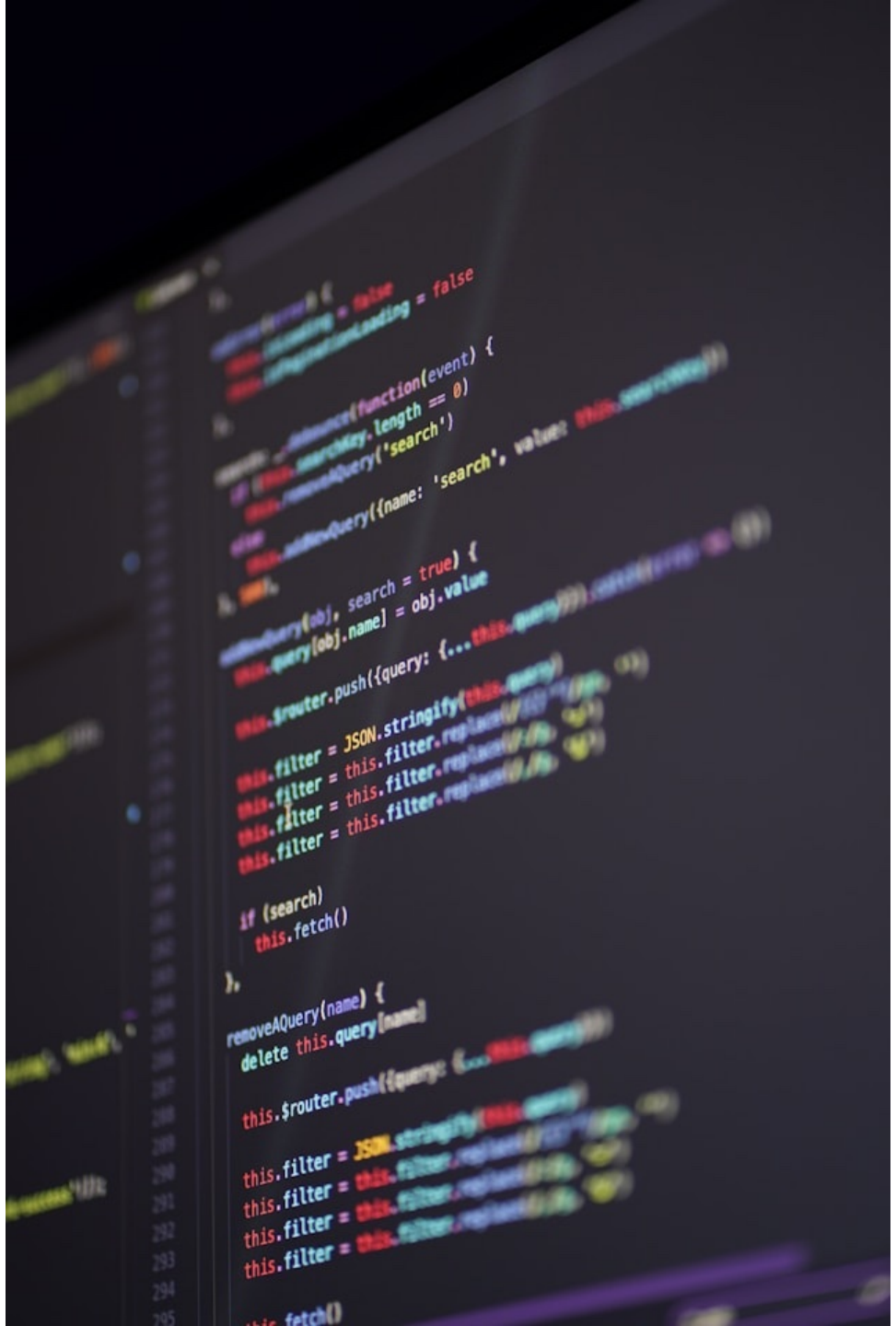


5954025 - Sistemas Distribuídos

Aula 12a - Transações

Prof. Dr. Denis M. L. Martins
DCM | FFCLRP | USP

Objetivos de Aprendizagem

- Explicar o conceito de transação como uma unidade lógica de trabalho.
- Compreender por que transações são necessárias em sistemas com acesso concorrente e possibilidade de falhas.
- Identificar problemas causados por execuções simultâneas.
- Descrever o ciclo de vida de uma transação.
- Explicar as propriedades ACID: atomicidade, consistência, isolamento e durabilidade.
- Compreender, em alto nível, como transações são gerenciadas em sistemas.

