

Persistência de dados

Rodrigo Moura J. Ayres

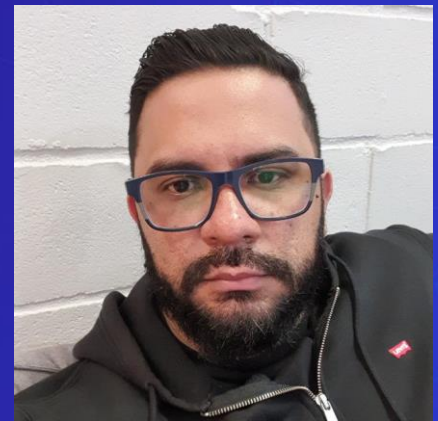
Bacharel em Ciência da Computação @ Univem

Mestre em Ciência da Computação @ UFSCar

Doutorando em Ciência da Computação @ UNESP

Professor de Ensino Superior @ Fatec Ourinhos

Professor de Ensino Superior @ Unifio



rmayres@gmail.com

O que é persistência?

Algo que continua a existir

Tipos de persistência

- Memória
- Disco

Modelo de Dados

- Bancos de dados são projetados com base nas regras e nos conceitos de um **modelo de dados**.
- Existem modelos para diferentes níveis de abstração de representação de dados:
 - Modelos conceituais
 - Modelos lógicos
 - Modelos físicos

Modelos Conceituais

- Representação com alto nível de abstração
 - Modela de forma mais natural os fatos do mundo real, suas propriedades e seus relacionamentos.
 - independente de BD
 - preocupação com a semântica da aplicação
 - exemplo: modelo entidade-relacionamento

Modelos Lógicos

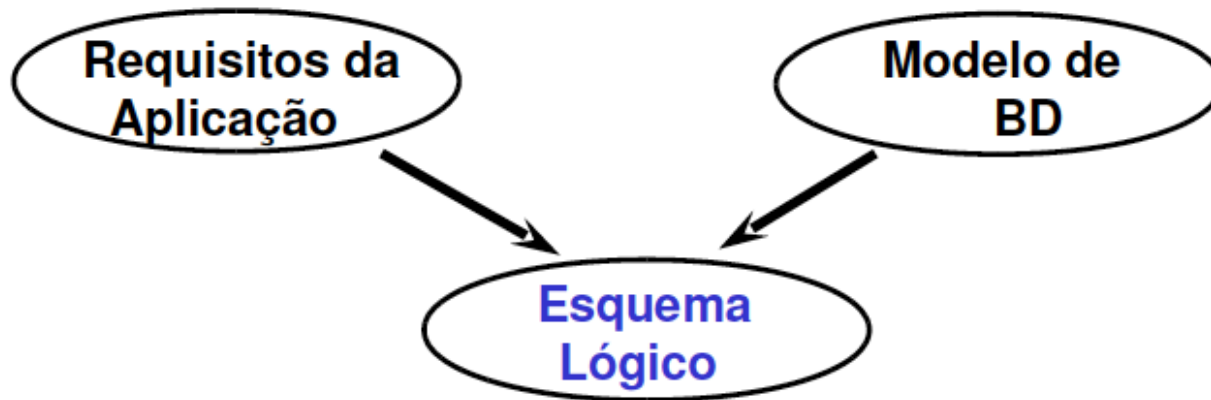
- Representam os dados em alguma estrutura (lógica) de armazenamento de dados
- também chamados de modelos de BD
- dependente de BD
- Exemplos:
 - modelo relacional (tabelas)
 - modelos hierárquico e XML (árvore)
 - modelo orientado a objetos (classes - objetos complexos)

Modelos Lógicos

- Suporte a métodos de acesso
 - especificação dos conceitos do modelo (DDL)
 - dados, seus domínios, relacionamentos e restrições
- Manipulação de conceitos modelados (DML)

Esquema de BD

- Resultado da especificação dos dados de um domínio de aplicação em um modelo de BD



Tipos de Modelos de Dados

- Modelo de banco de dados hierárquico
- Modelo relacional
- Modelo de rede
- Modelo de banco de dados orientado a objetos
- Modelo entidade-relacionamento
- Modelo documental
- Modelo multidimensional

Modelos de BD

- **1a geração:** Modelos pré-relacionais
 - *modelos hierárquico e de rede*
- **2a geração:** Modelo relacional
- **3a geração:** Modelos pós-relacionais
 - *modelos orientado a objetos, objeto-relacional temporal, geográfico, ...*

Modelos Pré-Relacionais

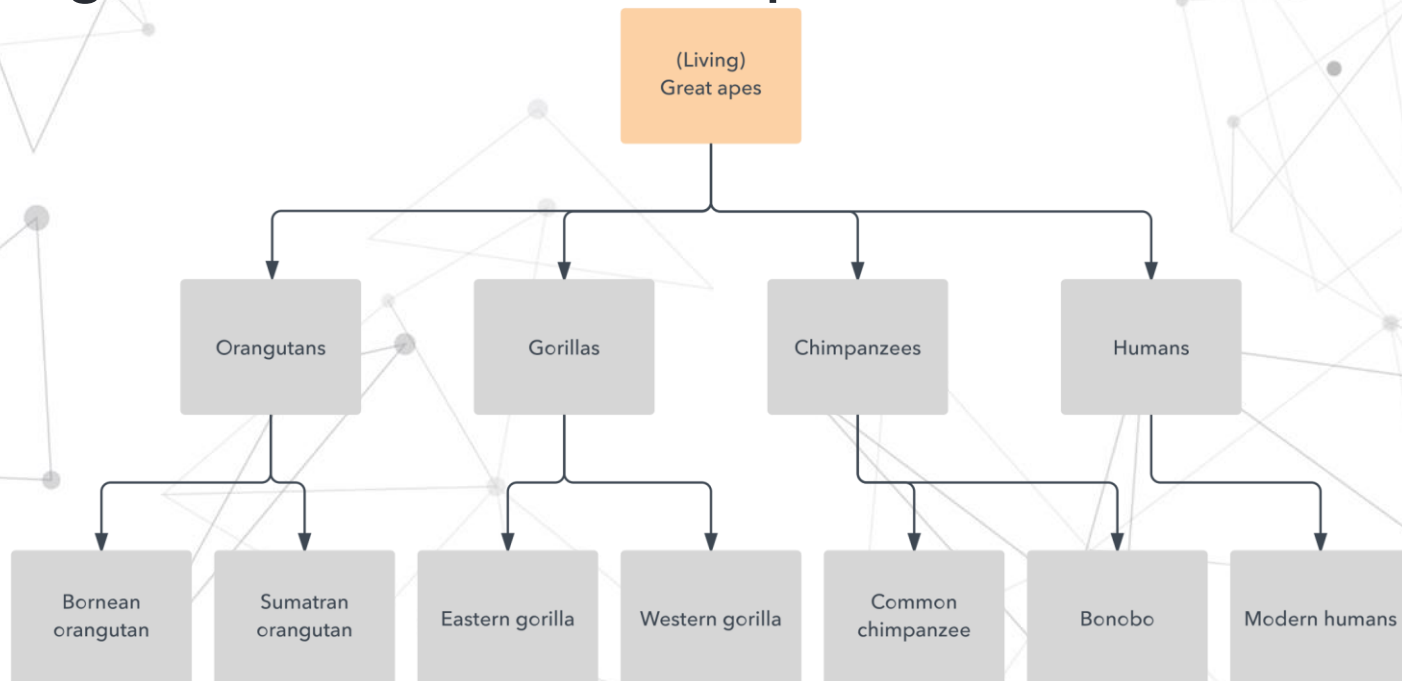
- Modelos com várias limitações:
 - Problemas de Performance. Ex. Varredura em grafo (rede)
 - Inexistência de uma linguagem de consulta declarativa.
 - Consultas exigem programação pela aplicação
 - Manipulam um registro por vez
 - Baixa performance de acesso.

