Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

Fevereiro, 2020

Seção 21

Processamento de Transações

Sistemas de monousuário *versus* multiusuário

- SGDB **monousuário**: um usuário de cada vez pode utilizar o sistema. Restrito a sistemas de computador pessoal.
- SGBD **multiusuário**: muitos usuários podem utilizar o sistema **simultaneamente**.

  São exemplos, banco de dados usados em *bancos, agências de seguros, mercado de ações, supermercados,* etc.

Processamento de transações

Multiprogramação: permite que um sistema operacional execute vários programas (ou **processos**) ao mesmo tempo.

Sistemas operacionais de multiprogramação, para uma CPU:

- Executam alguns comandos de um processo, depois o suspendem e executam alguns comandos do processo seguinte, e assim por diante.
- A execução simultânea do processo é **intercalada**.
- E a intercalação impede que a CPU permaneça ociosa durante o tempo de E/S, e que um processo longo atrase os demais processos.

Se o sistema possuir múltiplos processadores e hardware (CPUs), o **processamento paralelo** é possível.

Sistemas de processamento de transação

- Consistem em sistemas com grandes bancos de dados e centenas de usuários simultâneos executando transações.
- Exigem alta disponibilidade e tempo de resposta rápido para centenas de usuários simultâneos.

Transação

- Programa em execução que forma uma unidade lógica de processamento de banco de dados.
- Inclui uma ou mais operações de acesso ao banco de dados.
- Embutidas em um programa de aplicação ou especificadas por uma linguagem de consulta (SQL).

Processamento de transações

Transação somente de leitura: suas operações não atualizam o banco de dados, apenas recuperam dados.

Transação leitura-gravação: suas operações atualizam o banco de dados.

Transação confirmada (*commited*): suas operações foram concluídas com sucesso e seu efeito é registrado permanentemente no banco de dados.

Transação abortada: suas operações não possuem qualquer efeito no banco de dados ou em quaisquer outras transações.

Processamento de transações

Begin transaction e end transaction

- Especificam os limites de uma transação em um programa de aplicação.
- Um programa de aplicação pode conter mais de uma transação se tiver vários limites.

Conjunto de leitura

- Conjunto de todos os itens que a transação lê.

Conjunto de gravação

- Conjunto de todos os itens que a transação grava.

Itens de banco de dados

- **Coleção de itens de dados nomeados:** representação de um banco de dados.
- **Item de dados:** registro de banco de dados, bloco de disco inteiro ou valor de campo (atributo) individual de algum registro. Cada item possui um nome único.
- **Granularidade:** tamanho de um item de dados.

Operações básicas de uma transação

- ***read_item(x)*:** Lê um item do BD para uma variável do programa.
  1. Ache o endereço do bloco de disco que contém o item X.
  2. Copie esse bloco para um buffer na memória principal.
  3. Copie o item X do buffer para a variável de programa X.

Operações básicas de uma transação

- ***write_item(x)*:** Grava o valor da variável de programa no item de banco de dados.
  1. Ache o endereço do bloco de disco que contém o item *X*.
  2. Copie esse bloco para um buffer na memória principal.
  3. Copie o item *X* da variável de programa *X* para o local correto no buffer.
  4. Armazene o bloco atualizado do buffer de volta no disco.

Controle de concorrência

Problemas encontrados se duas transações forem executadas simultaneamente:

- **O problema da atualização perdida:**

  Ocorre quando duas transações que acessam os mesmos itens de BD têm suas operações intercaladas de modo que isso torna o valor de alguns itens da base de dados incorreto.

- **O problema da atualização temporária (leitura suja):**

  Uma transação atualiza um item do BD e depois a transação falha por algum motivo. Nesse meio tempo, o item atualizado é acessado (lido) por outra transação, antes de ser alterado de volta para seu valor original.

Controle de concorrência

Problemas encontrados se duas transações forem executadas simultaneamente:

- **O problema do resumo incorreto:**

  Se uma transação está calculando uma função de resumo de agregação em itens do BD, enquanto outras transações estão atualizando alguns desses itens, a função de agregação pode calcular valores antes que eles sejam atualizados e outros, depois que eles forem atualizados.

