INF / UFG

Disciplina

Banco de Dados

Conteúdo

Processamento de Transações.

Preâmbulo

O conceito de **operação** fornece um mecanismo para descrever unidades lógicas de processamento de dados.

Sistemas de **processamento de transações** são sistemas com grandes bases de dados e centenas de usuários simultâneos executando operações de banco de dados.

- Exemplos: reservas de passagens aéreas, bancos, processamento de cartão de crédito, compras de varejo online, caixas de supermercado, e muitas outras aplicações.

Estes sistemas requerem alta disponibilidade e tempo de resposta rápida para centenas de usuários simultâneos.

A **transação** é normalmente implementada por um programa de computador, que inclui comandos de banco de dados, tais como consultas, inserções, exclusões e atualizações.

Transação

Uma transação forma uma unidade lógica de processamento de dados. A **transação inclui uma ou mais operações** de acesso ao banco estes podem incluir inserção, eliminação, modificação ou operações de recuperação.

As operações que formam uma transação podem estar dentro de um programa de aplicação, ou ser especificados de forma interativa através de uma linguagem de consulta de alto nível, tal como SQL.

Em um programa de aplicação, é possível explicitar os limites de uma transação.

Se as operações em uma transação não atualizar o banco de dados, mas apenas recuperar os dados, a transação é chamada de **transação somente leitura**, caso contrário, ela é conhecida como uma **transação de leitura e escrita**.

Classificação de SGBDs

Classificação de acordo com o número de usuários, que podem usar o sistema ao mesmo tempo.

Um SGBD é **monousuário** se, no máximo, um usuário por vez pode usar o sistema, e é **multiusuário** se muitos usuários podem usar o sistema e, portanto, acessar o banco de dados ao mesmo tempo.

Em sistemas multiusuários, centenas ou milhares de usuários, tipicamente, operam na base de dados pela submissão simultânea de transações ao sistema:

>> nosso foco de interesse.

Operações de Acesso

Em um modelo simplificado, as operações de acesso ao banco de dados de base podem ser:

read_item (X). Lê um item de banco de dados chamado **X** em uma variável do programa. Para simplificar, vamos supor que a variável de programa também é chamado **X**.
write_item (X). Escreve o valor da variável de programa **X** no item **X** da base de dados.

X pode ter qualquer granularidade: endereço do bloco, endereço do registro, ...

Operações de Acesso

A execução de **read_item (X)** resulta em:

(R1) encontrar o endereço do bloco de disco que contém o Item **X**;
(R2) copiar esse bloco de disco para um *buffer* na memória principal (se tal bloco não estiver em algum *buffer* de memória principal);
(R3) copiar o Item **X** do *buffer* para a variável de programa chamada **X**.

A execução de **write_item (X)** resulta em:

(W1) encontrar o endereço do bloco de disco que contém o Item **X**;
(W2) copiar esse bloco de disco para um *buffer* na memória principal (se tal bloco não estiver em algum *buffer* de memória principal);
(W3) copiar o Item **X** da variável programa chamada **X** para sua localização correta no *buffer*;
(W4) armazenar o bloco atualizado do *buffer* de volta para o disco (imediatamente ou em algum momento posterior).

Operações de Acesso

A decisão sobre quando armazenar um bloco de disco modificado, cujo conteúdo está em um *buffer* de memória principal, é tomada pelo subsistema de recuperação do SGBD em cooperação com o sistema operacional.

O SGBD gerenciará um número de *buffers* de dados na memória principal (cache de banco de dados).

Cada *buffer* tipicamente suporta o conteúdo de um bloco de disco, e contém alguns dos itens do banco de dados que estão sendo processados.

Quando esses *buffers* estão todos ocupados, e algum(ns) bloco(s) precisa(m) ser copiado(s) para a memória, alguma política de substituição de *buffer* é utilizada:

>> se o *buffer* escolhido para substituição tiver sido modificado, ele deve ser escrito de volta para o disco antes que seja reusado.

Quando controle de concorrência é necessário

Problemas podem ocorrer quando operações são executadas de forma simultânea... . Exemplo – **reservas de passagens aéreas**.

A Figura 21.2 (a) mostra uma transação T1 que transfere N reservas, de um voo cujos assentos reservados é armazenado no item X, para outro voo cujos reservados é armazenado em Y.

A figura 21.2 (b) mostra uma Transação T2 que apenas reserva assentos M no primeiro vôo (X) referenciado na transação.