Modelos Pós-Relacionais

- Novos modelos de dados para atender requisitos de algumas categorias de aplicações
- BDOO
 - dados com representação complexa
- Exemplos de áreas de aplicação
 - Engenharia, arquitetura ...

Modelos hierárquico

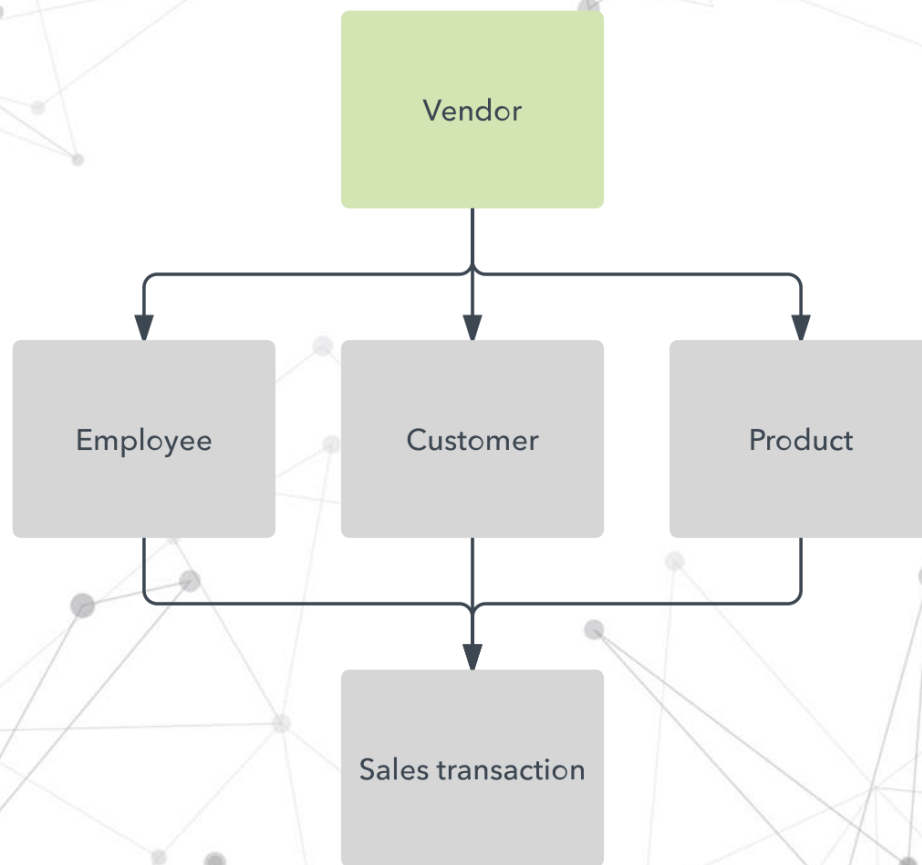
- O modelo hierárquico organiza dados em uma estrutura do tipo árvore, onde cada registro tem um único "pai" ou raiz.



Modelo de Rede

- Se baseia no modelo hierárquico, permite relações muitas para muitas entre registros vinculados, implicando em vários registros "pai".
- Um registro pode ser um membro, ou "filho", em vários conjuntos.
- Foi mais popular nos anos 70, depois de ter sido formalmente definido pela Conferência sobre Linguagens de Sistemas de Dados (CODASYL).

Modelo de Rede



BD Orientados a Objetos

- Define o banco de dados como uma coleção de objetos, ou elementos de software reutilizáveis, com recursos e métodos associados.
- Um banco de dados multimídia incorpora mídia, como imagens, que não podem ser armazenadas em um banco de dados relacional.

BD Orientados a Objetos

- É pós-relacional, pois ele incorpora tabelas, mas não se limita a elas.

Object 1: Sales report

Month	
Product code	
Vendor	
Revenue	

**Object 1
instance**

01-15-16
54
154-234
\$887

Object 2: Sales activity

Customer	
Product code	
Product name	
Sales associate	
Date of sale	
Price	



Modelo Multidimensional

- OLAP (on-line analytical processing) foi introduzido em 1993 por Codd et al.
- Define a categoria de processamento analítico sobre um banco de dados histórico voltado para os processos de gerência e tomada de decisão

