

Sumário

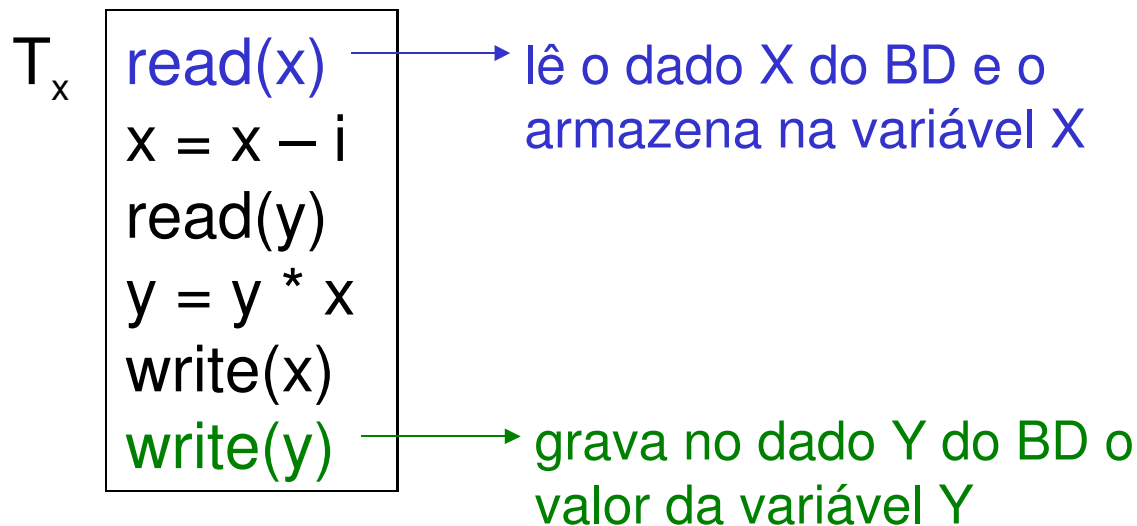
- 1 Introdução ao Processamento de Consultas
- 2 Otimização de Consultas
- 3 Plano de Execução de Consultas
- 4 Introdução a Transações**
- 5 Recuperação de Falhas
- 6 Controle de Concorrência
- 7 Fundamentos de BDs Distribuídos
- 8 SQL Embutida

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
- Conjunto de operações é chamado **transação**

