Banco de Dados

Sistemas NoSQL

(BD de Agregados)

Valéria Cesário Times
vct@cin.ufpe.br

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

Introdução a Sistemas NoSQL

- Por que BD relacionais se tornaram dominantes?
  - Capacidade de armazenar grandes quantidades de dados persistentes
    - Permite acessar partes do BD de maneira rápida e fácil
  - Controle de concorrência por meio de transações
    - Provê acesso concorrente de múltiplas aplicações ou vários usuários
  - Recuperação após falhas
    - Restaura o BD para um estado anterior ao estado da falha
  - Integração compartilhada de BD
    - Aplicações armazenam seus dados em uma única BD
  - Uso de um modelo de dados padrão
    - Dialetos SQL usados por diversos fornecedores são similares

Introdução a Sistemas NoSQL

- Por que existe uma ascensão de BD NoSQL?
  - Incompatibilidade de impedância entre o modelo relacional e as estruturas de dados na memória principal

Modelo Relacional

Relações (Tabelas) : conjunto de tuplas

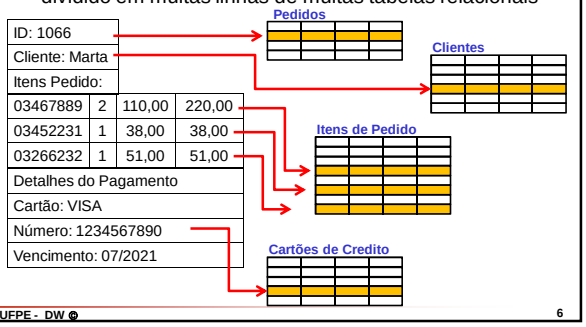
Tuplas (Linhas): conjunto de pares *nome-valor*

Álgebra Relacional: operações SQL consomem e retornam relações

- Valores de uma tupla relacional têm de ser simples
- Para armazenar dados em disco, é preciso traduzir a estrutura de dados não atômica da memória para a representação relacional
- Incompatibilidade de impedância: duas representações diferentes que requerem tradução

Introdução a Sistemas NoSQL

- Exemplo de Incompatibilidade de Impedância
  - Um pedido que aparenta ser uma estrutura única na GUI é dividido em muitas linhas de muitas tabelas relacionais



Introdução a Sistemas NoSQL

- Incompatibilidade de Impedância
  - Na década de 1990, houve o crescimento de LP OO e com elas, surgiram os SGBD OO
    - Mas enquanto o uso de LP OO cresceu, os SGBD OO caíram na obscuridade
  - Frameworks de mapeamento objeto-relacional facilitaram a convivência com a incompatibilidade de impedância
    - Exemplo: Hibernate e iBATIS
    - Implementam padrões de mapeamento conhecidos e poupam as pessoas de trabalho mas seu uso incorreto pode comprometer o desempenho do sistema
  - SGBD relacionais dominaram o mercado na década de 2000, mas alguns problemas começaram a surgir em seu domínio

Introdução a Sistemas NoSQL

- BD de integração x BD de aplicativos
  - Por que SGBDR prevaleceram sobre SGBDOO?
    - Uso de SQL como mecanismo de integração entre as aplicações
  - BD de integração: várias aplicações desenvolvidas por equipes distintas armazenam seus dados num único local
    - Melhora a comunicação porque as aplicações operam sobre um conjunto consistente de dados persistentes
    - Possui algumas desvantagens....

Introdução a Sistemas NoSQL

- Desvantagens do BD de integração
  - Uma estrutura projetada para integrar muitas aplicações acaba se tornando mais complexa do que qualquer necessidade de uma única aplicação
  - Se uma aplicação precisar modificar seu armazenamento de dados precisará coordenar essa modificação com as demais que usam o BD
  - Aplicações diferentes possuem requisitos estruturais e de desempenho diferentes
    - Um índice requerido por uma aplicação pode causar problemas nas inserções da outra
  - Como cada aplicação pertence a uma equipe diferente significa que o SGBD precisa assumir a responsabilidade de manutenção da integridade dos dados

Introdução a Sistemas NoSQL

- BD de integração x BD de aplicativos
  - BD de aplicativos: só é acessado diretamente por uma única aplicação que é gerenciada por apenas uma equipe
    - Facilidade de manutenção e desenvolvimento de esquema porque apenas uma equipe precisará conhecer sua estrutura
    - Uma vez que essa equipe controla o BD e o código da aplicação, a responsabilidade pela integridade do BD é dada a esse código
  - Há um movimento na direção ao uso de BD de aplicativos
    - Defende-se o encapsulamento de BD em aplicativos e a integração por meio de serviços
  - Uso de serviços web como mecanismo de integração:
    - Desacopla o BD e os serviços responsáveis pela comunicação com o mundo exterior
    - Não importa aos serviços como os dados são armazenados, permitindo que opções não relacionais sejam consideradas

Introdução a Sistemas NoSQL

- Aspectos BD de aplicativos
  - Provê mais flexibilidade na estrutura dos dados trocados
    - Por meio de um serviço, pode-se usar estruturas de dados mais ricas: registros aninhados ou listas (documentos XML ou JSON)
    - O uso de uma estrutura rica de informações em uma única solicitação ou resposta reduz o custo de comunicações remotas
  - Porém, a maioria das equipes que optou pela abordagem de BD de aplicativos permaneceu com BD relacionais:
    - SGBDR possuem familiaridade, estabilidade e vários recursos
    - Não está claro que BD de aplicativos tenha influenciado formas alternativas de armazenamento de dados.

Introdução a Sistemas NoSQL

- Qual o fator vital para mudança no armazenamento de dados?
  - Necessidade de suporte a grandes volumes de dados por meio da execução em clusters
    - BDR não foram projetados para serem executados em clusters
  - O termo NoSQL é um neologismo acidental
    - Não há uma descrição oficial
    - Representa características comuns de BD NoSQL
      - Não usam o modelo relacional nem SQL
      - Executam eficientemente em clusters, mas nem todos os BD NoSQL almejam a execução em clusters
      - Possuem código aberto
      - Não têm esquema de dados
      - Foram criados para propriedades na web do século XXI
    - O resultado mais importante do surgimento de NoSQL é a persistência poliglota

Introdução a Sistemas NoSQL

- O que é a persistência poliglota?
  - Consiste em usar diferentes armazenamentos de dados em diferentes circunstâncias
    - Corresponde a uma mistura de tecnologias de armazenamentos de dados para diferentes situações
  - É preciso entender a natureza dos dados a serem armazenados e como serão manipulados
  - Para fazer o mundo poliglota funcionar, é preciso migrar de BD de integração para BD de aplicativos