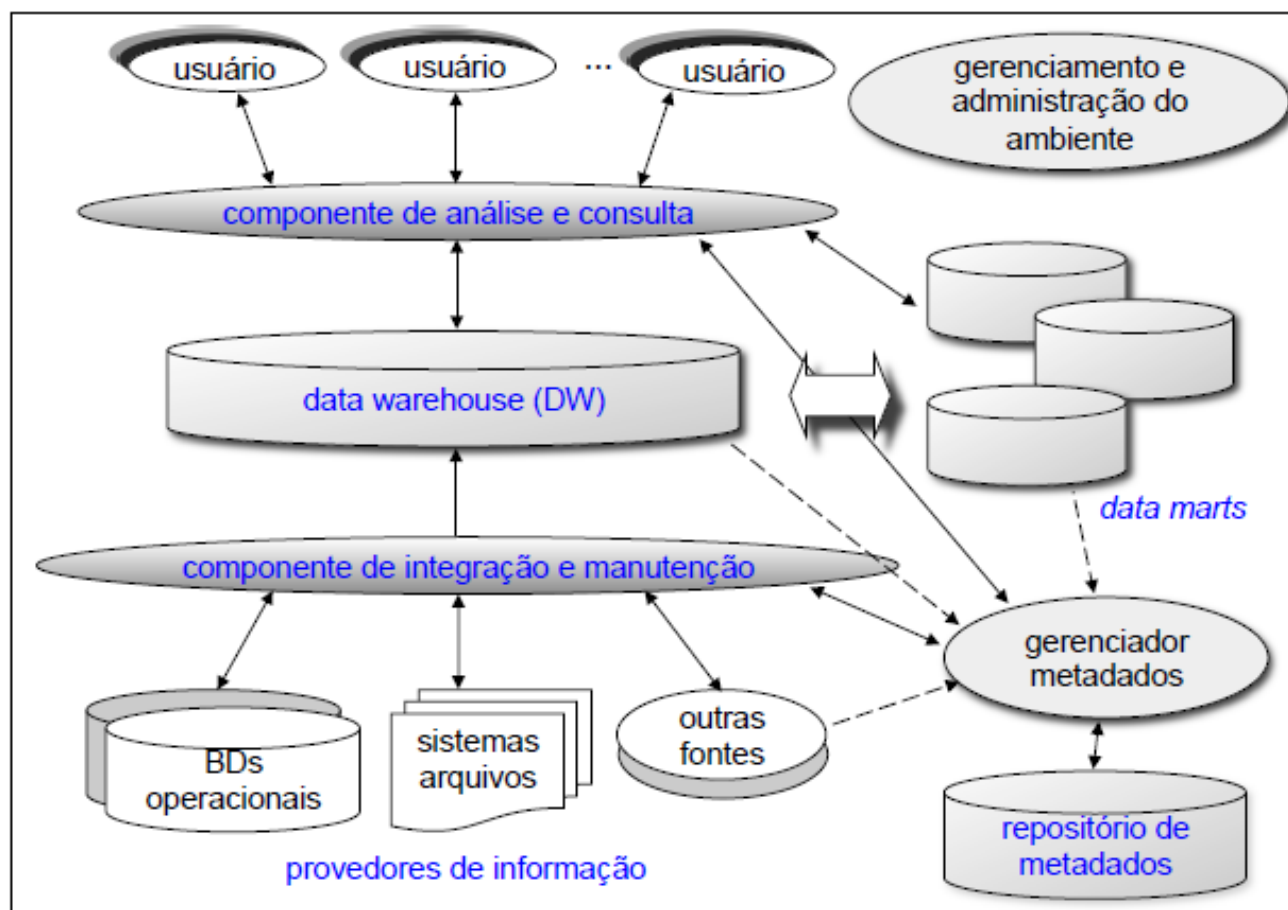
Modelo Multidimensional

- Data Warehousing:
 - Transforma dados operacionais em informação voltada à tomada de decisão estratégica
- Funcionalidades:
 - possibilita que dados de diferentes provedores de informação sejam extraídos, traduzidos, filtrados, integrados e armazenados no data warehouse (DW)

Modelo Multidimensional

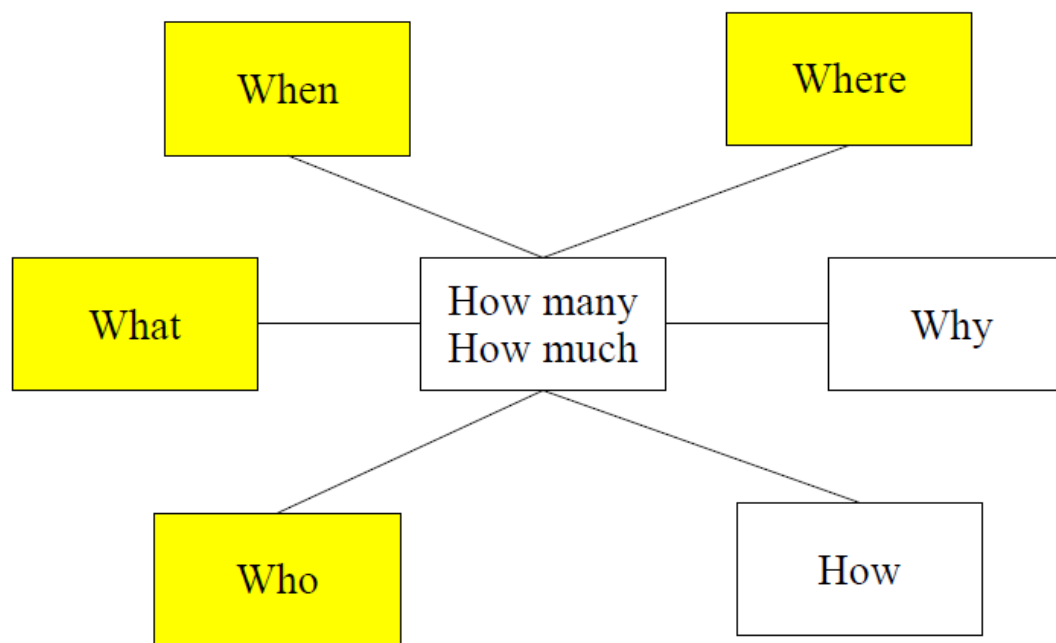
- Análises de tendências:
 - *Quais as vendas mensais de um certo produto no ano de 2021?*
- Análises Comparativas:
 - Quais as vendas mensais dos produtos de uma determinada marca nos últimos 3 anos?
- Análises de tendências múltiplas:
 - *Quais as vendas mensais dos produtos de uma data marca nos últimos 3 anos, de acordo com as promoções de Natal?*


Modelo Multidimensional



Modelo Multidimensional

- Modelo Estrela:



 Tipos de dimensão mais comuns

Modelo Multidimensional

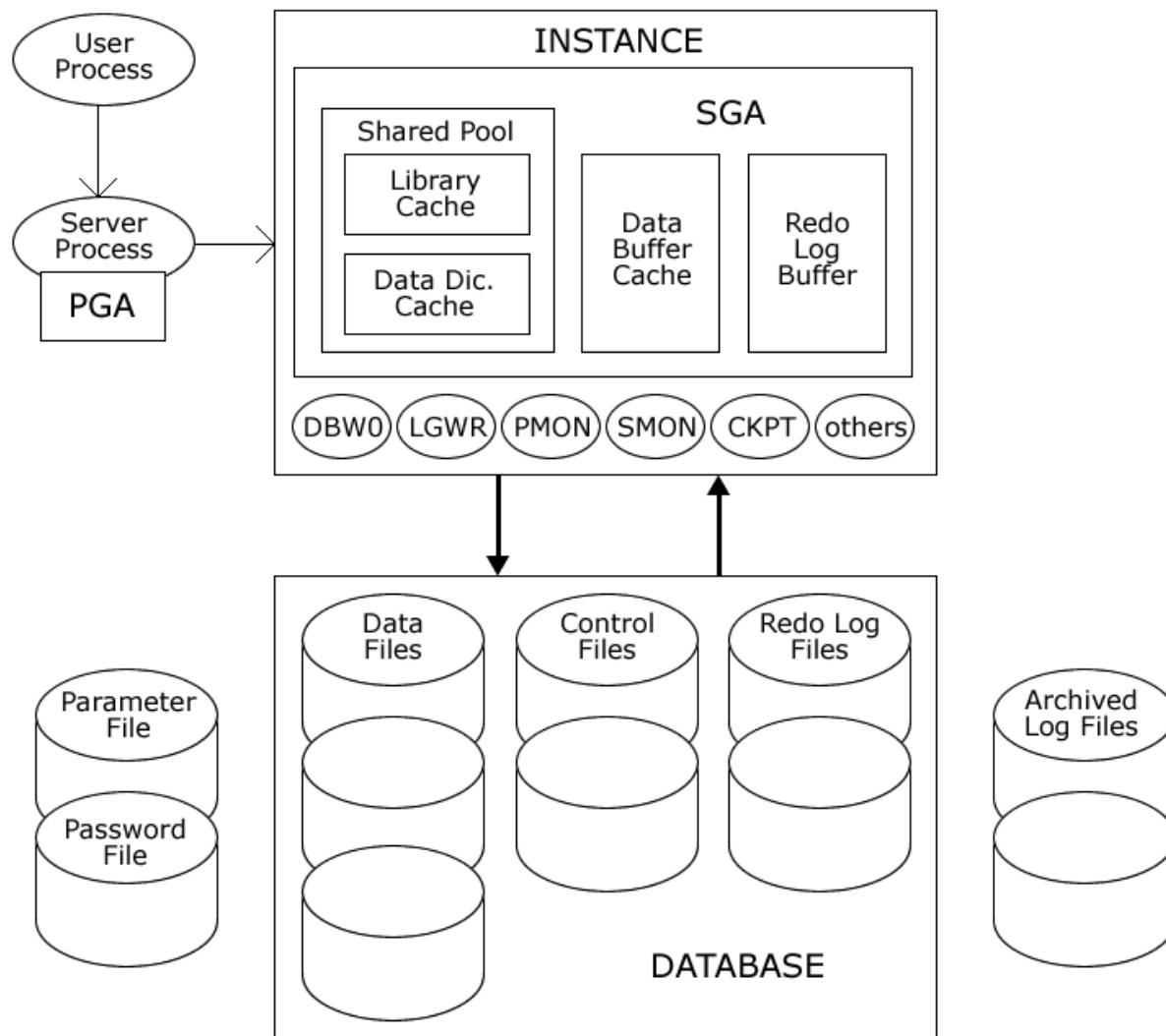
- Fato:
 - Tudo que pode ser representado por um valor aditivo, ou seja, por meio de valores numéricos.
 - O conjunto de valores numéricos é denominado métricas ou medidas.
- Por exemplo:
 - “Os índices de criminalidade aumentam no ano atual de 50% sobre os últimos dois anos”

Modelo Relacional

- Definido em 1970 (E. Codd – IBM/Califórnia)
- Modelo com uma sólida base formal
 - teoria dos conjuntos
- Modelo simples
 - estruturas tabulares
 - poucos conceitos
- Linguagens declarativas para a manipulação de dados
 - álgebra relacional e cálculo relacional (formais)
 - SQL (comercial)

Estrutura Banco de Dados Relacional

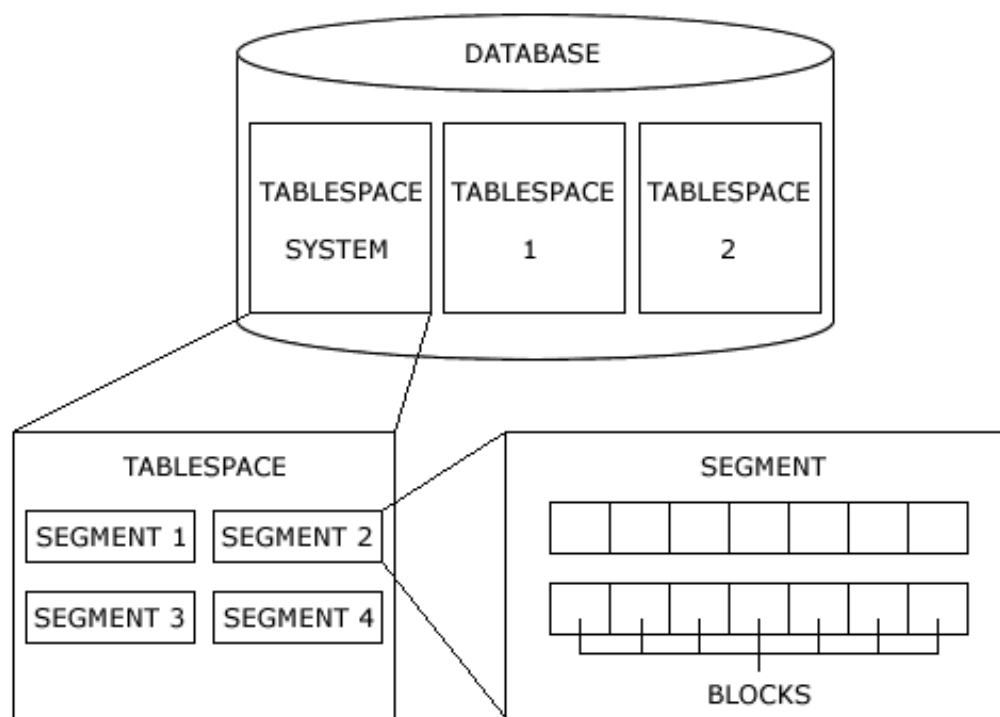
- O Banco de Dados possui uma **estrutura física** e uma **estrutura lógica**:
 - As **estruturas lógicas** representam os componentes que é possível ver no Banco de Dados (tabelas, índices, etc.)
 - As **estruturas físicas** representam os métodos de armazenamento utilizado internamente pelo BD (os arquivos físicos).



Estrutura Banco de Dados

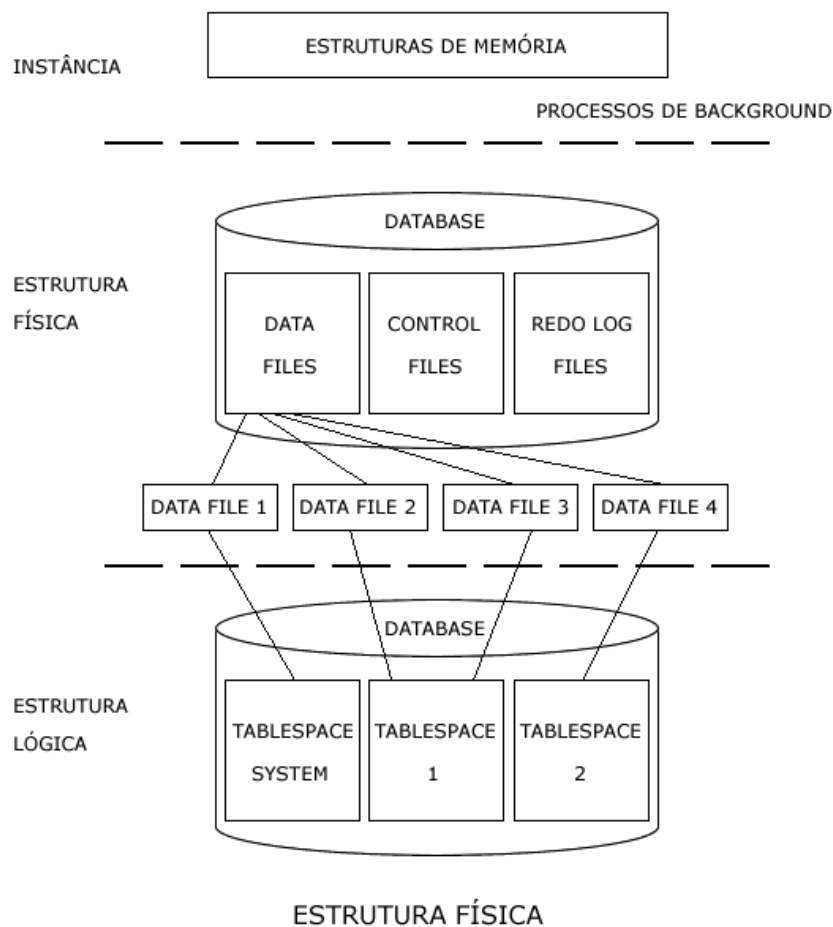
- **Fisicamente**, é um conjunto de arquivos em algum lugar do disco.
 - O local físico desses arquivos é irrelevante para as funções do banco de dados, mas não para seu funcionamento.
- **Logicamente**, o banco de dados é dividido em um conjunto de contas de usuário conhecido como *schemas*.