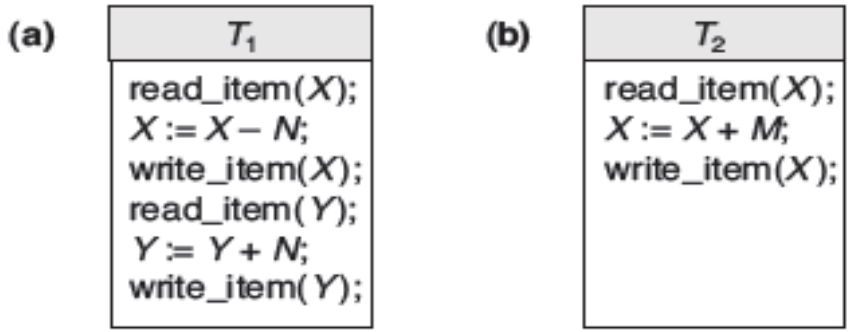
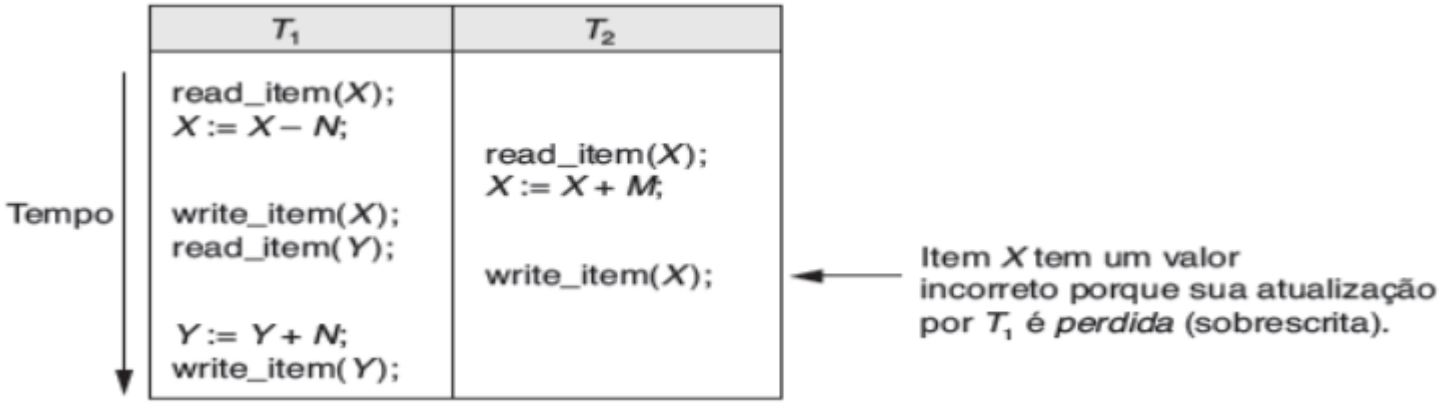


Figura 21.2

Duas transações de exemplo. (a) Transação T_1. (b) Transação T_2.

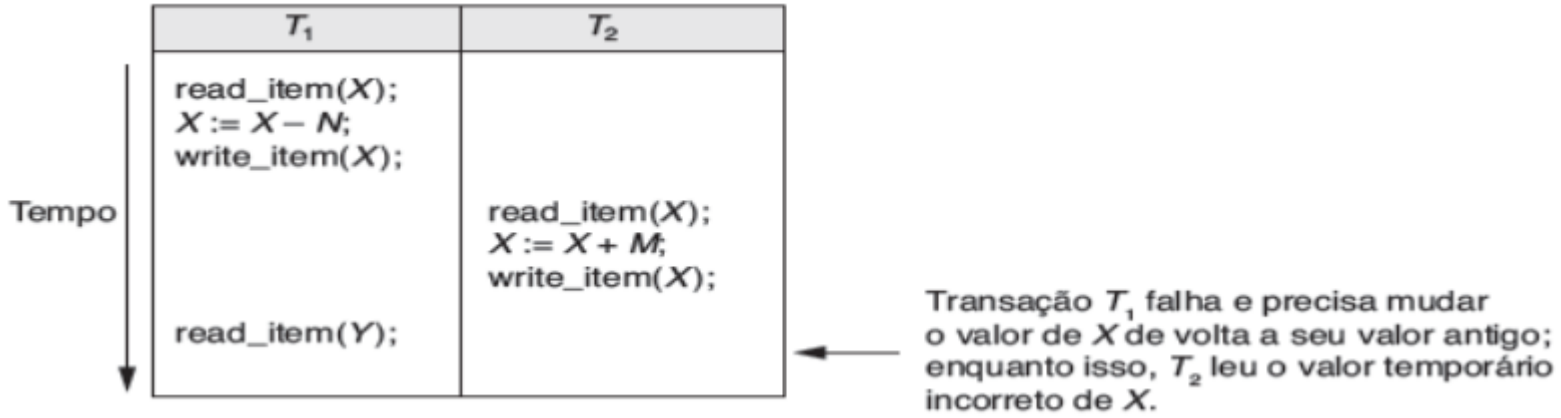
Quando controle de concorrência é necessário

O Problema de Atualização Perdida. Pode ocorrer quando duas operações, que acessam os mesmos itens de banco de dados, tem suas operações intercaladas.



Quando controle de concorrência é necessário

Problema da atualização temporária (ou leitura suja). Pode ocorrer quando uma transação atualiza um item de banco de dados e, em seguida, a transação falha por algum motivo. Enquanto isso, o item atualizado é acessado (leitura) por outra transação antes de ser retornado ao seu valor original.



Quando controle de concorrência é necessário

O Problema do resumo incorreto. Se uma transação estiver computando uma função agregada (cálculo de resumo, exemplos: soma, valor médio, etc.), enquanto outras transações estão atualizando alguns desses itens, a função pode calcular alguns valores antes que eles são atualizados e outros depois que eles são atualizados.

T_1	T_3

	sum := 0;
	read_item(*A*);
	sum := *sum* + *A*;
	. . .

read_item(*X*);
X := *X* − *N*;
write_item(*X*);	

	read_item(*X*);
	sum := *sum* + *X*;
	read_item(*Y*);
	sum := *sum* + *Y*;

read_item(*Y*);
Y := *Y* + *N*;
write_item(*Y*);	

← T_3 lê *X* depois que *N* é subtraído e lê *Y* antes que *N* seja somado; um resumo errado é o resultado (defasado por *N*).