Introdução a Sistemas NoSQL

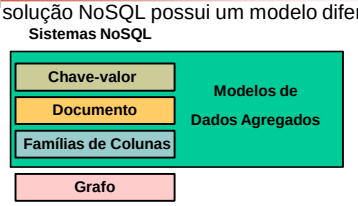
- Motivos principais para adotar BD NoSQL:
  - Simplifica o acesso ao BD, mesmo que não haja a necessidade de escalar para além de uma única máquina
  - Melhora a produtividade de desenvolvimento de aplicativos usando um estilo de interação de dados mais conveniente
  - Permite lidar eficientemente com o acesso a dados cujo tamanho e desempenho demandam um cluster

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

Modelos de Dados de BD NoSQL

- Cada solução NoSQL possui um modelo diferente:



- A orientação agregada reconhece que frequentemente deseja-se trabalhar com dados na forma de unidades
  - Tais unidades possuem uma estrutura mais complexa do que um conjunto de tuplas
- Apesar de não existir um *termo comum* que identifique tais unidades, BD dos tipos chave-valor, documentos e famílias de colunas fazem uso da unidade de agregados

Modelos de Dados de BD NoSQL

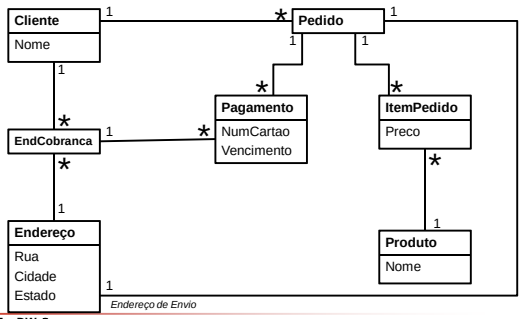
- O que é um agregado?
  - É um conjunto de objetos relacionados que desejamos tratar como uma unidade
  - É uma unidade de manipulação de dados e de gerenciamento de consistência
    - Comunicação com o gerenciador de armazenamento de dados do sistema é feita em termos de agregados
    - Agregados são atualizados com operações atômicas
  - Lidar com agregados facilita a execução de BD NoSQL em clusters, já que o agregado constitui uma unidade natural para replicação e fragmentação
  - Agregados também são mais simples de serem manipulados por programadores de aplicativos

Modelos de Dados de BD NoSQL

- Agregado é a base para BI em tempo real
  - Agregados também podem ser usados para obter estatísticas
  - Exemplo: Agregado pode conter informações sobre quais pedidos possuem um dado produto
    - Sempre que o pedido for realizado pelo cliente a estatística mantida no agregado é atualizada
  - Essa desnormalização permite o acesso rápido aos dados
  - Empresas tornam-se independentes da execução em lotes ao final do dia para povoar DW e gerar análises

Modelos de Dados de BD NoSQL

- Exemplo de Relações x Agregados
  - Considere uma aplicação de comércio eletrônico



Modelos de Dados de BD NoSQL

- Exemplo de Relações x Agregados
  - Alguns dados da aplicação de comércio eletrônico

Cliente

ID	Nome
1	Ana

Pagto

ID	NPedido	Cartao	Vencimento	EndCobran
33	99	4595-1	10/10/2021	55

Produto

ID	Nome
27	GoPro H4

Pedido

ID	Cliente	EndEnvio
99	1	77

Endereco

ID	Cidade
77	Rec

ItemPed

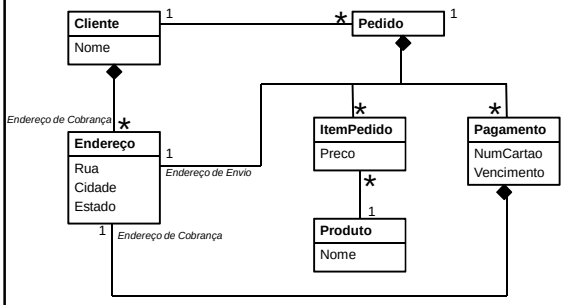
ID	NPedido	ProdID	Preco
100	99	27	500,00

EndCobranca

ID	Cliente	Endereco
55	1	77

Modelos de Dados de BD NoSQL

- Exemplo de Relações x Agregados
  - Considere uma aplicação de comércio eletrônico



Modelos de Dados de BD NoSQL

- Exemplo de Relações x Agregados
  - Alguns dados da aplicação de comércio eletrônico

```
// em  Clientes
{
"ID" :  1,
"Nome" :  "Ana",
"EndCobranca" :  [ { "Cidade": "Rec" } ]
}
```


Modelos de Dados de BD NoSQL

- Exemplo de Relações x Agregados
  - Alguns dados da aplicação de comércio eletrônico

```
// em  Pedidos  {
"ID" :  99,
"ClienteID" :  1,
"ItemPed" :  [
{  "ProdutoID":  27,
   "Preco":  500,00 ,
   "Nome":  "GoPro H4"  }
],
```


Modelos de Dados de BD NoSQL

- Exemplo de Relações x Agregados
  - Alguns dados da aplicação de comércio eletrônico

```
"EndEnvio":  [  { "Cidade" : "Rec" } ]
"Pagto":  [
 {
   "Cartao": "4595-1" ,
   "Vencimento":  "10/10/2021" ,
   "EndCobran": { "Cidade" : "Rec" }   }
] ,
}
```

Modelos de Dados de BD NoSQL

- Consequências da orientação a agregados:
  - Sistema de BD pode usar o seu conhecimento sobre a estrutura agregada para armazenar e distribuir os dados
  - Auxilia na execução de um cluster
    - Minimiza o número de nós a serem lidos quando dados estiverem sendo processados
  - BD não possui transações ACID que executem por múltiplos agregados
    - Garantem a manipulação atômica em um único agregado por vez
    - Se vários agregados precisarem ser manipulados de forma atômica, o código do aplicativo precisará gerenciar isso

Modelos de Dados de BD NoSQL

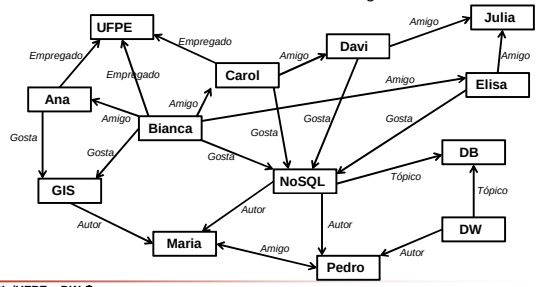
- Quando volumes de dados aumentam, torna-se mais difícil e custoso fazer uma expansão
- Uma opção interessante é a execução do BD em um cluster de servidores
  - Orientação a agregados é útil porque o agregado é uma unidade natural para ser usada de forma distribuída
  - A execução do BD em um cluster introduz complexidade
    - Não é algo para ser feito a menos que os benefícios sejam muitos
- Existem sistemas de BD NoSQL que não executam em clusters ou não são orientados a agregados
  - BD orientados a grafos: Neo4J