Estrutura Banco de Dados



ESTRUTURA LÓGICA

Estrutura Banco de Dados



Como fazer persistência?

- Comunicação com SGBD (Sistema de Gerenciamento de Banco de Dados)
- Utilização de linguagens específicas para comunicação (SQL, NoSQL)

Persistência em sistemas não distribuídos

Sistemas não distribuídos

- Necessidade de confiabilidade nas informações

Confiabilidade

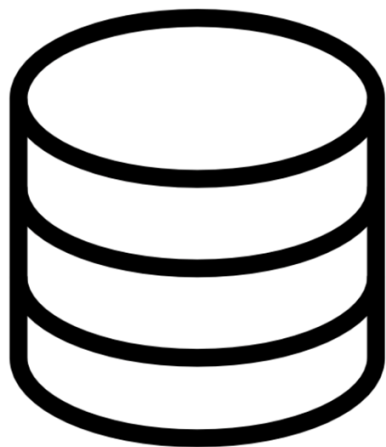
Uso de transações

Transações

Sequência de operações executadas em uma única unidade lógica de trabalho

Independente entre outras transações

Propriedades de transações



A

Atomicidade

C

Consistência

I

Isolamento

D

Durabilidade

Transações - Atomicidade

Todas as operações da mesma unidade lógica devem ser executadas com sucesso para que a transação seja efetivada

Ex.: Transações - Atomicidade

Transferência de dinheiro em um sistema bancário

- Débito de dinheiro de uma conta
- Crédito de dinheiro em outra conta

Transações - Consistência

Todas as regras e restrições definidas devem ser obedecidas

Um banco de dados deve sair de um estado de consistência para outro estado de consistência

Ex.: Transações - Consistência

Banco que guarda CPF de um cliente

A inclusão ou alteração de um dado não pode ocasionar em uma duplicidade de CPFs (integridade de chaves primárias)

Utilização de valores inválidos para CPF (integridade lógica)

Transações - Isolamento

Cada transação funciona completamente a parte de outras transações

Ex.: Transações - Isolamento



Campo: salario
Valor inicial: 100
Valor final: 121

TRANSAÇÃO 1	TRANSAÇÃO 2
BEGIN	
LÊ O CAMPO SALARIO: 100	
APLICA AUMENTO DE 10: 110	
COMMIT	
	BEGIN
	LÊ O CAMPO SALARIO: 110
	APLICA AUMENTO DE 10%: 121
	COMMIT

Ex.: Transações - Isolamento



Campo: salario
Valor inicial: 100
Valor final: 120

TRANSAÇÃO 1	TRANSAÇÃO 2
	BEGIN
	LÊ O CAMPO SALARIO: 100
	APLICA AUMENTO DE 10%: 110
	COMMIT
BEGIN	
LÊ O CAMPO SALARIO: 110	
APLICA AUMENTO DE 10: 120	
COMMIT	

Ex.: Transações - Isolamento



Campo: salario
Valor inicial: 100
Valor final: 110

TRANSAÇÃO 1	TRANSAÇÃO 2
LÊ O CAMPO SALARIO: 100	LÊ O CAMPO SALARIO: 100
	APLICA AUMENTO DE 10%: 110
APLICA AUMENTO DE 10: 110	

Transações - Durabilidade

O resultado de uma transação é permanente.

O resultado só pode ser desfeito com uma próxima transação

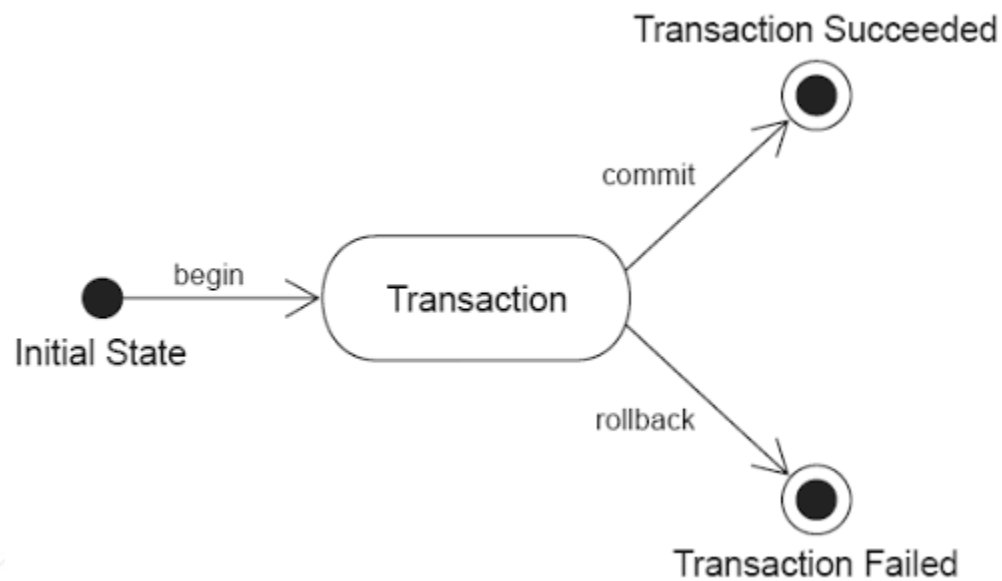
Ex.: Transações - Durabilidade

Houve uma queda de energia, a transação não pode ficar incompleta

Como garantir ACID?

Controle de transação

Como garantir ACID?



