MAC0439 – Laboratório de Bancos de Dados

Aula 19 – Parte B

Transações

Controle de comportamentos concorrentes

04 de novembro de 2015

Profa. Kelly Rosa Braghetto

(Adaptação dos slides do prof. Jeffrey Ullman, da *Stanford University*)

Por que Transações?

- Sistemas de BD geralmente são acessados por muitos usuários ou processos ao mesmo tempo
  - Esses acessos podem ser de consulta ou modificação de dados
- Diferentemente dos sistemas operacionais, que suportam a interação entre processos, um SGBD precisa evitar os problemas causados pela interação entre os processos

Exemplo: Interação ruim

- Um casal saca R$100 de uma conta corrente conjunta em diferentes caixas eletrônicos ao mesmo tempo
  - O SGBD deve garantir que nenhum dos débitos na conta sejam perdidos
- Comparação: Um SO permite que duas pessoas editem um mesmo documento ao mesmo tempo. Se ambas gravarem as alterações, as mudanças de uma das duas serão perdidas

Transações

- *Transação* = processo envolvendo consultas e/ou modificações no BD
- Geralmente, possui algumas propriedades rígidas relacionadas à controle de concorrência
- Em SQL, é formada por comandos simples ou pelo controle explícito do programador

Transações ACID

Transações ACID garantem:

- *Atomicidade*: ou a transação completa é feita, ou nada é feito
- Consistência: as restrições do BD devem ser garantidas
- Isolamento: para o usuário, deve parecer que há somente o seu processo em execução num dado momento
- Durabilidade: os efeitos do processo devem "sobreviver" a uma falha

Opcional: com frequência, formas mais "fracas" de transações também são suportadas

Transações na SQL

- Não existe no padrão SQL um comando do tipo “begin transaction” para começar uma transação de um modo explícito
  - O início de uma transação é implícito, quando instruções SQL começam a ser executadas
  - Mas muitos dialetos ou SGBDs implementam o “begin transaction” (ou algo parecido)
- Mas cada transação precisa ter uma instrução de fim explícita (ou seja, um COMMIT ou um ROLLBACK)

COMMIT

- O comando **COMMIT** da SQL causa o encerramento da transação
  - Depois do COMMIT, as modificações feitas no BD na transação se tornam permanentes

ROLLBACK

- A instrução **ROLLBACK** da SQL também causa o encerramento da transação, mas *abortando-a*
  - Efeito: nenhuma modificação é feita no BD de fato
- Falhas como divisão por 0 ou uma violação de restrição também podem causar um *rollback*, mesmo que o programador não o tenha solicitado

Exemplo: Processos Interativos

- Considere a relação Vendas(nome_lanche,nome_refri,preco) e suponha que o Sujinhos vende apenas Fanfa por R$2.50 e Sprife por R$3.00
- Sara está consultando Vendas, para saber o maior e o menor preço cobrado por um refrigerante no Sujinhos
- O gerente do Sujinhos decide parar de vender Fanfa e Sprife, e passar a vender só Cola-Coca por R$3.50

Programa da Sara

- Sara executa os dois comandos SQL a seguir, com rótulos (min) e (max) para nos ajudar a lembrar o que eles fazem

(max)

```
SELECT MAX(preco) FROM Vendas
WHERE nome_lanch = 'Sujinhos';
```

(min)

```
SELECT MIN(preco) FROM Vendas
WHERE nome_lanch = 'Sujinhos';
```

Programa do Gerente do Sujinhos

- Ao mesmo tempo, o gerente executa os seguintes passos: (del) e (ins).

(del)

```
DELETE FROM Vendas
WHERE nome_lanch = 'Sujinhos';
```

(ins)

```
INSERT INTO Vendas
VALUES('Sujinhos', 'Cola-Coca', 3.50);
```

Entrelaçamentos dos Comandos

- Embora (max) deva vir antes de (min), e (del) deva vir antes de (ins), não há nenhuma outra restrição com relação à ordem desses comandos (a menos que agrupemos os comandos da Sara e o comandos do gerente em transações diferentes)

Exemplo: Entrelaçamento Estranho

- Suponha que os passos sejam executados na seguinte ordem (max)(del)(ins)(min).