Modelos de Dados de BD NoSQL

- BD de grafos são não agregados:
  - Isto não é uma desvantagem
  - Muitas vezes é difícil definir corretamente o limite de agregados
    - Isto ocorre principalmente quando os mesmos dados são usados em muitos contextos diferentes
  - Uma estrutura de agregado pode ser útil em algumas interações de dados, mas um obstáculo em outras
  - Um modelo de dados não agregado permite o acesso a qualquer tipo de organização de dados
    - É a melhor escolha quando não há uma forma primária de manipulação de dados
  - Gerenciam transações ACID de forma semelhante aos SGBDR

Modelos de Dados de BD NoSQL

- Modelo de BD de grafos representam registros pequenos com conexões complexas:
  - *Encontre os tópicos de BD que meus amigos gostam ou cujos autores são conhecidos de meus amigos*



Modelos de Dados de BD NoSQL

- Modelo de dados de um BD de grafos é simples: nós conectados por arestas. Mas há variações:
  - FlockDB é composto por nós e arestas (sem atributos)
  - Neo4J armazena objetos Java como propriedades de nós e arestas
- SGBR podem implementar relacionamentos por meio de chaves estrangeiras, mas:
  - Junções necessárias para navegar por eles podem ser muito custosas. Por isso:
- BD de grafos transferem a maior parte do trabalho do momento da consulta para o momento de inserção
  - Útil em situações onde o desempenho da consulta é mais importante que a velocidade de inserção

Modelos de Dados de BD NoSQL

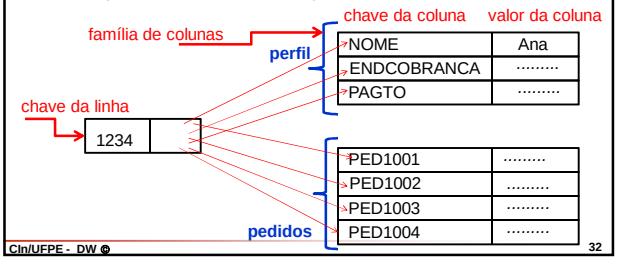
- Modelos Chave-valor x Orientados a Documentos:
  - Ambos são orientados a agregados
  - Cada agregado tem um ID ou chave de acesso
  - Diferem no fato de que:
    - No BD chave-valor, o agregado é um grande amontoado de bits sem significado
      - A vantagem é que é possível armazenar qualquer coisa
    - Um BD de documentos impõe limites sobre o que pode ser inserido, definindo estruturas e tipos permitidos
      - A vantagem é a flexibilidade de acesso
  - No entanto, alguns sistemas chave-valor permitem:
    - Uso de metadados para indexação de agregados: Riak
    - Agregado seja dividido em listas ou conjuntos: Redis

Modelos de Dados de BD NoSQL

- Modelos de Famílias de Colunas:
  - Voltados para cenários onde gravações são raras e é preciso ler colunas de muitas linhas de uma só vez
  - Armazena grupos de colunas para todas as linhas como a unidade básica de armazenamento
    - São chamados de BD colunar ou de família de colunas
    - Ex: BigTable e seus descendentes (Cassandra, Hbase)
  - É uma estrutura agregada de dois níveis
    - Cada linha é um agregado (cliente 1234) de famílias de colunas - representando partes úteis de dados (perfil, histórico de pedidos) dentro do agregado
    - Cada família de colunas define um tipo de registro (perfil de clientes)

Modelos de Dados de BD NoSQL

- Modelos de Famílias de Colunas:
  - Cada coluna pertence a uma única família de colunas
  - Cada coluna atua como uma unidade de acesso
    - Assume-se que dados de uma família de colunas são geralmente acessados juntos



Modelos de Dados de BD NoSQL

- Mais detalhes:
  - BD de agregados tornam os relacionamentos entre agregados mais difíceis de modelar do que relacionamentos intra-agregados
  - BD de grafos organizam os dados em grafos com nós e arestas
    - Funcionam melhor com dados que tenham estruturas de relacionamento complexas
  - BD sem esquema permitem que campos sejam adicionados livremente aos registros
  - BD de agregados frequentemente criam visões materializadas para disponibilizar dados organizados de um modo diferente de seus agregados primários
    - Isso é frequentemente feito por computações map-reduce

Modelos de Dados de BD NoSQL

- BD sem esquema:armazena dados informalmente
  - BD do tipo chave-valor permite manter quaisquer dados sob uma chave
  - BD de documentos faz o mesmo já que não restringe a estrutura de documentos armazenados
  - BD de família de colunas armazena quaisquer dados na coluna escolhida
  - BD de grafos adiciona livremente novos nós, novas arestas e novas propriedades aos nós e às arestas
- Vantagens:
  - Facilita o tratamento de dados não uniformes
  - Flexibiliza a alteração de esquemas
- Mas novos problemas surgem.....

Modelos de Dados de BD NoSQL

- O que é um esquema implícito?
  - Todo programa que acessa dados quase sempre baseia-se em alguma forma de esquema implícito, pois assume:
    - Determinados nomes de campos existem
    - Determinados campos possuem certos tipos de dados
  - É um conjunto de suposições sobre a estrutura dos dados feitas pelos códigos de programa que a manipula
  - BD sem esquema transfere a definição do esquema para o código do aplicativo que o acessa
  - Torna-se um problema quando múltiplos aplicativos, desenvolvidos por pessoas diferentes, acessam o BD
- Quais são as soluções? Existem várias abordagens

Modelos de Dados de BD NoSQL

- Possíveis soluções para o esquema implícito:
- Acesso ao BD por meio de um único aplicativo
  - Encapsular toda a interação do BD em um único aplicativo e integrá-lo com outros aplicativos usando webservices
- Controlar o acesso aos agregados por aplicativos
  - Delinear áreas distintas de um agregado para acesso por diferentes aplicativos
    - Em um BD de documentos: seções diferentes
    - Em um BD colunar: diferentes famílias de colunas

Modelos de Dados de BD NoSQL

- Visões materializadas em BD NoSQL:
  - São consultas pré-computadas por computações do tipo *map-reduce* e mantidas em cache
- Abordagem antecipada:
  - A visão materializada e os dados básicos relacionados a ela são atualizados ao mesmo tempo
  - Por exemplo, adicionar um pedido também atualiza o histórico de compras do agregado para cada produto
  - Boa abordagem quando há mais leituras da visão materializada do que gravações e visões precisarem estar o mais atualizadas possível

