# Transações

Prof. Márcio Bueno

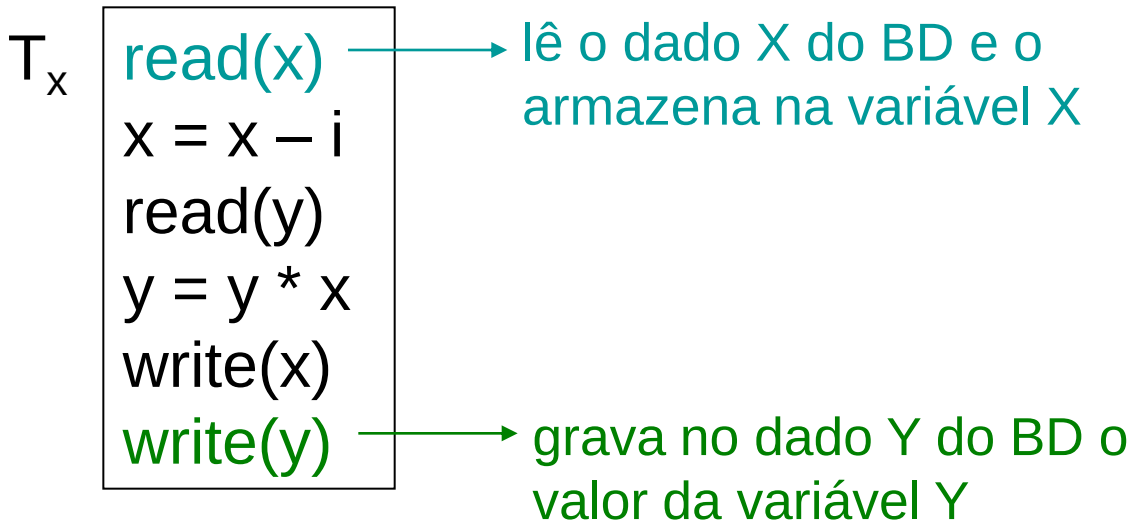
{bd2tarde,bd2noited}@marciobueno.com

# Introdução a Transações

- SGBD
  - sistema de processamento de operações de acesso ao BD
- SGBDs são em geral multi-usuários
  - processam simultaneamente operações disparadas por vários usuários
    - deseja-se alta disponibilidade e tempo de resposta pequeno
  - execução intercalada de conjuntos de operações
    - exemplo: enquanto um processo *i* faz I/O, outro processo *j* é selecionado para execução
- Operações são chamadas transações

# Transação

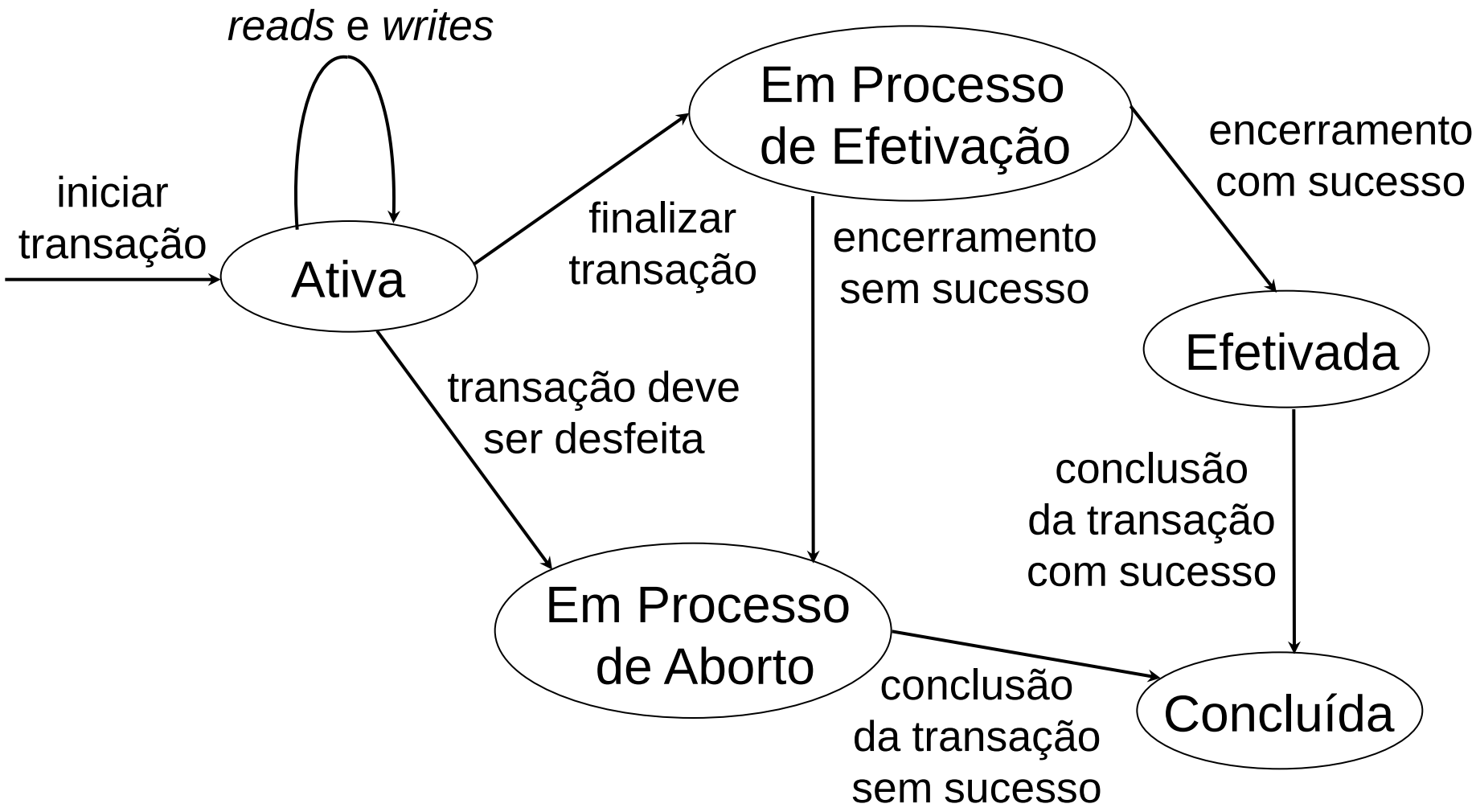
- Unidade lógica de processamento em um SGBD
- Composta de uma ou mais operações
  - seus limites podem ser determinados em SQL
- De forma abstrata e simplificada, uma transação pode ser encarada como um conjunto de operações de leitura e escrita de dados