Por que estudar transações?

Sistemas reais executam várias operações ao mesmo tempo.

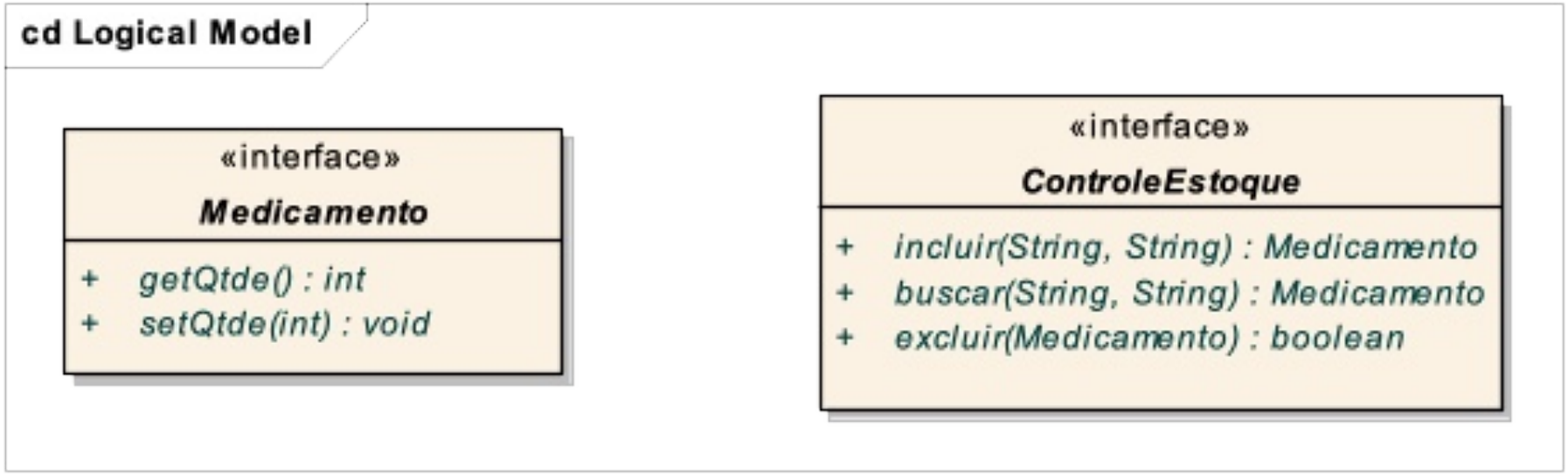
Exemplos:

- Transferência bancária
- Atualização de estoque
- Compra em e-commerce

Problema central: Como garantir consistência mesmo com concorrência e falhas?

Exemplo de Motivação

Considere um hospital com estoques distribuídos entre diferentes unidades, onde implementação de um serviço de controle de estoque tem a seguinte difinição:



Como diferentes aplicações (clientes) podem usar este serviço, como evitar **inconsistências** em operações **simultâneas**?

Exemplo de Motivação (cont.)

Considere o estoque inicial de *Diuretil* na Cardiologia: 10 unidades.
Considere ainda as operações concorrentes C1 e C2 a seguir:

Tempo	C1 — Transferir 5 unidades	C2 — Adicionar 10 unidades	Estoque na Cardiologia
T0	—	—	10
T1	Lê estoque da Cardiologia: **10**	—	10
T2	Calcula novo estoque: **10 - 5 = 5**	—	10
T3	Grava estoque da Cardiologia: **5**	—	5
T4	—	Lê estoque da Cardiologia: **5**	5
T5	—	Calcula novo estoque: **5 + 10 = 15**	5
T6	—	Grava estoque da Cardiologia: **15**	15

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exemplo de Motivação (cont.)