Modelos de Dados de BD NoSQL

- Abordagem adiada:
  - Evita a sobrecarga acrescida a cada atualização do BD para atualização da visão
  - Visões são atualizadas em lotes e intervalos regulares
  - É preciso entender os requisitos de dados do usuário para avaliar o quão desatualizadas as visões podem ficar

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

Modelos de Distribuição

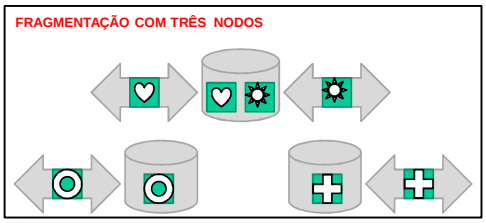
- Há dois estilos de distribuição de dados:
  - Fragmentação: distribui dados diferentes em múltiplos servidores, de modo que cada servidor atua como a única fonte de um subconjunto de dados
  - Replicação: copia os dados para múltiplos servidores de modo que cada parte dos dados possa ser encontrada em vários lugares ao mesmo tempo
- São ortogonais: Um sistema de BD pode usar uma delas ou ambas técnicas de distribuição de dados
- Replicação tem duas formas:
  - Mestre-escravo
  - Ponto a ponto

Modelos de Distribuição

- Um único servidor:
  - Opção mais simples e recomendada com mais frequência é: não usar distribuição alguma
  - BD é executado em uma única máquina que lida com todas leituras e gravações
  - Vantagens:
    - Elimina a complexidade das demais opções
    - É mais facilmente gerenciada por DBAs
    - É melhor compreendida por desenvolvedores de aplicativos
  - BD de Grafos é a melhor opção aqui pois trabalham melhor em uma configuração com servidor único
  - Se o objetivo maior da aplicação for processar agregados:
    - Armazenamento de documentos ou chave-valor em um único servidor pode valer a pena pois facilita o desenvolvimento

Modelos de Distribuição

- Fragmentação (*Data Sharding*):
  - Acessos concorrentes ao BD feitos por vários usuários pode degradar o desempenho do sistema
    - Colocar partes diferentes dos dados em servidores diferentes promove a escalabilidade horizontal
    - Cada fragmento lê e grava seu próprio dado



Modelos de Distribuição

- Fragmentação (*Data Sharding*):
  - Agrupar os dados de modo que um usuário obtenha a maior parte de seus dados de um único servidor
    - Agregados → unidades de distribuição
  - Manter a carga distribuída igualmente entre os nós
    - Agregados → tenham quantidades iguais de carga
  - Juntar agregados se eles forem lidos em sequência
  - Sistemas de BD NoSQL oferecem a auto-fragmentação:
    - Sistema é responsável por alocar os dados nos fragmentos e por garantir que o acesso aos dados seja feito no fragmento correto

Modelos de Distribuição

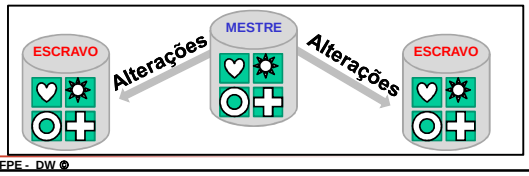
- Replicação com cache melhora o desempenho de leituras, mas pouco faz para otimizar muitas gravações
- Fragmentação (*Data Sharding*):
  - Pode melhorar o desempenho de leituras e gravações
    - Provê uma forma de escalar horizontalmente as gravações
  - Faz pouco para melhorar a tolerância a falhas quando usada isoladamente
    - Embora os dados estejam em nós diferentes, a falha em um nó torna indisponíveis os dados do fragmento
    - O único benefício é que apenas os usuários dos dados do nó que sofreu a falha serão prejudicados
    - Mas não é bom ter um BD com partes de seus dados faltando

Modelos de Distribuição

- Fragmentação (*Data Sharding*):
  - Clusters geralmente usam máquinas menos confiáveis, aumentando as chances de falha por nó
    - Quando se tem um único servidor é mais fácil bancar o esforço e o custo para mantê-lo funcionando
  - É melhor iniciar com um servidor único e apenas usar a fragmentação quando as projeções de carga indicarem que os recursos estão terminando
  - Por outro lado, é importante usar a fragmentação bem antes de precisar dela → quando os recursos forem suficientes para executá-la

Modelos de Distribuição

- Replicação Mestre-Escravo:
  - Torna um nó a cópia oficial, a qual lida com gravações, enquanto os escravos sincronizam-se com o mestre e podem lidar com as leituras
  - Reduz a chance de conflitos de atualização
  - Todas as atualizações são feitas apenas no mestre
  - Leituras podem ser feitas no mestre ou nos escravos
  - Alterações são propagadas para os escravos



Modelos de Distribuição

- Replicação Mestre-Escravo:
  - É mais últil para a escalabilidade quando há um número maior de operações de leitura
  - É possível escalar horizontalmente para lidar com mais leituras por meio da:
    - Adição de mais nós escravos
    - Envio de todas as solicitações de leitura apenas para os escravos
  - Mas existe a limitação da capacidade do mestre de processar as atualizações e transmiti-las adiante
  - Não é adequado quando existem muitas gravações
  - Possui tolerância a falhas de operações de leitura:
    - Se o mestre falhar, os escravos lidam com as leituras

Modelos de Distribuição

- Replicação Mestre-Escravo:
  - A falha no mestre elimina a capacidade de lidar com gravações até que ele seja restaurado ou um novo mestre seja escolhido
  - Ter escravos como replicações do mestre acelera a recuperação após falhas, já que um escravo pode ser designado como um novo mestre muito rapidamente
  - A capacidade de escolher um escravo para substituir um mestre que falhou indica que a replicação mestre-escravo é útil mesmo que não haja necessidade da escalabilidade
    - Toda solicitação de leitura e gravação pode ir para o mestre, enquanto o escravo atua como uma cópia de segurança ativa
    - Assemelha-se a um sistema com armazenamento de servidor único com backup ativo → útil para lidar com falhas no servidor

Modelos de Distribuição

- Replicação Mestre-Escravo:
  - Mestres podem ser designados manual ou automaticamente
    - Manual: usuário escolhe um nó como mestre na configuração do cluster
    - Automática:usuário cria um cluster de nós os quais elegem um deles como mestre
  - Vantagens da configuração automática:
    - Simplifica a configuração do cluster por si só
    - Cluster pode escolher automaticamente um novo mestre na ocorrência da falha → reduzindo o tempo fora do ar