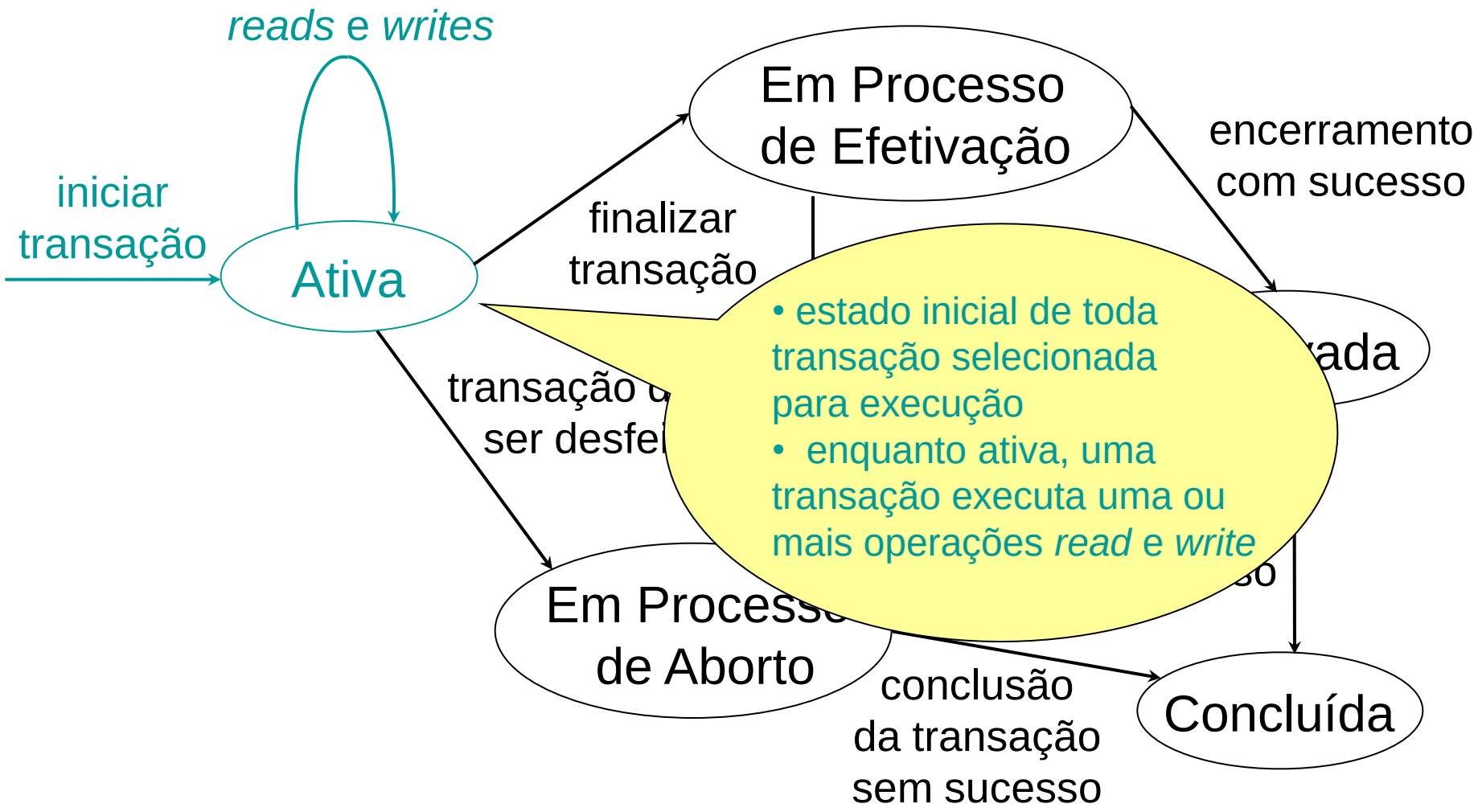
# Estados de uma Transação

- Uma transação é sempre monitorada pelo SGBD quanto ao seu estado
  - que operações já fez? concluiu suas operações? deve abortar?
- Estados de uma transação
  - Ativa, Em processo de efetivação, Efetivada, Em processo de aborto, Concluída
  - Respeita um Grafo de Transição de Estados