Ex. de controle de transação



```
DB::beginTransaction();
```

```
try {  
    codigo();  
    maisCodigo();  
    DB::commitTransaction();  
} catch (\Exception $exception) {  
    DB::rollbackTransaction();  
}
```

Controle de transações concorrentes

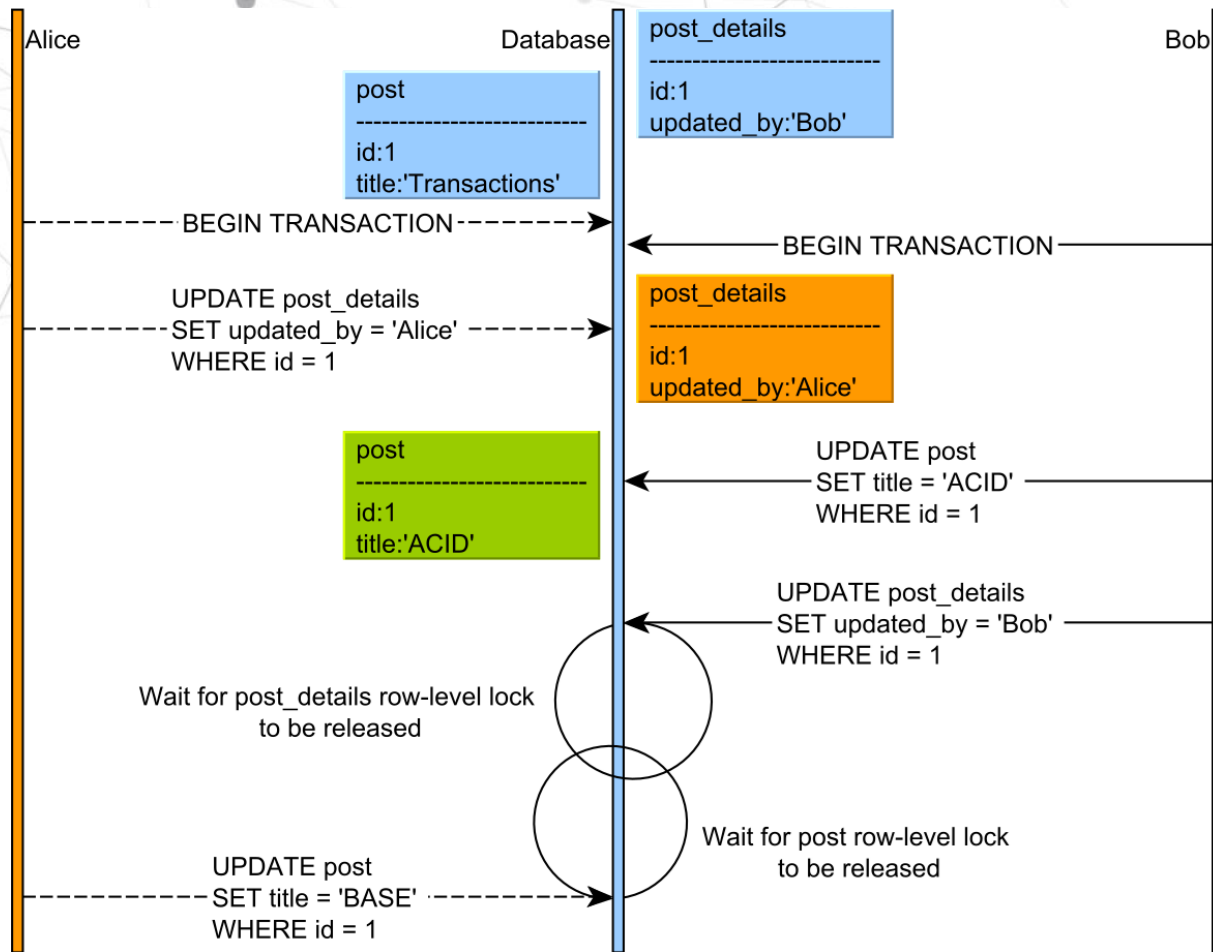
Utilização de locks

- Utilizado para evitar inconsistência quando tem transações concorrentes

Controle de transações concorrentes

- Lock compartilhado
 - utilizado para transações de leitura; não é possível atualizar o dado enquanto ele é lido
- Lock exclusivo
 - utilizado para operações de exclusão e alteração; não é possível ler o dado enquanto ele está reservado

Deadlock



Precisamos nos preocupar?

Atualmente os ORMs (Mapeamento objeto-relacional), utilizados em frameworks, fazem esse tratamento

ORM em frameworks?

LARAVEL
Eloquent ORM



Dapper
a simple object mapper for .Net

Persistência em sistemas distribuídos

Sistemas Distribuídos

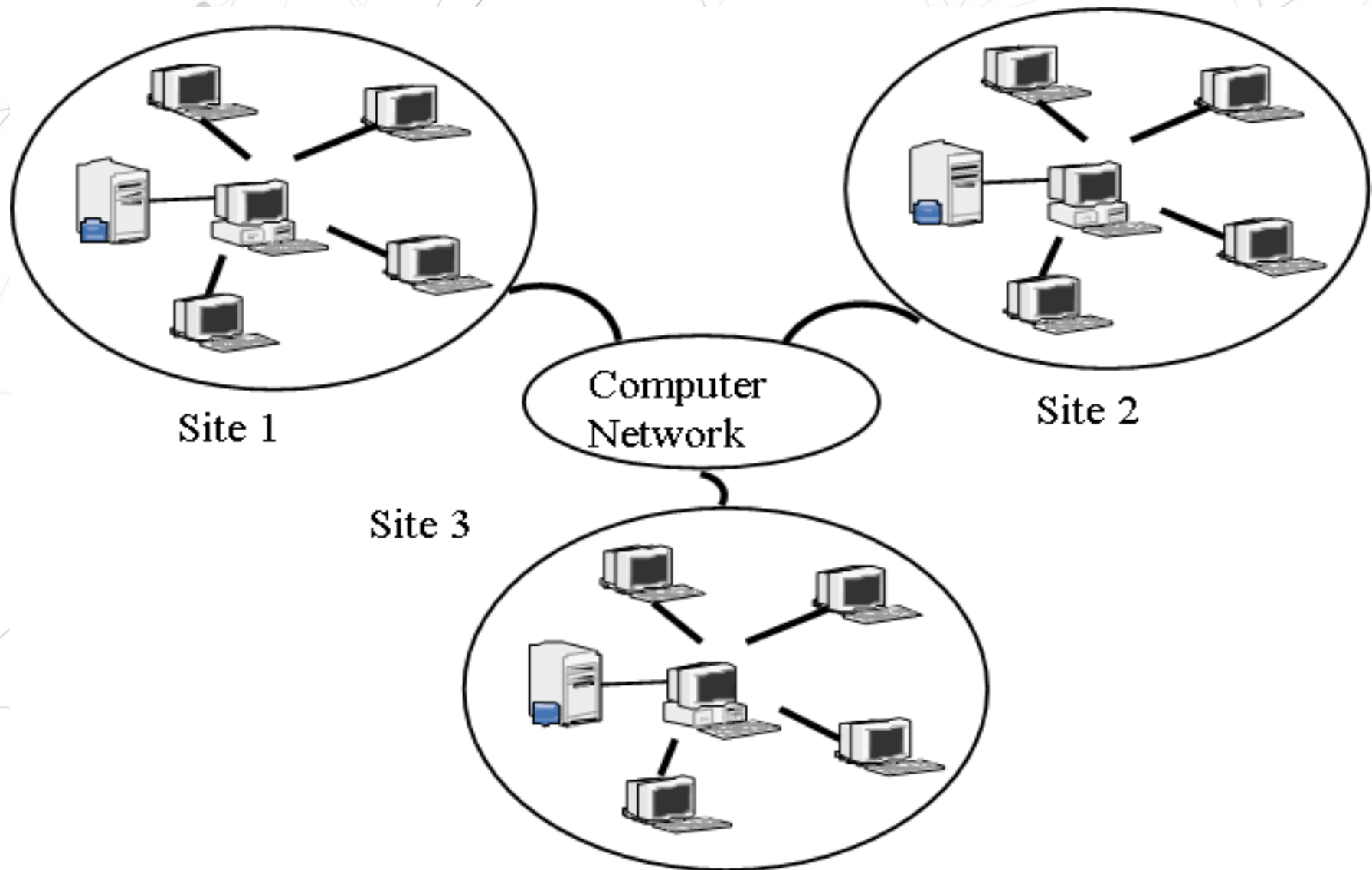
- Grupo de elementos autônomos de processamento (*não necessariamente homogêneos*) que estão interconectados por uma rede de computadores e que cooperam na realização de tarefas a eles atribuídas.

