pares de chave-valor. Por exemplo, em um banco de dados Redis:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

* + **Modelo Orientado a Documentos**

**Documentos** Os dados são armazenados em documentos JSON ou BSON. Por exemplo, em um banco de dados MongoDB:

Texto, Carta

Descrição gerada automaticamente

* + **Modelo Orientado a Colunas**

**Colunas** Os dados são armazenados em colunas em vez de linhas. Por exemplo, em um banco de dados Cassandra:

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

* + **Modelo Orientado a Grafos**

**Grafos** Os dados são armazenados como nós e arestas. Por exemplo, em um banco de dados Neo4j:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

* **Esquema**: Relacionais têm esquemas rígidos; NoSQL tem esquemas flexíveis.
  + **Esquema Rígido (Banco de Dados Relacional)**
    - Em um **banco de dados relacional**, o esquema é definido antes de inserir os dados e deve ser seguido rigorosamente.

Texto

Descrição gerada automaticamente

* + Se quisermos adicionar um novo campo, como "Endereço", precisamos alterar o esquema da tabela:



* + **Esquema Flexível (Banco de Dados NoSQL)**
    - Em um banco de dados NoSQL, como o MongoDB, não há necessidade de definir um esquema rígido. Podemos inserir documentos com diferentes campos sem alterar a estrutura do banco de dados

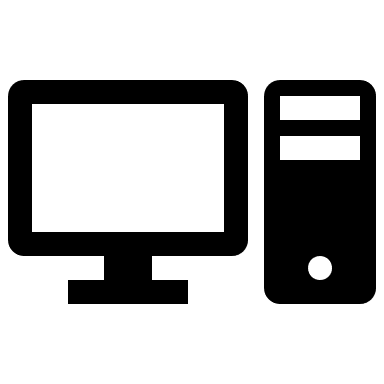
Texto, Carta

Descrição gerada automaticamente

* **Escalabilidade**: Relacionais escalam verticalmente (aumentando a capacidade do servidor); NoSQL escala horizontalmente (adicionando mais servidores).
  + **Escalabilidade Vertical (Banco de Dados Relacional)**

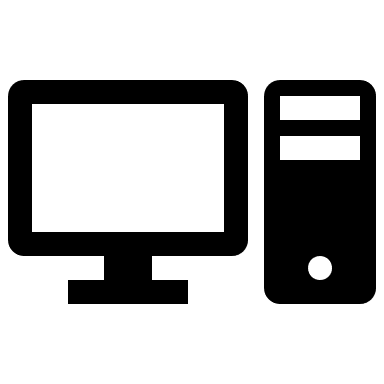
**Escalabilidade Vertical** significa aumentar a capacidade de um único servidor, adicionando mais recursos como CPU, memória e armazenamento. Imagine que você tem um servidor que está ficando sobrecarregado com o aumento de dados e transações. Para melhorar o desempenho, você pode:

* + - **Adicionar mais memória RAM** para suportar mais operações simultâneas.
    - **Atualizar o processador** para um modelo mais rápido.
    - **Aumentar o armazenamento** para acomodar mais dados.



**Configuração antes upGrade**

Servidor A: 16 GB RAM, 4 CPUs, 1 TB de armazenamento



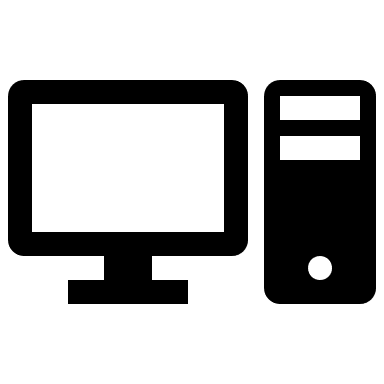
**Configuração depois upGrade**

Servidor A: 32 GB RAM, 8 CPUs, 2 TB de armazenamento

* + **Escalabilidade Horizontal (Banco de Dados NoSQL)**

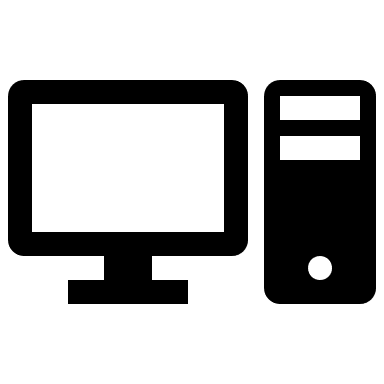
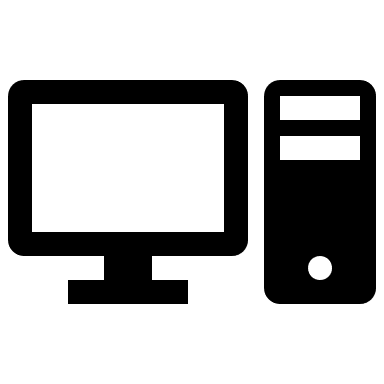
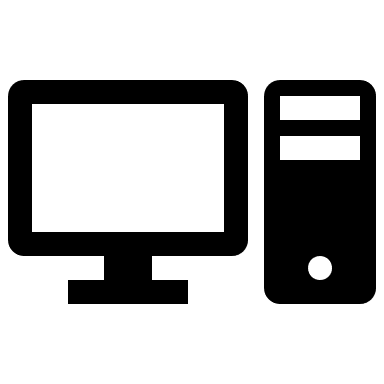
**Escalabilidade Horizontal** significa adicionar mais servidores para distribuir a carga de trabalho. Em vez de melhorar um único servidor, você adiciona mais servidores ao sistema. Isso é especialmente útil para lidar com grandes volumes de dados e altas taxas de transações.