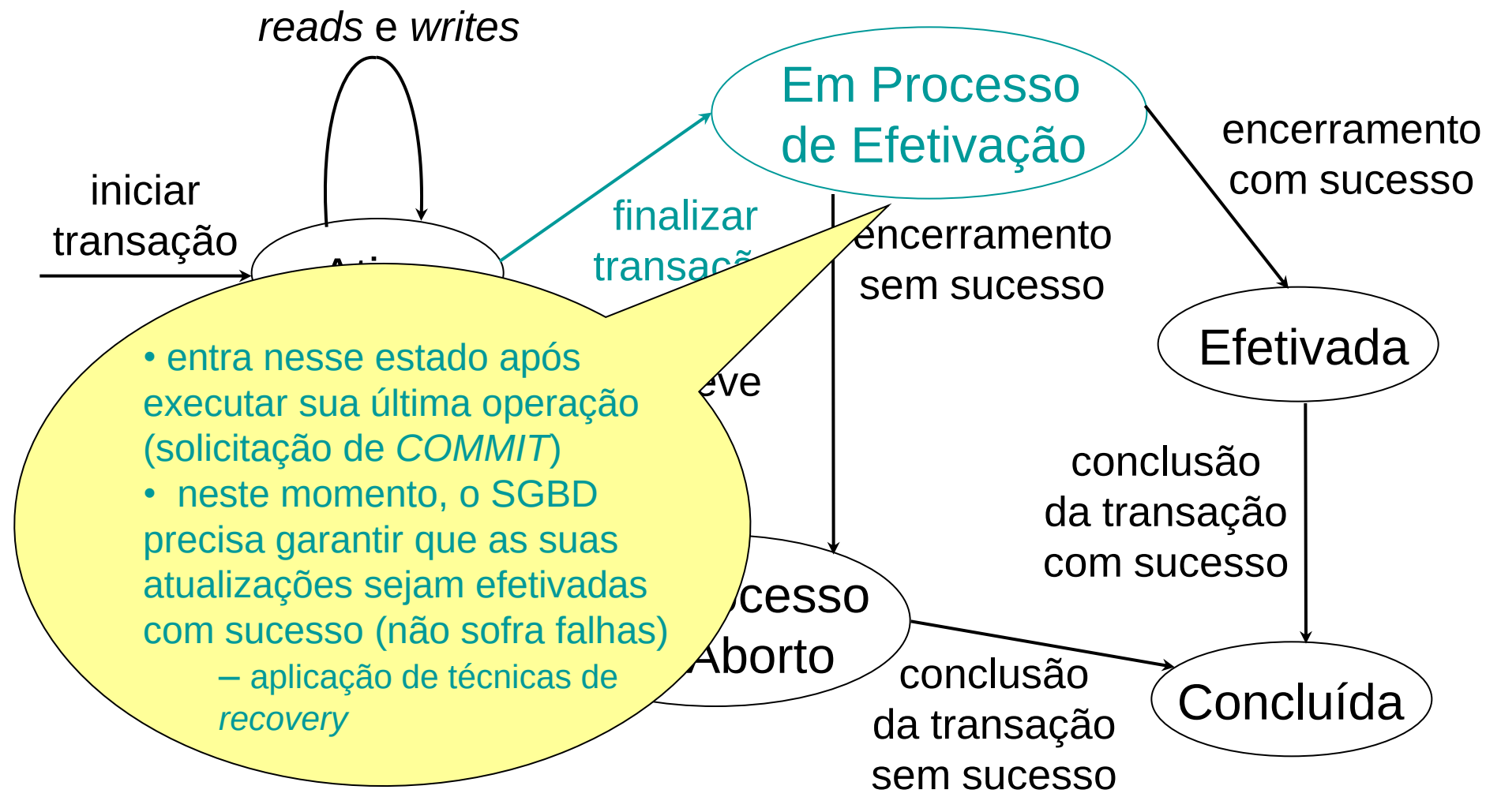
# Transição de Estados de uma Transação



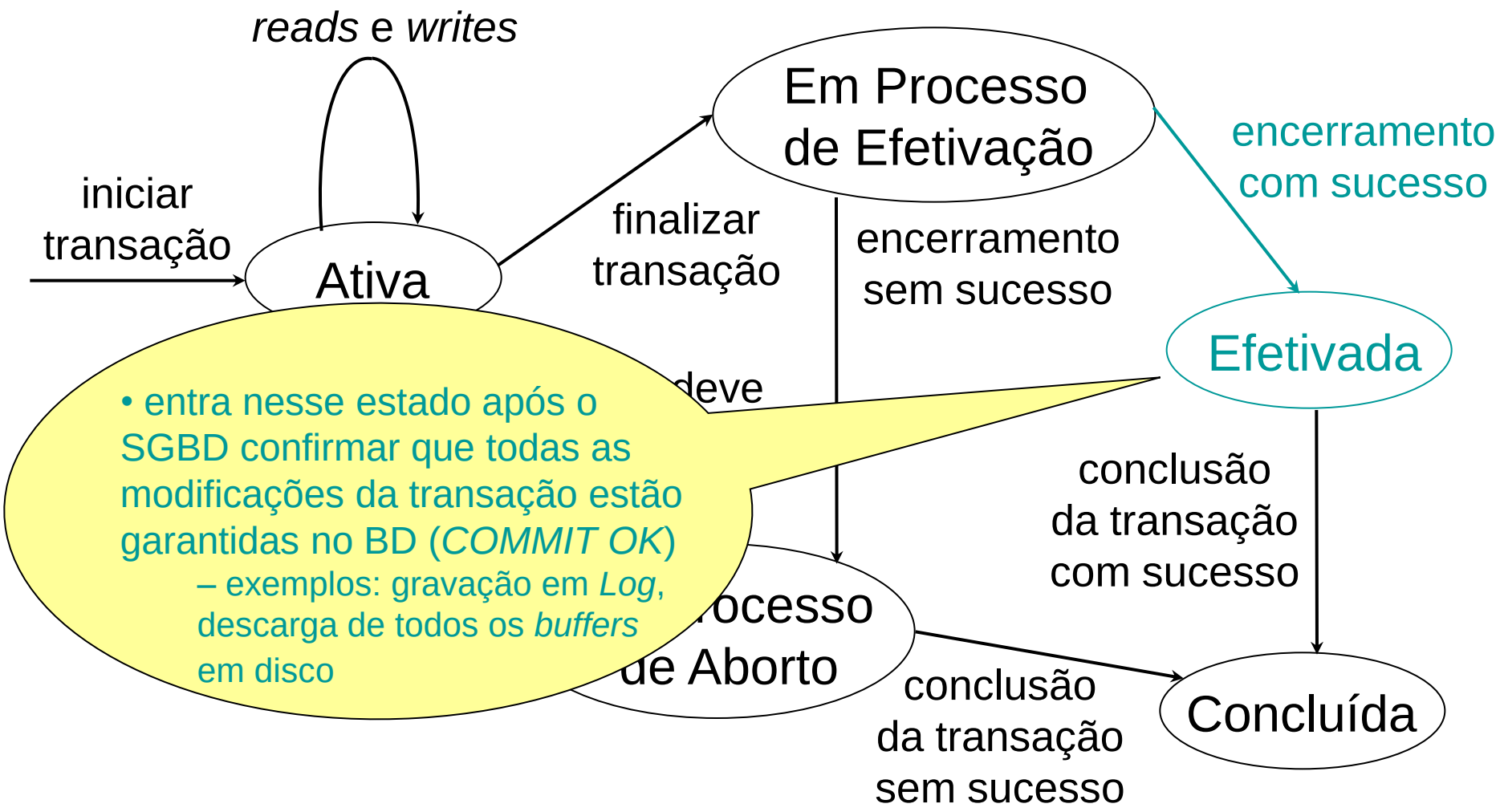
# Transição de Estados de uma Transação



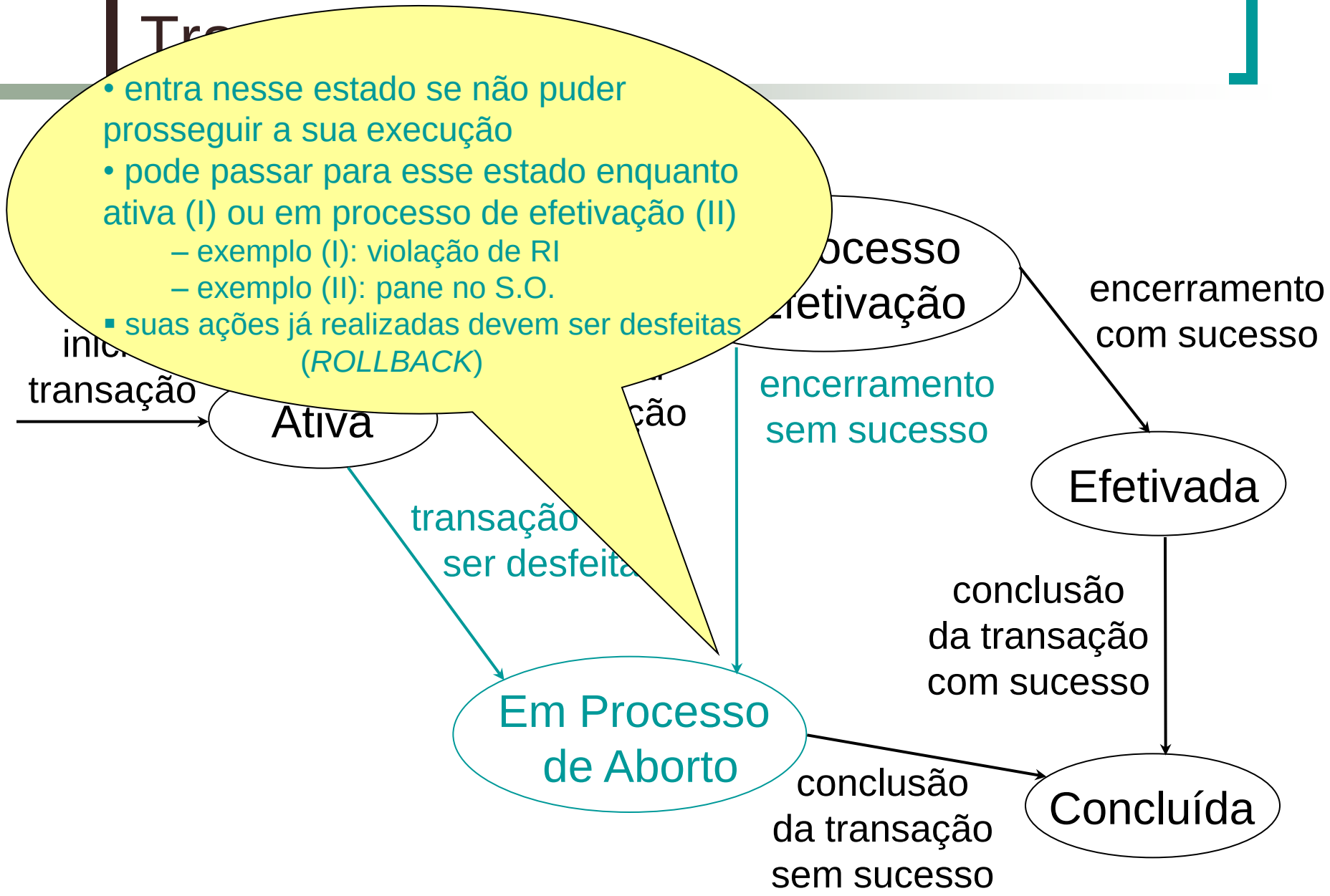
# Transição de Estados de uma Transação



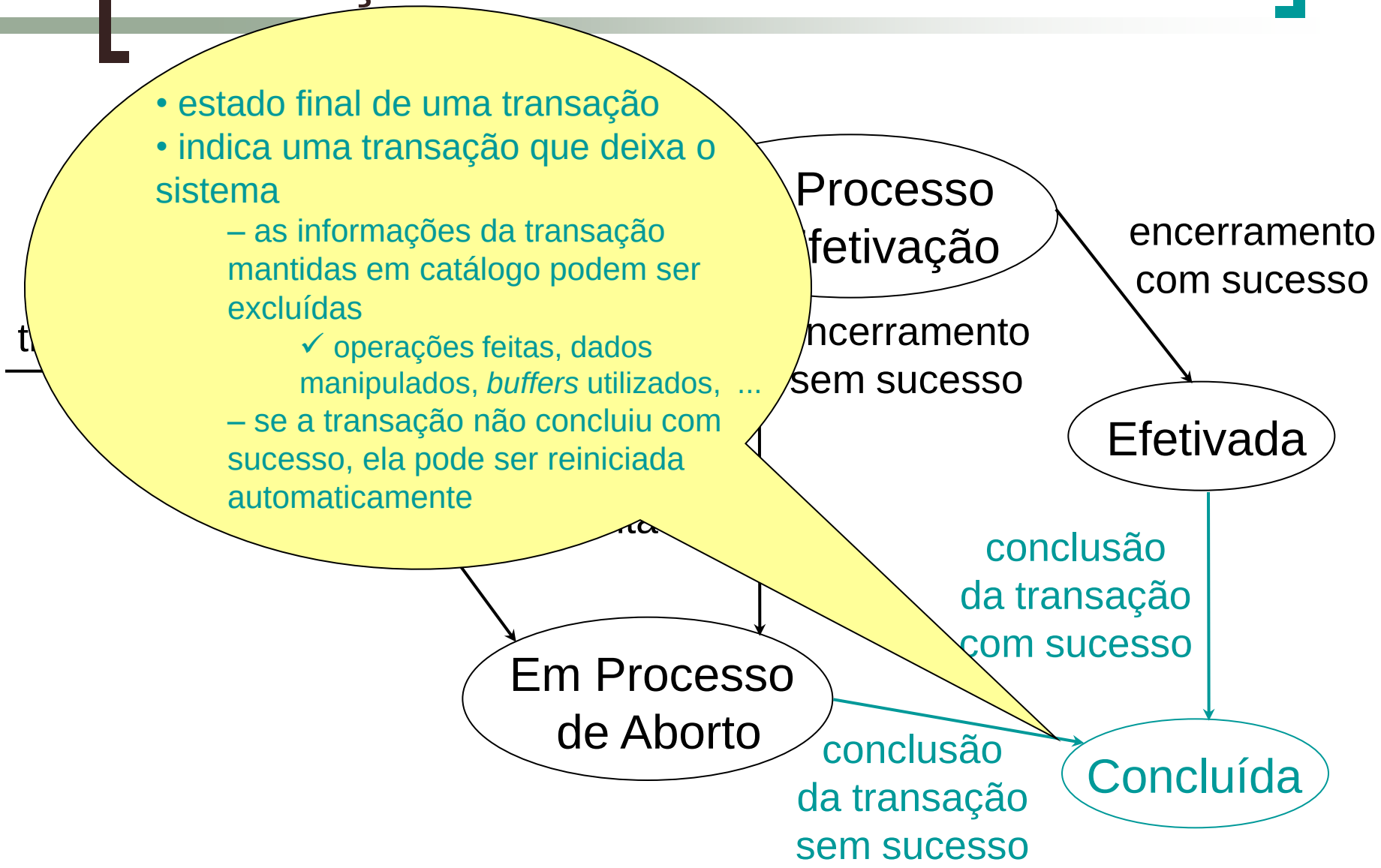
# Transição de Estados de uma Transação



# Transição de Estados de uma Tr



# Transição de Estados de uma Transação



# Propriedades de uma Transação

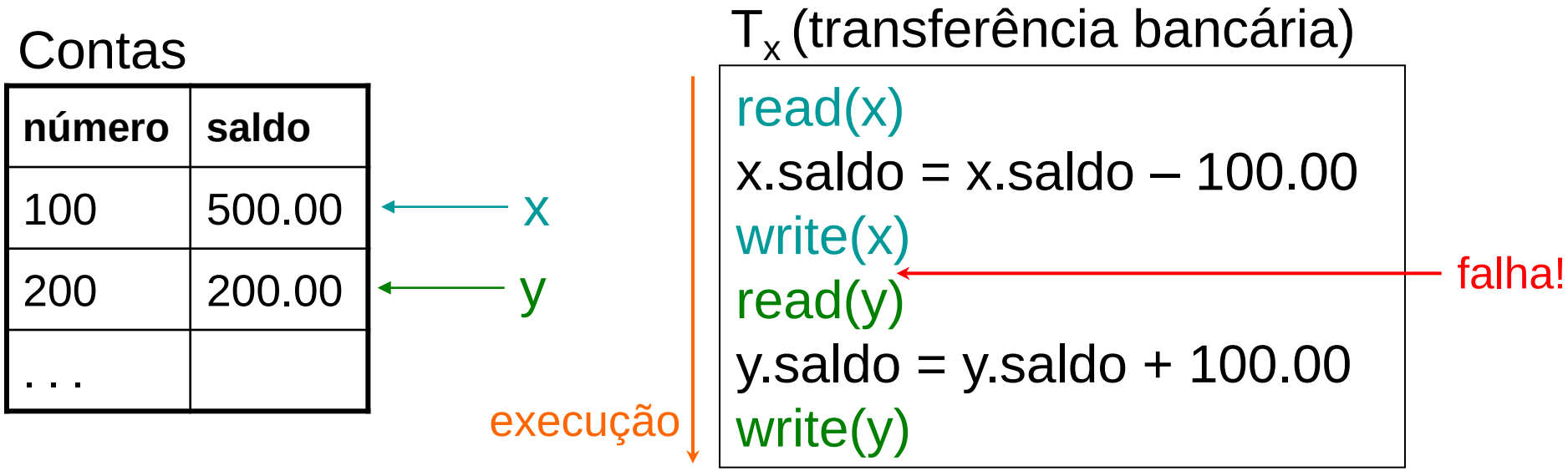
- Requisitos que sempre devem ser atendidos por uma transação
- Chamadas de propriedades ACID
  - Atomicidade
  - Consistência
  - Isolamento
  - Durabilidade ou Persistência

# Atomicidade

- Princípio do *"Tudo ou Nada"*