Modelos de Distribuição

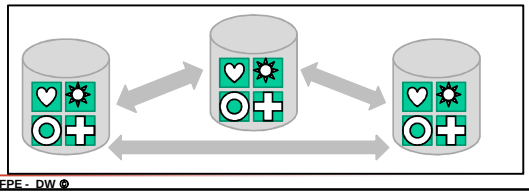
- Replicação Mestre-Escravo:
  - A replicação traz alguns benefícios interessantes, mas:
    - Tem um lado negativo inevitável → inconsistência
    - O risco é que usuários diferentes, lendo diferentes escravos, vejam valores diferentes, se todas as alterações não tiverem sido propagadas aos escravos
    - Mesmo que a replicação mestre-escravo use uma cópia de segurança ativa, o problema persiste:
      - Se o mestre falhar, quaisquer atualizações, que não tenham sido enviadas para a cópia de segurança, estarão perdidas

Modelos de Distribuição

- Replicação Mestre-Escravo:
  - Em resumo:
    - Ajuda com a escalabilidade de leitura, mas não com a escalabilidade de gravação
    - Provê tolerância a falhas de um escravo, mas não de um mestre
    - O mestre é um gargalo e um ponto único de falhas
  - Por isso, a replicação ponto a ponto, vista a seguir, resolve estes problemas pois não tem um mestre:
    - Todos os nós têm peso igual
    - Todos os nós podem receber gravações
    - A perda de um deles não impede o acesso aos dados

Modelos de Distribuição

- Replicação Ponto a Ponto (p2p):
  - Permite gravações em qualquer nó
  - Nós são coordenados para sincronizar suas cópias de dados → nós comunicam suas garvações
  - Evita manter todas as gravações em um único ponto de falha
  - Todos os nós lêm e gravam todos os dados



Modelos de Distribuição

- Replicação Ponto a Ponto (p2p):
  - É possível contornar falhas nos nós sem perder o acesso aos dados
  - É possível adicionar novos nós para melhorar o desempenho
  - Mas existe a possibilidade de geração de inconsistência de dados:
    - Ao permitir a gravação em dois lugares diferentes, existe o risco de que dois usuários tentem atualizar o mesmo registro ao mesmo tempo → gerando um conflito de gravação
  - É possível garantir que sempre que dados são gravados, as réplicas sejam atualizadas para evitar conflitos
    - Há um custo de tráfego de rede para coordenar as gravações
    - Não há necessidade de todas as réplicas concordarem com a gravação → apenas uma maioria pode ser considerada

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

Consistência

- Tipos de Conflitos:
  - Gravação: ocorrem quando dois usuários tentam gravar os mesmos dados ao mesmo tempo
    - Gera o problema da atualização perdida
  - Leitura-gravação (ou leitura incopnsistente): ocorrem quando um usuário lê dados inconsistentes durante a gravação feita por outro usuário
- Abordagens de controle de concorrência:
  - Pessimista:Evita que os conflitos ocorram
  - Otimista:Permite que os conflitos ocorram, mas os detecta e depois os corrige

Consistência

- Abordagem Pessimista :
  - Bloqueios: para alterar um valor é necessário obter um bloqueio de gravação e o sistema assegura que apenas um usuário por vez pode obtê-lo
  - Pode produzir conflitos de bloqueio (deadlocks) difíceis de evitar e identificar
- Abordagem Otimista:
  - Atualização condicional: qualquer usuário que execute uma atualização testa o valor antes de atualizá-lo para ver se ele foi alterado desde sua última leitura
  - Atualização dupla: gravar ambas as atualizações, registrar as que estão em conflito e conciliá-las
    - Mostrando ambos valores ao usuário para que ele decida
    - Concilia automaticamente, com base em informações de domínio

Consistência

- Tanto a Pessimista quanto a Otimista baseiam-se na serialização consistente das atualizações
  - Se existir mais de um servidor (como na p2p): dois nós podem atualizar dados em uma ordem diferente → resultando em um valor diferente para cada nó
- Replicação contribui para ocorrência de conflitos de gravação
  - Se diferentes nós tiverem cópias diferentes dos mesmos dados e forem atualizados independentemente → haverá conflitos de gravação se não forem evitados
  - Usar um único nó como responsável por todas as gravações facilita a manutenção da consistência de atualização
    - Dos modelos de distribuição vistos, todos fazem isso – menos a replicação p2p

Consistência

- Janela de Inconsistência:
  - Nem todos os dados podem ser colocados em um único agregado
  - Qualquer alteração que afete múltiplos agregados deixa aberto um tempo no qual usuários podem realizar leituras inconsistentes → chamado de janela de inconsistência
- Sistemas distribuídos:
  - Geram conflitos do tipo leitura-gravação porque alguns nodos recebem atualizações e outros não
  - Consistência eventual: em algum momento, o sistema se tornará consistente, assim que as gravações tiverem sido propagadas para todos os nós
  - Para obter boa consistência, muitos nós devem ser envolvidos em operações de dados → aumenta latência

Consistência

- Teorema CAP:
  - No mundo NoSQL, o teorema CAP corresponde ao motivo pelo qual pode-se precisar relaxar a consistência
  - Baseia-se em três propriedades: Consistência, Disponibilidade e Tolerância a partições
  - Definição: Dadas as 3 propriedades, somente é possível obter duas delas.
    - Um sistema com um único servidor → é um exemplo de CA
  - Como clusters têm de ser tolerantes a partições de rede, então o teorema CAP na prática diz que:
    - Como um sistema distribuído pode sofrer partições, é preciso balancear a consistência com a disponibilidade
  - O sistema resultante não é perfeitamente consistente nem perfeitamente disponível

Consistência

- Relaxando a durabilidade:
  - É possível trocar um pouco da durabilidade por um desempenho melhor
  - A durabilidade pode ser balanceada com a latência
  - Se um BD puder ser executado em sua maior parte na memória e mover periodicamente as alterações para o disco, um desempenho melhor pode ser obtido
  - O custo é que se o servidor falhar, quaisquer atualizações feitas desde a última transferência dos dados para o disco serão perdidas.

