Transação, Concorrência e Recuperação de Banco de Dados

INF 220-Banco de Dados I – Prof. Jugurta Lisboa Filho

Apresentação elaborada por Glauber Costa (Estágio em Ensino 2011 – PPGCC/UFV)

Agenda

- Conceitos fundamentais e processamento de transações
- Transações
- Técnicas de controle de concorrência
  - Baseadas em Locks
  - Outras técnicas de controle de concorrência
  - Ocorrência de deadloks
- Níveis de isolamento
- Recuperação de Falhas

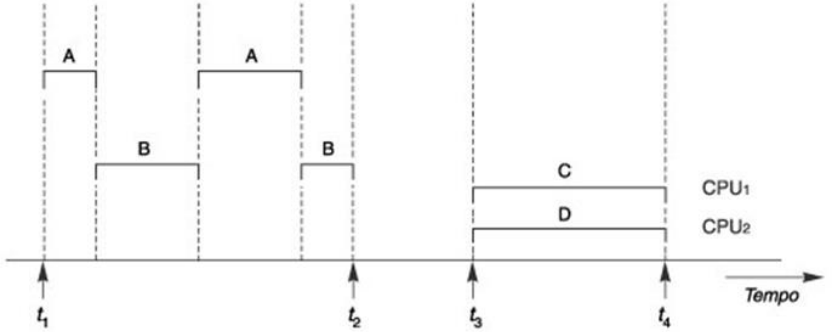
Conceitos fundamentais e Processamento de Transações

Transação, Concorrência e Recuperação à Falhas

- Ambiente multi-tarefa
  - Vários acessos ao BD são realizados simultaneamente.
  - É preciso controlar a sequência desses acessos, de modo a garantir que o BD sempre esteja em um estado consistente.

Processamento intercalado X Processamento paralelo

- Classificação de um SGBD:
  - Monousuário
  - Multiusuário
    - Vários processos rodam ao mesmo tempo



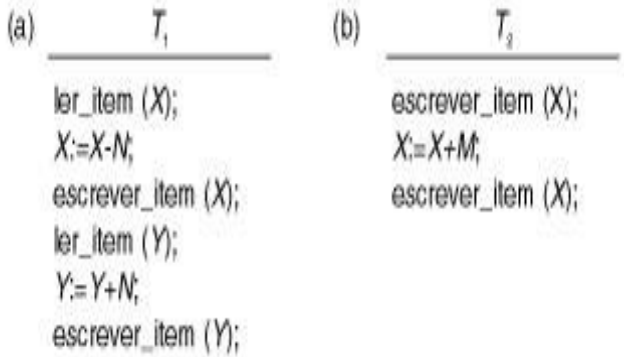
Processamento intercalado *versus* processamento paralelo de transações concorrentes.

Transação, Concorrência e Recuperação à Falhas

- Existem atualizações no BD que envolvem várias operações
  - se ocorrer alguma falha que não permita a execução total de todas as operações o BD ficará em um estado inconsistente.
- Para impedir que isso aconteça é necessário algum mecanismo de recuperação de falhas.

Transação, Concorrência e Recuperação à Falhas

- Transação
  - Uma unidade de processamento do BD que inclui uma ou mais operações de acesso, como recuperação, inserção, remoção ou alteração de dados.



Duas transações simples.

(a) Transação T_1.

(b) (b) Transação T_2.

Transação, Concorrência e Recuperação à Falhas

- **Exemplos da necessidade de controle de concorrência.**
  - Suponha a tabela I contendo informações sobre saldo de contas bancárias:
  - Suponha também duas operações de atualização sobre a conta 10189-9:
    - I) acrescentar R$50,00
    - II) acrescentar R$70,00 à Conta.
  - As operações devem ser feitas de modo a levar o BD de um estado consistente à outro.

Estado antes das operações

NumCC	Saldo
10189-9	200,00
20645-7	300,00

Estado desejado após as operações

NumCC	Saldo
10189-9	320,00
20645-7	300,00

Lost Update

- Cada uma dessas operações é composta por duas suboperações de acesso:
  - uma de leitura e
  - uma de escrita do valor atualizado.

NumCC	Saldo
10189-9	270,00
20645-7	300,00

Operação 1	Operação 2
→ leitura do saldo	→ leitura do saldo
→ Escrita do Saldo+50,00	→ Escrita do Saldo+70,00

A atualização da operação 1 foi perdida!
Este problema é conhecido (*Lost Update*)

Dirty Read

- Exemplo de Dirty Read
  - Suponha a mesma tabela. Uma operação deve transferir R$50,00 da conta 10189-9 para a conta 20645-7 e uma segunda operação deve somar R$70,00 à conta 10189-9.