Quando controle de concorrência é necessário

O problema de leitura "irrepetível". Uma transação T1 lê o mesmo item duas vezes e o item é alterado por outra transação T2 entre as duas leituras. Assim, T1 recebe valores diferentes para as duas leituras do mesmo item.

Isto pode ocorrer, por exemplo, se durante uma operação de reserva de passagens aéreas, o cliente pergunta sobre a disponibilidade de assentos em vários voos. Quando o cliente decide por um voo particular, a transação lê novamente o número de assentos disponíveis do vôo antes de completar a reserva, o que pode resultar em um valor diferente para o item.

Operações para Recuperação

BEGIN TRANSACTION. Marca o início da execução da transação.

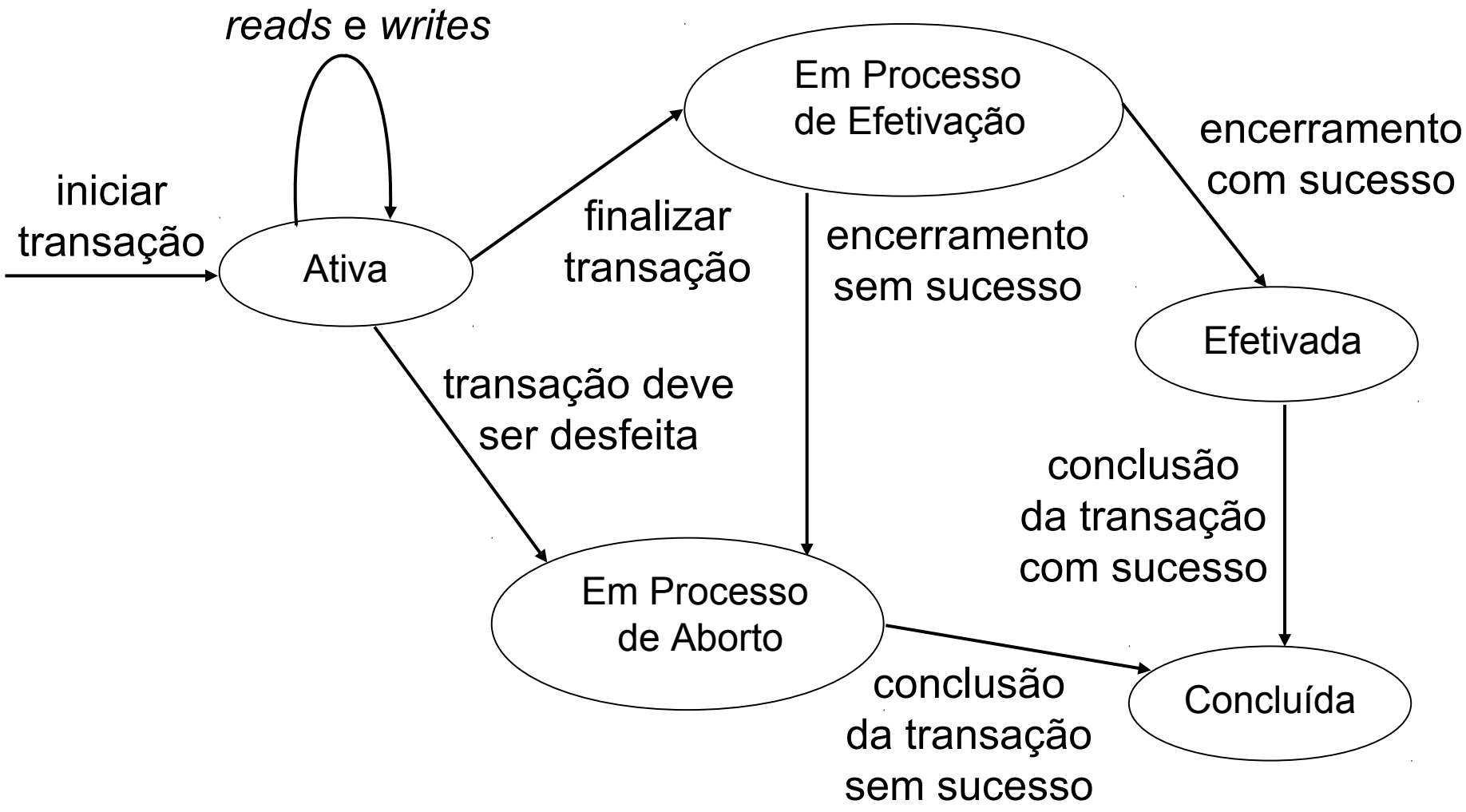
READ / WRITE. Operações para a leitura e a escrita no banco de dados.

END TRANSACTION. Especifica que as operações de leitura e de escrita finalizaram e marca o fim da execução da transação.
Neste ponto, pode ser necessário verificar se as alterações introduzidas pela transação podem ser permanentemente aplicada ao banco de dados, ou se a transação tem que ser abortado (exemplo, viola serialização).

