Processamento de Transações

Banco de Dados II

Prof. Guilherme Tavares de Assis

Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM

Transação

- Transação é uma unidade lógica de processamento de operações sobre um banco de dados.
  - Uma transação é formada por uma sequência de operações que precisam ser executadas integralmente para garantir a consistência e a precisão dos dados de um banco de dados.

- Geralmente, uma transação consiste de uma das seguintes instruções:
  - uma operação DDL (*Data Definition Language*);
  - uma operação DCL (*Data Control Language*);
  - um conjunto de operações DML (*Data Manipulation Language*).

Transação

- Uma transação começa quando for executada a 1ª operação SQL executável e termina com um dos seguintes eventos:
  - comando COMMIT ou ROLLBACK é emitido;
  - operação DDL ou DCL é executada (*commit* automático);
  - o usuário desconecta do banco de dados (*commit* automático);
  - o sistema falha (*rollback* automático).

- Quando uma transação termina, o próximo comando SQL inicia automaticamente a próxima transação.

Transação

- As operações de controle de transações são:
  - **COMMIT**: finaliza a transação atual tornando permanentes todas as alterações de dados pendentes.
  - **SAVEPOINT <nome_savepoint>**: marca um ponto de gravação dentro da transação atual, sendo utilizado para dividir uma transação em partes menores.
  - **ROLLBACK [TO SAVEPOINT <nome_savepoint>]**: ROLLBACK finaliza a transação atual, descartando todas as alterações de dados pendentes.

    ROLLBACK TO SAVEPOINT descarta o ponto de gravação determinado e as alterações seguintes ao mesmo.

Transação

- Exemplo:

```
INSERT INTO Departamento (ID_Depto, NomeDepto,
ID_Gerente) VALUES (10, 'Marketing', 3);

UPDATE Funcionario SET salario = salario * 1.05
WHERE ID_Depto = 5;

COMMIT;

DELETE FROM Funcionario;

ROLLBACK;
```

No exemplo, o departamento 10 é inserido, os salários dos funcionários do departamento 5 são atualizados, mas nenhum funcionário é excluído.

Transação

- Exemplo:

```
INSERT INTO Departamento (ID_Depto, NomeDepto,
ID_Gerente) VALUES (11, 'RH', 2);

SAVEPOINT ponto1;

DELETE FROM Funcionario;

ROLLBACK TO SAVEPOINT ponto1;

UPDATE Funcionario SET salario = salario * 1.05
WHERE ID_Depto = 5;

COMMIT;
```

No exemplo, o departamento 11 é inserido, os salários dos funcionários do departamento 5 são atualizados, mas nenhum funcionário é excluído.

Transação

- Quando uma transação altera o banco, é feita uma cópia dos dados anteriores à alteração no segmento de *rollback*.
  - Se uma outra transação quiser acessar estes dados, serão acessados os dados que estão no segmento de *rollback*.
- Quando uma transação efetua um COMMIT, os dados são definitivamente alterados no banco de dados e a cópia dos dados anteriores é eliminada do segmento de *rollback*.
  - Assim, as outras transações passam a ter acesso aos próprios dados já alterados.
- Quando uma transação efetua um ROLLBACK, os dados anteriores são copiados novamente para o banco de dados, retornando assim à versão anterior dos dados.

Processamento de Transações

- Quando várias transações são emitidas ao mesmo tempo (transações concorrentes), ocorre um entrelaçamento de operações das mesmas.
  - Algumas operações de uma transação são executadas; em seguida, seu processo é suspenso e algumas operações de outra transação são executadas.
  - Depois, um processo suspenso é retomado a partir do ponto de interrupção, executado e interrompido novamente para a execução de uma outra transação.

Processamento de Transações

- As propriedades (**ACID**) de uma transação são:
  - **Atomicidade**: uma transação é uma unidade atômica de processamento; é realizada integralmente ou não é realizada.
  - **Consistência**: uma transação é consistente se levar o banco de dados de um estado consistente para outro estado também consistente.
  - **Isolamento**: a execução de uma transação não deve sofrer interferência de quaisquer outras transações que estejam sendo executadas concorrentemente.
  - **Durabilidade (ou persistência)**: as alterações aplicadas ao banco de dados, por meio de uma transação confirmada, devem persistir no banco de dados, não sendo perdidas por alguma falha.

