Bases de Dados

Transações

Profa. Elaine Parros Machado de Sousa

ICMC USP SÃO CARLOS SCC

Grupo de Bases de Dados e Imagens
USP - São Carlos

Transações

- **Transação**: Unidade lógica de trabalho
  - abrange um conjunto de operações de manipulação de dados que executam uma única tarefa

Conecta ao Banco de Dados

Começa transação

Operações de consulta/atualização

...

Finaliza transação

Começa transação

Operações de consulta/atualização

...

Aborta transação

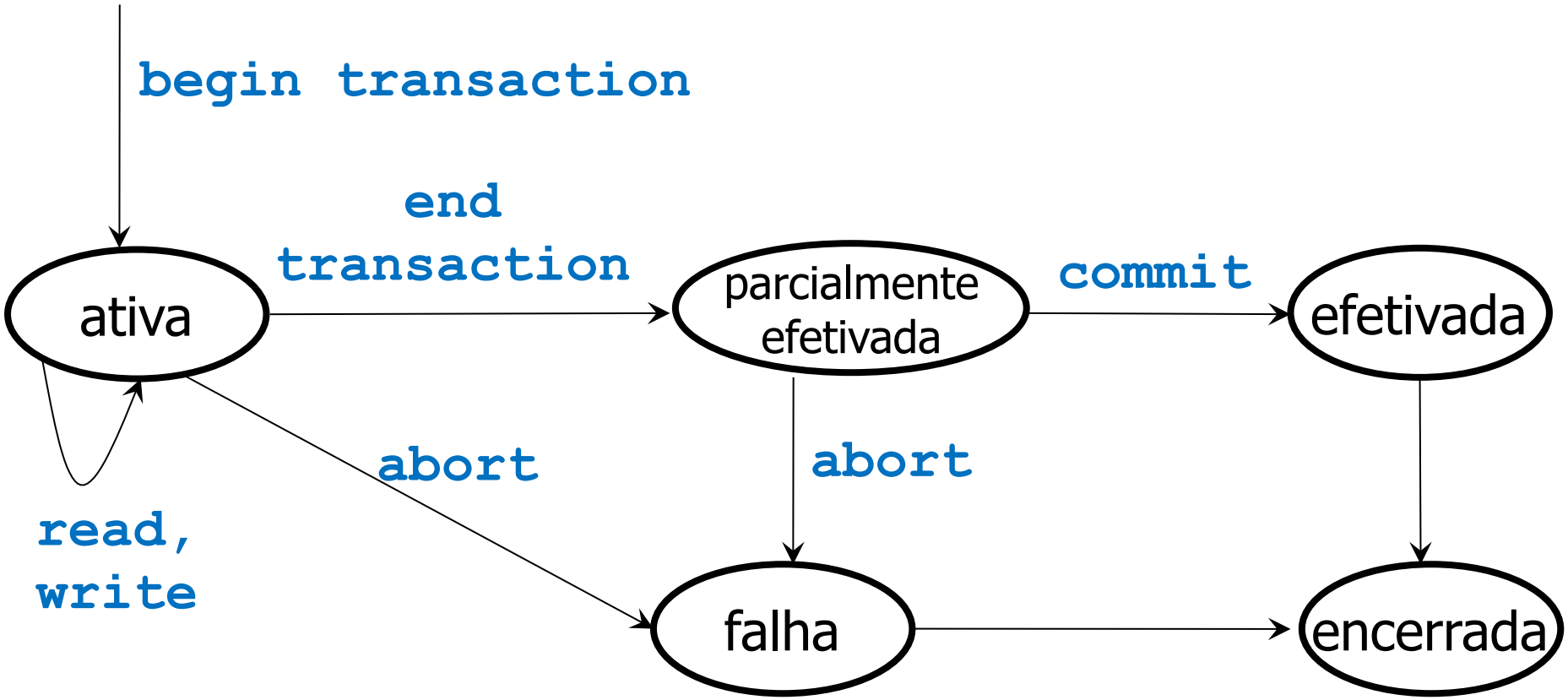
Desconecta

Transações

- Operações
  - **`begin_transaction`**
  - **`read/write`**
    - operações de leitura/escrita nos dados
  - **`end_transaction`**
    - verifica se a transação executará *commit* ou *rollback*
  - **`commit_transaction (commit)`**
    - transação finalizada com sucesso
    - torna alterações permanentes banco
  - **`abort_transaction (rollback)`**
    - transação finalizada sem sucesso
    - desfaz as alterações realizadas no banco