Transação

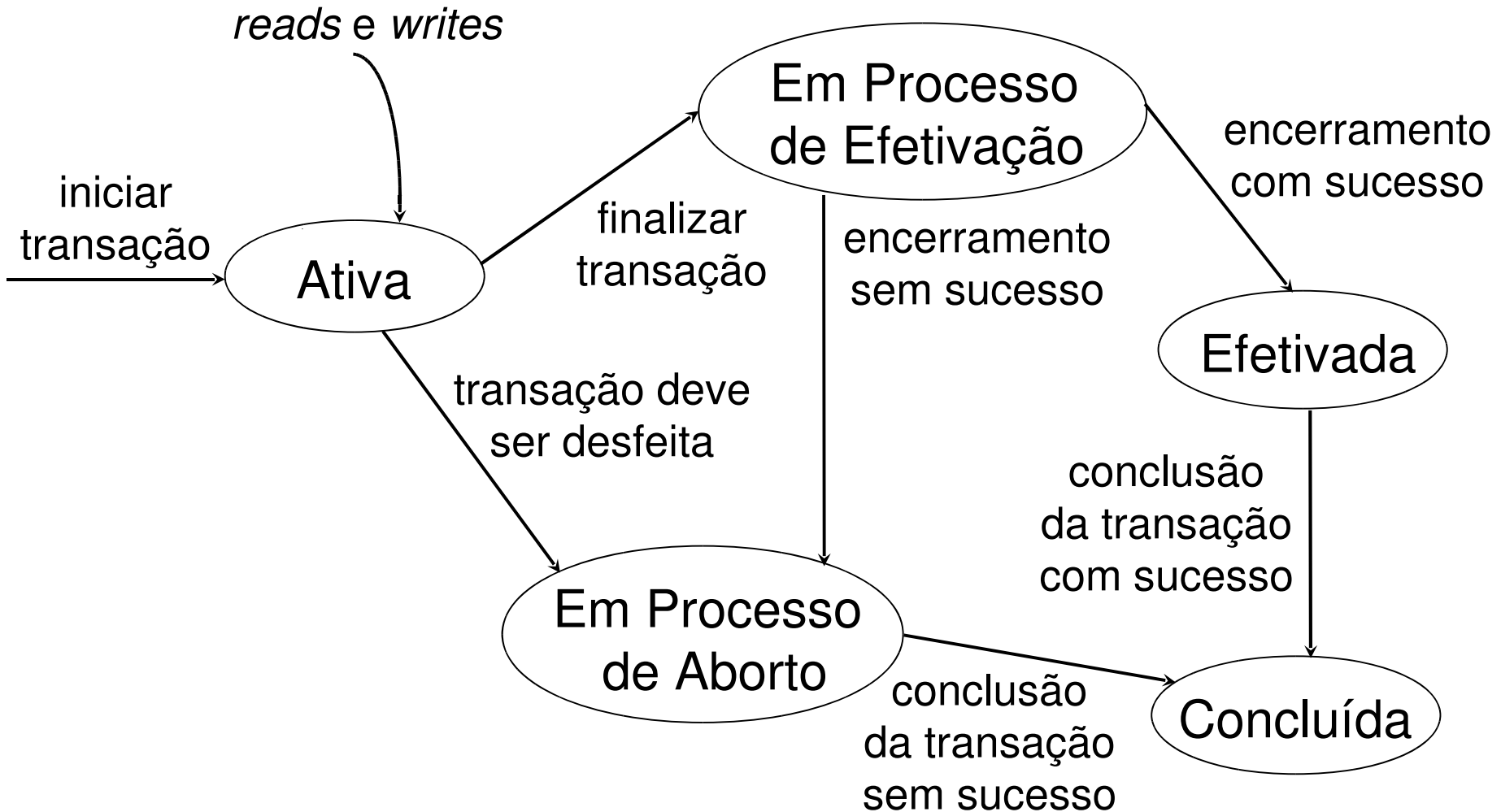
- Unidade lógica de processamento em um SGBD
- Composta de uma ou mais operações
 - seus limites podem ser determinados em SQL
- De forma abstrata e simplificada, uma transação pode ser encarada como um conjunto de operações de leitura e escrita de dados