BD Distribuído

- Coleção de diversas bases de dados, interligadas logicamente através de uma rede de computadores.
- Para o usuário, um DB distribuído se parece e comporta exatamente como um banco de dados não distribuído.

Possíveis Arquiteturas

- **Bancos de Dados Distribuídos Homogêneos**
 - O mesmo **software / esquema** em todos os sites, os dados podem ser divididas entre sites
 - **Objetivo:** fornecer a visão de um único banco de dados, escondendo detalhes de distribuição (*transparência de distribuição*)
- **Bancos de Dados Distribuídos Heterogêneos**
 - **Software / esquema** diferentes em locais diferentes
 - **Objetivo:** integrar bases de dados existentes para fornecer funcionalidades úteis



Promessas de SBDDs

- **Transparência** na Gerência dos Dados Distribuídos, Fragmentados e Replicados
- **Confiabilidade** através de Transações Distribuídas
- **Aumento de Desempenho**
- **Escalabilidade**

Confiabilidade

- Espera-se que os SBDDs ofereçam ***confiabilidade*** por trabalharem com componentes replicados, eliminando assim pontos únicos de falha
- A ***confiabilidade*** é, portanto, associada a **tolerância à falhas**

Aumento de Desempenho

- Proximidade dos dados de seus pontos de uso
- **Execução Paralela**
 - *Paralelismo entre consultas*

Escalabilidade

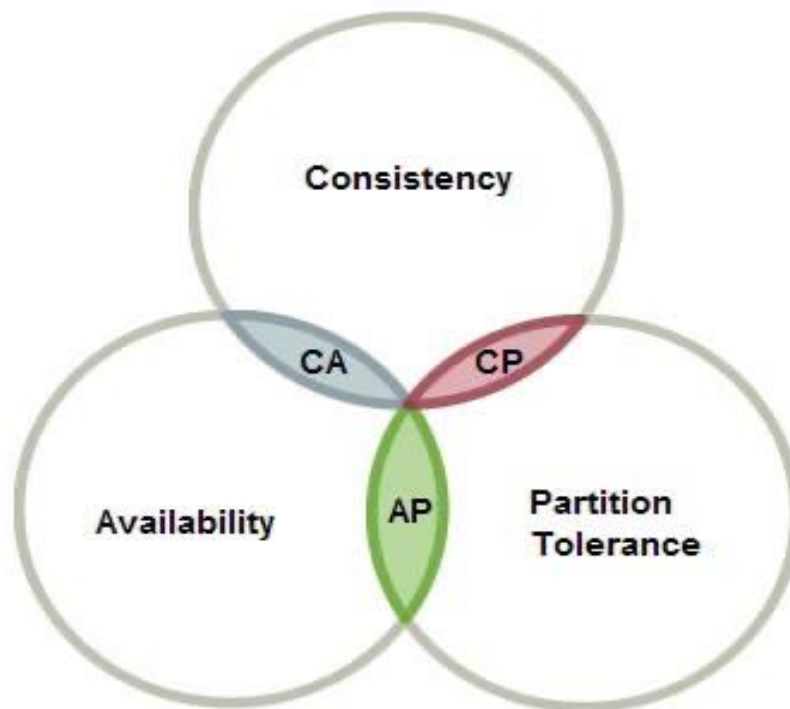
- Aumento da capacidade de processamento e armazenamento da rede através da adição de *elementos processadores, equipamentos de rede e aprimoramento de software de BD*

Teorema CAP

Teorema CAP afirma que apenas dois dos três itens abaixo podem ser satisfeitos concorrentemente:

- Consistência (consistency)
- Disponibilidade (availability)
- Tolerância a falhas (partition)

Teorema CAP



Teorema CAP - Consistência

Cada leitura sempre recebe o que foi escrito mais recentemente (ou um erro)

Teorema CAP - Disponibilidade

Sempre acessível por todos

Toda leitura recebe uma resposta sem erro

Sem a garantia de que é o que foi escrito mais recentemente

Teorema CAP - Tolerância

Tudo continua a funcionar apesar da indisponibilidade de algum servidor onde os dados estão contidos

Problemas

- Como tratar atualizações?
 - Atualizações sobre dados replicados implica na implementação de controle de concorrência distribuído e protocolos de finalização (commit protocols)

Controle de transação

- 2PC - Two Phase Commit
- SAGA

Two Phase Commit

- Duas fases:
 - Preparação
 - Confirmação
- Dois papéis:
 - Coordenador
 - Servidores de banco de dados

Two Phase Commit

- Coordenador cria uma transação global
- Cada servidor inicia sua própria transação
- Coordenador espera a resposta de cada um dos servidores
 - Caso todos tenham respondido OK, uma mensagem de "commit" é enviada para todos os servidores
 - Caso alguma tenha respondido com falha, uma mensagem de "rollback" é enviada para todos os servidores

SAGA

- Transações
- Compensações

SAGA

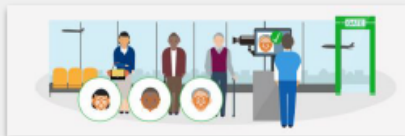
- **Saga** é um conceito antigo de bancos de dados, proposto em 1987 por Hector Garcia-Molina e Kenneth Salem.
- Dois métodos:
 - Orquestrada
 - Coreografada