- **O problema da leitura não repetitiva:**

  Uma transação T lê o mesmo item duas e o item é alterado por uma outra transação T' entre as duas leituras. Logo, T recebe valores diferentes para as duas leituras do mesmo item.

Processamento de transações

As falhas são classificadas como falhas de transação, sistema e mídia.

Tipos de falhas:

1. **Falha do computador (falha do sistema):**

   Um erro de hardware, software ou rede no sistema de computação durante a execução da transação.

2. **Erro de transação ou do sistema:**

   Alguma operação na transação pode fazer que esta falhe ou valores de parâmetro errôneos ou erro lógico de programação.

   Além disso, o usuário também pode interromper a transação durante sua execução.

Tipos de falhas:

3. **Erros locais ou condições de exceção detectadas pela transação:**

   Durante a execução da transação, podem ocorrer condições que necessitam de cancelamento. Essa exceção poderia ser programada na própria transação e não seria considerada uma falha.

4. **Imposição de controle de concorrência:**

   O método de controle de concorrência pode abortar uma transação porque ela viola a serialização ou pode abortar uma ou mais transações para resolver um deadlock.

Tipos de falhas:

5. **Falha de disco:**

   Blocos de disco podem perder seus dados devido a um defeito de leitura, gravação ou por causa de uma falha da cabeça de leitura/gravação.

6. **Problemas físicos e catástrofes**:

   Falha de energia, incêndio, roubo, regravação de discos ou fitas por engano e montagem da fita errada pelo operador.

Operações adicionais

O sistema precisa registrar quando cada transação *começa*, *termina* e *confirma* ou *aborta*. O gerenciador de recuperação do SGBD acompanha as seguintes operações:

- ***BEGIN_TRANSACTION*:** Início da execução da transação.
- ***READ* ou *WRITE*:** Operações de leitura ou gravação nos itens de banco de dados de uma transação.
- ***END_TRANSACTION*:** Final da execução da transação. Nesse ponto é necessário verificar se a transação será confirmada ou abortada.

Operações adicionais

O sistema precisa registrar quando cada transação *começa*, *termina* e *confirma* ou *aborta*. O gerenciador de recuperação do SGBD acompanha as seguintes operações:

- ***COMMIT_TRANSACTION*:** Atualizações executadas pela transação podem ser confirmadas (*commited*) ao banco de dados e não serão desfeitas.
- ***ROLLBACK* (ou *ABORT*):** Mudanças ou efeitos que a transação possa ter aplicado ao banco de dados precisam ser desfeitos.

Estados de transação

- **Estado ativo:** após iniciar a execução, onde pode executar suas operações *READ* e *WRITE*.
- **Estado parcialmente confirmado:** transação terminada.
- **Estado confirmado:** alguns protocolos de recuperação precisam garantir que não há falhas.

  Caso não ocorra falhas, a transação alcançou seu ponto de confirmação e todas as suas mudanças precisam ser gravadas permanentemente no BD.

Estados de transação

- **Estado de falha:** se uma das verificações falhar ou se a transação for abortada. A transação pode ter de ser cancelada para desfazer o efeito de suas operações. As transações com falha ou abortadas podem ser reiniciadas depois.
- **Estado terminado:** corresponde a transação que sai do sistema.

Log do sistema

- Arquivo sequencial, apenas para inserção, que é mantido no disco.
- Registra todas as operações de transação que afetam os valores dos itens de banco de dados, bem como outras informações que podem ser necessárias para permitir a recuperação de falhas.
- A noção de recuperação de uma falha de transação equivale a desfazer ou refazer operações de transação individualmente com base no *log*.

Conceitos de transação e sistema

Registros de log:

1. [***start_transaction, T***]**.** Indica que a transação *T* iniciou sua execução.

2. [***write_item, T, X, valor_antigo, valor_novo***]**.** Indica que a transação *T* mudou o valor do item do banco de dados *X* de *valor_antigo* para *valor_novo*.

3. [***read_item, T, X***]**.** Indica que a transação *T* leu o valor do item de banco de dados *X*.