Estado antes das operações

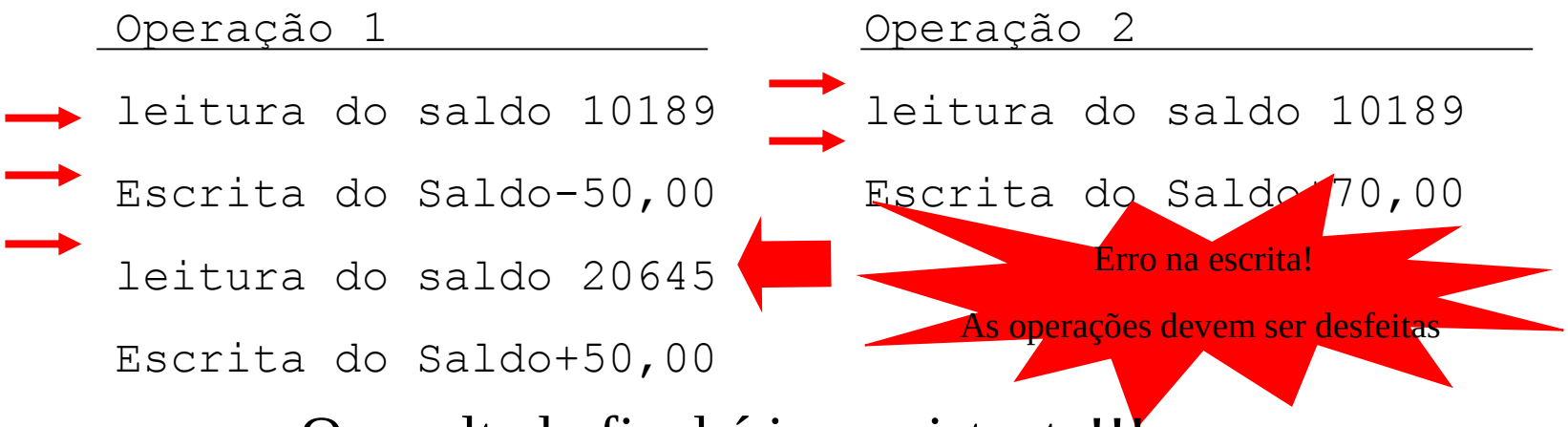
NumCC	Saldo
10189-9	200,00
20645-7	300,00

Estado desejado após as operações

NumCC	Saldo
10189-9	220,00
20645-7	350,00

Dirty Read

NumCC	Saldo
10189-9	220,00
20645-7	300,00



O resultado final é inconsistente!!!

Problemas com funções de agregação

- Outro problema pode ocorrer quando se trabalha funções de agregação sobre tabelas.
  - Ex.: uma operação pode estar realizando a soma do valor total das contas correntes enquanto outra pode estar adicionando novas contas.
  - Sem controle de concorrência o valor da totalização pode não refletir a situação anterior e nem a posterior às inserções das novas contas.

Unrepeatable read

- Problema de leitura de valores não repetidos de um mesmo atributo em uma mesma operação (*unrepeatable read*).
- Isso acontece quando uma transação lê duas vezes um mesmo item mas o item é alterado por outra transação entre as duas leituras.
- Exemplo: reserva de passagens.

Processamento de Transações

Propriedades ACID

- ***Atomicidade***
  - Uma transação é uma unidade de processamento; é feita totalmente ou nada é feito.
- ***Preservação da Consistência***
  - Uma transação deve levar o BD de um estado consistente a outro.
- ***Isolamento***
  - Uma transação não deve fazer visível para outra as suas atualizações até que seja finalizada.
- ***Durabilidade***
  - As mudanças realizadas por uma transação, após finalizada, devem ser permanentes.

Estados e Operações

- BEGIN_TRANSACTION
  - Marca o início de uma transação.
- END_TRANSACTION
  - Marca o fim de uma transação.
- COMMIT
  - Sinaliza o fim bem sucedido de uma transação - as modificações feitas pela transação serão tornadas permanentes.
- ROLLBACK
  - Sinaliza o fim mal sucedido de uma transação - as modificações feitas pela transação serão desfeitas.
- Operações de leitura e escrita no Banco de Dados