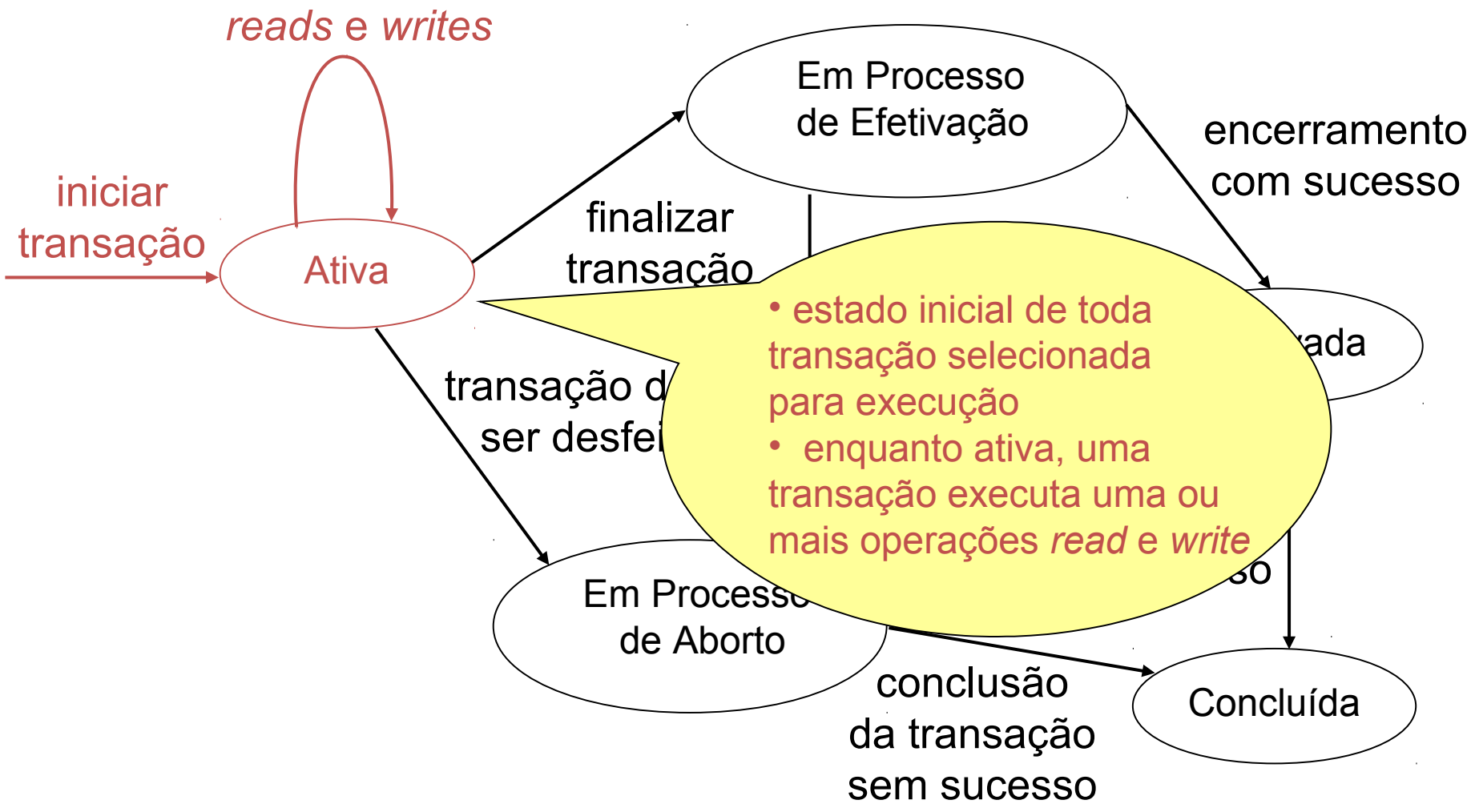
COMMIT TRANSACTION. Sinaliza um **final bem-sucedido** da transação, para que quaisquer alterações (atualizações) executadas pela transação possam ser seguramente ***commited*** no banco de dados (a transação não será desfeita).

ROLLBACK (ou anular). Sinaliza que a transação foi **encerrada sem sucesso**, para que quaisquer alterações aplicadas (e seus efeitos) ao banco de dados sejam desfeitas.