Transações

- Transição de Estados de Execução



Transações

- Quando uma transação começa?
  - explicitamente com a operação de **begin_transaction**
  - implicitamente quando uma sessão é iniciada no SGBD
  - implicitamente após a execução de um comando **DDL**, **commit** ou **rollback**

Transações

- Quando uma transação termina?
  - explicitamente com **commit** ou **rollback**
  - implicitamente quando um processo de usuário é finalizado
    - com sucesso – ex: disconnect (**commit**)
    - sem sucesso – ex: falha de sistema (**rollback**)
  - usuário (aplicação) executa comando DDL

Transações

- **`Read (X)`**
  - Lê o item de dado X (em geral para uma variável de programa)
  - Passos básicos
    - encontrar o endereço do bloco de disco que contém X
    - copiar o bloco para um *buffer* da memória principal
    - copiar o valor de X do *buffer* para uma variável de programa

Transações

- **`Write (X)`**
  - Grava um valor em um item de dado X
    - em geral a partir de uma variável de programa
  - Passos básicos
    - encontrar o endereço do bloco de disco que contém X
    - copiar o bloco para um *buffer* da memória principal
    - copiar o valor de uma variável de programa (ou de uma posição de memória) para o local correto no *buffer* (posição de X)
    - copiar o bloco do *buffer* de volta para o disco (imediatamente ou posteriormente)

Transações

- SGBDs multi-usuários
  - vários usuários/aplicações usando o SGBD concorrentemente
    - mesmo usuário/aplicação pode executar várias transações
  - transações concorrentes
    - transações podem acessar/atualizar os mesmos dados
      - **granularidade** dos dados (**item de dado**): campo, tupla, tabela, base de dados

Transações

- O que acontece, por exemplo:
  - quando duas transações executam simultaneamente manipulando o mesmo item de dado?
  - se a energia acabar no meio de uma transação, ou se houver um problema com o disco?

⇒ O banco de dados pode ser levado a um estado inconsistente...

Transações – **Propriedades ACID**

Atomicidade

Consistência

Isolamento

Durabilidade

Transações – **Propriedades ACID**

- **Atomicidade**: todas as operações de uma transação devem ser efetivadas. Ou, na ocorrência de uma falha, nada deve ser efetivado
  - "**tudo ou nada**"

- **Consistência**: transações preservam a consistência da base
  - Estado inicial consistente ⇒ Estado final consistente

Transações – **Propriedades ACID**

- **Isolamento**: uma transação A não vê o efeito de uma transação B até que B termine

- **Durabilidade**: uma vez terminada a transação, as alterações realizadas permanecem no banco até que outras alterações sejam explicitamente realizadas

Garantindo as Propriedades ACID

- ***Logging* e Recuperação de falhas**
  - Garantem a **Atomicidade** e a **Durabilidade**
    - Módulos de *Log*
- **Controle de Concorrência**
  - Garante a **Consistência** e o **Isolamento**, (dada a atomicidade das transações)
    - Módulos de *Locks*

Recuperação de Falhas

- Falhas locais
- Falhas globais
  - falhas de sistema
  - falhas de meio físico

Recuperação de Falhas

- **Falhas locais**
  - afetam apenas a transação corrente
  - exemplos:
    - *bugs* de software
    - transação em *deadlock*
    - transação finalizada pelo usuário
    - ...
  - como evitar danos?
    - Tratamento de Erros e Exceções

Recuperação de Falhas

- Falhas globais:
  - falha de sistema (*soft crash*)
    - afeta todas as transações em andamento, mas não danifica os dados permanentemente
    - exemplos:
      - falhas do SO
      - falhas de hardware
      - falta de energia
      - .....
    - como evitar danos?
      - **Registros de *Log***

Recuperação de Falhas

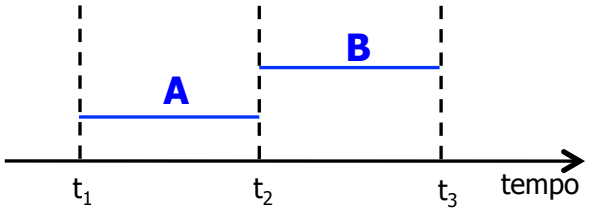
- *Log*: arquivo histórico
  - cadastra todas as atualizações realizadas no banco de dados
  - permite desfazer/refazer ações de transações inacabadas em caso de falhas.
- Registram:
  - identificação das transações
  - arquivos manipulados
  - registros atualizados
  - operações executadas
  - **valores atuais e anteriores**
  - ...