Diagrama de transição

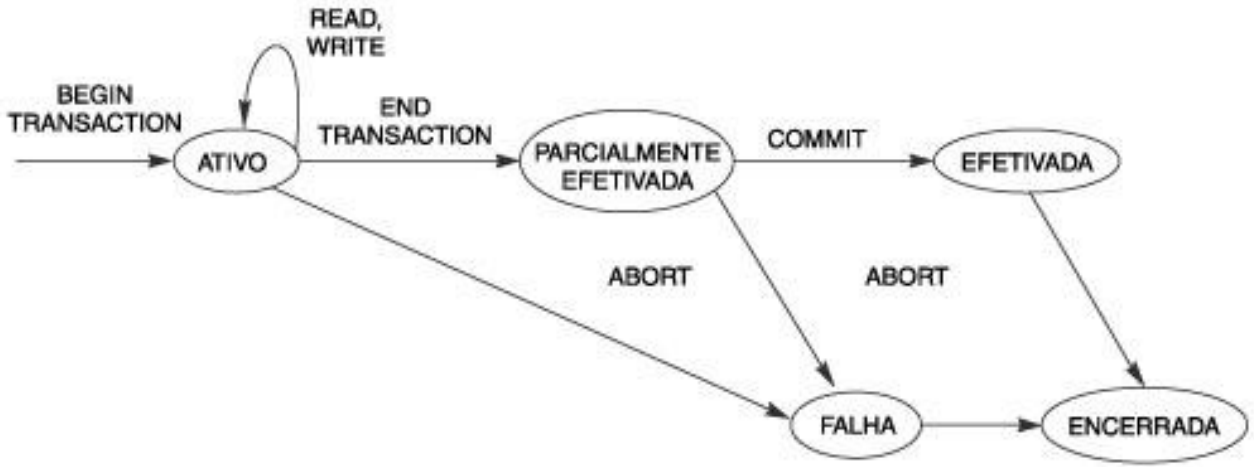


Diagrama de transição de estado ilustrando os estados de execução de uma transação.

Técnicas de controle de Concorrência

Locks

- Um *Lock* garante uso exclusivo de um item de dados a uma transação.
- Granularidade
  - Nível do banco de dados
  - Nível de tabela
  - Nível de página
  - Nível de linha
  - Nível de campo

Tipos de *Locks*

- **Lock binário**
  - Possui apenas dois estados: livre ou trancado (locked).
  - Um item trancado por uma transação não pode ser acessado por outra.

Operações

Lock_Item(X)

```
A:if (LOCK(X) == 0) LOCK(X) = 1;
   else
   {
      wait (até LOCK(X)==0 e o
            gerenciador acordar a
            transação)
      goto A;
   }
```

unlock_Item(X)

```
LOCK(X) = 0;
if (existe transação
esperando o item)
   acorda a primeira
transação;
```

Locks Compartilhado/exclusivo

- **Lock compartilhado/exclusivo**
  - O lock binário é muito restritivo.
  - O método de que usa Lock compartilhado/exclusivo permite que um dado seja lido por mais de uma transação simultaneamente.

Operações de read e write

read_lock(X)

```
{
   A: if (LOCK(X) == UNLOCKED) {
            LOCK(X) = READ_LOCKED;
            no_of_reads(X)++;
       }
       else if (LOCK(X)==READ_LOCKED)
            no_of_reads(X)++;
       else {
            wait (até LOCK(X)== UNLOCKED
e o gerenciador acordar a transação)
          goto A;
       }
}
```

write_lock(X)

```
{
   A: if (LOCK(X) == UNLOCKED) {
            LOCK(X) =
WRITE_LOCKED;
       }
       else {
            wait (até LOCK(X)==
UNLOCKED e o gerenciador acordar
a transação)
          goto A;
       }
}
```

Operação de unlock

```
unlock_lock(X) {
   A: if (LOCK(X) == WRITE_LOCKED)  {
         LOCK(X) = UNLOCKED;
         if (existe transação) acorda transação;
        }
        else if (LOCK(X)==READ_LOCKED) {
             no_of_reads(X)--;
             if (no_of_reads(X)==0)   {
                 LOCK(X) = UNLOCKED;
                 if (existe transação) acorda transação;
              }
        }
}
```

Protocolos de Locking

- **Locking em duas fases básico**
  - Todas as operações de locking (read_lock, write_lock) precedem o primeiro unlock.
  - Dividido em fase de expansão e retração.
    - Na fase de expansão - os locks são adquiridos.
    - Na fase da retração - os locks são liberados.
  - Uso de Protocolo em duas fases garante um escalonamento *serializável*.

Protocolos de Locking

- ***Escalonamento serializável***
  - Um escalonamento de transações é dito serializável se o resultado da execução das transações possui o mesmo efeito de uma execução de todas as transações seqüencialmente em alguma ordem (escalonamento serial).
  - Um escalonamento serializável leva o BD de um estado consistente a outro.