Considere esta outra ordem de operações:

Tempo	C1 — Transferir 5 unidades	C2 — Adicionar 10 unidades	Estoque na Cardiologia
T0	—	—	10
T1	Lê estoque da Cardiologia: **10**	—	10
T2	—	Lê estoque da Cardiologia: **10**	10
T3	Calcula novo estoque: **10 - 5 = 5**	—	10
T4	—	Calcula novo estoque: **10 + 10 = 20**	10
T5	Grava estoque da Cardiologia: **5**	—	5
T6	—	Grava estoque da Cardiologia: **20**	20

Necessidade de transações

Precisamos garantir que um conjunto de operações seja executado:

- De forma atômica
- Sem interferência indevida
- Preservando consistência
- Mesmo na presença de falhas

- Esse é o papel das **transações**.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Conceito de transação

Definição: Uma transação é **um conjunto de operações** tratado como uma única **unidade lógica** de trabalho.

Exemplo:

```
Transferência bancária:
1. Debitar R$ 100 da conta A
2. Creditar R$ 100 na conta B
```

Ou as duas operações acontecem, ou nenhuma acontece.

Transação como unidade indivisível

Uma transação deve terminar em apenas um dos estados:

- **Commit**
  - A transação foi confirmada
  - Suas alterações se tornam permanentes
- **Abort/Rollback**
  - A transação foi cancelada
  - O sistema retorna ao estado anterior

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Ciclo de vida de uma transação

```
openTransaction()
    operação 1
    operação 2
    operação 3
closeTransaction()
```

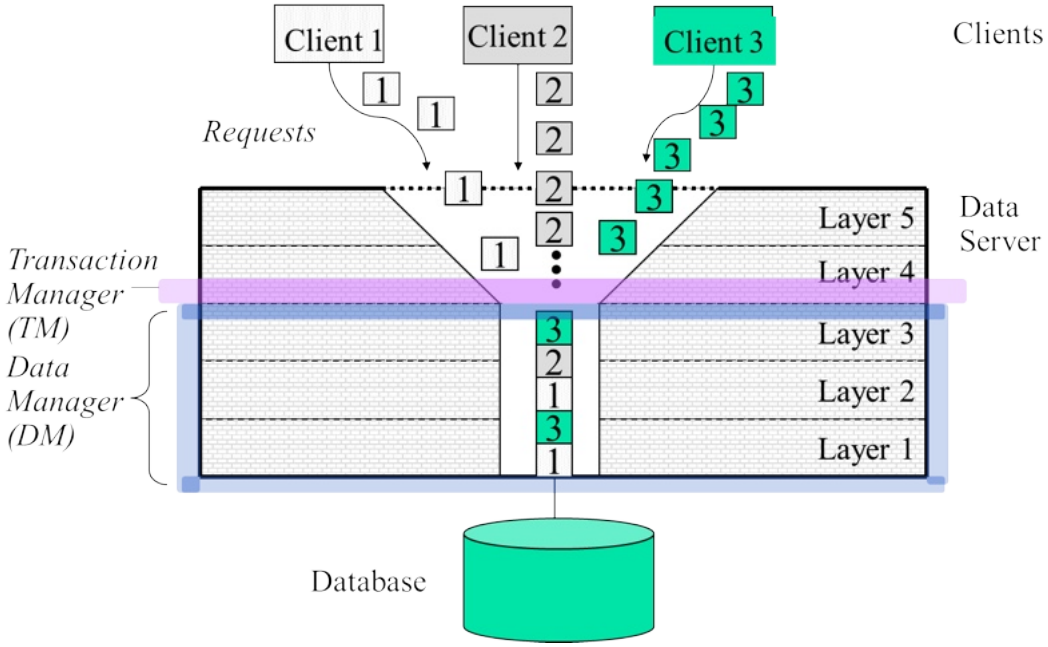
Possíveis resultados:

```
commit → alterações confirmadas
abort  → alterações desfeitas
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Transações em Bancos de Dados

- O acesso concorrente aos mesmos dados pode induzir diferentes tipos de anomalias.
- Em um SGBD Transações passam por camadas responsáveis por:
  - receber comandos da aplicação;
  - analisar e otimizar comandos SQL;
  - controlar concorrência;
  - registrar operações no log;
  - acessar páginas de dados em memória e em disco.



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Delimitação de Transações e Ciclo de Vida

- Os limites de uma transação podem ser especificados de forma implícita ou explícita.
- **Explicitamente**: **begin_transaction** ... **end_transaction**
- **Implicitamente**: primeira instrução SQL executável.
- Quando a primeira operação é executada, a transação passa ao estado ativo.
- Se a transação for concluída com sucesso, ela pode ser confirmada.
- Caso contrário, precisa ser desfeita por rollback.

Exemplo: Transferência Bancária