4. [***commit, T***]**.** Indica que a transação *T* foi concluída com sucesso, e afirma que seu efeito pode ser confirmado.

5. [***abort, T***]**.** Indica que a transação *T* foi abortada.

Ponto de confirmação

- Uma transação alcança seu **ponto de confirmação** quando todas as suas operações tiverem sido executadas com sucesso e registradas no *log*.
- É comum manter um ou mais blocos do arquivo de *log* nos buffers da memória principal (*buffer de log*), até que eles sejam preenchidos com entradas de *log* e, depois, gravá-los de volta ao disco apenas uma vez.
- Antes que uma transação alcance seu ponto de confirmação, qualquer parte do *log* que ainda não tenha sido gravada no disco deve agora sê-lo (**gravação forçada**).

Propriedades desejáveis das transações

*Propriedades **ACID** devem ser impostas pelos métodos de controle de concorrência e recuperação do SGBD.*

- **Atomicidade.** Uma transação é uma unidade de processamento atômica; ela deve ser realizada em sua totalidade ou não ser realizada em sua totalidade ou não ser realizada de forma alguma.
- **Durabilidade ou permanência.** As mudanças aplicadas ao banco de dados pela transação confirmada precisam persistir (não devem ser perdidas).

Propriedades desejáveis das transações

*Propriedades **ACID** devem ser impostas pelos métodos de controle de concorrência e recuperação do SGBD.*

- **Preservação da consistência.** Se uma transação for completamente executada do início ao fim sem interferência deve-se levar o banco de dados de um estado consistente para outro.
  - **Estado de bando de dados**: coleção de todos os itens de dados armazenados no banco de dados em determinado período.
  - **Estado consistente:** banco de dados satisfaz as restrições especificadas no esquema.

Propriedades desejáveis das transações

*Propriedades **ACID** devem ser impostas pelos métodos de controle de concorrência e recuperação do SGBD.*

- **Isolamento.** A execução de uma transação não deve ser interferida por quaisquer outras transações que acontecem simultaneamente.
  - **Isolamento nível 0 (zero)**: transação não grava sobre as leituras sujas das transações de nível mais alto.
  - **Isolamento nível 1 (um)**: não possui atualizações perdidas.
  - **Isolamento nível 2 (dois)**: não possui atualizações perdidas ou leituras sujas.
  - **Isolamento nível 3 (três)**: além das propriedades de nível 2, leituras repetitivas.

Escalonamento de transações

Escalonamentos (históricos) de transações

- Ordem da execução das operações que estão executando simultaneamente em um padrão intercalado.
- As operações de uma transação precisam aparecer na mesma ordem em que ocorrem.
- A ordem das operações é uma ordenação total, se para duas operações quaisquer no schedule uma precisa ocorrer antes da outra.

Escalonamento de transações

Escalonamentos (históricos) de transações

- Notação para schedule: *begin_transaction* (***b***), *read_item* (***r***), *write_item* (***w***), *end_transaction* (***e***), *commit* (***c***) ou *abort* (***a***).
- E acrescenta como **subscrito** a *id* da transação a cada operação no schedule.

$$S: r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); a_1$$

- Note que a transação foi cancelada após sua operação *read_item(Y)*.

Escalonamento de transações

- Note que a transação foi cancelada após sua operação *read_item(Y)*.
- Duas operações em um schedule estão em conflito se satisfazerem todas as condições a seguir:
  - Pertencem a diferentes transações;
  - Acessam o mesmo item X;
  - Pelo menos uma das operações é um *write_item(X)*.
- Duas operações estão em conflito se a mudança de sua ordem puder resultar em algo diferente.
  - **Conflito de leitura-gravação**
  - **Conflito de gravação-gravação**

Escalonamento de transações

Escalonamento completo

Condições:

1. As operações são exatamente as das transações, incluindo uma operação de confirmação ou cancelamento como última operação em cada transação;

2. Para qualquer par de operações da mesma transação, sua ordem de aparecimento relativa na schedule é a mesma que a ordem de aparecimento na transação;

3. Para duas operações quaisquer em conflito, uma das duas precisa ocorrer antes da outra schedule.

Escalonamento de transações

Escalonamento completo

- Como cada transação é confirmada ou cancelada, um schedule completo **não terá quaisquer transações ativas** ao final do schedule;

- Difícil encontrar schedules completos em um sistema de processamento de transação.
  - Novas transações estão sendo continuamente submetidas ao sistema;
  - Convém definir a **projeção confirmada C(S)**:
    - Inclui apenas as operações em **S** que pertencem a transações confirmadas.