Protocolos de Locking

- **Protocolo de Locking em duas fases conservativo**:
  - todos os itens são trancados antes da transação começar.
  - Evita dead-lock.
  - Os recursos podem ficar trancados durante muito tempo.

Protocolos de Locking

- **Protocolo de Locking em duas fases estrito**
  - os itens são trancados de acordo com a necessidade
  - são liberados somente no final da transação
- Não é um protocolo livre de dead-lock

Técnicas não baseadas em Lock

- Baseados em timestamps
  - Neste tipo de técnica as operações seguem uma determinada ordem que garante a seriabilidade das transações.
- Multiversão
  - Usa várias versões de um mesmo item.
- Otimista
  - As transações são certificadas após a execução de suas operações.

Deadlock

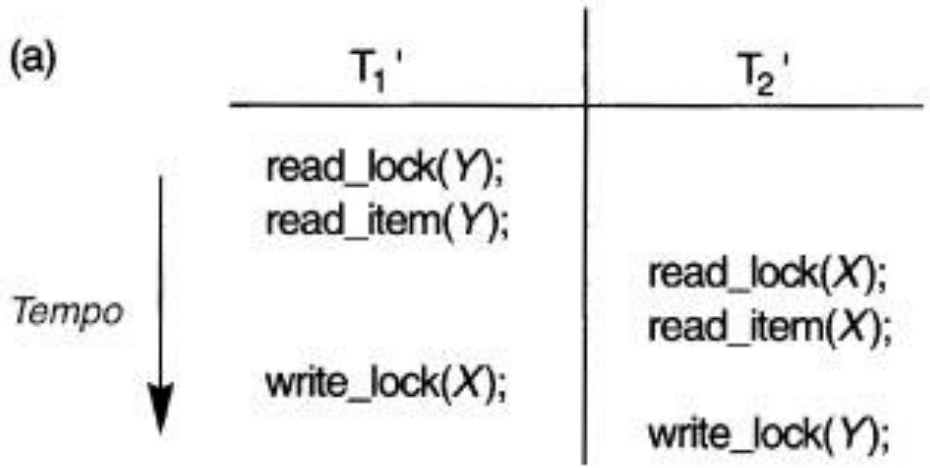


Ilustração do problema de deadlock: plano de execução parcial de T_1' e T_2' que está em um estado de deadlock.

Prevenção de deadlocks

- **Prevenção**
  - Neste caso pode-se usar um protocolo livre de dead-lock. Esta opção limita a concorrência.
- **Uso de time-stamps**
  - Neste caso as transações são ordenadas pelo time-stamp. A transação mais nova é desfeita.
- **Tempo esgotado**
  - Espera por tempo determinado. Se não conseguir desfaz e reinicia.

Níveis de Isolamento

Níveis de Isolamento

- O nível de isolamento define como as alterações feitas por uma transação são vistas por outra.
  - Quanto maior a invisibilidade das ações de uma transação em relação a outra maior o nível de isolamento.
  - Quanto maior o nível de isolamento menor será o desempenho do SGBD no tratamento de transações concorrentes.
- O número de níveis de isolamento, sua nomenclatura e características depende do SGBD utilizado.

Tipos de violação

Nível de Isolamento	Leitura Suja	Não repetível	Fantasma
READ UNCOMMITTED	Sim	Sim	Sim
READ COMMITTED	Não	Sim	Sim
REPEATABLE READ	Não	Não	Sim
SERIALIZABLE	Não	Não	Não

Níveis de Isolamento

- Read uncommitted
  - É o nível menos restritivo.
  - Pode ocorrer leituras de registros não *committed (dirty reads)*.
  - Usados onde não existe concorrência ou não existem alterações em registros ou quando essas alterações não são relevantes.

Níveis de Isolamento

- Read committed
  - Somente registros *committed* podem ser lidos.
  - Evita o problema de *dirty reads*
  - Duas leituras de um mesmo item em uma mesma transação podem possuir valores diferentes, uma vez que o valor pode ser mudado por outra transação entre duas leituras.

Níveis de Isolamento

- Repeatable Read
  - Somente registros *committed* podem ser lidos
  - Impede a alteração de um item lido pela transação.
  - Evita o problema de *dirty reads* e o problema do *non-repeatable read* .

Níveis de Isolamento

- Serializável
  - É o nível mais restritivo.
  - Impede *dirty reads* e *non-repeatable reads*.
  - Impede o problema de phantom reads onde um conjunto de registros satisfazendo a condição WHERE é lido enquanto outra transação insere novos registros que satisfazem a condição.