```
<begin_transaction>

UPDATE account
SET balance = balance - :amount
WHERE accountnumber = :account_to_debit;

UPDATE account
SET balance = balance + :amount
WHERE accountnumber = :account_to_credit;

<end_transaction>
```

Se houver erro, a transação deve ser desfeita.

Propriedades de Transações: ACID

Toda transação tradicional deve buscar quatro propriedades (ACID):

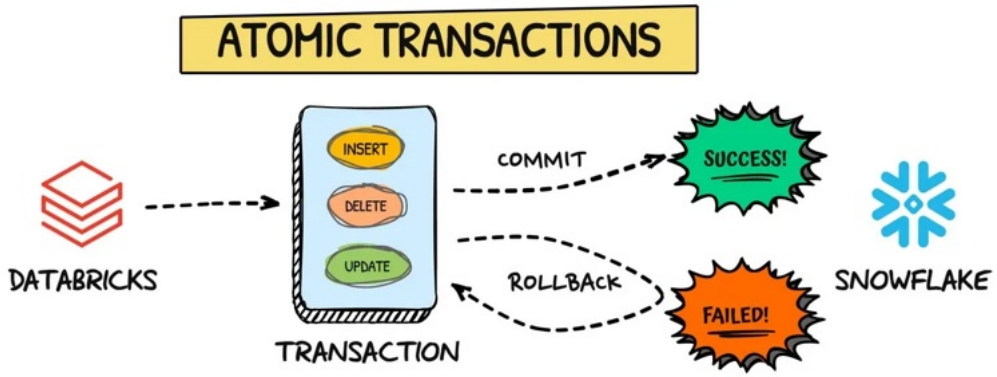
- **Atomicidade**
- **Consistência**
- **Isolamento**
- **Durabilidade**

> O termo ACID foi cunhado por Andreas Reuter and Theo Härder, fundamentados no trabalho de Jim Gray (na foto ao lado).
> Veja em https://dl.acm.org/doi/10.1145/289.291.
> Fonte da Imagem: ACM
> Veja mais sobre ACID em Julia Evans' Drawing.

Atomicidade

Esta propriedade garante que cada transação ou acontece completamente ou simplesmente não acontece.

- Adicionalmente, se a transação acontece, esta é indivisível e “instantânea” não há estados intermediários
- Se uma transação possui várias operações, o sistema não pode deixar apenas parte delas efetivada.



Fonte da Imagem: Vigneshraja Palaniraj @Medium.

Consistência

Esta propriedade garante que a transação não afeta as invariantes do sistema, i.e., a execução de uma transação **não deve introduzir inconsistências**.

- Uma transação leva o sistema de um estado consistente para outro estado consistente.
- **Exemplo**: O saldo total antes e depois de uma transferência deve permanecer correto.

Isolamento

Isolamento significa que transações concorrentes não devem interferir incorretamente entre si.

- Efeitos intermediários de uma transação não devem ser visíveis por outras transações.
- Indica que, quando múltiplas transações são executadas concorrentemente, o resultado deve ser o mesmo que se cada transação fosse executada isoladamente (**serialização**).
  - Mesmo que, internamente, elas sejam executadas de forma intercalada.

Durabilidade

Indica que os efeitos de uma transação confirmada (após o **commit**) devem sempre ser persistidos no banco de dados.

Mesmo se ocorrer:

- Queda de energia
- Falha do sistema operacional
- Falha do banco de dados
- Reinicialização do servidor

> É responsabilidade do gerenciador de recuperação, por exemplo, por meio de operações **REDO** ou redundância de dados.

Classificação de Transações

Transações planas são formadas por um conjunto de operações executadas de forma sequencial e não hierárquica

- Não possuem subtransações
- Não permitem o comprometimento de resultados parciais
- Ou toda a transação termina com sucesso ou o estado do sistema retorna à situação anterior ao início da transação

Transações aninhadas permitem que uma transação seja composta por outras transações.

- Subtransações podem ser efetivadas ou canceladas independentemente
- Subtransações de um nível (e suas descendentes) podem ser executadas concorrentemente com outras subtransações de mesmo nível na hierarquia.

Transações Planas: Exemplo