Processamento de Transações

- O modelo simplificado para processamento de transações concorrentes envolve as seguintes operações:
  - **read_item(X)**: lê um item X do banco de dados e transfere para uma variável X de memória;
  - **write_item(X)**: escreve o valor de uma variável X de memória em um item X do banco de dados.

- Exemplo:

T_1	T_2
read_item(X);	read_item(X);
X := X - N;	X := X + M;
write_item(X);	write_item(X);
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Controle de Concorrência

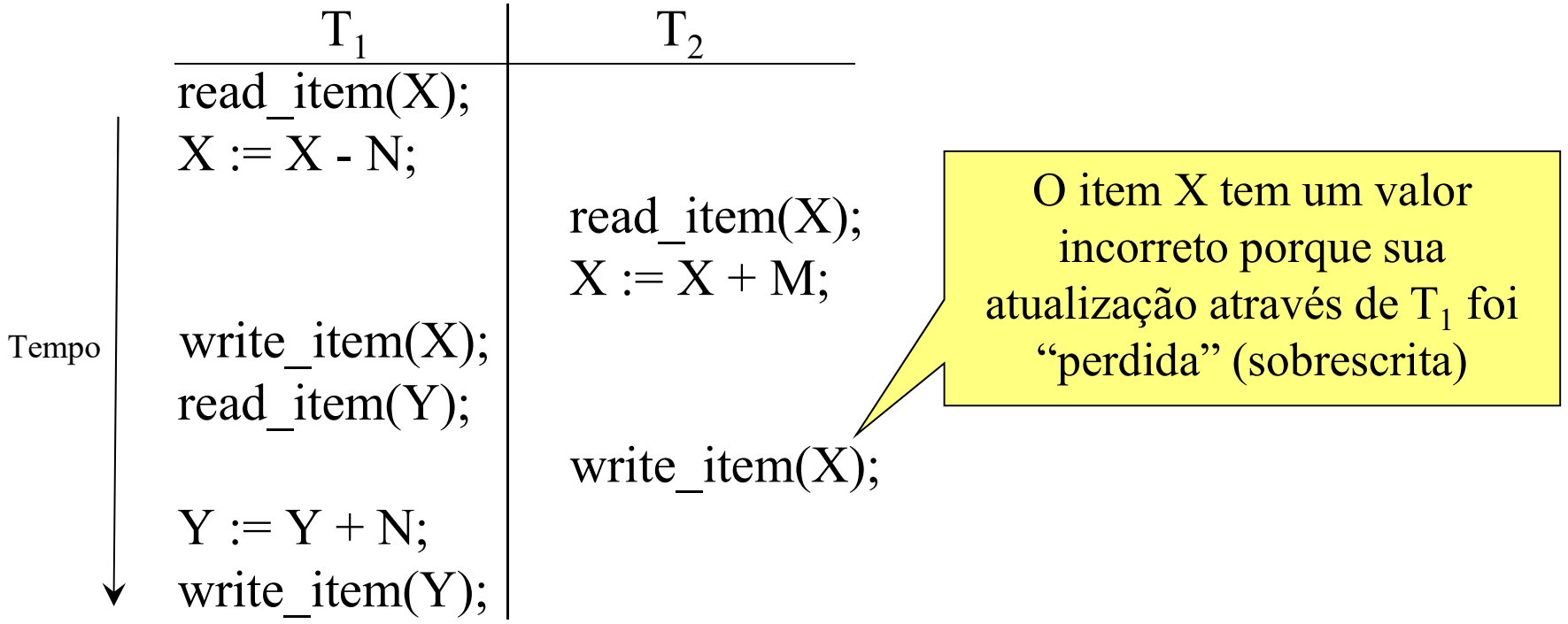
- Técnicas de controle de concorrência são utilizadas para garantir que transações concorrentes sejam executadas adequadamente.

- Muitos problemas podem ocorrer quando transações concorrentes são executadas sem controle, a saber:
  - problema da perda de atualização;
  - problema da atualização temporária (leitura suja);
  - problema da agregação incorreta;
  - problema da leitura não-repetitiva.