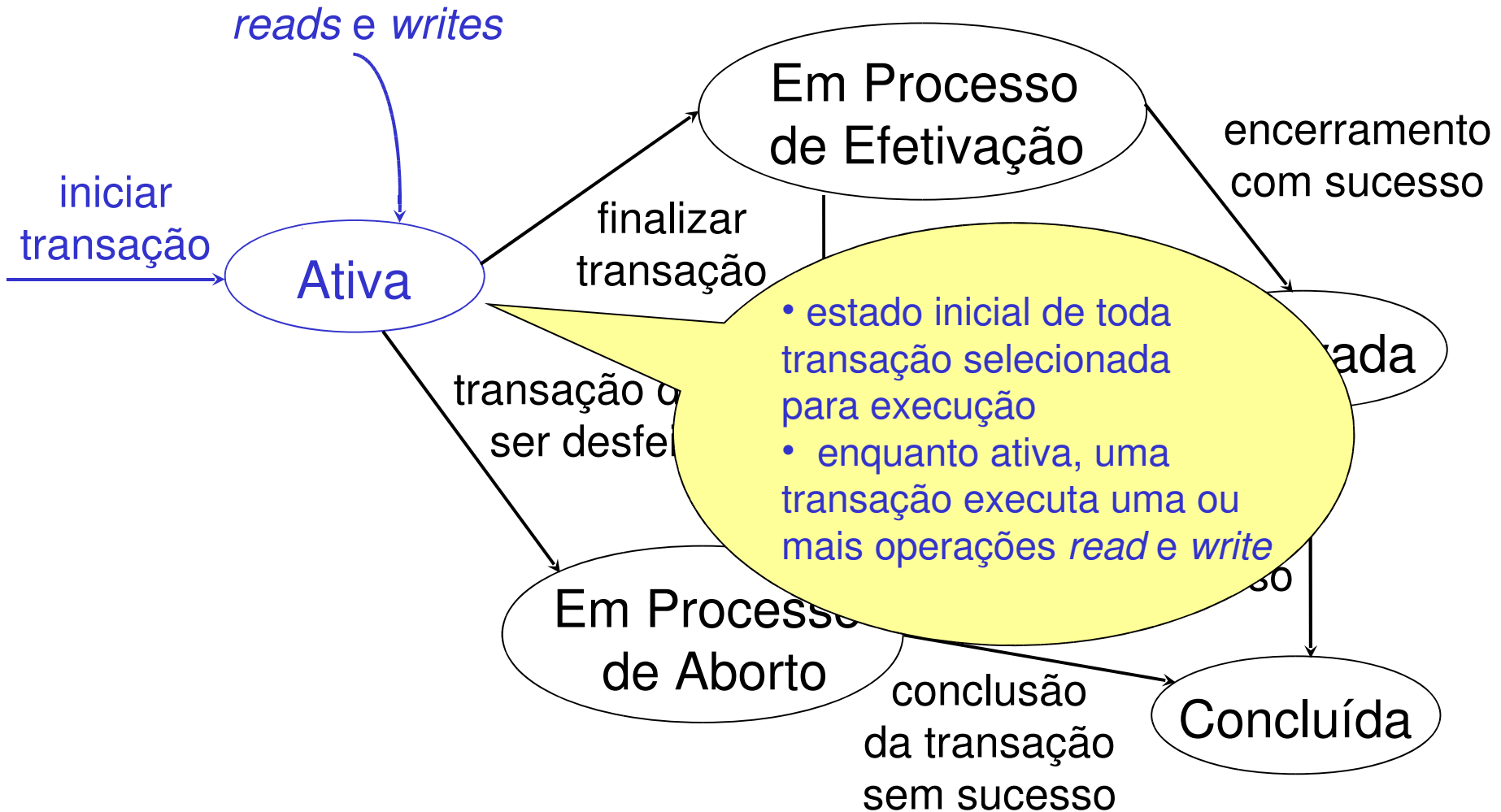
Estados de uma Transação

- 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

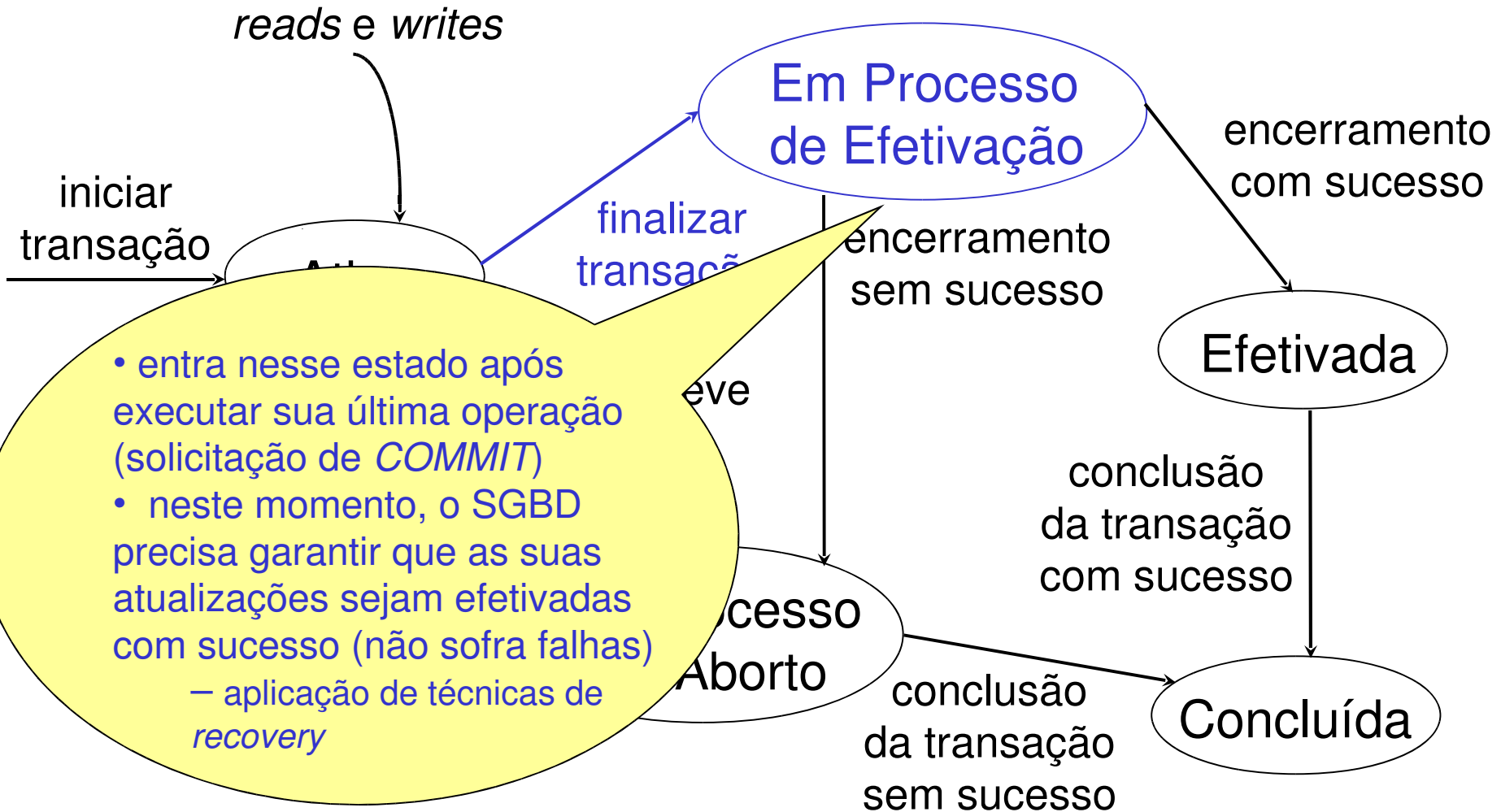
Transição de Estados de uma Transação



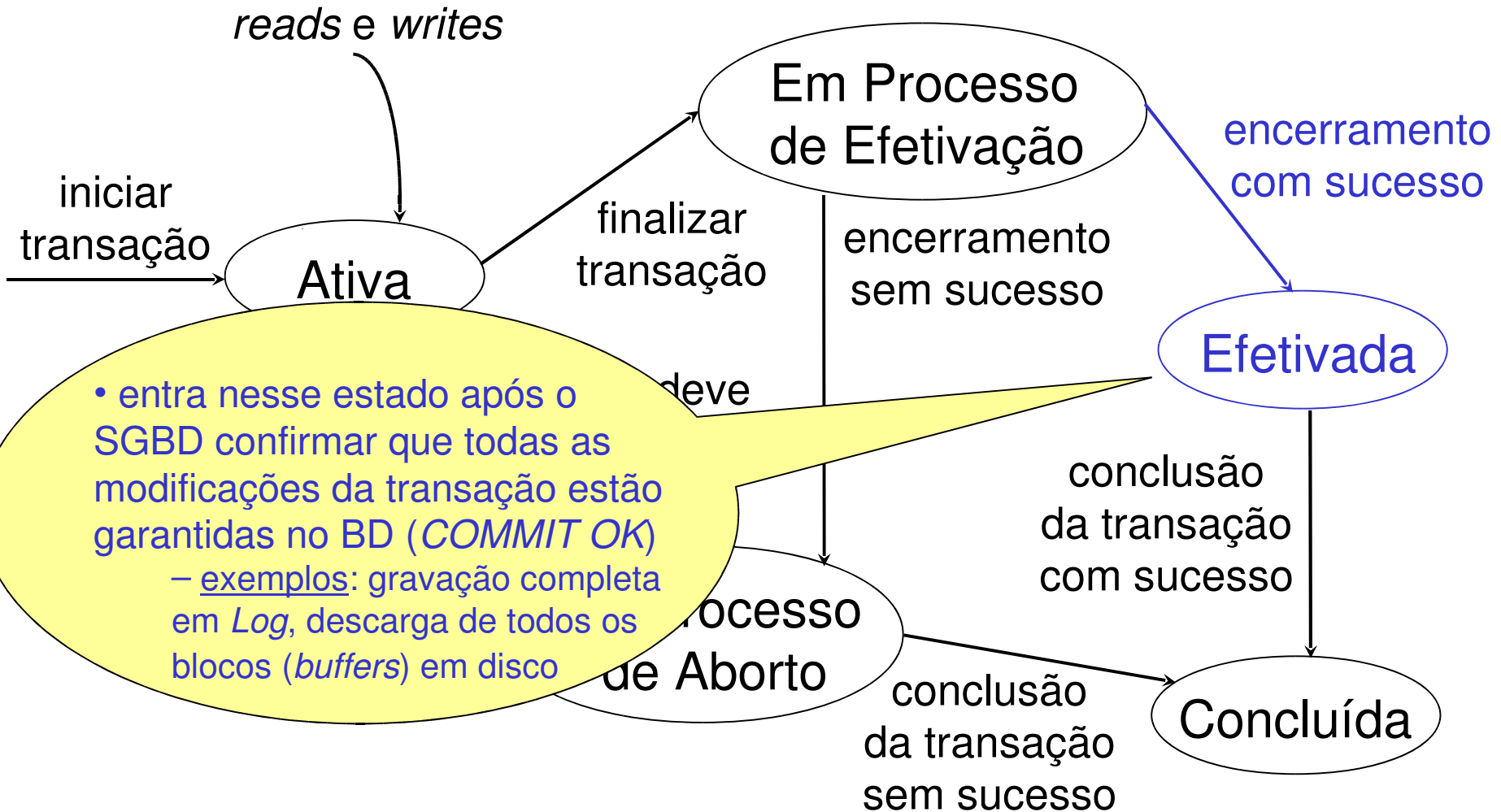
Transição de Estados de uma Transação



Transição de Estados de uma Transação

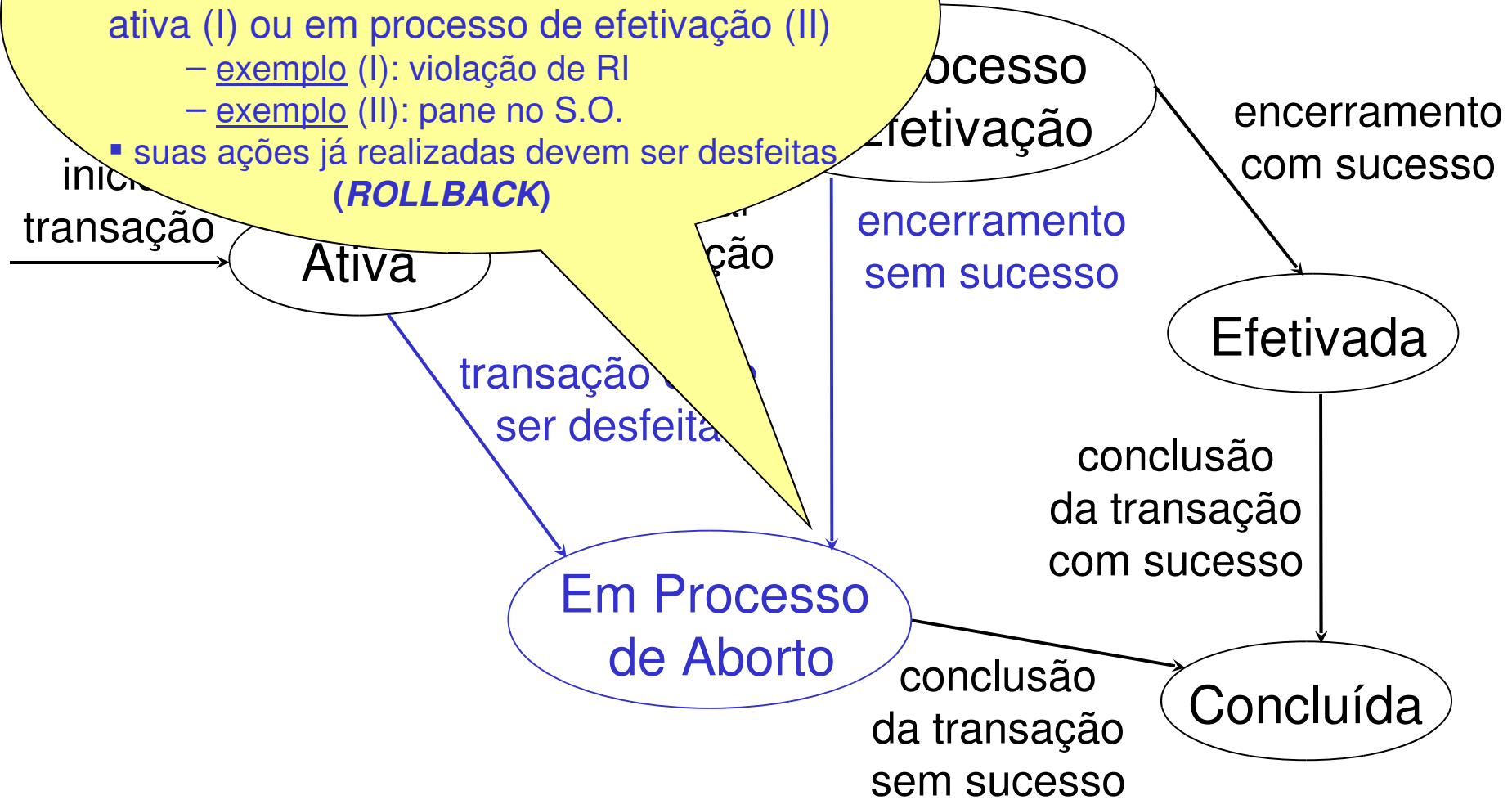


Transição de Estados de uma Transação



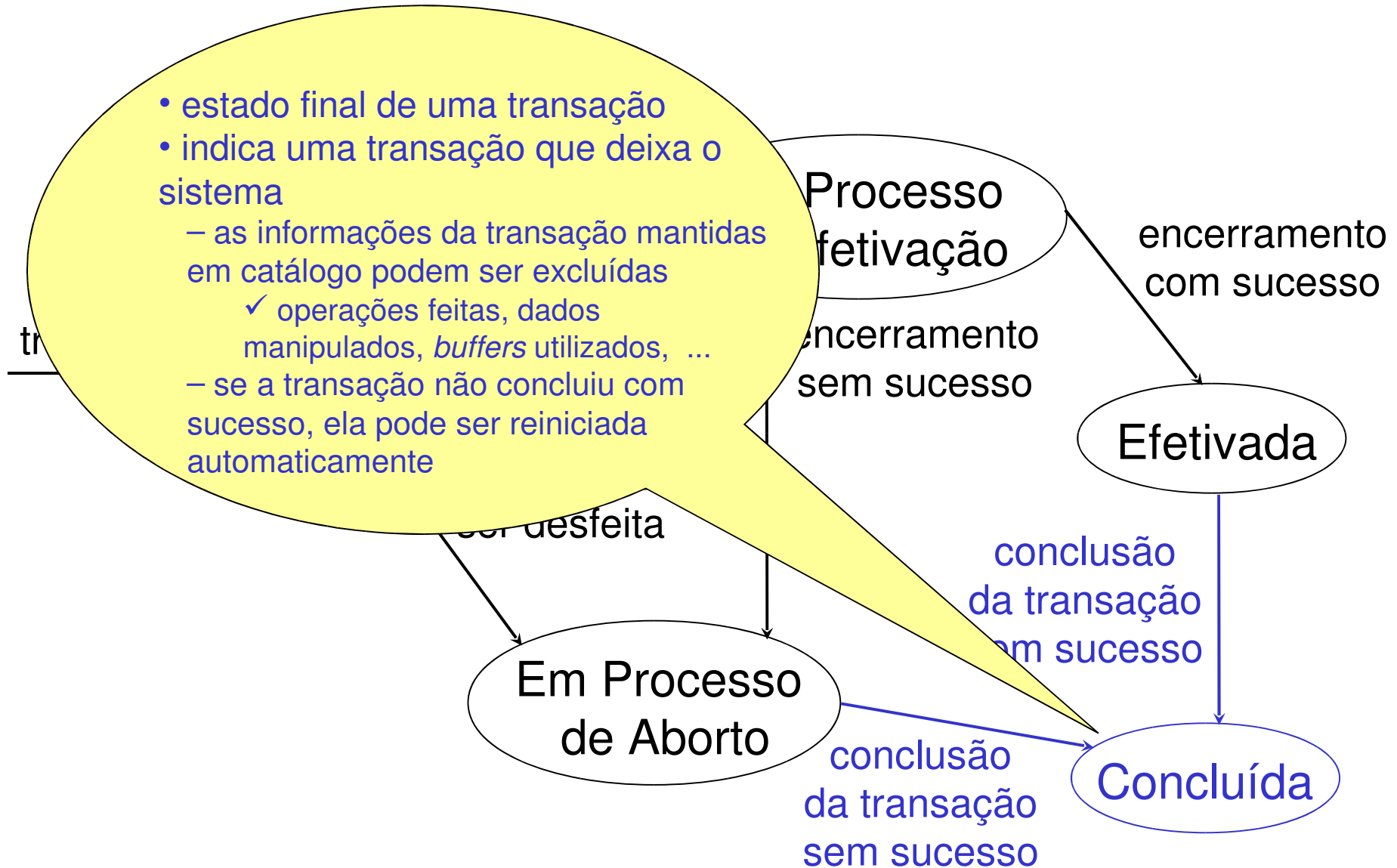
Transição de Estados de uma Transação

- entra nesse estado se não puder prosseguir a sua execução
- pode passar para esse estado enquanto ativa (I) ou em processo de efetivação (II)