  - ou todas as operações da transação são efetivadas com sucesso no BD ou nenhuma delas se efetiva
    - preservar a integridade do BD
- Responsabilidade do subsistema de recuperação contra falhas (subsistema de *recovery*) do SGBD
  - desfazer as ações de transações parcialmente executadas

# Atomicidade

- Deve ser garantida, pois uma transação pode manter o BD em um estado inconsistente durante a sua execução



# Propriedade: Atomicidade

- Considere as seguintes tabelas
  - EMPREGADO(EmpNo, Nome, DepNo, ...)
  - DEPARTAMENTO(DepNo, Nome, ... )

# Unidade Lógica de Trabalho

1) A Transação $T_1$ começa

   BEGIN TRANSACTION

2) Atualiza o código do departamento

   UPDATE Departamento SET DepNo = ‘INF’
   WHERE DepNo = ‘DEI’

3) Atualiza os empregados

   UPDATE Empregado SET Depto = ‘INF’
   WHERE DepNo = ‘DEI’

4) Efetiva

   COMMIT TRANSACTION

# Exemplo 1

1) A Transação T1 começa

   BEGIN TRANSACTION

2) Atualiza o código do departamento

   UPDATE Departamento SET DepNo = 'INF'
   WHERE DepNo = 'DEI'

3) Atualiza os empregados

   UPDATE Empregado SET Depto = 'INF'
   WHERE DepNo = 'DEI'

4) FALHA: (ROLLBACK ou ignora)

➔ os efeitos da transação não devem ser gravados em disco

# Exemplo 2

1) A Transação T1 começa

   BEGIN TRANSACTION

2) Atualiza o código do departamento

   UPDATE Departamento SET DepNo = 'INF'
   WHERE DepNo = 'DEI'

3) Atualiza os empregados

   UPDATE Empregado SET Depto = 'INF'

4) Efetiva

   COMMIT TRANSACTION

➔ Todos os efeitos da transação devem ser gravados em disco

# Consistência

- Uma transação sempre conduz o BD de um estado consistente para outro estado também consistente
- Responsabilidade conjunta do
  - DBA
    - definir todas as RIs para garantir estados e transições de estado válidos para os dados
      - exemplos: salário > 0; salário novo > salário antigo
  - subsistema de *recovery*
    - desfazer as ações da transação que violou a integridade

# Consistência

- Transação 1:

  READ (A,A1)
  A1= A1 – 50
  WRITE(A,A1)
  READ (B,B1)
  B1= B1 + 50
  WRITE(B,B1)