Controle de Concorrência

- Problema da perda de atualização:
  - Ocorre quando duas transações que acessam os mesmos itens do banco de dados possuem operações entrelaçadas, de modo que torne incorreto o valor de algum item do banco de dados.

Controle de Concorrência

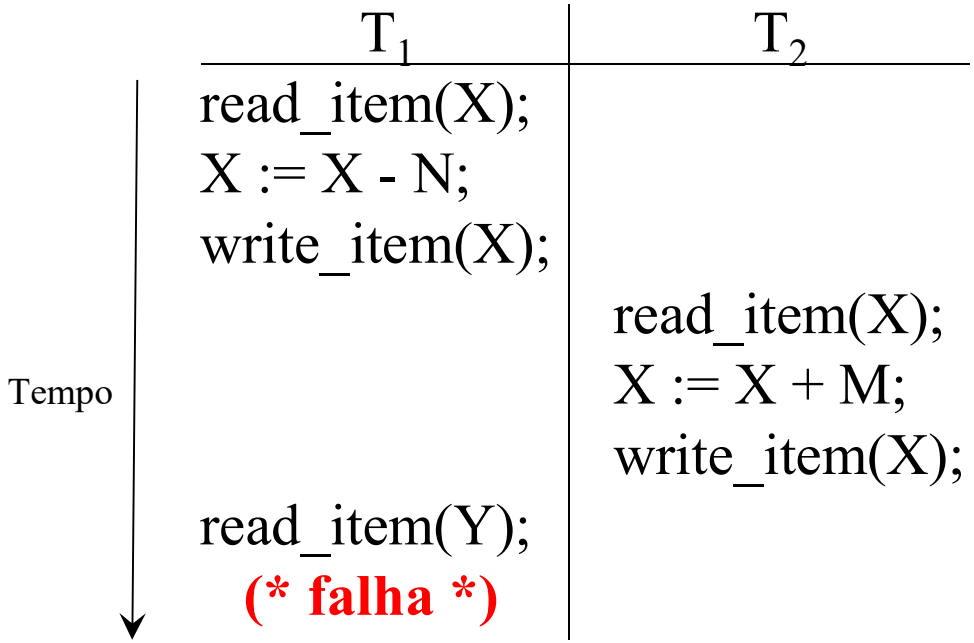


- O valor final do item X em T_2 estará incorreto, porque T_2 lê o valor de X antes que T_1 o altere no banco de dados e, portanto, o valor atualizado resultante de T_1 será perdido.

Controle de Concorrência

- Problema da atualização temporária (leitura suja):
  - Ocorre quando uma transação atualiza um item do banco de dados e, por algum motivo, a transação falha; no caso, o item atualizado é acessado por uma outra transação antes do seu valor ser retornado ao valor anterior.