 - exemplo (I): violação de RI
 - exemplo (II): pane no S.O.
- suas ações já realizadas devem ser desfeitas (**ROLLBACK**)



Transição de Estados de uma Transação

- estado final de uma transação
- indica uma transação que deixa o sistema
 - as informações da transação mantidas em catálogo podem ser excluídas
 - ✓ operações feitas, dados manipulados, *buffers* utilizados, ...
 - se a transação não concluiu com sucesso, ela pode ser reiniciada automaticamente



Propriedades de uma Transação

- Requisitos que sempre devem ser atendidos por uma transação
- Chamadas de **propriedades ACID**
 - **A**tomicidade
 - **C**onsistência
 - **I**solamento
 - **D**urabilidade

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

Contas

número	saldo
100	500.00
200	200.00
...	

← x

← y

execução ↓

T_x (transferência bancária)

read(x)

x.saldo = x.saldo - 100.00

write(x)

read(y)

y.saldo = y.saldo + 100.00

write(y)

falha!

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

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(A)

escalonamento válido

T ₁	T ₂
read(A) A = A - 50	
	A = A+A*0.1 write(A) read(B)
write(A) read(B) B = B + 50 write(B)	
	B = B - A write(B)

T₁ interfere em T₂

T₂ interfere em T₁

escalonamento inválido

Durabilidade

- 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

Gerência Básica de Transações

Ações da Aplicação ou Usuário

T_1 inicia →
 T_1 submete →
operações ao BD
 T_1 termina →

Ações do SGBD

← inicia ações para garantir **Atomicidade** de T_1
← executa as operações, garantindo **Isolamento** de T_1 , e testa RIs imediatas, com possível *rollback* e msg erro, para garantir **Consistência** de T_1
← testa RIs postergadas, com possível *rollback* e msg erro, para garantir **Consistência** de T_1
← executa ações para garantir **Durabilidade** de T_1
← confirma o término de T_1 para a aplicação/usuário

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 (várias configurações possíveis)
 - SERIALIZABLE (T_i executa com completo isolamento - *default*)
 - ...
 - READ COMMITTED (T_i só lê dados efetivados, mas outras transações podem escrever em dados lidos por T_i) – aceitável, por exemplo, em casos onde transações que lêem dados não irão escrever esses dados posteriormente
 - ...

Transações em SQL

- Exemplo (SQL embutida)

```
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 (script SQL)

```
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

...
```