Recuperação de Falhas

- Falhas globais:
  - falha do meio físico (*hard crash*)
    - causa danos ao banco de dados (ou parte dele) de maneira irrecuperável
    - afeta todas as transações em andamento que estavam utilizando a parte danificada, e possivelmente pode causar perda no banco de dados
    - Como evitar danos?
      - **BACKUP**

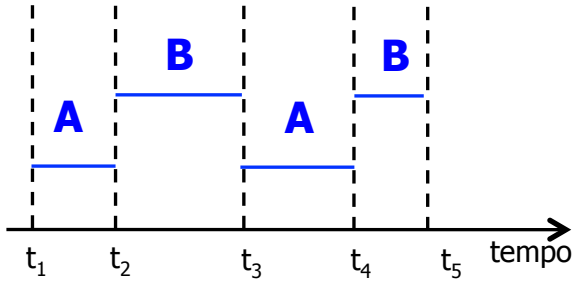
Controle de Concorrência

- **Execução Serial**: diversas transações executadas em sequência
  - deixa a base de dados em estado correto e consistente
  - maior isolamento
  - menor concorrência



A e **B** são transações

- **Execução Intercalada**: comandos de diversas transações são intercalados
  - pode levar a inconsistências
  - menor isolamento
  - maior concorrência



Execução Serial X Intercalada

- **Execução serial**
  - estado final da base de dados após **execução serial** pode variar dependendo da ordem em que as transações são executadas
    - mas **todos são estados corretos e consistentes** (na perspectiva do SGBD)

Execução Serial X Intercalada

- **Execução Intercalada**
  - Estado final da base de dados após **execução intercalada** é consistente se for igual ao resultado obtido por uma execução serial (qualquer uma)
    - esta execução é dita **SERIALIZÁVEL**

Execução Intercalada

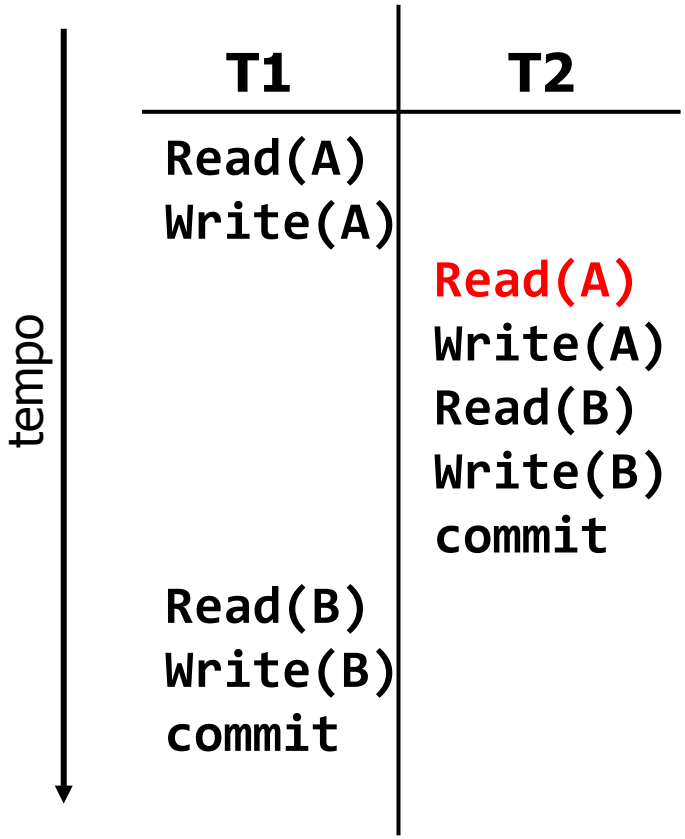
- O **PROBLEMA**.... Dependendo da ordem em que os comandos de duas ou mais transações são intercalados:
  - o resultado final pode ser inconsistente ⇒ ou seja, não existe uma sequência de transações executadas em série que leve a esse resultado
- Ocorrência de anomalias.
  - **leitura inválida**
  - **leitura não repetível**
  - **leitura fantasma**
  - ....