Controle de Concorrência

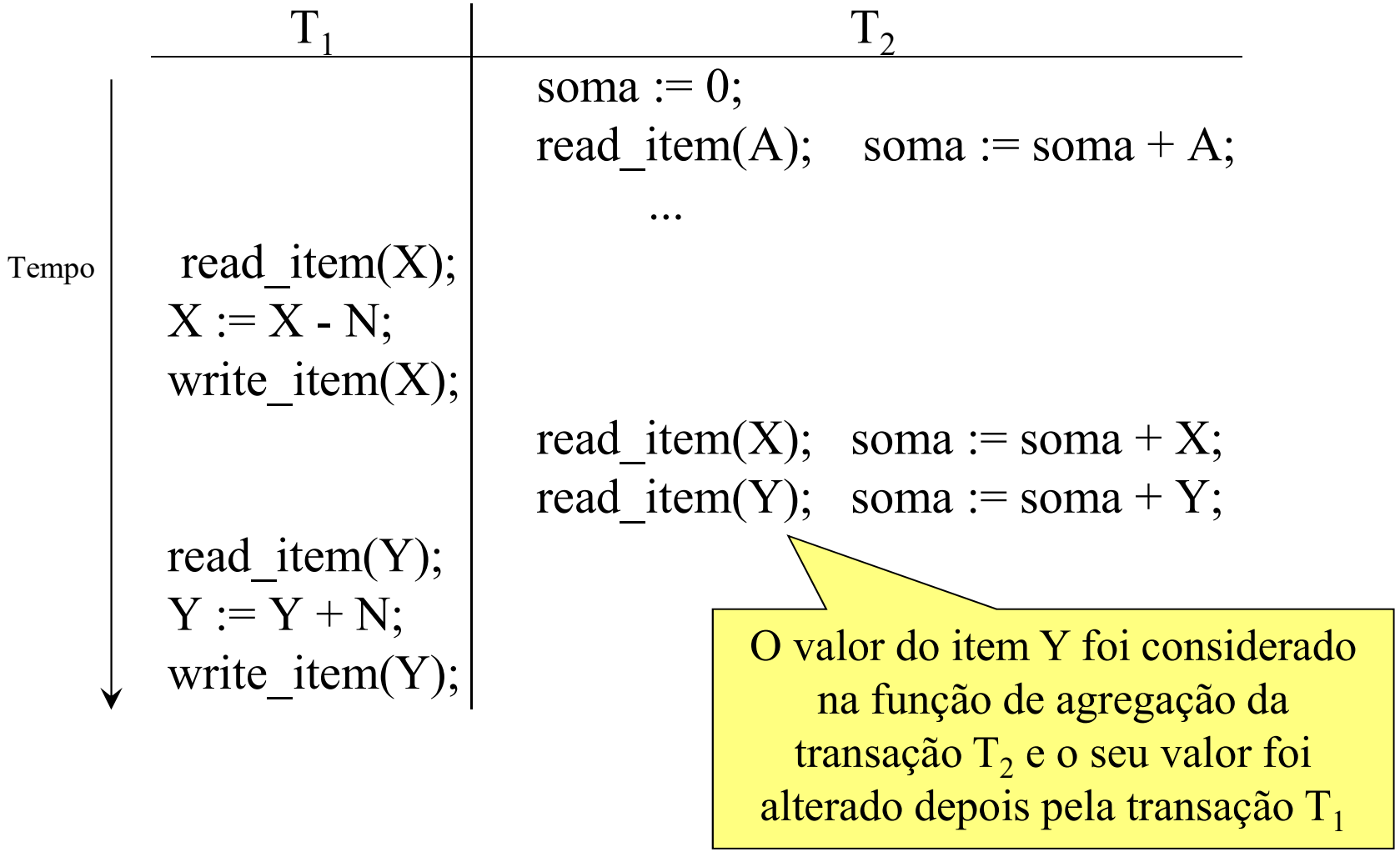


- A transação T_1 falha e o sistema deve alterar o valor do item X para o seu valor anterior; porém, T_2 já leu o valor temporário "incorreto" do item X.

Controle de Concorrência

- Problema da agregação incorreta:
  - Se uma transação estiver calculando uma função de agregação em um número de itens, enquanto outras transações estiverem atualizando alguns desses itens, a função agregada pode considerar alguns itens antes que eles sejam atualizados e outros depois que tenham sido atualizados.

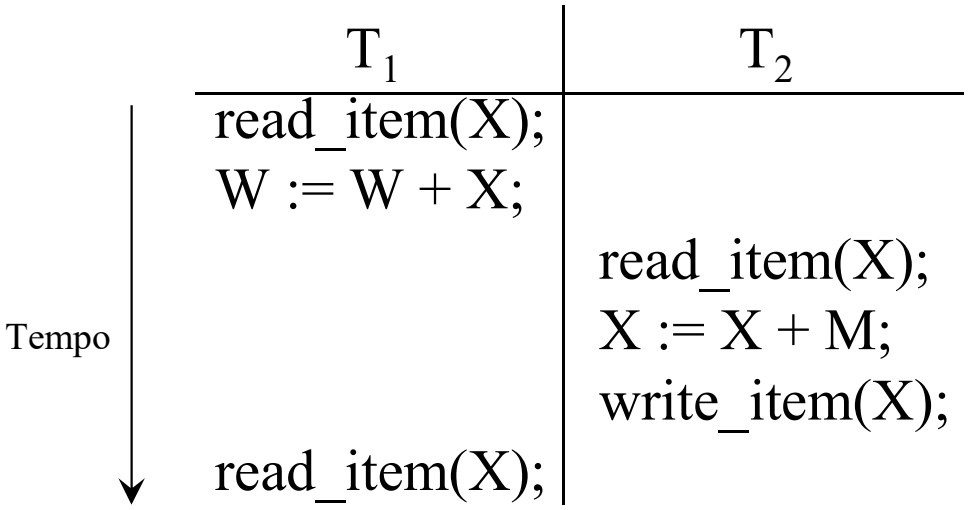
Controle de Concorrência



Controle de Concorrência

- Problema da leitura não-repetitiva:
  - Ocorre quando uma transação T lê um item duas vezes e o item é alterado por uma outra transação T' entre as duas leituras de T. Portanto, T recebe diferentes valores para suas duas leituras do mesmo item.