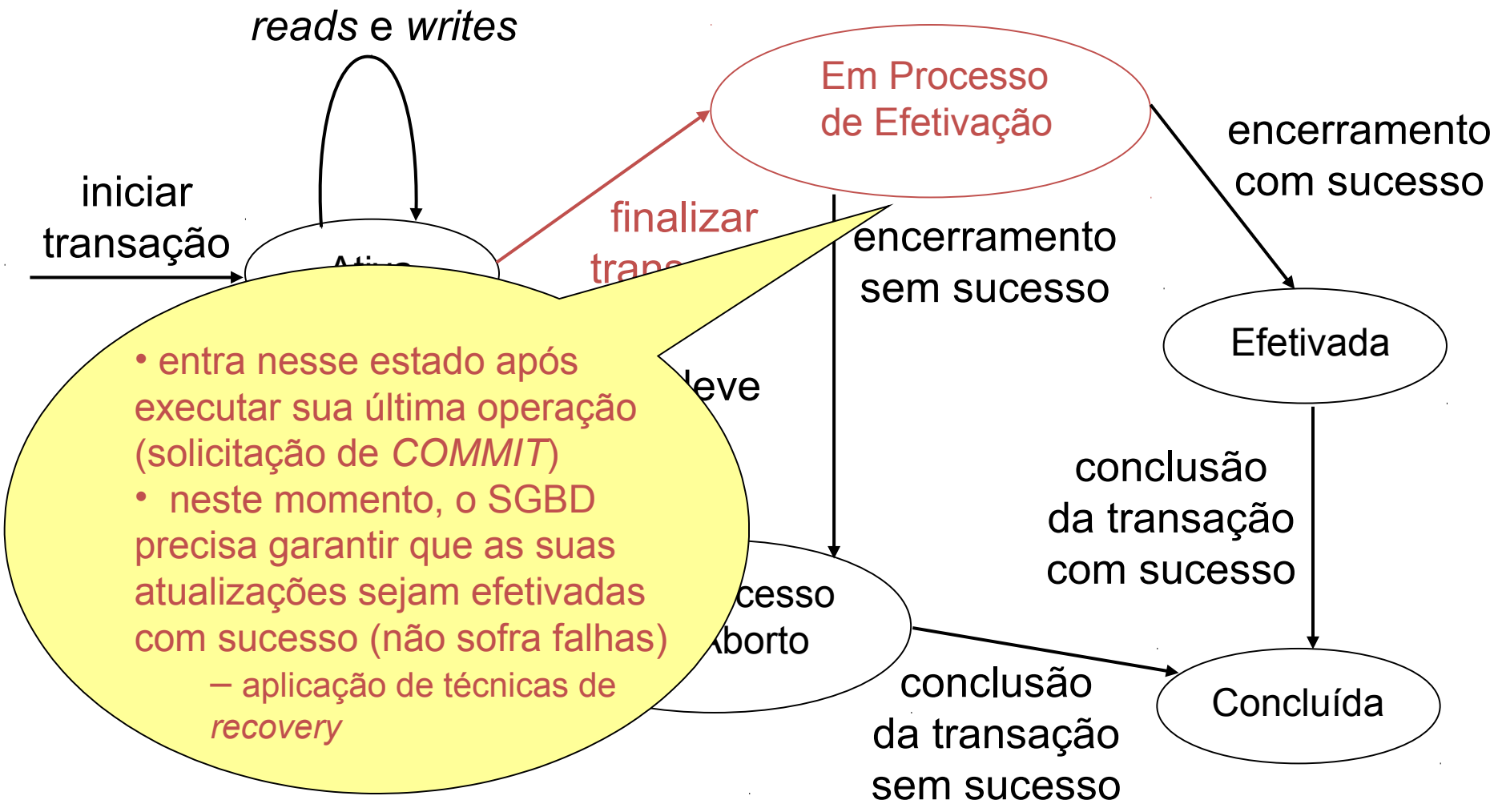
Transição de Estados de uma Transação



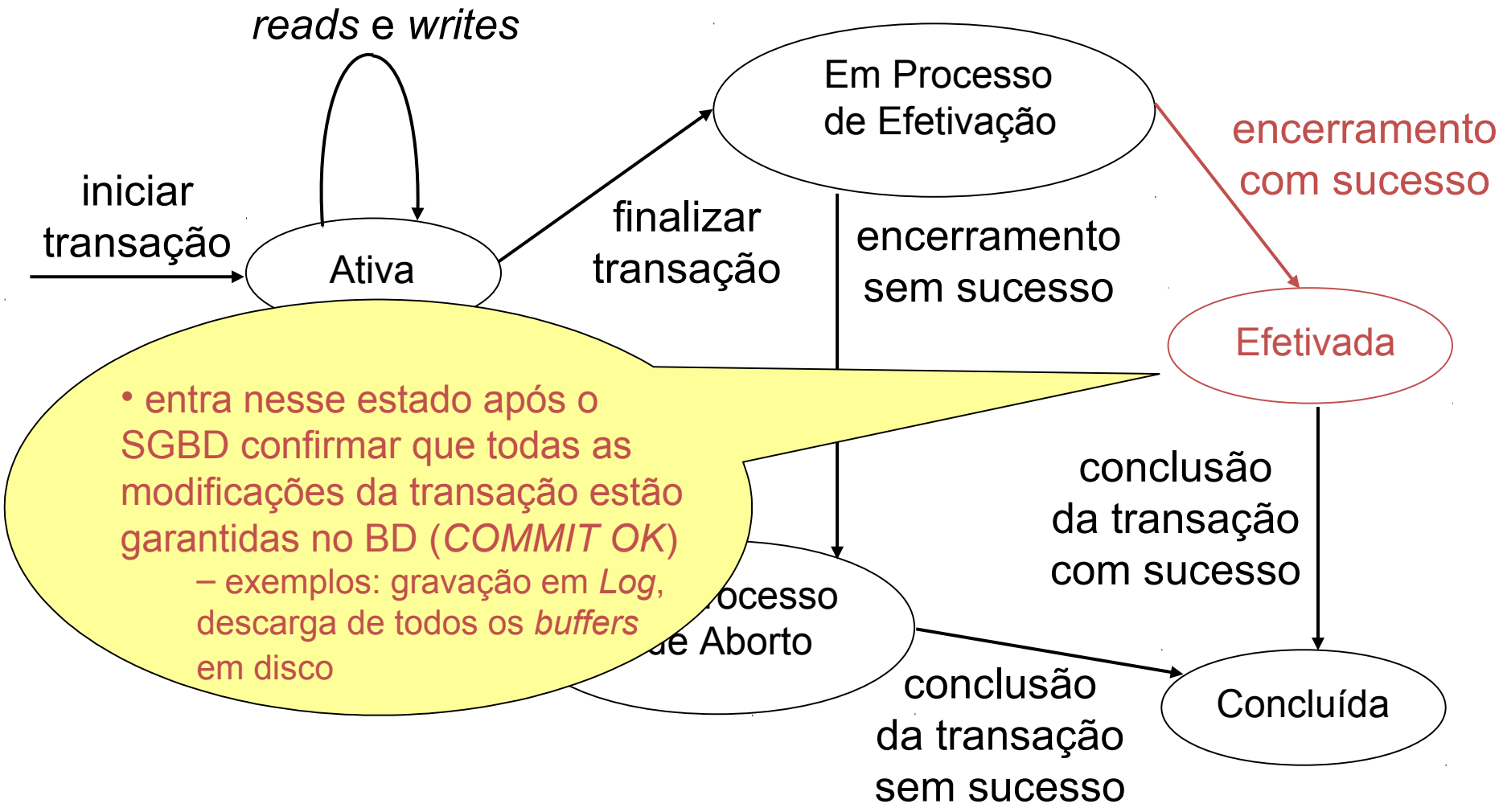
Transição de Estados de uma Transação



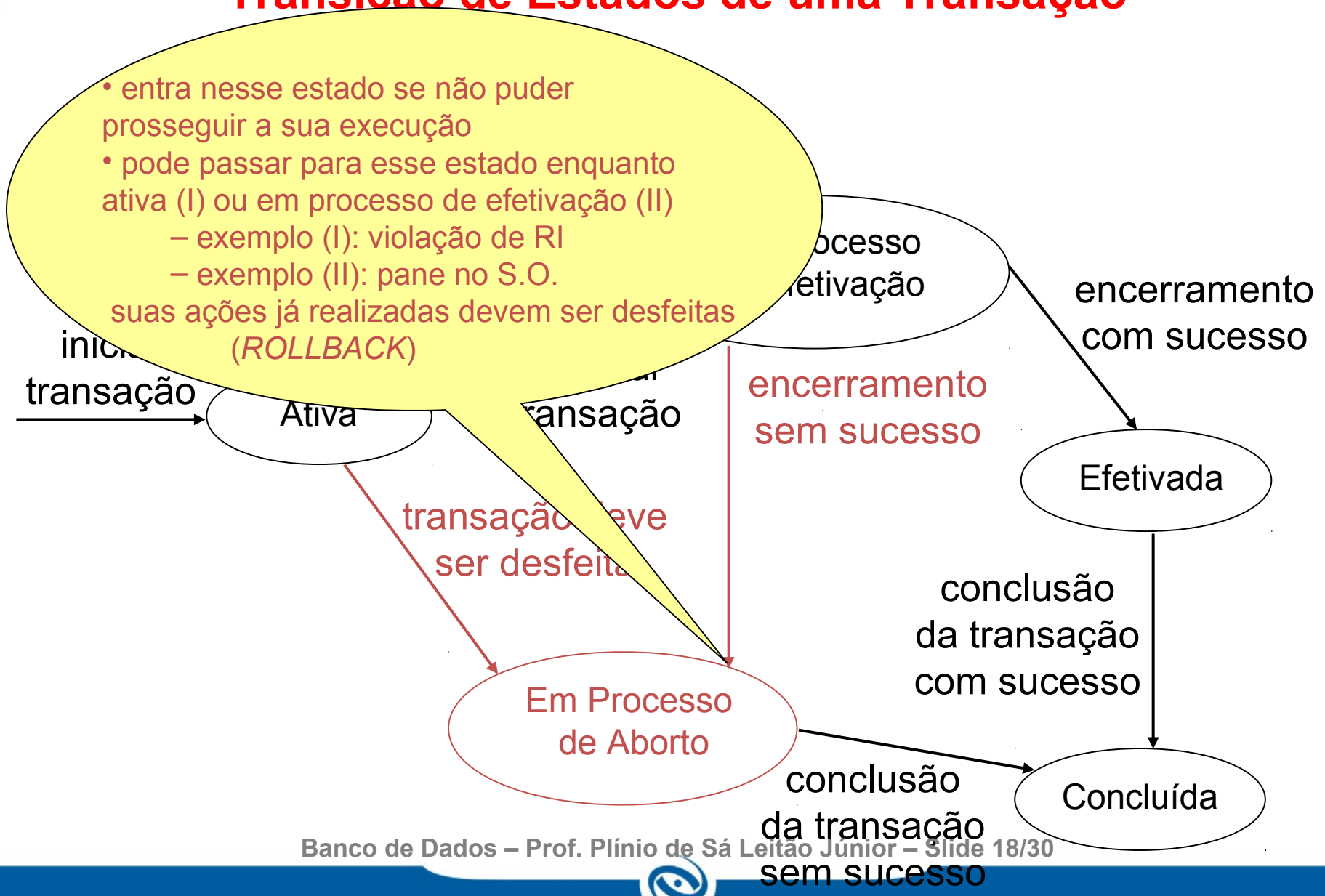
Transição de Estados de uma Transação



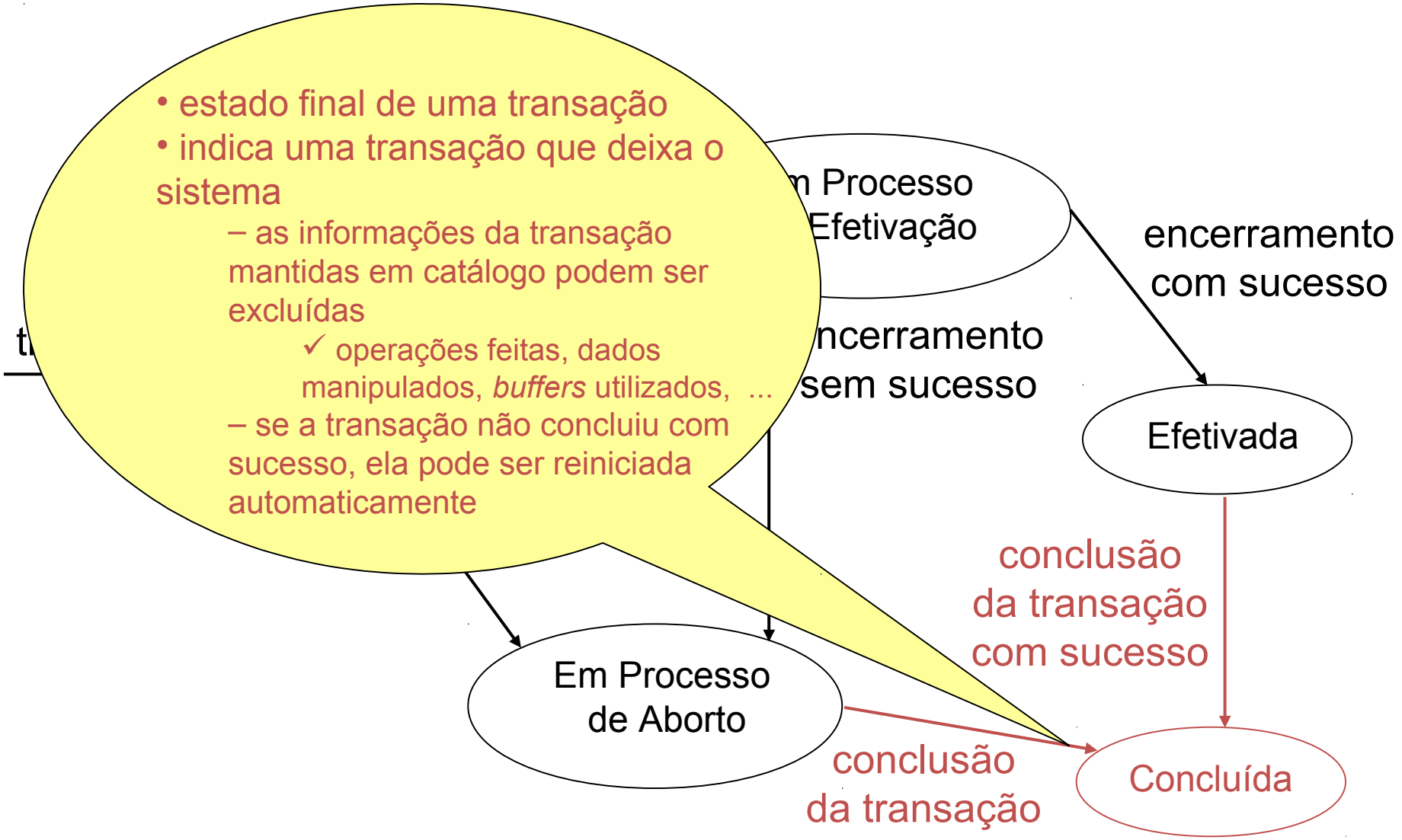
Transição de Estados de uma Transação



Transição de Estados de uma Transação



Transição de Estados de uma Transação



Propriedades de uma Transação

Requisitos que devem ser atendidos.

Atomicidade
Consistência
Isolamento
Durabilidade ou Persistência

ACID

Atomicidade

Princípio do "Tudo ou Nada":

- ou todas as operações da transação são efetivadas com sucesso no BD ou nenhuma delas é efetivada;
- preservar a integridade do BD.

Responsabilidade:

- subsistema de recuperação contra falhas do SGBD (subsistema de *recovery*):
- garantia: desfazer as ações de transações parcialmente executadas.

A atomicidade deve ser garantida, pois uma transação pode manter o BD em um estado inconsistente durante a sua execução.

Atomicidade

Exemplo

Contas

número	saldo
1001-2	500.00
2002-3	200.00
. . .	

x

y

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!

execução

Consistência

Estado consistente:

- uma transação sempre conduz o BD de um estado consistente para outro estado também consistente.

Responsabilidade conjunta:

1) Projetista (e DBA):

- definir todas as regras de integridade para garantir estados e transições de estado válidos para os dados:
    exemplos:
        salário > 0
        salário novo < salário antigo

2) subsistema de *recovery*:

- desfazer as ações da transação que violou a integridade.

Isolamento

Transações concorrentes:

- no contexto de um conjunto de transações concorrentes, a execução de uma transação **Tx** deve funcionar como se **Tx** executasse de forma isolada;

- Tx não deve sofrer interferências de outras transações executando concorrentemente.