Suporte a transações em SQL

- Não há necessidade declarar BEGIN-TRANSACTION
- É necessário COMMIT/ROLLBACK.
- Características da transação SET TRANSACTION
  - Modo de acesso (READ ONLY ou READ WRITE)
  - Tamanho área diagnóstico
  - Nível de isolamento
- DBA ou programador pode tirar vantagem pelo relaxamento do nível de isolamento.

Uma transação SQL simples

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSACTION
   READ WRITE
   DIAGNOSTIC SIZE 5
   ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO TABLE EMPREGADO(NOME,CPF,
   SALARIO) VALUES ('Roberto', 9899998,510,00);
EXEC SQL UPDATE EMPREGADO SET SALARIO=SALARIO*1.1 WHERE DPTO='dpi';
EXEC SQL COMMIT;
GOTO FIM;
UNDO: EXEC SQL ROLLBACK;
FIM: END:...;
```

Recuperação de Falhas

Tipos de falha

- **Falha de transação**
  - pode ser subdividida em dois tipos:
    - **Erro lógico**
      - A transação não pode prosseguir devido a alguma condição interna, como entrada errada, dado não encontrado, overflow, etc.
    - **Erro de Sistema**
      - O sistema entra em um estado indesejável, como Deadlock, e não pode prosseguir, mas a transação pode ser executada mais tarde.

Tipos de falha

- **Queda do Sistema**
  - Falha de Hardware ou um bug no software do SGBD ou do Sistema Operacional que cause perda do conteúdo da memória volátil.
  - O conteúdo da memória não volátil é preservado.
- **Falha de Disco**
  - Um ou mais blocos do disco perdem seu conteúdo.

Recuperação baseada em log do sistema

- Para ser capaz de se recuperar de falhas nas transações o sistema mantém um arquivo log.
- No log são mantidos registros de todas as operações que afetam os valores dos itens no BD.
- O log é mantido no disco de modo a não ser afetado por falhas, a não ser falhas de disco e catástrofes.

Recuperação baseada em log do sistema

- Um arquivo de log é composto por vários registros de log, contendo em geral as seguintes informações
  - formato $<T_i, X_j, V_1, V_2>$, *onde*
    - T_i - ID da transação que realizou uma operação de escrita
    - X_j - ID do item escrito.
    - V_1 - Valor Antigo
    - V_2 - Valor novo

Recuperação baseada em log do sistema

- Outros registros de log seriam:
  - <T_i **start**> A transação T_i iniciou.
  - <T_i **commit**> A transação T_i realizou um commit.
  - <T_i **abort**> A transação T_i abortou.
- Sempre que uma transação realiza uma escrita é essencial que o registro de log seja inserido antes da modificação do BD.

Recuperação baseada em log do sistema

- Técnica: Adiamento de atualização (*deferred update*)
  - Todas as modificações são registradas no log, mas as atualizações no BD são realizadas apenas quando a transação atinge o estado de parcialmente committed.
  - Após as atualizações serem realizadas, a transação entra em estado committed.

<t0 **start**>

<t0, A, 900, 850>

<t0, B, 1000, 1050>

<t0 **commit**>

Parttialy committed

Recuperação baseada em log do sistema

- Atualização Imediata (Immediate update)
  - As atualizações no BD são feitas a medida que são determinadas pelas transações.
  - Em caso de falha usa-se o registro de log para recuperar os valores antigos.

Recuperação baseada em log do sistema

- Checkpoints
  - O objetivo do checkpoint é indicar um ponto no log que garante que todas as modificações estão salvas.
  - Periodicamente o SGBD realiza checkpoints que requerem as seguintes ações:
    - Gravar em memória não volátil todos os registros de log.
    - Gravar toda as modificações no disco.
    - Gravar um registro <checkpoint> no log.

Recuperação baseada em Shadow Paging

- Neste tipo de recuperação as modificações são feitas em uma página de cópia. Somente após a confirmação que a transação executado com sucesso é que a página de cópia passa a ser a página do BD.

Recuperação de catástrofes

- No caso de falha de disco e catástrofes a solução é realizar backups periódicos de toda a base de dados. Para não perder as transações o arquivo de log também deve ser copiado com uma maior freqüência.

Bibliografia

- Elmasri, R.; Navathe, S. – Sistemas de Banco de Dados – 6ª. Edição. São Paulo – 2011
- Oliveira, A. P. – Notas de aula – Universidade Federal de Viçosa
- Lisboa Filho, J. – Notas de aula – Universidade Federal de Viçosa.