```
T:
1. Ler saldo da conta A
2. Debitar conta A
3. Creditar conta B
4. Confirmar
```

Limitação: Não permitem confirmação parcial de resultados.

Transações Aninhadas: Exemplo

```
T
├── T1: reservar voo
├── T2: reservar hotel
└── T3: reservar carro
```

Vantagem:

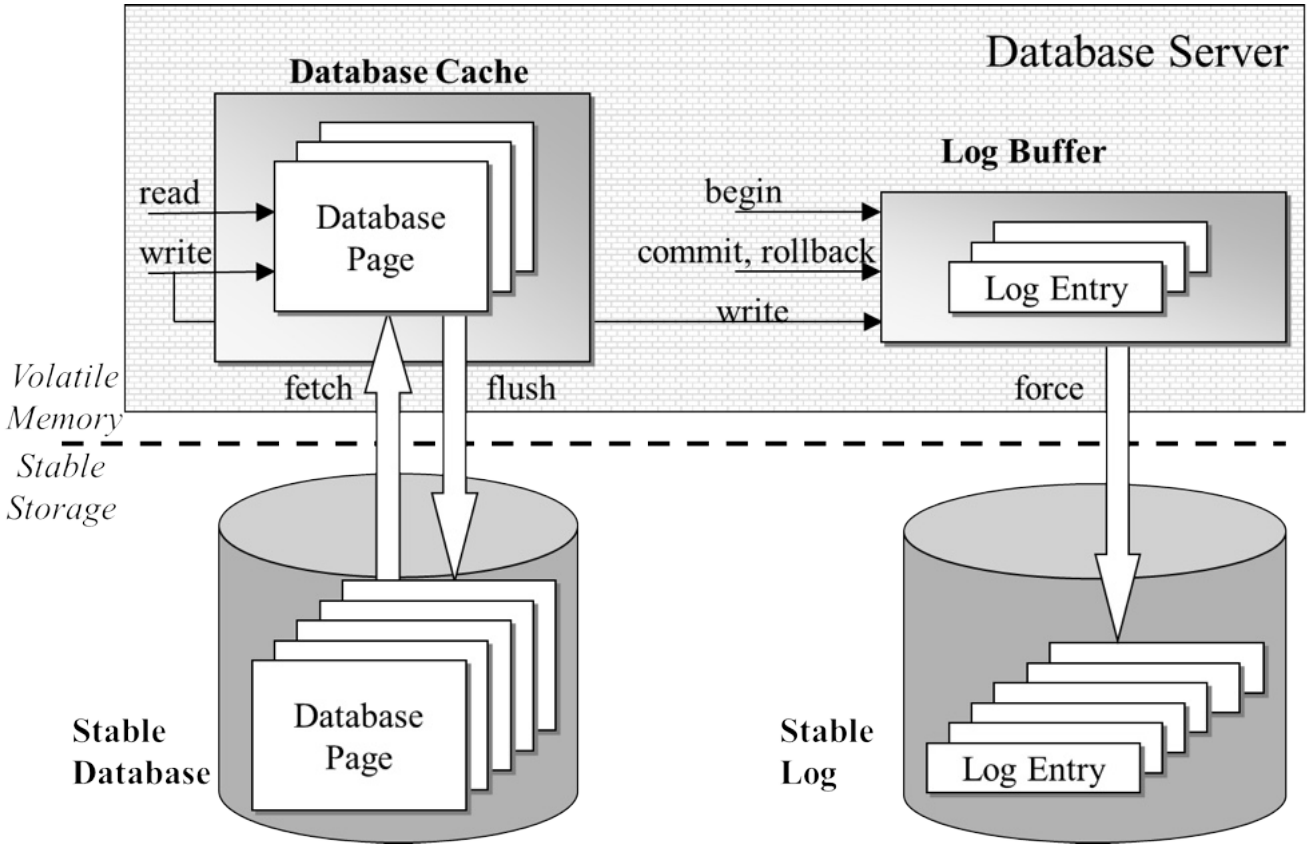
- Subtransações podem executar em paralelo
- Cada subtransação pode ser tratada separadamente
- Útil em sistemas distribuídos

> **Note que**: quando uma transação ascendente é cancelada, todas as suas subtransações são canceladas. Quando uma subtransação é cancelada, a transação ascendente pode decidir se vai ser cancelada ou não.
> Se transação de nível superior é efetivada, então todas as subtransações que foram efetivadas provisoriamente também podem ser efetivadas desde que nenhuma de suas ascendentes tenha sido cancelada.

Transações em SGBDs

No gerenciamento de transações, os principais componentes de um SGBD são:

- Gerenciador de transações
- Gerenciador de recuperação
- Gerenciador de concorrência ou escalonador
- Gerenciador de buffer
- Gerenciador de armazenamento
- Arquivo de log



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Espaço de trabalho privado

A transação não altera diretamente os dados reais.

Ela trabalha sobre cópias ou versões temporárias.