Consistência

- Quóruns:
  - Quanto mais nós forem envolvidos em uma solicitação, maior a chance de evitar uma inconsistência.
  - Quantos nós precisam ser envolvidos para produzir uma alta consistência?
    - Se as gravações forem conflitantes, apenas uma delas pode obter a maioria → quórum de gravação (QG)
    - QG é expresso pela desigualdade: W > N/2
    - Significa que o número de nós participantes da gravação (W) deve ser maior que a metade do número de nós replicados (N)
  - Quantos nós devem ser acessados para garantir que se tem a alteração mais atualizada?
    - Quórum de leitura (QL): R + W > N

Consistência

- Quóruns:
  - Lista de parâmetros de configuração de consistência:
    - N é o número de nós usados para armazenar as réplicas do BD chave-valor
    - R é o número de nós cujos dados devem ser recuperados antes da leitura ser considerada bem sucedida
    - W é o número de nós nos quais a gravação deve ser executada antes dela ser considerada bem sucedida
  - Estas configurações nos permite gerenciar a ocorrência de falhas nos nós em operações de leitura ou gravação
  - É possível alterar os valores dos parâmetros para obter mais disponibilidade de leitura ou gravação

Consistência

- Quóruns:
  - Considere um cluster Riak com cinco nós:
    - Se N = 3 significa que todos os dados serão replicados em pelo menos 3 nós
    - Se R = 2 indica que dois nós quaisquer devem responder a uma solicitação GET para que esta seja considerada bem sucedida
    - Se W = 2 garante que a solicitação PUT seja gravada em 2 nós antes da gravação ser considerada bem sucedida

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

Map-reduce

- Motivação
  - A ascensão de BD orientados a agregados deve-se em grande parte ao crescimento dos clusters
  - Executar em clusters implica em equilibrar o armazenamento de dados de maneira diferente da execução feita em uma única máquina
  - Clusters mudam regras de armazenamento de dados e regras de computação
    - Distribuir os dados entre os vários nodos do cluster
    - Reduzir a quantidade de dados que deve ser transferida pela rede, processando tanto quanto for possível no nodo onde estiverem os dados necessários

Map-reduce

- Objetivos
  - Organizar o processamento para usar as várias máquinas de um cluster e ao mesmo tempo,
  - Manter o processamento dos dados na mesma máquina que contém os dados
- Definição
  - É um padrão que permite que computações sejam paralelizadas em um cluster
    - Uma implementação de código aberto faz parte do projeto Hadoop
    - Apache Pig é uma linguagem criada especificamente para facilitar a escrita de programas map-reduce
    - Diversos sistemas de BD NoSQL têm suas próprias implementações

Map-reduce

- Possui duas tarefas principais:
  - Mapeamento
    - Lê dados de um agregado e os agrupa em pares de *chave-valor* relevantes
    - Somente lê um único registro de cada vez e pode, assim, ser paralelizado e executado no nodo que armazena o registro
  - Redução
    - Recebe muitos valores de uma única chave de saída, a partir do mapeamento, e os resume em uma única saída
    - Cada redutor trabalha sobre o resultado de uma única chave, de modo que pode ser paralelizado por chave

Map-reduce

- Exemplo de Aplicação – Cenário:
  - Suponha que *pedidos de clientes* formem um agregado, onde pedidos inteiros são recuperados em um só acesso
  - Cada pedido possui itens do pedido
  - Cada item de pedido possui um ID do produto, a quantidade e o preço
  - Como existem muitos pedidos, eles são fragmentados em várias máquinas
  - Porém, deseja-se ver os *dados do produto* e sua receita total nos últimos 7 dias!!
    - Esta análise não se adapta à estrutura do agregado existente → isto é uma desvantagem do uso de agregados
  - Para obter a receita dos produtos, será preciso recuperar vários registros de cada máquina do cluster

Map-reduce

- Etapa de Mapeamento (Função MAP):
  - Função cuja entrada é um único agregado e cuja saída são alguns pares de chave-valor
  - No exemplo, a entrada seria um pedido e a saída seria pares de chave-valor correspondendo aos itens
  - Cada par teria o ID do produto como chave, e a quantidade e o preço como valores
  - Cada aplicação da função é independente de todas as outras
    - Isto permite que elas sejam paralelizáveis com segurança
  - Cria tarefas de mapeamento em cada nodo e aloca cada pedido a uma tarefa de mapeamento
    - Isto produz bastante paralelelismo e localidade no acesso aos dados

Map-reduce

- Etapa de Mapeamento:
  - Lê registros do BD e gera pares de chave-valor

ID: 1066			
Cliente: Marta			
Itens Pedido:			
Refrigerante	8	8,50	68,00
Suco	4	37,50	150,00
Água Mineral	10	1,99	19,90
Endereço de Entrega			
Detalhes do Pagamento			

MAP

Refrigerante:
Preço: 68,00
Quantidade: 8

Suco:
Preço: 150,00
Quantidade: 4

Água Mineral:
Preço: 19,90
Quantidade: 10

Map-reduce

- Etapa de Redução (Função REDUCE):
  - Recebe múltiplos resultados do mapeamento com a mesma chave e combina seus valores
  - Utiliza todos os valores enviados para uma única chave
  - Recebe diversos pares de chave-valor com a mesma chave e os agrega

SUCO

Preço: 187,50
Quantidade: 5

Preço: 150,00
Quantidade: 4

Preço: 375,00
Quantidade: 10

REDUCE

Preço: 712,50
Quantidade: 19

Map-reduce

- Aspectos importantes
  - Funções redutoras que tenham o mesmo formato de entrada e saída podem ser combinadas em pipelines
    - Isto melhora o paralelelismo e reduz a quantidade de dados a serem transferidos
  - Operações de map-reduce podem ser compostas em pipelines, nas quais a saída de uma redução é a entrada do mapeamento de outra operação
  - Se o resultado de uma computação map-reduce for amplamente utilizado, ele pode ser armazenado como uma visão materializada
  - Visões materializadas podem ser atualizadas por meio de operações map-reduce que apenas computem alterações na visão, em vez de computar tudo desde o início

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

BD Chave-valor

- Características:
  - É o tipo de BD NoSQL mais simples de usar a partir da perspectiva de uma API. O usuário pode:
    - Obter o valor de uma determinada chave
    - Inserir um valor para uma dada chave
    - Remover uma chave do BD
  - Valor é um *blob ou texto ou JSON ou XML* que o BD guarda sem se preocupar ou saber o que há dentro dele
    - É responsabilidade da aplicação entender o que está armazenado
  - Já que o acesso é sempre feito pela chave primária:
    - Um bom desempenho pode ser alcançado
    - Podem ser escaláveis facilmente

BD Chave-valor

- Baseia-se em uma *hash table* contendo uma chave única e um apontador para um item particular de dado
- Exemplos:
  - Riak
  - Redis
  - HamsterDB
  - Amazon DynamoDB (que não é open-source)
  - Project Voldemort (versão open-source do DynamoDB)
- Riak:
  - Permite associar chaves a *namespaces ou buckets*
  - *Buckets de domínio* são usados para guardar dados específicos