Escalonamento de transações

- Importante caracterizar os tipos de escalonamentos para os quais a recuperação é possível;
- Quando uma transação T é confirmada, nunca deve ser necessário cancelar T.
  - Garante a não violação a propriedade de durabilidade no SGBD.

1. Schedules **recuperáveis**;
2. Schedules **não recuperáveis**;

Escalonamento de transações

1. Escalonamentos recuperáveis

- Um schedule **S** é recuperável se nenhuma transação **T** em **S** for confirmada até que todas as transações **T′**, que tiverem gravado algum item **X** que **T** lê, sejam confirmadas;
- Uma transação **T** lê a transação **T′** em um schedule **S** se algum item **X** for gravado primeiro por **T′** e depois lido por **T**;
- **T′** não deve ser cancelado antes que **T** leia o item **X**.

1. Escalonamentos recuperáveis

- Considere o escalonamento $\mathbf{S_a}'$ a seguir:

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

- S_a' é recuperável, embora sofra do problema da atualização (tratado pela teoria da serialização).

2 Escalonamentos não recuperáveis

- Considere os escalonamentos (parciais) $\mathbf{S_c}$ e $\mathbf{S_d}$ a seguir:

  S_c: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; c_2; a_1;

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

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

- $\mathbf{S_c}$ não é recuperável porque $\mathbf{T_2}$ lê o item $\mathbf{X}$ de $\mathbf{T_1}$, mas $\mathbf{T_2}$ confirma antes que $\mathbf{T_1}$ confirme.

- O problema ocorre se $\mathbf{T_1}$ abortar depois da operação $\mathbf{c_2}$ em $\mathbf{S_c}$, então o valor de X que $\mathbf{T_2}$ lê não é mais válido e $\mathbf{T_2}$ precisa ser abortado **depois** de ser confirmado, levando a um schedule que **não é recuperável**.

Escalonamento de transações

- Em um escalonamento recuperável, nenhuma transação confirmada precisa ser cancelada.
  - A definição de transação confirmada como durável não é violada;
  - Possibilidade do **rollback em cascata**:
    - Uma transação não confirmada é cancelada porque leu um item de uma transação que falhou;
    - Exemplificado em S_c - a transação R_2 foi cancelada porque leu o item X de T_1, e T_1 então foi cancelada.

Escalonamento de transações

- **Escalonamento sem cascata** (evita rollback em cascata)
  - Cada transação nele ler apenas itens que tenham sido gravadas por transações;
  - Todos os itens lidos não serão descartados;
  - Neste caso, o comando $r_2(X)$ nos schedules S_d e S_c devem ser adiados até depois que T_1 tiver sido confirmada.

Escalonamento de transações

- **Escalonamento estrito**
  - As transações não podem ler nem gravar um item X até que a última transação que gravou X tenha sido confirmada (ou cancelada);
  - Simplificam o processo de recuperação;
  - O processo de desfazer uma operação **write_item(X)** de uma transação abortada serve apenas para restaurar a **imagem anterior** do item de dados X;
  - Pode não funcionar para schedules recuperáveis ou sem cascata.

Escalonamento de transações

- A seguir os tipos de escalonamentos serão caracterizados corretos quando transações concorrentes estão sendo executadas.
  - **Escalonamentos serializáveis.**
- Suponha que dois usuários submetam as transações no SGBD T_1 e T_2 :

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

T_2
read_item(X);
X := X+M;
write_item(X);

Escalonamento de transações

- Se nenhuma intercalação de operações for permitida, os resultados possíveis são:

  1. Executar todas as operações de T_1 seguidas por todas as operações de T_2;
  2. Executar todas as operações de T_2 seguidas por todas as operações de T_1.