**Cenário antes da Escalabilidade Horizontal**



Servidor A: 16 GB RAM, 4 CPUs, 1 TB de armazenamento

**Cenário depois da Escalabilidade Horizontal**



Servidor A: 16 GB RAM, 4 CPUs, 1 TB de armazenamento

Servidor C: 16 GB RAM, 4 CPUs, 1 TB de armazenamento

Servidor B: 16 GB RAM, 4 CPUs, 1 TB de armazenamento

* **Transações**: Relacionais suportam transações ACID; NoSQL pode priorizar a consistência eventual.

**3. Exemplos de uso no mercado**

* **Redes Sociais**: Facebook usa Cassandra para gerenciar grandes volumes de dados de usuários.
* **E-commerce**: Amazon usa DynamoDB para gerenciar catálogos de produtos e transações.

**4. História e Evolução**

**Surgimento dos bancos de dados NoSQL**

Os bancos de dados NoSQL surgiram no final dos anos 2000 como resposta à necessidade de gerenciar grandes volumes de dados gerados por aplicações web e móveis.

**Principais marcos históricos**

* **2007**: Lançamento do Cassandra pelo Facebook.
* **2009**: Lançamento do MongoDB.
* **2010**: Google lança o Bigtable.

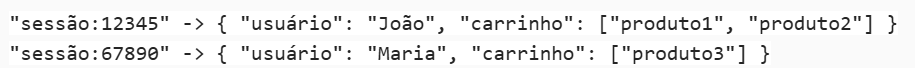
**5. Tipos de Bancos de Dados Não Relacionais**

**Chave-valor**

* **Exemplo**: Redis.
* **Uso**: Armazenamento de sessões, cache.

Imagine que você tem um site de e-commerce e precisa armazenar as sessões dos usuários para que eles não precisem fazer login novamente a cada visita. Você pode usar Redis para armazenar essas sessões como pares de chave-valor.

**Exemplo Visual:**



**Orientados a documentos**

* **Exemplo**: MongoDB.
* **Uso**: Aplicações web, gerenciamento de conteúdo.

Suponha que você está desenvolvendo um blog e precisa armazenar artigos com diferentes campos, como título, autor, data e conteúdo. MongoDB permite armazenar esses artigos como documentos JSON.

**Exemplo Visual:**

Texto

Descrição gerada automaticamente

**Orientados a colunas**

* **Exemplo**: Cassandra.
* **Uso**: Análise de dados, redes sociais.

Imagine que você está gerenciando uma rede social e precisa armazenar grandes volumes de dados de usuários e suas atividades. Cassandra permite armazenar esses dados em colunas, facilitando a análise e a consulta.

**Exemplo Visual:**

Uma imagem contendo Texto

Descrição gerada automaticamente

**Orientados a grafos**

* **Exemplo**: Neo4j.
* **Uso**: Redes sociais, recomendações.

Suponha que você está desenvolvendo uma rede social e precisa armazenar as conexões entre os usuários, como amizades e recomendações. Neo4j permite armazenar esses dados como nós e arestas em um grafo.

**Exemplo Visual:**

Texto

Descrição gerada automaticamente com confiança baixa

**6. Vantagens e Desvantagens**

**Vantagens**

* **Escalabilidade**: Fácil de escalar horizontalmente.
* **Flexibilidade de esquema**: Adapta-se a mudanças nos dados.
* **Desempenho**: Alta velocidade de leitura e escrita.

**Desvantagens**

* **Consistência**: Pode não garantir consistência imediata.
* **Complexidade**: Pode ser mais complexo de gerenciar.

**7. Estudo de Caso: Uso do MongoDB no e-commerce**

**Contexto**

Uma grande empresa de e-commerce precisava de um banco de dados que pudesse lidar com grandes volumes de dados de produtos e transações.

**Solução**

Implementaram o MongoDB para gerenciar catálogos de produtos, permitindo uma busca rápida e eficiente, além de flexibilidade para adicionar novos atributos aos produtos sem necessidade de alterar o esquema do banco de dados.

**Resultados**

* **Desempenho**: Melhorou a velocidade de busca e recuperação de dados.
* **Escalabilidade**: Facilitou a adição de novos servidores para suportar o crescimento.