BD Chave-valor

- Consistência no Riak
  - Modelo de *consistência eventual* é implementado
  - Quando o valor é replicado em outros nós, conflitos de atualização podem ser resolvidos de duas maneiras:
    - Gravação mais recente prevalece sobre as mais antigas
    - Todos os valores são retornados para o usuário resolver o conflito
  - Estas opções podem ser configuradas na criação do *bucket* → valores padrões podem ser fornecidos
  - Por exemplo: uma gravação pode ser efetivada apenas quando os dados forem consistentes para todos nós que mantiverem estes dados

BD Chave-valor

- Consistência no Riak
  - Exemplo de configuração de parâmetros de consistência na criação do *bucket:*

```
Bucket bucket = connection
.createBucket (bucketName)
.withRetrier (attempts(3))
.allowSiblings (siblingsAllowed)
.nVal (numberOfReplicasOfTheData)
.w (numberOfNodesToRespondToWrite)
.r (numberOfNodesToRespondToRead)
.execute();
```


BD Chave-valor

- Consultas
  - Todos BD chave-valor podem ser consultados pela chave
  - Se for preciso consultar algum atributo da coluna de valores, o sistema NoSQL não disponibiliza o recurso
    - Aplicação do usuário se encarregará de fazer esta busca
  - Realizar consultas por chave também causa problemas se a chave não for conhecida
    - A maioria dos sistemas NoSQL não fornece uma lista de todas as chaves primárias
  - O projeto da chave primária é importante:
    - Pode ser gerada por meio de algum algoritmo?
    - Pode ser fornecida pelo usuário?
    - Pode ser derivada do relógio do sistema?

BD Chave-valor

- Casos de uso apropriados
  - Armazenando informações de sessão
    - Toda sessão web é única e recebe um valor único de *sessionId*
    - Tudo sobre a sessão pode ser inserido por um único PUT ou recuperado usando GET
  - Perfis de usuário, preferências
    - Quase todos os usuários têm um único *userId* (ID do usuário), um único *username* (nome de usuário) ou algum outro atributo único, além de preferências, cor, idioma, etc
    - Tudo isso pode ser colocado em um objeto e obtido pelo GET
  - Carrinhos de compra
    - Websites de comércio eletrônico possuem carrinhos de compras associados ao usuário
    - Todas as informações de compras podem ser colocadas no valor onde a chave é *userId*

BD Chave-valor

- Quando não usar:
  - Relacionamentos entre dados
    - Modelar relacionamentos entre diferentes conjuntos de dados ou correlacionar dados entre diferentes conjuntos de chaves
  - Transações com múltiplas operações
    - Gravar múltiplas chaves e ocorrer uma falha na gravação de alguma delas, e for preciso reverter ou desfazer
  - Consulta por dados
    - Pesquisar chaves com base no valor de pares chave-valor
  - Operações de conjuntos
    - Executar operações sobre múltiplas chaves por vez

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

BD de Família de Colunas

- Armazena chaves mapeadas para valores onde:
  - Valores são agrupados em múltiplas famílias de colunas
- Famílias de Colunas:
  - São grupos de dados relacionados que frequentemente são acessados juntos
- Exemplos:
  - Cassandra, Hbase, Hypertable

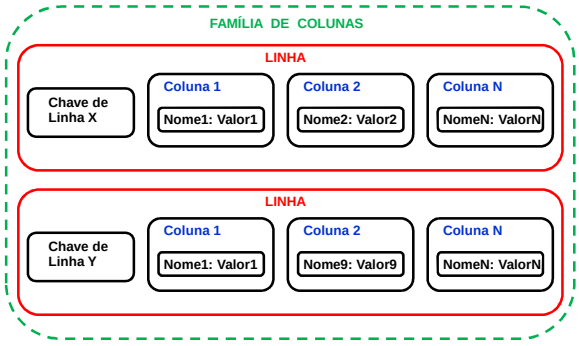
SGBDR	Cassandra
Instância do BD	Cluster
BD	Keyspace
Tabela	Família de Colunas
Linha	Linha
Coluna	Coluna

BD de Família de Colunas

- Cassandra:
  - É rápido e de fácil crescimento escalar
  - Não possui um nó mestre de modo que tanto a leitura quanto a gravação podem ser feitas em qualquer nó
  - Coluna: Unidade básica de armazenamento que consiste em um par chave-valor
    - Cada coluna tem um valor de *timestamp* usado para:
      - Expirar dados
      - Resolver conflitos de gravação
      - Lidar com dados desatualizados
  - Linha: coleção de colunas associadas a uma chave
  - Família de colunas: coleção de linhas semelhantes

BD de Família de Colunas

- Modelo de Dados do Cassandra



BD de Família de Colunas

- O que é uma coluna?

```
{ Nome :  “PrimeiroNome”,        CHAVE
  Valor : “Mateus”,              VALOR
  Timestamp : 1234567890
}
```


BD de Família de Colunas

- O que é uma linha?

```
// família de colunas
{
// linha
“pedro-souza” :  { Nome : “Pedro”,
                   Sobrenome : “Souza”,
                   UltimaVisita : “12/12/2012”  }
// linha
“marta-alves” :  { Nome : “Marta”,
                   Sobrenome : “Alves”,
                   Local : “Recife”  }   }
```


BD de Família de Colunas

- O que é uma supercoluna?
  - Consiste em uma coluna cujo valor é um conjunto de colunas

```
{
      Nome :  “livro:978-0767905923”,
      Valor :  {
      Autor:  “Joel da Silva”,
      Titulo : “ A Linguagem GeoMDQL”,
      Isbn : “978-0767905923”  }
}
```


BD de Família de Colunas

- O que é uma família de supercolunas?

```
// família de supercolunas
{  // linha
Nome : “conta:pedro-souza”,
Valor :  {
      Endereço:  {
            Nome : “endereço:default”,
            Valor:  { nomeCompleto : “ Pedro Souza”,
                      rua : “Av. Boa Viagem 4200” ,
                      complemento: “ Apto 301” }
                    } ,
```


BD de Família de Colunas

```
Conta :  {
      Nome : “conta:default”,
      Valor:  {
            numeroCartao : “ 8888-4444-2222-1111”,
            vencimento: “12/2018”
            }
        }
}
```


BD de Família de Colunas

```
// linha
Nome : “conta:marta-alves”,
Valor :  {
      Endereço:  {
            Nome : “endereço:default”,
            Valor:  {
                  nomeCompleto : “ Marta Alves”,
                  rua : “Rua Serinhaem 44” ,
                  complemento: “ Apto 1801”
                  }
                } ,
```