Escalonamento de transações

- Escalonamentos possíveis:

calonamento A:

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

Escalonamento de transações

- Escalonamentos possíveis:

calonamento B:

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

Escalonamento de transações

- Escalonamentos possíveis:

calonamento C:

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

Escalonamento de transações

- Escalonamentos possíveis:

calonamento D:

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

Escalonamento de transações

- Se a intercalação de operações for permitida, haverá muitas ordens possíveis para execução das operações individuais.
- **Serialização de escalonamentos**
  - Identifica quais escalonamentos estão corretos quando as execuções da transação tiverem intercalação de suas operações nos escalonamentos.

Escalonamento de transações

Escalonamentos seriais

- Os escalonamentos de exemplo **a** e **b** são seriais;
- As operações de cada transação são executadas consecutivamente, sem intercalações;
- Somente uma transação ativa por vez;
  - O commit (ou abort) da transação ativa inicia a execução da próxima transação.
- Desvantagem: Limitam a concorrência - desperdício de tempo de CPU.
  - Os tornam inaceitáveis na prática.

Escalonamentos não seriais

- Os escalonamentos de exemplo **c** e **d** são não seriais;
- Cada sequência intercala operações das duas transações;
- Necessário determinar quais schedules sempre produzem o resultado correto e quais podem gerar resultados errôneos:
  - Processo chamado de **serialização** de um schedule.

Escalonamento de transações

Escalonamento serializável

- Um escalonamento **S** de **n** transações é serializável se for equivalente a algum escalonamento serial das mesmas **n** transações;
- Existem **n!** escalonamentos seriais possíveis de **n** transações e muito mais escalonamentos não seriais possíveis.
- Pode-se formar dois grupos dos escalonamentos não seriais:
  1. Equivalentes a um (ou mais) dos escalonamentos seriais - são serializáveis;
  2. Não são equivalentes a qualquer escalonamento serial - não são serializáveis.

Escalonamento serializável

- Dizer que um escalonamento é serializável quer dizer que ele é correto.

Quando dois escalonamentos são considerados equivalentes?

As operações aplicadas a cada item de dados afetado pelos escalonamentos devem ser aplicadas a esse item nos dois escalonamentos, **na mesma ordem**. Assim, verificaremos se estes **produzirão o mesmo estado final do banco de dados**.

Escalonamento de transações

Escalonamento serializável

- Equivalência de conflito
  - Dois escalonamentos são considerados **equivalentes em conflito** se a ordem de duas operações for a mesma nos dois escalonamentos;
  - Escalonamento S é **serializável de conflito** se ele for equivalente (em conflito) a algum escalonamento serial S′;
    - Reordena as operações **não em conflito** em **S** até formar o escalonamento serial equivalente **S′**.

Escalonamento serializável

- Equivalência de conflito
  - O escalonamento **D** é equivalente ao escalonamento serial **A**;
  - O escalonamento **C** não é equivalente a qualquer um dos possíveis escalonamentos seriais **A** e **B**, e, portanto, não é serializável.

Escalonamento de transações

Testando a serialização de conflito

- Existe um algoritmo simples para determinar se um escalonamento é serializável de conflito ou não;
- Verifica apenas as operações **read_item** e **write_item** para construir um **grafo de precedência**;
  - Grafo direcionado $G = (N, E)$;
  - Conjunto de nós $N = T_1, T_2, ..., T_N$;
  - Conjunto de arestas direcionadas $A = a_1, a_2, ..., a_n$;
  - Um nó para cada transação T_i no escalonamento.

