# Banco de Dados

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

## Transaction / Transação

**Sistemas monousuário *versus* multiusuário**

1. **Monousuário**: no máximo um usuário pode acessar o sistema por vez. Restritos a sistemas de computador pessoal.
2. **Multiusuário**: muitos usuários acessam o sistema simultaneamente. Grande maioria dos sistemas atuais.

## Transaction / Transação

### Sistemas monousuário *versus* multiusuário

| MONOUSUÁRIO | MULTIUSUÁRIO |
|---|---|
| • Sistemas Operacionais;<br>• Editor de texto, planilhas eletrônicas;<br>• Antivírus;<br>• Games offline; etc | • Sistema de:<br>Reservas Aéreas;<br>Bancos;<br>Supermercados;<br>Redes Sociais; etc…<br>• Games online |

## Transaction / Transação

**ACID**

Representa quatro princípios básicos de qualquer SGBD:

**A** = Atomicidade: tudo ou nada

**C** = Consistência: coerência nas operações

**I** = Isolamento: um "processo" por vez

**D** = Durabilidade: garantir que a operação foi feita

## Transaction / Transação

**Transação**

- É um programa em execução que forma uma unidade lógica de processamento de banco de dados.
- Inclui uma ou mais operações de acesso ao banco de dados – SELECT, UPDATE, INSERT, DELETE.
- Muito utilizada quando se executa operações críticas no BD.
- Também pode ser especificada através de PLSQL ou de uma Linguagem de Programação de alto nível (Java, PHP, C#, etc).

## Transaction / Transação

**Há dois tipos de transações:**

1. **Transação somente de leitura**: quando as operações realizadas não atualizarem o banco de dados, mas apenas recuperarem dados (select);
2. **Transação de leitura-gravação**: o contrário da "somente de leitura". Ocorre quando há atualização/ inserção/remoção de dados.

## Transaction / Transação

Criação (com SQL):

**begin transaction;** ou somente **begin**;

/*instruções a serem executadas pela transação*/

**end transaction;** ou somente **end**;

OBS1: É possível aninhar transações, ou seja, colocar uma transação dentro da outra.

## Transaction / Transação

Após a submissão de uma transação, o SGBD é responsável por garantir que TODAS as operações na transação sejam concluídas com sucesso e seu efeito seja registrado permanentemente no BD.

O SGBD **não** deve permitir que somente algumas operações de uma transação sejam aplicadas ao BD e outras não. **(ATOMICIDADE)**

**"TUDO OU NADA"**

## Transaction / Transação

É necessário sempre fechar a transação no final do processamento;

Caso uma transação fique aberta, as tabelas ficam “presas” pela transação que não foi finalizada **(ISOLAMENTO)**

| COMMIT | ROLLBACK |
|---|---|
| Comando para **confirmar** a execução da transação. | Comando para **cancelar** a execução da transação. |

OBS: Ambos os comandos também fecham a transação.

## Transaction / Transação

Exemplo:

```
begin;
   insert into compra (produto, valor)
values (‘SSD 256GB’, 390.90);
   update estoque set qtd = qtd - 1
where cod_produto = 20;
commit;
```

Início da transação

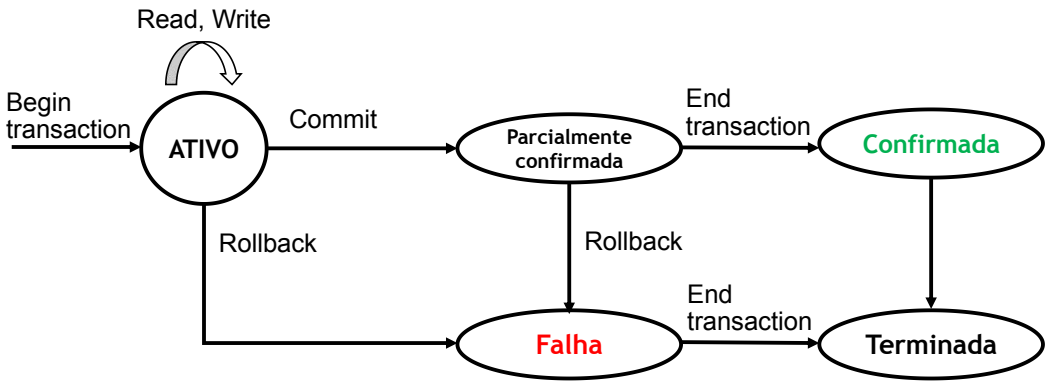
Se falhar, volta ao que era antes

Se falhar, volta ao que era antes

Se nada falhar, executa as mudanças no BD

# Transaction / Transação

## Estados de Transação




# Transaction / Transação

## Controle de Concorrência

O problema da atualização perdida:

Ocorre quando duas transações que acessam os mesmos itens do BD têm suas operações **intercaladas** de modo que isso torna o valor de alguns itens incorreto.

*operações intercaladas

| **T1:** | **T2:** |
|---|---|
| X = 80 | |
| X = 80 – 5 | |
| | X = 80 |
| | X = 80 + 4 |
| X = 75 <updated> | |
| | X = 84 <updated> |

# Transaction / Transação

**Controle de Concorrência**

O problema da atualização temporária (ou leitura suja):

Ocorre quando uma transação atualiza um item do BD e depois a transação falha. Nesse meio tempo, o item atualizado é lido por outra transação, antes de ser alterado de volta para seu valor original.

| T1: | T2: |
|---|---|
| X = 10 | |
| X = 10 – 3 | |
| X = 7 | |
| <falha> | |
| LEITURA SUJA | X = 7 |
| | X = 7 + 2 |
| | X = 9 |

# Transaction / Transação

**Controle de Concorrência**

O problema do resumo incorreto:

Se uma transação estiver calculando valores quantitativos através de **funções de agregação**, enquanto outras transações estão atualizando alguns registros, então a função de agregação pode calcular alguns valores antes que eles sejam atualizados e outros depois da atualização.

| T1: | T2: |
|---|---|
| A = 5; | |
| A = 5 – 2; | |
| | sum(col_A) |
| A = 3 <updated> | |

## Transaction / Transação

**Controle de Concorrência**

O problema da leitura não repetitiva:

Uma transação lê o mesmo item duas vezes e o item é alterado por outra transação entre as duas leituras. Logo, a transação inicial receberia valores diferentes para suas duas leituras do mesmo item.

**Exemplo:**

Reserva aérea:

1) Cliente consulta a disponibilidade de assentos em vários voos;
2) Cliente demora para decidir sobre um voo em particular;
3) Quando clica para comprar, não há mais vagas naquele voo.

## Transaction / Transação

**Assistir o vídeo que resume o conteúdo de ACID:**

**youtube.com/watch?v=NtOBPtInK8w**

## Transaction / Transação

**Exemplo beeem simples (didático):**

- Criar um BD qualquer;
- Criar a tabela T apenas com uma coluna C do tipo INT;
- Abrir uma transação;
- Inserir o número 1 na coluna C da tabela T;
- Selecionar todos os dados da tabela T;
- Executar um rollback;
- Selecionar todos os dados da tabela T novamente;
- Analisar o resultado!

## Transaction / Transação

**Exemplo beeem simples (didático):**

- Repetir os passos anteriores, a partir da abertura da transação, mas usando duas conexões distintas;
- MySQL Workbench e PHPMyAdmin;
- Analisar o resultado!