BD de Família de Colunas

```
Conta : {
        Nome : "conta:default",
        Valor:  {
                numeroCartao : " 9999-7777-5555-1111",
                vencimento: "12/2020"
                }
          }
    }
}
```


BD de Família de Colunas

- Consultas e Manipulação de Dados usando CQL:
  - Para criar uma família de colunas:

```
CREATE COLUMNFAMILY Cliente (
    KEY varchar PRIMARY KEY,
    nome varchar,
    cidade varchar,
    web varchar ) ;
```

  - Para inserir os mesmos dados:

```
INSERT INTO Cliente (KEY, nome, cidade, web)
  VALUES ( 'pedro-souza', 'Pedro Souza', 'Recife',
                                  'www.pedrosouza.com');
```


BD de Família de Colunas

- Consultas e Manipulação de Dados usando CQL:
  - Para ler todas as colunas:

```
SELECT * FROM Cliente ;
```

  - Para selecionar apenas as colunas desejadas:

```
SELECT nome, web FROM Cliente;
SELECT nome, web FROM Cliente
        WHERE cidade = 'Recife' ;
```

- Escalabilidade:
  - Para fazer um cluster crescer em escala é preciso apenas adicionar mais nós
    - Como nenhum nó é mestre, a adição de nós aumenta a capacidade de execução de mais leituras/gravações

Tópicos

- Introdução
- Modelos de Dados de BD NoSQL
- Modelos de Distribuição
- Consistência
- Map-reduce
- BD Chave-valor
- BD de Família de Colunas
- BD de Documentos

BD de Documentos

- Documentos: armazenados na parte do valor
  - São o conceito principal pois armazenam e recuperam documentos (XML, JSON, BSON, etc)
  - São árvores hierárquicas e autodescritivas contendo coleções de valores e valores escalares
  - São semelhantes entre si mas não têm de ter exatamente a mesma estrutura

ORACLE	MongoBD
Instância do BD	Instância MongoDB
Esquema	Banco de Dados
Tabela	Coleção
Linha	Documento
Rowid	_id
Junção	DBRef

BD de Documentos

- O que é um BD de documentos?

```
{ "Nome" :  "Marta",
"Gosta" :  [  "Ciclismo", "Fotografia"  ] ,
"UltimaCidade" :  "Recife"
}
```

BD de Documentos

- O que é um BD de documentos?

```
{ “Nome” : “Pedro”,
“CidadesVisitadas” : [ “Recife”, “Olinda” , “Natal”, “Fortaleza” ] ,
“Enderecos” : [
 { “Estado” : “PB”,
   “Cidade” : “Patos”,
   “Tipo” : “R” } ,
 { “Estado” : “SP”,
   “Cidade” : “Campinas”,
   “Tipo” : “R” } ] ,
“UltimaCidade” : “Recife” }
```


BD de Documentos

- Documentos podem ter:
  - Atributos diferentes entre si
    - Campos que aparecem em um documento mas não aparecem em outro
    - Campos com nomes diferentes
  - Atributos compostos (por exemplo, listas e arrays)
  - Diferentes representações de dados e pertencerem à mesma coleção de documentos
  - Objetos complexos: vários documentos dentro de um outro documento
- Documentos não possuem atributos vazios
  - Se um dado atributo não for encontrado, supõe-se que ele não é relevante para o documento

BD de Documentos

- Documentos permitem que novos atributos sejam criados sem a necessidade de:
  - Definição prévia
  - Alteração nos documentos já existentes
- Exemplos de sistemas que usam armazenamento de documentos:
  - MongoDB
  - CouchDB
  - Terrastore
  - OrientDB
  - RavenDB
  - Lotus Notes

BD de Documentos

- Características:
  - Cada instância do MongoDB possui zero ou mais BD
  - Cada BD pode ter zero ou mais coleções de documentos
  - Coleções são compostas de zero ou mais documentos e podem ser indexadas
  - Um documento possui um ou mais campos
  - Ao armazenar um documento, é preciso escolher em qual BD e em qual coleção ele ficará
  - Quando dados são solicitados, MongoDB retorna um ponteiro para um *result set* (chamado de cursor)
- Por que usar uma terminologia diferente?
  - SGBDR define colunas ao nível de tabelas
  - MongoDB define seus campos ao nível de documentos

BD de Documentos

- Consistência
  - É configurada usando as réplicas e esperando que as gravações sejam replicadas:
    - Em todos os escravos ou
    - Em um determinado número de escravos
  - Cada gravação define o número de escravos cuja atualização deve ser propagada antes dela ser considerada bem-sucedida
  - É possível informar ao sistema o quão alta é a consistência desejada. Por exemplo:
    - Único servidor com W = “majority”: gravação será concluída imediatamente já que há apenas um nó
    - Três nós no conjunto de réplicas com W = “majority”: a gravação terá de se completar em pelo menos dois nós antes de ser informada como bem-sucedida

BD de Documentos

- Consistência
  - Parâmetros de gravação/leitura podem ser configurados:
    - Na conexão ou
    - Na criação do BD ou
    - Na criação da coleção de documentos ou
    - Individualmente, para cada operação
  - Por padrão, uma gravação é informada como bem-sucedida assim que o BD é atualizado
    - Por padrão, todas as gravações são reportadas como bem-sucedidas
    - Isso pode ser alterado para que se espere até as gravações serem sincronizadas em disco ou propagadas para 2 ou mais escravos

```
DBCollection shopping = database.getCollection (“shopping”);
shopping.setWriteConcern (REPLICAS_SAFE);
```

BD de Documentos

- Consistência
  - É também possível configurar a consistência desejada por operação de gravação:

```
DBCollection shopping = database.getCollection (“shopping”);
WriteResult resultado =
        shopping.insert (pedido, REPLICAS_SAFE);
```

  - É também possível indicar que leituras poderão ser feitas nos nós escravos:
    - Na conexão com o BD:

```
Mongo mongo = new Mongo (“localhost:27017”);
mongo.slaveOK();
```


BD de Documentos

- Consistência
  - Transações no nível de um único documento são conhecidas como atômicas
  - Transações não têm mais de uma operação
    - Porém, o RavenDB processa transações com múltiplas operações
  - Níveis diferentes de consistência/segurança nas gravações e leituras podem ser escolhidos
    - Menor nível de segurança: WriteConcern.NONE
  - Existe um equilíbrio que precisa ser cuidadosamente pensado com base nos requisitos de negócio da aplicação
    - Identificar quais configurações fazem sentido na leitura
    - Definir o nível de segurança desejado durante a gravação