Testando a serialização de conflito - Algoritmo

1. Para cada transação T_i participante no schedule **S**, crie um nó rotulado T_i no grafo de precedência;

2. Para cada caso em **S** onde T_i executa um **read_item(X)** depois de T_i executar um **write_item(X)**, crie uma aresta T_i - T_j;

3. Para cada caso em **S** onde T_i executa um **write_item(X)** após T_i executar um **read_item(X)**, crie uma aresta T_i - T_j;

Testando a serialização de conflito - Algoritmo

4 Para cada caso em **S** onde T_i executa um **write_item(X)** após T_i executar um **write_item(X)**, crie uma aresta T_i - T_j;

5 O escalonamento **S** é serializável se, e somente se, o grafo de precedência não tiver ciclos.

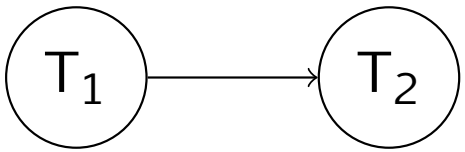
Testando a serialização de conflito - Observações

- Uma aresta de T_i para T_j significa que a transação T_i precisa vir antes da transação T_j em qualquer escalonamento serial que seja equivalente a **S**;
- Vários escalonamentos seriais podem ser equivalentes a **S** se o grafo de precedência para **S** não tiver ciclo;
- Se o grafo tiver ciclo, é fácil mostrar que não pode-se criar qualquer escalonamento serial equivalente, de modo de **S** não é serializável.

Testando a serialização de conflito

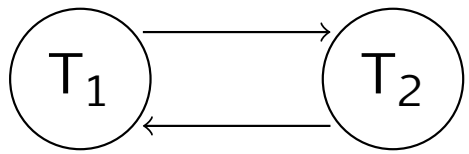
- Grafo de precedência criado para o escalonamento **A**:

- Grafo de precedência criado para o escalonamento **D**:



Testando a serialização de conflito

- Grafo de precedência criado para o escalonamento **C**:



Observe que o grafo para o escalonamento **C** tem um ciclo e, portanto, não é serializável.

Escalonamento de transações

- Pontos de atenção:
  - **Um escalonamento ser serializável é diferente de ser serial.**
  - Um escalonamento serial representa um processamento ineficiente;
  - Um escalonamento serializável oferece os benefícios da execução concorrente sem abrir mão de qualquer exatidão.
- Difícil, na prática, testar a serialização de um escalonamento.
  - Fatores definidos e iniciados pelo Sistema Operacional.

Escalonamento de transações

- Técnica para obtenção de resultado de "teste"da serialização de um escalonamento:
  - Determinação de métodos ou protocolos que garantem a serialização, sem ter de testar os próprios schedules.
- Quando as transações são submetidas, é difícil determinar quando um escalonamento começa e termina.
  - Solução: adaptação da teoria da serialização;
  - Considera-se somente a projeção confirmada de um escalonamento **S**.

Escalonamento de transações

- Adaptação da teoria da serialização:
  - Pode-se definir um um escalonamento **S** para ser serializável se sua projeção confirmada **C(S)** for equivalente a algum escalonamento serial.
    - Apenas transações confirmadas são garantidas pelo SGBD.
  - Técnica de bloqueio de duas fases:
    - Mais comum para controle de concorrência, que garantem a serialização;
    - Consiste no **bloqueio de itens de dados** para impedir que transações concorrentes infiram umas com as outras. Sempre na imposição de uma condição adicional que garanta a serialização.

Equivalência de visão

- Outra definição de equivalência de escalonamentos;
- Dois escalonamentos **S** e **S′** são considerados **equivalentes de visão** se:

1. O mesmo conjunto de transações participa em **S** e **S′**, e **S** e **S′** incluem as mesmas operações dessas transações;

2. Para qualquer operação $\mathbf{r_i(X)}$ de $\mathbf{T_i}$ em **S**, se o valor de X lido pela operação tiver sido gravado por uma operação $\mathbf{w_i(X)}$ de $\mathbf{T_i}$, a mesma condição deve ser mantida para o valor de X lido pela operação $\mathbf{r_i(X)}$ de $\mathbf{T_i}$ em **S′**;

Equivalência de visão

- Outra definição de equivalência de escalonamentos;
- Dois escalonamentos **S** e **S′** são considerados **equivalentes de visão** se:

  3 Se a operação $w_k(Y)$ de T_k for a última operação a gravar o item Y em S, então $w_k(Y)$ de T_k também deve ser a última operação a gravar o tem Y em **S′**.