SAGA

- Uma saga representa uma coleção de sub transações locais

Saga - Caso de Sucesso

Boarding Service



Passenger On Board ↻



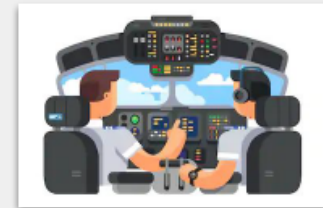
Passenger Security Service



Passenger Safe ↻



Airplane Check Service



Passenger On Flight! ↻

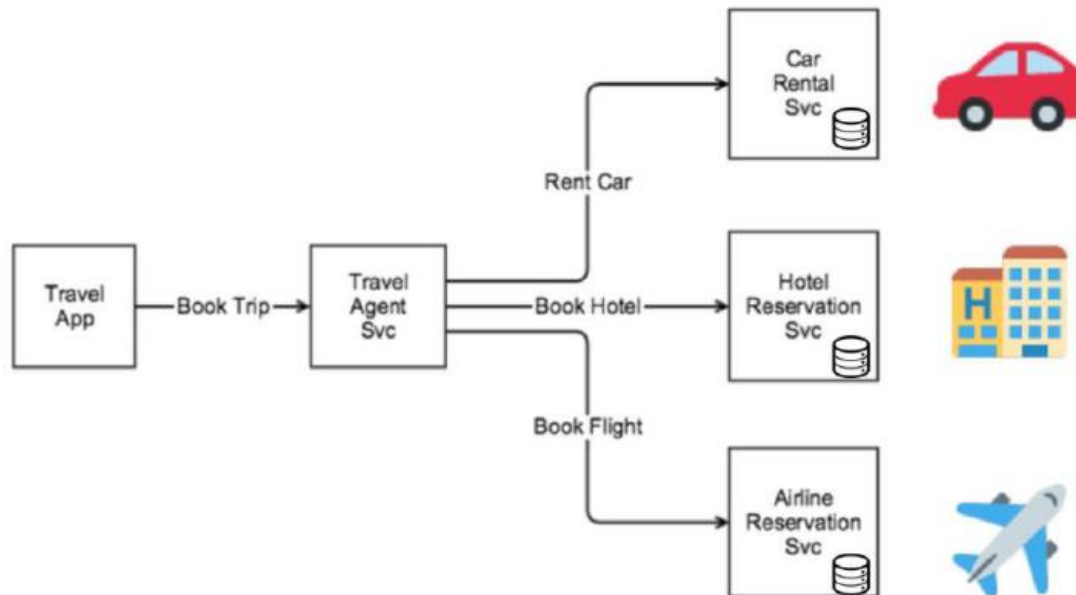
SAGA

- Cada sub-transação deve possuir uma transação de compensação
 - A transação de compensação desfaz o que foi feito na sub-transação local.



SAGA - Orquestrada

- Orquestrada: Neste caso, temos um orquestrador que conhece os participantes e coordena a execução da saga.

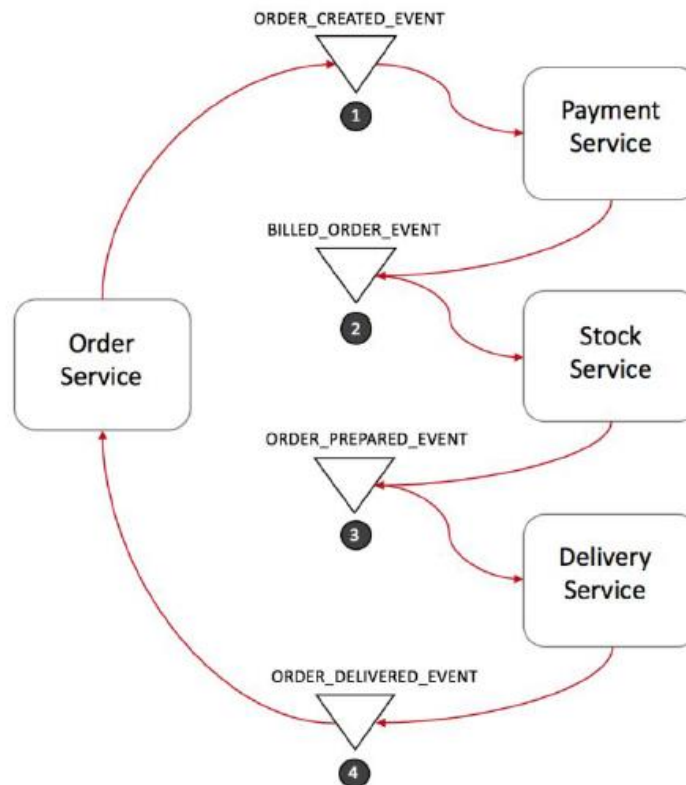


SAGA - Orquestrada

- Existe um sistema orquestrador que faz o papel de "avisar" o próximo, em caso de sucesso, ou o anterior, em caso de falha
- O orquestrador serve como uma "fonte da verdade" em relação ao status da execução

SAGA - Coreografada

- Coreografada: Um participante invoca o outro ao final de sua própria transação através de eventos normalmente.



SAGA - Coreografada

- Em caso de sucesso, o serviço é responsável por dizer ao próximo que ocorreu sucesso ao concluir uma operação
- Em caso de falha, o serviço é responsável por dizer ao anterior que ocorreu falha na conclusão da sua operação

SAGA - Implementação

- Síncrona: requisições diretas onde o orquestrador espera a resposta do serviço chamado
- Assíncrona: uso de eventos com mensageria

Bancos de dados relacionais (SQL)

Características:

- ACID
- Rigidez de esquema
- Inflexibilidade no formato de dados

Ex.: bancos de dados relacionais

- MySQL
- SQL Server
- Oracle
- Postgre SQL
- ...

Bancos de dados não relacionais (NoSQL)

Quando não há necessidade de relações entre as entidades

Muito eficaz em escalabilidade e alto desempenho

Bancos de dados não relacionais (NoSQL)

Características:

- Alta performance
- Facilidade em escalar
- Flexibilidade

Tipos de bancos de dados não relacionais (NoSQL)

- Colunar
- Documentos
- Chave-valor
- Grafos

Ex.: Bancos de dados não relacionais (NoSQL)

- Colunar
 - Cassandra
- Documentos
 - MongoDB
- Chave-valor
 - Redis

Quando usar bancos do tipo Colunar?

- Query antes de entidade

Quando usar bancos do tipo Documentos?

- Alta quantidade de registros
- Alta taxa de requisição
- Dados são agrupados em documentos (geralmente JSON), sem relações

Quando usar bancos de chave-valor?

- Frequentemente utilizados para cache
- Frequentemente utilizado para feature-flag

Qual usar? SQL ou NoSQL?

- Posso abrir mão de consistência?
 - Qual é meu sistema?
 - Um dado momentaneamente errado, faz diferença?
- Meus dados precisam estar estritamente estruturados?
- Preciso manter relação entre meus dados?

Demo!

Prática

Criação de aplicações simples, da linguagem de preferência:

- Persistência de escrita em um banco relacional
- Uso manual de controle de transação (begin, commit, rollback)

Extra: atualização de dados de leitura usando persistência através de eventos

Obs.: pode ser tudo feito em uma aplicação apenas ou em aplicações pequenas separadas

Obrigado!