Controle de Concorrência



Na 2ª leitura do item X pela transação T_1, o valor lido é diferente do valor da 1ª leitura, devido à alteração feita pela transação T_2 no item X entre as duas leituras da transação T_1

Recuperação de Falhas

- O SGBD não deve permitir que algumas operações de uma transação T sejam aplicadas ao banco de dados enquanto outras operações de T não sejam.
  - Isso pode acontecer quando uma transação falha após executar algumas de suas operações (e não todas).

Recuperação de Falhas

- Os tipos de falhas possíveis são:
  - falha no computador;
  - erro de transação ou de sistema;
  - imposição do controle de concorrência;
  - falha no disco;
  - problemas físicos e catástrofes.

- Para os 3 primeiros tipos de falhas, o sistema deve manter informações suficientes para se recuperar da falha.

Recuperação de Falhas - *Log* do Sistema

- Para manter a consistência do banco de dados, o gerenciador de recuperação registra no histórico (*log*), para cada transação, as operações que afetam os valores dos itens do banco:
  - **[start_transaction, T]**: indica que a transação T iniciou sua execução.
  - **[write_item, T, X, *old_value*, *new_value*]**: indica que a transação T alterou o valor do item X do banco de dados de *old_value* (valor antigo) para *new_value* (novo valor).
  - **[read_item, T, X]**: indica que a transação T leu o valor do item X do banco de dados.
  - **[commit, T]**: indica que a transação T foi finalizada com sucesso.
  - **[abort, T]**: indica que a transação T foi abortada.

Recuperação de Falhas - *Log* do Sistema

- Quando ocorre uma falha:
  - as transações inicializadas, mas que não gravaram seus registros de *commit* no *log*, devem ser desfeitas;
  - as transações que gravaram seus registros de *commit* no *log* podem ter que ser refeitas a partir dos registros do *log*.

- Para tanto, no processo de recuperação de falhas relativa a uma transação T, usam-se as operações:
  - UNDO (desfazer): desfaz a transação, ou seja, percorre o *log* de forma retroativa, retornando todos os itens alterados por uma operação *write* de T aos seus valores antigos.
  - REDO (refazer): refaz a transação, ou seja, percorre o *log* para frente, ajustando todos os itens alterados por uma operação *write* de T para seus valores novos.

Escalonamento e Recuperabilidade

- Um escalonamento S de n transações é uma ordenação das operações dessas transações sujeita à restrição de que, para cada transação T_i que participa de S, as operações de T_i em S devem aparecer na mesma ordem em que ocorrem em T_i.

- Notação simplificada para escalonamento:
  - $\mathbf{r_i(X)}$: read_item(X) na transação T_i.
  - $\mathbf{w_i(X)}$: write_item(X) na transação T_i.
  - $\mathbf{c_i}$: commit na transação T_i.
  - $\mathbf{a_i}$: abort na transação T_i.

Escalonamento e Recuperabilidade

- Exemplos de escalonamento:
  - S_a: $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$;
  - S_b: $r_1(X)$; $w_1(X)$; $r_2(X)$; $w_2(X)$; $r_1(Y)$; a_1;

- Duas operações em um escalonamento são ditas conflitantes se:
  - pertencem a diferentes transações;
  - possuem acesso ao mesmo item X;
  - pelo menos uma delas é uma operação write_item(X).

Escalonamento e Recuperabilidade

- Um escalonamento S é dito ser recuperável se nenhuma transação T em S entrar no estado confirmado até que todas as transações T', que tenham escrito um item que T tenha lido, entrem no estado confirmado.

  - S_a: $r_1(X)$; **$w_1(X)$; $r_2(X)$;** $r_1(Y)$; $w_2(X)$; $w_1(Y)$; **c_1;** $r_2(Y)$; c_2;
    S_a é recuperável.