Responsabilidade:

- responsabilidade do subsistema de controle de concorrência do SGBD (*scheduler*), para 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

Durabilidade (ou Persistência)

Modificações devem persistir:

- deve-se garantir que as modificações realizadas por uma transação que concluiu com sucesso persistam no BD;

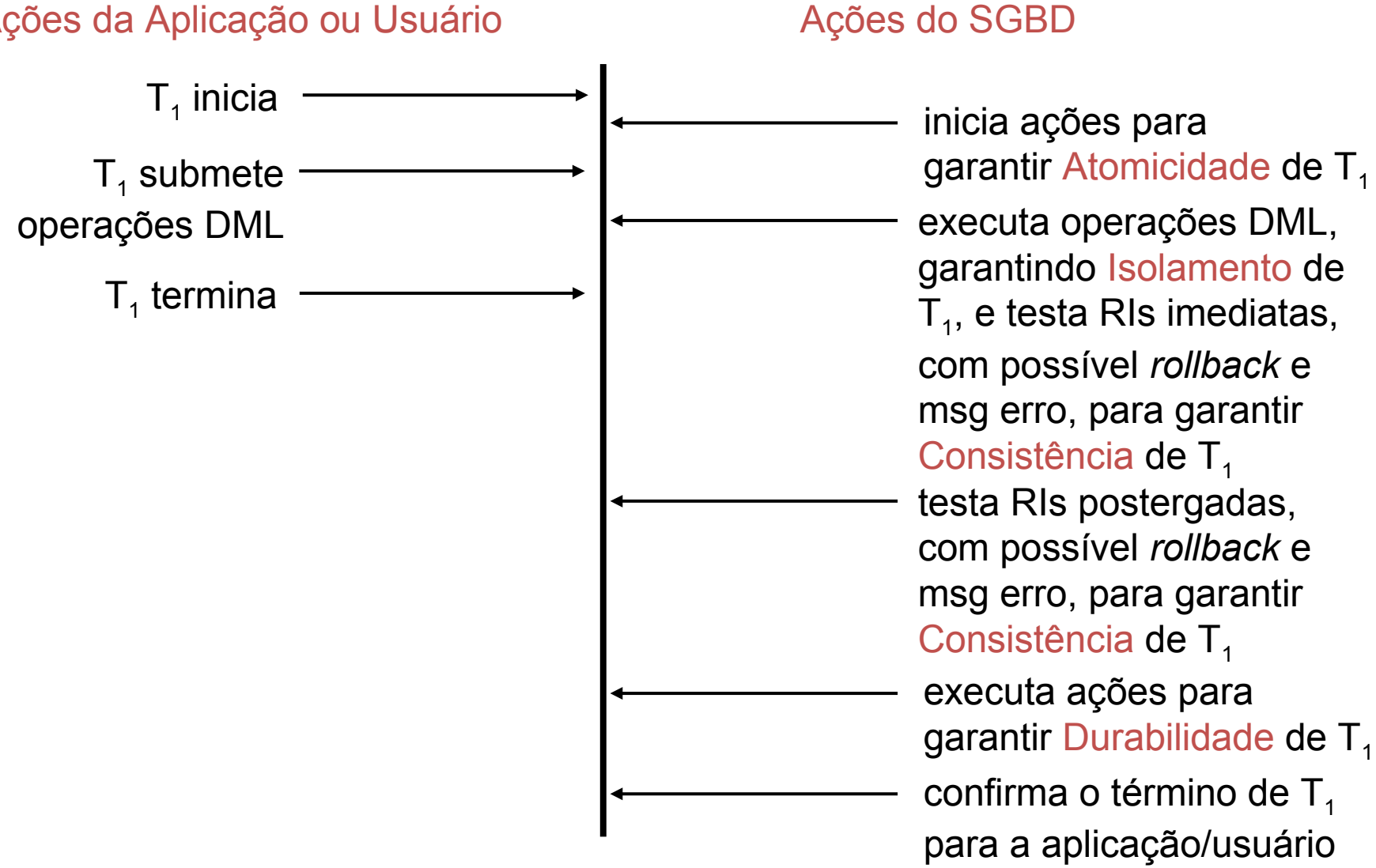
- nenhuma falha posterior ocorrida no BD deve perder essas modificações.

Responsabilidade:

- responsabilidade do subsistema de *recovery*;

- “refazer” transações que executaram com sucesso em caso de falha no BD.

Gerência Básica de Transações



Transações em SQL

Transação implícita (*default*)

- por *default*, todo comando individual é considerado uma transação;
- por exemplo, o comando

```
DELETE FROM Pacientes
```

exclui todas ou não exclui nenhuma tupla de pacientes.

Transação explícita:

- inicia uma transação

START TRANSACTION (ou BEGIN WORK, ou BEGIN TRANSACTION)

- salva o estado do banco de dados em um ponto da transação

SAVE TRANSACTION (ou SAVEPOINT)

- torna permanentes as modificações das operações da transação

COMMIT

- descarta as modificações ocorridas desde o último COMMIT

ROLLBACK

Transações em SQL

Exemplo 1:

```
START TRANSACTION;
  UPDATE TABELA SET Valor = Valor - 200 WHERE Chave = 1234;
  UPDATE TABELA SET Valor = Valor + 200 WHERE Chave = 2345;
IF ERRORS = 0 COMMIT;
IF ERRORS <> 0 ROLLBACK;
```

Exemplo 2:

```
CREATE TABLE TABELA (Valor INT);
INSERT INTO TABELA (Valor) VALUES (1);
INSERT INTO TABELA (Valor) VALUES (2);
COMMIT;
UPDATE TABELA SET Valor = 100 WHERE Valor = 1;
SAVEPOINT Save1;
UPDATE TABELA SET Valor = 200 WHERE Valor = 2;
ROLLBACK TO Save1;
SELECT Valor FROM TABELA;
```


Pesquise ...

O que é nível de isolamento entre transações de banco de dados ?

Descreva os níveis de isolamento abaixo:

Isolation level Serializable
Isolation level Repeatable Read
Isolation level Read Committed
Isolation level Read Uncommitted