Anomalias

- **Leitura inválida (*Dirty Read*)**:
  - transação T_2 lê um dado **modificado** por uma transação T_1 que ainda **não terminou**
    - leitura de dado que ainda não foi efetivado

Anomalias

- **Ex: Leitura inválida (*Dirty Read*):**



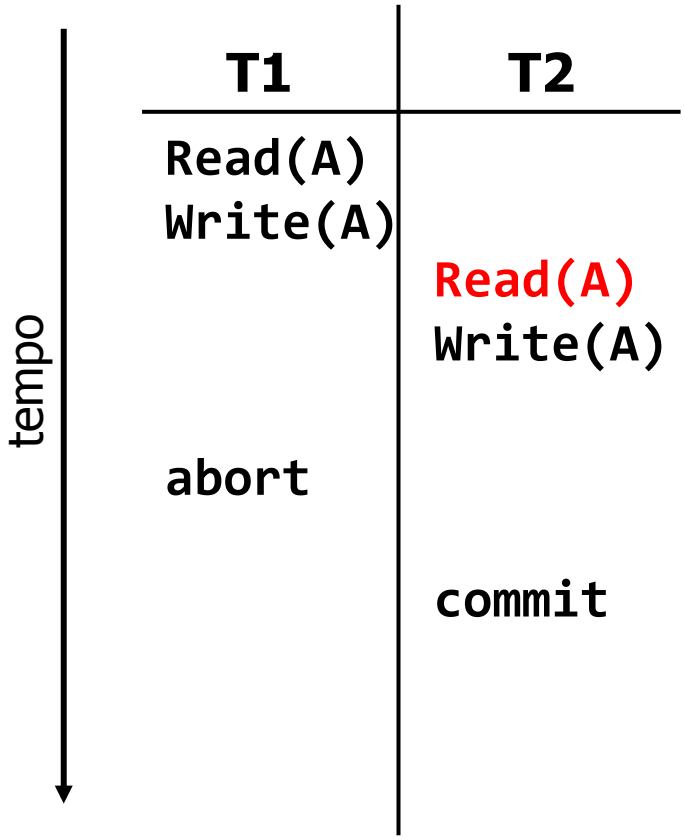
Exemplo 1:

- Transação T_1: transfere R$100,00 da conta A para a conta B.
- Transação T_2: incrementa A e B em 1% (juros).

Execução **não serializável** ⇒ estado final inconsistente!

Anomalias

- **Ex: Leitura inválida (*Dirty Read*):**



Exemplo 2:

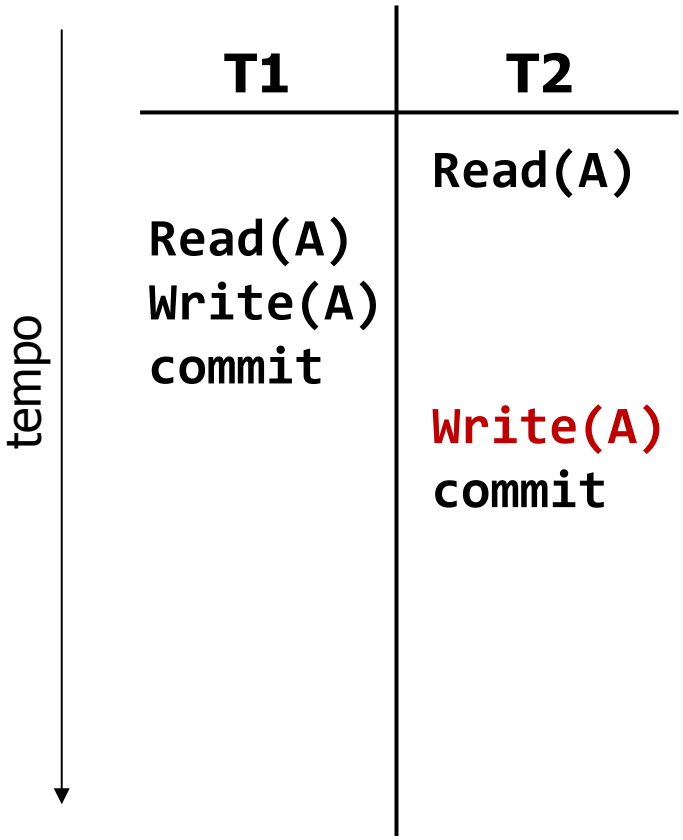
- Transação T_1: deposita R$100,00 na conta A.
- Transação T_2: saca tudo de A.
- T_1 é cancelada (*abort* => *rollback*)