- A restrição de consistência é que a soma de A e B é imutável pela execução da transação

# Isolamento

- No contexto de um conjunto de transações concorrentes, a execução de uma transação $T_x$ deve funcionar como se $T_x$ executasse de forma isolada
  - $T_x$ não deve sofrer interferências de outras transações executando concorrentemente
- Responsabilidade do subsistema de controle de concorrência (*scheduler*) do SGBD
  - garantir escalonamentos sem interferências

# Isolamento

| $T_1$ | $T_2$ |
|---|---|
| read(A) | |
| A = A – 50 | |
| write(A) | |
| | read(A) |
| | A = A+A*0.1 |
| | write(A) |
| read(B) | |
| B = B + 50 | |
| write(B) | |
| | read(B) |
| | B = B - A |
| | write(B) |

escalonamento válido

| $T_1$ | $T_2$ |
|---|---|
| read(A) | |
| A = A – 50 | |
| | read(A) |
| | A = A+A*0.1 |
| | write(A) |
| | read(B) |
| write(A) ← | |
| read(B) | |
| B = B + 50 | |
| write(B) | |
| | B = B - A |
| | write(B) ← |

$T_1$ interfere em $T_2$

$T_2$ interfere em $T_1$

escalonamento inválido

# Problema 1: Leitura de dados Inválidos

- Transação T1 começa

…

2) BEGIN TRANSACTION
3) UPDATE Departamento SET DepNo = ‘INF’ WHERE DepNo = ‘DEI’
4) UPDATE Empregado SET Depto = ‘INF’
5) COMMIT TRANSACTION

- Transação T2 começa

1) SELECT DepNo FROM Departamento WHERE Nome = ‘Informática’

(Encontra DepNo = ‘DEI’)

6) SELECT * FROM Empregado WHERE Depto = ‘DEI’

(Não encontra empregados do Depto = ‘DEI’)

***$T_2$ teve uma visão irreal do BD***

# Problema 2: Leitura Não Repetida

- Transação T1 começa

…

2) BEGIN TRANSACTION
3) UPDATE Departamento
   SET DepNo = 'INF'
   WHERE DepNo = 'DEI'
4) UPDATE Em'1pregado
   SET Depto = 'INF'
   WHERE DepNo = 'DEI'
5) COMMIT TRANSACTION

- Transação T2 começa

1) SELECT DepNo
   FROM Departamento
   WHERE Nome = 'Informática'
   (Encontra DepNo = DEI')
6) SELECT DepNo
   FROM Departamento
   WHERE Nome = 'Informática'
   (Encontra DepNo = INF')

**$T_2$ teve uma visão esquisita do BD**

# Problema 3: Atualizações Perdidas

- Transação T3
- Atualiza Conta #1500
  - 1) Lê conta #1500
  - 3) SaldoTmp += 100
  - 5) Grava Conta no BD

- Transação T4
- Atualiza Conta #1500
  - 2) Lê conta #1500
  - 4) SaldoTmp += 200
  - 6) Grava Conta no BD

***A atualização de T4 foi perdida***

# Problema 4: Transação Lê outra que Falha

- Transação T1

...

1) BEGIN TRANSACTION
2) UPDATE Departamento
   SET DepNo = 'INF'
   WHERE DepNo = 'DEI'
3) UPDATE Empregado
   SET Depto = 'INF'
6) FALHA

- Transação T2

4) Lê DepNo = 'INF'
5) $T_2$ termina (COMMIT TRANSACTION)