Preços do Sujinhos	{2.50, 3.00}	{2.50, 3.00}		{3.50}
Comando	(max)	(del)	(ins)	(min)
Resultado	3.00			3.50

- Sara verá MAX < MIN!

Corrigindo o Problema Usando Transações

- Se agrupássemos os comandos (max) (min) da Sara em uma transação, então não teríamos essa inconsistência
- Sara veria os preços do Sujinhos em um momento fixo do tempo
  - Ou antes, ou depois, ou no meio da mudança de preços feita pelo gerente do Sujinhos, mas o MAX e MIN seriam computados a partir do mesmo conjunto de preços

Outro Problema: *Rollback*

- Suponha que o gerente do Sujinhos execute (del)(ins), não como uma transação, mas depois de executar esses comandos, pense melhor sobre eles e execute uma instrução de ROLLBACK
- Se Sara executar seus comandos depois do (ins) mas antes do *rollback*, ela verá o valor 3.50, que nunca existiu de fato no banco de dados

Solução

- Se o gerente executar (del)(ins) como uma transação, seu efeito não pode ser visto por outros antes que a transação execute um COMMIT
  - Se a transação executar ROLLBACK em vez de COMMIT, seus efeitos nunca serão vistos

Propriedades das Transações na SQL

- Na SQL, é possível atribuir propriedades a uma transação por meio do comando

  SET TRANSACTION
- Essas propriedades podem ser de dois tipos:
  - **Modo de acesso** (READ WRITE ou READ ONLY)
  - **Nível de isolamento** (SERIALIZABLE, READ COMMITED, REPEATABLE READ ou READ UNCOMMITED)

Transações do Tipo "READ WRITE"

- Tipo de transação "padrão"
  - Lê dados, faz algum processamento e então grava dados no BD
- Esse tipo de transação está mais sujeito a problemas de seriação
- Para indicar que uma transação faz leituras e escritas, podemos usar o comando a seguir (mas isso já é o padrão):

SET TRANSACTION READ WRITE;

Transações do Tipo "READ WRITE"

- Exemplo: reserva de assentos em um avião – envolve 2 etapas

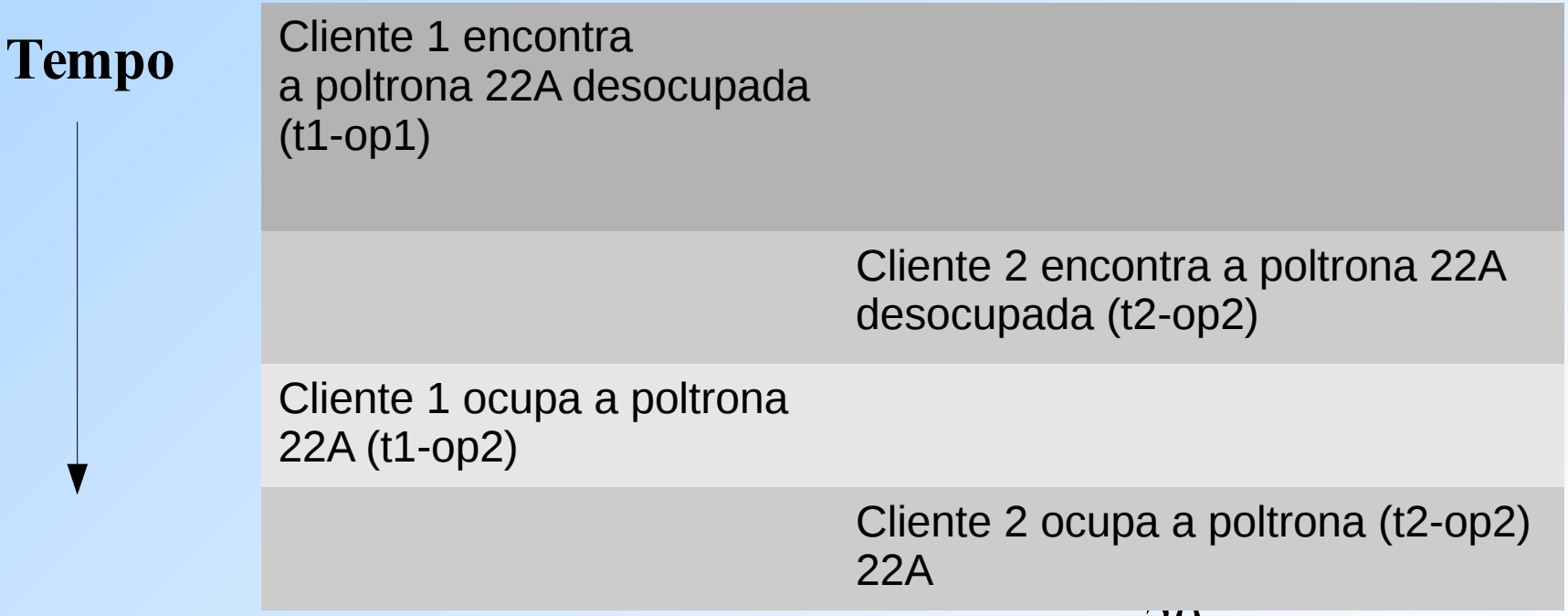
(op1)

```
EXEC SQL SELECT ocupado INTO :ocup
FROM Voos WHERE NumVoo = :voo AND
        DtVoo = :data AND PoltronaVoo = :poltr;

If (!ocup)
(op2)
EXEC SQL  UPDATE Voo SET ocupado = TRUE
              WHERE NumVoo = :voo AND
                      DtVoo = :data AND PoltronaVoo = :poltr;
```