Anomalias

- **Leitura não repetível (*Nonrepeatable Read*)**:
  - transação T_2 lê um dado **válido**
  - transação T_1, que **começou depois de T_2**, lê o mesmo dado (**válido**) e o modifica
  - T_1 é efetivada
  - o dado que T_2 tem está desatualizado
    - se T_2 tentar reler o mesmo dado, terá dois valores diferentes (*nonrepeatable read*) em **leituras válidas**

Anomalias

- **Ex: Leitura não repetível (*Nonrepeatable Read*)**:



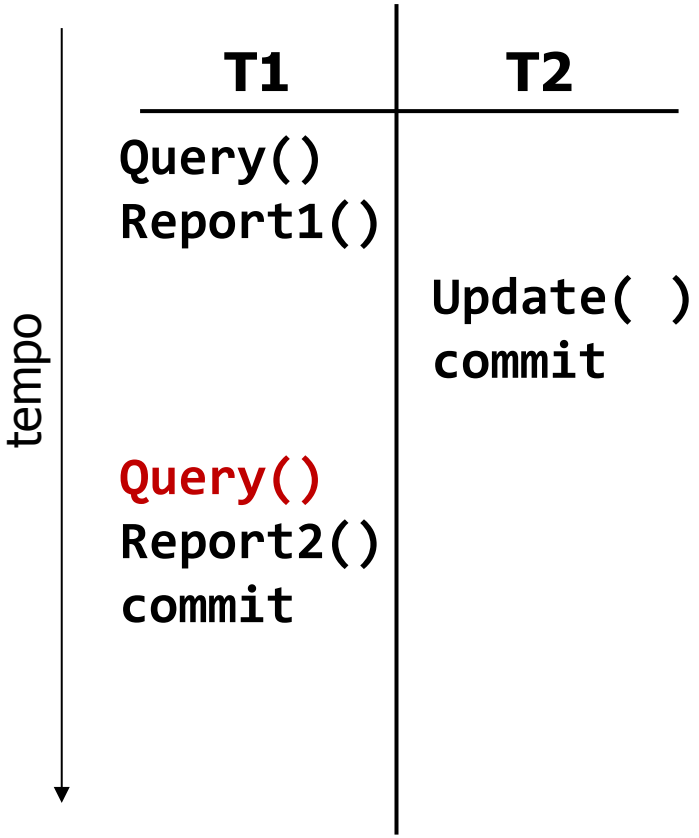
- **Transação T_2**: lê reservas de um vôo e verifica que há apenas um lugar disponível (**leitura válida**)
- **Transação T_1**: lê a mesma coisa (**leitura válida**).
- T_1 reserva o último lugar e é efetivada.
- T_2 tenta reservar o lugar e ocorre um erro ou um *overbooking*.

Anomalias

- **Leitura fantasma (*Phantom Read*)**:
  - transação T_1 lê um **conjunto de tuplas válidas** que atendam a uma **condição de consulta**
  - transação T_2 **insere/remove/atualiza** uma tupla que afetaria a resposta da consulta de T_1 e é efetivada
  - se T_1 refizer a mesma consulta, obterá um conjunto diferente de tuplas (*phantom read*)
    - consultas em **dados válidos** dentro da mesma transação com resultados distintos

Anomalias

- **Ex: Leitura fantasma (*Phantom Read*):**



Transação T_1: faz uma consulta que retorna NUSP e média geral dos alunos que têm ponderada acima de 5.0, e gera um relatório R1.

- Transação T_2: atualiza as notas de vários alunos, causando alteração na média geral de alguns deles, e é efetivada
- T_1 refaz a consulta para gerar um relatório R2 com quantidade de alunos por faixa de média

⇒ relatórios inconsistentes.

Níveis de Isolamento em SQL99

Nível de isolamento	Anomalias **EVITADAS**		
	Leitura inválida	**Leitura não repetível**	**Leitura fantasma**
Read uncommitted	**Não**	**Não**	**Não**
Read committed	**Sim**	**Não**	**Não**
Repeatable read	**Sim**	**Sim**	**Não**
Serializable	**Sim**	**Sim**	**Sim**

Níveis de Isolamento em SQL99

- **Read Uncommitted**
  - transação T **pode** ler dados modificados por transações em andamento
  - transação T **não** obtém bloqueios (*locks*) de leitura
  - SQL proíbe T de executar operações de escrita (READ ONLY)

Níveis de Isolamento em SQL99

- **Read Committed**
  - transação T lê apenas dados modificados por transações já efetivadas
    - **evita leitura inválida**
  - transação T obtém e mantém bloqueios exclusivos dos objetos que precisa escrever até terminar
  - transação T obtém bloqueios compartilhados de objetos que precisa ler e os **libera logo após a leitura**
    - **permite leitura fantasma e leitura não repetível**