***Execução não - Recuperável : $T_2$ leu de $T_1$ e $T_1$ "abortou"***

# Durabilidade ou Persistência

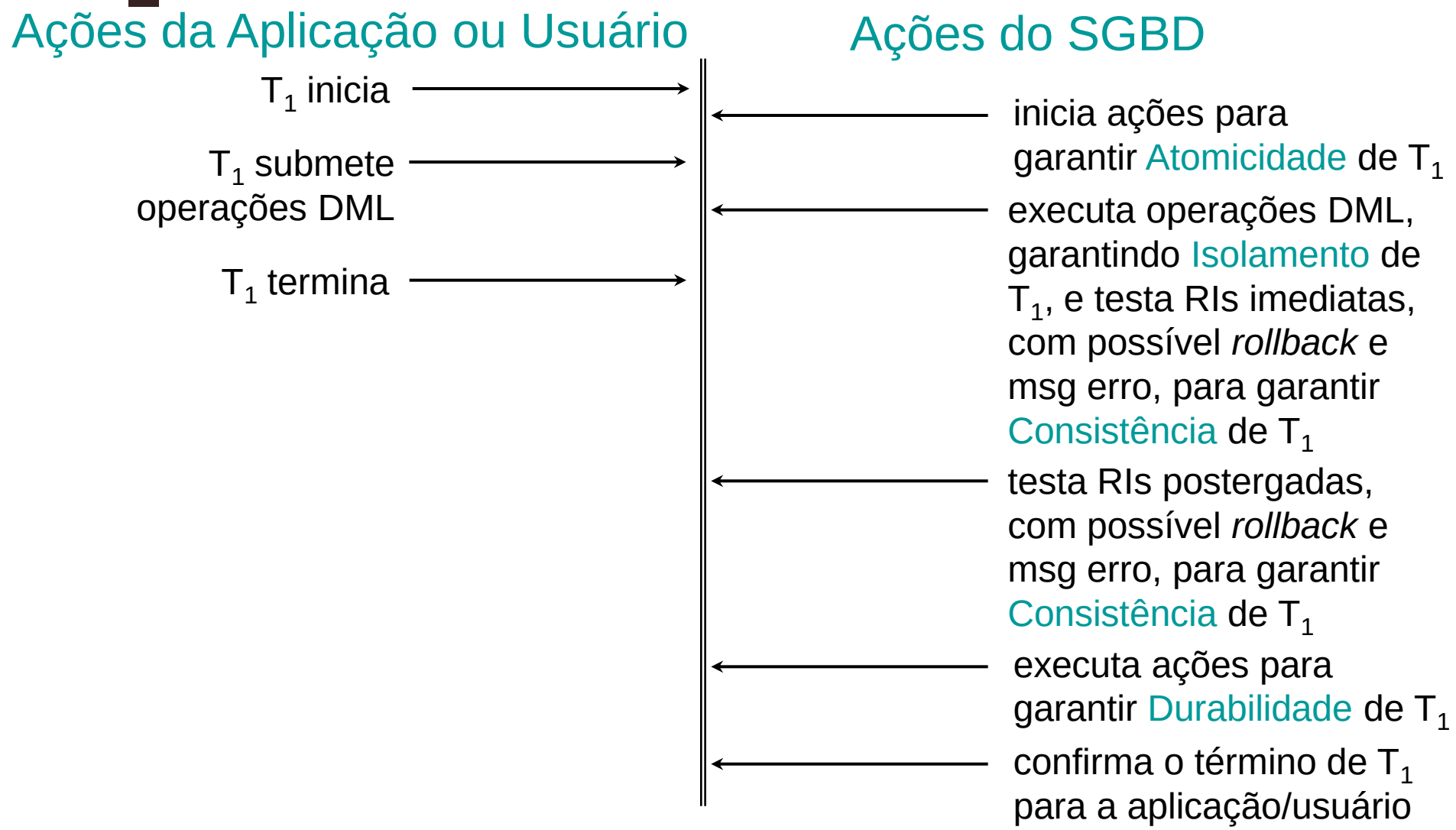
- Deve-se garantir que as modificações realizadas por uma transação que concluiu com sucesso persistam no BD
  - nenhuma falha posterior ocorrida no BD deve perder essas modificações
- Responsabilidade do subsistema de *recovery*
  - refazer transações que executaram com sucesso em caso de falha no BD

# Propriedade: Durabilidade

1) BEGIN TRANSACTION
2) ...
3) ...
4) ...
5) COMMIT (Até este momento os efeitos de T1 não foram gravados no BD)

Falha ➔ Na re-iniciação todos os efeitos de T1 são gravados no BD (Valores encontrados no LOG (Durabilidade)

# Gerência Básica de Transações



# Transações em SQL

- Por *default*, todo comando individual é considerado uma transação
  - exemplo: `DELETE FROM Pacientes`
    - exclui todas ou não exclui nenhuma tupla de pacientes, deve manter o BD consistente, etc
- SQL Padrão (SQL-92)
  - `SET TRANSACTION`
    - inicia e configura características de uma transação
  - `COMMIT [WORK]`
    - encerra a transação (solicita efetivação das suas ações)
  - `ROLLBACK [WORK]`
    - solicita que as ações da transação sejam desfeitas

# Transações em SQL

- Principais configurações (`SET TRANSACTION`)
  - modo de acesso
    - `READ` (somente leitura), `WRITE` (somente atualização) ou `READ WRITE` (ambos - *default*)
  - nível de isolamento
    - indicado pela cláusula `ISOLATION LEVEL` *nível*
    - *nível* para uma transação $T_i$ pode assumir
      - `SERIALIZABLE` ($T_i$ executa com completo isolamento - *default*)
      - `REPEATABLE READ` ($T_i$ só lê dados efetivados e outras transações não podem escrever em dados lidos por $T_i$) – pode ocorrer que $T_i$ só consiga ler alguns dados que deseja
      - `READ COMMITTED` ($T_i$ só lê dados efetivados, mas outras transações podem escrever em dados lidos por $T_i$)
      - `READ UNCOMMITTED` ($T_i$ pode ler dados que ainda não sofreram efetivação)

# Transações em SQL

- Exemplo

```
EXEC SQL SET TRANSACTION
         WRITE
         ISOLATION LEVEL SERIALIZABLE;
...
for(;;)
{...
EXEC SQL INSERT INTO Empregados
           VALUES (:ID,:nome,:salario)
...
EXEC SQL UPDATE Empregados
         SET salário = salário + 100.00
         WHERE ID = :cod_emp
if (SQLCA.SQLCODE <= 0) EXEC SQL ROLLBACK;
...
}
EXEC SQL COMMIT;
...
```

# Variações do SQL Padrão

- Adotado por alguns SGBDs

```
BEGIN TRANSACTION T1

UPDATE Medicos
SET nroa = NULL
WHERE nroa = @nroAmb

IF @@ERROR <> 0 ROLLBACK TRANSACTION T1

DELETE FROM Ambulatorios
WHERE nroa = @nroAmb

IF @@ERROR <> 0 ROLLBACK TRANSACTION T1
ELSE COMMIT TRANSACTION T1
...
```