Equivalência de visão

- Desde que cada operação de leitura leia o resultado da mesma operação de gravação nos dois escalonamentos, as operações de gravação de cada transação devem produzir os mesmos resultados;
- As leituras veem a mesma visão nos dois escalonamentos;
- Um escalonamento **S** é considerado **serializável de visão** se for equivalente de visão a um escalonamento serial.

Suposição de gravação restrita

- Qualquer operação de gravação $w_i(X)$ em T_i é precedida por um $r_i(X)$ em T_i e que o valor gravado por $w_i(X)$ em T_i depende apenas do valor de X lido por $r_i(X)$;
- O cálculo do novo valor de X é uma função $f(X)$ baseada no valor antigo de X lido no banco de dados;
- As definições de **serialização de conflito** e **serialização de visão** se esta condição se mantiver em todas as transações no escalonamento.

Escalonamento de transações

Gravação cega

- Gravação em uma transação T em um item X que não depende do valor de X, de modo que não é precedida por uma leitura de X na transação T.

- A **serialização de visão** é menos restrita do que a **serialização de conflito** sob a **suposição de gravação irrestrita**;
- O valor gravado por uma operação $w_i(X)$ em T_i pode ser independente no banco de dados;
- É possível quando as gravações cegas são permitidas.

Suporte para transação em SQL

Transação SQL

- Unidade lógica de trabalho e tem garantias de ser atômica;
- Seu início é feito implicitamente quando instruções SQL são encontradas;
- Seu fim precisa ter uma instrução explícita:
  - Comando **COMMIT** ou **ROLLBACK**.
- Características: (definidas por **SET TRANSACTION**)
  1. Modo de acesso;
  2. Tamanho da área de diagnóstico;
  3. Nível de isolamento.

Suporte para transação em SQL

Transação SQL

1. Modo de acesso
   - **READ WRITE**
     - Modo default (exceto quando o nível de isolamento é READ UNCOMMITTED);
     - Permite a execução de comandos de seleção, atualização, inserção, exclusão e criação.
   - **READ ONLY**
     - Somente para recuperação de dados.

Transação SQL

2 Tamanho da área de diagnóstico (**DIAGNOSTIC SIZE n**)

- Especifica um valor inteiro **n** que indica o número de condições que podem ser mantidas de maneira simultânea na área de diagnóstico;
- Fornecem informações de feedback (erros ou exceções) ao usuário ou programas nas **n** instruções SQL executadas.

Suporte para transação em SQL

Transação SQL

3 Nível de isolamento (**ISOLATION LEVEL**)

- READ UNCOMMITTED
- READ COMMITTED
- SERIALIZABLE (REPEATABLE READ)
  - Modo default;
  - Se uma transação é executada em um nível de isolamento inferior a SERIALIZABLE, uma ou mais violações a seguir podem ocorrer: **Leitura suja**, **Leitura não repetitiva** e **Fantasmas**.

Suporte para transação em SQL

Transação SQL

3.1 Leitura suja

- Uma transação T_1 pode ler a atualização de uma transação T_2, que ainda não foi confirmada;
- Se T_2 falhar e for abortada, T_1 teria lido um valor que não existe e é incorreto.

3.2 Leitura não repetitiva

- Uma transação T_1 pode ler determinado valor de uma tabela;
- Se outra transação T_2 mais tarde atualizar esse valor e T_1 ler o valor novamente, T_1 verá um valor diferente.

Transação SQL

3.3 Fantasmas

- Uma transação T_1 pode ler um conjunto de linhas de uma tabela, talvez com base em alguma condição especificada na cláusula SQL WHERE;
- Se uma transação T_2 inserir uma nova linha que também satisfaça a condição da cláusula WHERE usada em T_1, na tabela usada por T_1;
- Se T_1 for repetida, então T_1 verá um fantasma, uma linha que anteriormente não exista.

Suporte para transação em SQL

Transação SQL

- Violações para diferentes níveis de isolamento:

Tipo de violação			
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

OBRIGADO

Wladmir Cardoso Brandão
www.wladmirbrandao.com

"Science is more than a body of knowledge. It is a way of thinking."
Carl Sagan