```
Dados reais
    ↓ cópia
Espaço privado da transação
```

Se houver **commit**: alterações são aplicadas

Se houver **abort**: alterações são descartadas

Arquivo de Log = Lista Ordenada de Anotações

O arquivo de log registra:

- número único de **sequência do log**;
- identificador único da **transação**;
- marcação de início da transação, com horário de início e indicação se a transação é somente leitura ou leitura/escrita;
- identificadores dos registros do banco de dados envolvidos na transação e das operações aplicadas a eles;
- imagens anteriores de todos os registros que participaram da transação;
- imagens posteriores de todos os registros alterados pela transação;
- estado atual da transação: ativa, confirmada ou abortada.

Arquivo de Log

- O arquivo de log também pode conter **checkpoints**.
- Checkpoints são momentos em que atualizações armazenadas em buffer por transações ativas são gravadas em disco de uma só vez.
- Estratégia Write-Ahead Logging:
  - todas as atualizações são registradas no log antes de serem gravadas em disco;
  - imagens anteriores são sempre registradas no log antes que os valores reais sejam sobrescritos nos arquivos físicos do banco de dados.

Write-Ahead Logging (WAL)

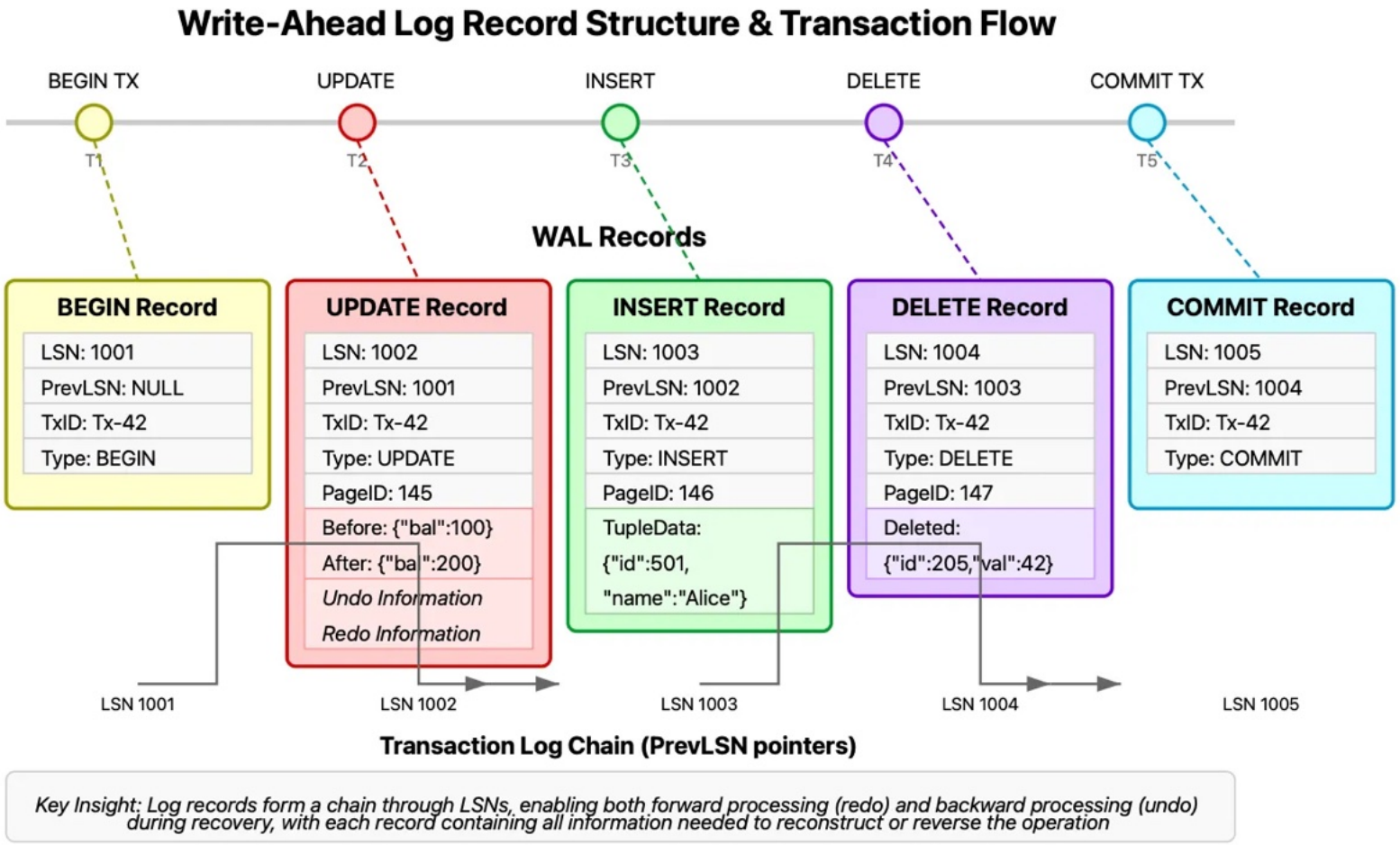
1. Força o log a ser gravado **antes** da alteração ser aplicada no banco.
2. Força **todas** as anotações de log de uma transação **antes** do **commit**.

Isso permite recuperação após falhas:

- **UNDO**: Desfaz alterações de transações abortadas ou incompletas.
- **REDO**: Refaz alterações de transações confirmadas, mas ainda não gravadas completamente.

> Na imagem: C. Mohan, criador do ARIES, um algoritmo popular de WAL, utilizado no IBM DB2 e no Microsoft SQL Server. Fonte da Imagem: https://www.comp.hkbu.edu.hk/v1/?page=news&id=595.