Níveis de Isolamento em SQL99

- **Repeatable Read**
  - transação T lê apenas dados modificados por transações já efetivadas
    - evita leitura inválida
  - transação T obtém e mantém bloqueios dos objetos que precisa ler (compartilhado) e/ou escrever (exclusivo) **até terminar**
    - evita leitura não repetível
    - permite leitura fantasma

Níveis de Isolamento em SQL99

- **Serializable**
  - transação T lê apenas dados modificados por transações já efetivadas
    - **evita leituras inválidas**
  - transação T obtêm e mantém bloqueios de todos os objetos que precisa ler (compartilhado) e/ou escrever (exclusivo) até terminar
    - **evita leituras não repetíveis**
  - se transação T lê um conjunto de valores baseado em uma condição de consulta, este conjunto não é modificado até que T termine
    - **evita leitura fantasma**

Leitura Recomendada

- R. Elmasri, S. Navathe: *Sistemas de Banco de Dados*
  - 6ª Edição
    - Capítulos 21, 22 e 23

- Oracle Documentation 19c
  - *Database Concepts*
    - *Transaction Management*
  - *SQL Language Reference*

EXERCÍCIOS

- Considere as transações T1 e T2, executadas sobre os itens de dado X e Y

T1
Read(X)
Read(Y)
Write(X)
commit

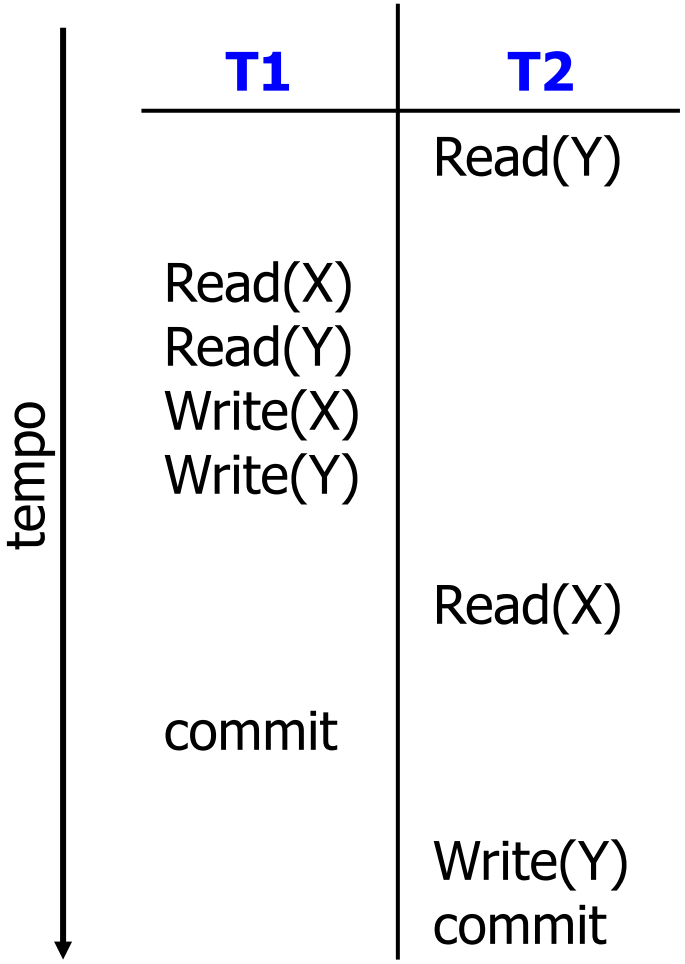
T2
Read(X)
Read(Y)
Write(X)
Write(y)
commit

1) Dê um exemplo de execução intercalada que resulte em uma **anomalia de leitura inválida** e explique o porquê.

2) Dê um exemplo de execução intercalada que resulte em uma **anomalia de leitura não repetível** e explique o porquê.

Exercício

- Considere as transações T1 e T2, executadas sobre os itens de dado X e Y, e a seguinte execução intercalada:



a) Indique onde ocorrem as anomalias de **leitura inválida** e **leitura não repetível.**

b) Explique como o nível de isolamento **read committed** evitaria a anomalia de leitura inválida para estas transações.

c) Explique como o nível de isolamento **repeatable read** evitaria a anomalia de leitura não repetível para estas transações.