  - S_c: $r_1(X)$; **$w_1(X)$; $r_2(X)$;** $r_1(Y)$; $w_2(X)$; **c_2;** a_1;
    S_c não é recuperável, porque T_2 lê o item X atualizado por T_1, e então T_2 é confirmado antes que T_1 se confirme.

Escalonamento e Recuperabilidade

- Em um escalonamento recuperável, pode ocorrer um fenômeno conhecido como *rollback* em cascata, no qual uma transação não-confirmada tenha que ser desfeita porque leu um item de uma transação que falhou.

  S_e: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; **a_1; a_2;**

- Um escalonamento evita *rollbacks* em cascata se todas as transações no escalonamento lerem somente itens que tenham sido escritos por transações já confirmadas.
  - No escalonamento S_e anterior, $r_2(X)$ deve ser adiada até que T_1 tenha sido confirmada (ou abortada), retardando T_2.

Escalonamento e Recuperabilidade

- Um escalonamento é denominado estrito se todas as suas transações não puderem ler nem escrever um item X até que a última transação que escreveu X tenha sido confirmada (ou abortada).

Seriabilidade de Escalonamentos

- Um escalonamento S é denominado serial se, para todas as transações T participantes do escalonamento, todas as operações de T forem executadas consecutivamente no escalonamento; caso contrário, o escalonamento é denominado não-serial.

- Um escalonamento serial:
  - possui somente uma transação ativa de cada vez;
  - não permite nenhum entrelaçamento de transações;
  - é considerado correto, independente da ordem de execução das transações;
  - limita a concorrência;
  - na prática, é inaceitável.

Seriabilidade de Escalonamentos

- Um escalonamento S de n transações é serializável se for equivalente a algum escalonamento serial das n transações.
  - Dizer que um escalonamento não-serial S é serializável equivale a dizer que ele é correto, já que equivale a um escalonamento serial que é considerado correto.
  - Dois escalonamentos são ditos equivalentes se a ordem de quaisquer duas operações conflitantes for a mesma nos dois escalonamentos.

S_a: $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; c_1; $r_2(X)$; $w_2(X)$; c_2; (serial)

S_b: $r_1(X)$; $w_1(X)$; $r_2(X)$; $w_2(X)$; $r_1(Y)$; $w_1(Y)$; c_1; c_2; (serializável)

S_c: $r_1(X)$; **$r_2(X)$; $w_1(X)$;** $r_1(Y)$; $w_2(X)$; $w_1(Y)$; c_1; c_2;
(não serializável)

Teste de Seriabilidade

- Uma forma de testar a seriabilidade de um escalonamento é através da construção de um grafo de precedência.

- Um grafo de precedência de um escalonamento S é um grafo dirigido G = (N, E), onde N é um conjunto de nodos $\{T_1, T_2, ..., T_n\}$ e E é um conjunto de arcos dirigidos $\{e_1, e_2, ...,e_m\}$ tal que:
  - cada nodo T_i corresponde a uma transação de S;
  - cada arco e_j liga uma transição T_j que possui uma operação conflitante com uma transição T_k.

- Um escalonamento S é serializável se e somente se o grafo de precedência não tiver nenhum ciclo.

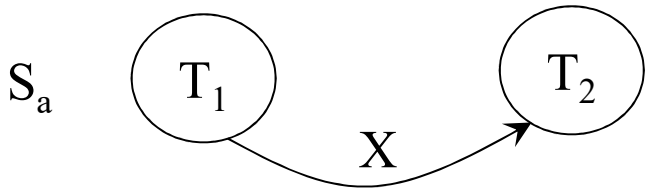
Teste de Seriabilidade

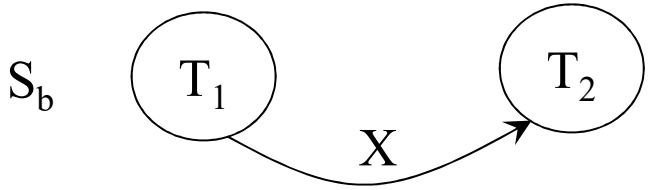
S_a: $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; c_1; $r_2(X)$; $w_2(X)$; c_2;