Write-Ahead Logging (WAL)



Fonte da Imagem: System Design Interview Roadmap.

Conclusão

- Transações são essenciais para manter a confiabilidade de sistemas que manipulam dados compartilhados.
- Em ambientes concorrentes, não basta executar operações corretamente de forma individual; é necessário garantir que a combinação das operações também produza um estado consistente.
- O gerenciamento de transações permite proteger o sistema contra falhas, inconsistências e interferências entre usuários ou aplicações.
- As propriedades ACID fornecem uma base conceitual para compreender o comportamento esperado de transações em bancos de dados e sistemas distribuídos.

Material Adicional

- Concurrency Control Theory (CMU Intro to Database Systems) by Andy Pavlo
- Intro to Write-Ahead Logging by Joe Hellerstein (UC Berkley)
- Write-Ahead Logs. The secret to fast database queries.

Dúvidas e Discussão

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Questões

Questão 1

A propriedade de consistência é frequentemente descrita como a garantia de que uma transação leva o sistema de um estado consistente para outro estado consistente. Discuta por que essa propriedade não depende apenas do SGBD, mas também da lógica da aplicação.

Questão 2

Explique o que poderia acontecer se um sistema implementasse apenas atomicidade e durabilidade, mas não garantisse isolamento adequado entre transações concorrentes. Use o exemplo do estoque hospitalar ou outro exemplo realista.

Questão 3

Explique por que transações aninhadas podem ser úteis em sistemas distribuídos. Em seguida, discuta o que acontece quando uma subtransação é confirmada provisoriamente, mas a transação de nível superior é posteriormente abortada.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br