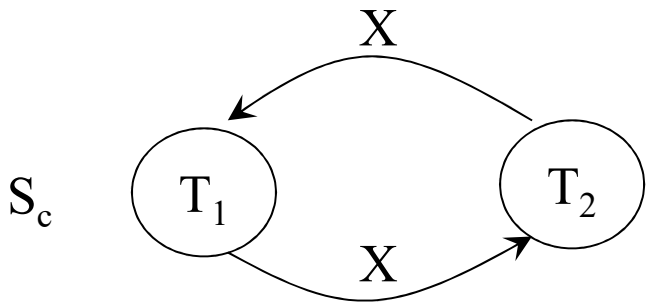
S_b: $r_1(X)$; $w_1(X)$; $r_2(X)$; $w_2(X)$; $r_1(Y)$; $w_1(Y)$; c_1; c_2;

S_c: $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; c_1; c_2;

- Para os escalonamentos S_a, S_b e S_c, os grafos de precedência são:







Suporte de Transações em SQL

- Em SQL, quanto à definição de uma transação, tem-se que:
  - não há uma inicialização explícita *(begin transaction)*;
  - o término é necessariamente explícito (*commit* ou *rollback*);
  - o comando SET TRANSACTION especifica as seguintes características de uma transação: modo de acesso, tamanho da área de diagnóstico e nível de isolamento.

- O modo de acesso pode ser READ ONLY (somente leitura) ou READ WRITE (leitura e gravação).
  - READ WRITE permite inserção, remoção e atualização de dados além de criação de comandos a serem executados;
  - READ ONLY permite apenas recuperação de dados;
  - O padrão é READ WRITE, a não ser que o nível de isolamento seja "leitura não efetivada".

Suporte de Transações em SQL

- O tamanho da área de diagnóstico (DIAGNOSTIC SIZE *n*) especifica um valor inteiro *n* que indica o número de condições que podem ser manipuladas simultaneamente na área de diagnóstico.
  - Essas condições fornecem informações sobre as condições de execução (erros ou exceções), ao usuário ou a um programa, para os comandos SQL executados mais recentemente.

Suporte de Transações em SQL

- O nível de isolamento (ISOLATION LEVEL) pode ser READ UNCOMMITTED (leitura não efetivada), READ COMMITED (leitura efetivada), REPEATABLE READ (leitura repetível) ou SERIALIZABLE (serializável).
  - O nível padrão é SERIALIZABLE que garante isolamento total.
  - Para os demais níveis, as seguintes violações podem ocorrer:
    - Leitura suja: T_1 pode ler uma atualização ainda não efetivada de T_2; se T_2 falhar, T_1 terá lido um valor que não existe e é incorreto.
    - Leitura não-repetitiva: T_1 pode ler um valor de uma tabela; se T_2, depois, atualizar esse valor e T_1 lê-lo novamente, T_1 verá um valor diferente.
    - Fantasma: T_1 pode ler várias linhas de uma tabela (condição do *where*); se T_2 inserir uma nova linha na tabela lida por T_1, que satisfaça a mesma condição da leitura feita, e se T_1 repetir a leitura, verá uma nova linha "fantasma" que antes não existia.

Suporte de Transações em SQL

- A tabela abaixo sintetiza as possíveis violações para os diferentes níveis de isolamento.

Nível de isolamento	Leitura suja	Leitura não-repetitiva	Fantasma
READ UNCOMMITTED	sim	sim	sim
READ COMMITTED	não	sim	sim
REPEATABLE READ	não	não	sim
SERIALIZABLE	não	não	Não

Suporte de Transações em SQL

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSACTION READ WRITE
         DIAGNOSTIC SIZE 5  ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPREGADO (PNOME, UNOME, SSN, DNO, SAL)
         VALUES ('Ana', 'Silva', '1233215', 2, 5000);
EXEC SQL UPDATE EMPREGADO SET SAL = SAL * 1.1 WHERE DNO = 2;
EXEC SQL COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END: ...;
```

- Se ocorrer erro em algum comando SQL, a transação inteira será revertida: restauração dos salários e remoção da 'Ana'.
- A SQL oferece várias facilidades para o tratamento de transações.
  - O DBA pode melhorar o desempenho de transações pelo relaxamento da serialização, caso seja aceitável na aplicação.