Transações do Tipo "READ WRITE"

- Exemplo: reserva de assentos em um avião
- Vamos considerar que há dois clientes tentando reservar a mesma poltrona num dado voo:



Transações do Tipo "READ WRITE"

- A menos que se indique explicitamente o contrário, uma transação geralmente é considerada pelo SGBD como sendo READ WRITE
  - **Esse tipo de transação é sempre executado de forma seriada**
  - A execução seriada do exemplo do slide anterior forçaria um dos seguintes tipos de execução: T1 seguido de T2, ou T2 seguido de T1.

Transações do Tipo "READ ONLY"

- São transações que apenas leem dados do BD
- Esse tipo de transação geralmente pode ser executado paralelamente a outras transações "READ ONLY" ou que não escrevem nos dados de interesse
- Para indicar que uma transação é "READ ONLY", usar o comando a seguir no início da transação:

SET TRANSACTION READ ONLY;

Níveis de Isolamento

- A SQL define quatro *níveis de isolamento*
- Eles determinam quais interações são permitidas entre transações que executam ao mesmo tempo
- O nível de isolamento de uma transação é especificado por meio do comando

  SET TRANSACTION ISOLATION LEVEL

  <isolamento>

Escolha do Nível de Isolamento

- Dentro de uma transação, nós podemos dizer:

SET TRANSACTION ISOLATION LEVEL *X;*

onde X =

1. SERIALIZABLE
2. REPEATABLE READ
3. READ COMMITTED
4. READ UNCOMMITTED

Transações do tipo “SERIALIZABLE”

- Se Sara = (max)(min) e gerente = (del) (ins) são transações, e Sara roda com nível de isolamento “SERIALIZABLE”, então Sara verá o BD **ou antes ou depois da execução da transação do gerente**, mas não no meio
- “Serializable” é o nível de isolamento padrão para transações

O Nível de Isolamento é uma Escolha “Pessoal”

- A escolha de um nível para uma transação, afeta somente a forma como **essa** transação vê o BD, e não como os outros o vêem
- Exemplo: Se a transação do gerente executar no modo “serializable”, mas a da Sara não, então a Sara **poderá** não ver preço algum para o Sujinhos
  - i.e., para a Sara, pode ocorrer a situação em que parecerá que ela está executando no meio da transação do gerente

Transações “READ COMMITED”

- Se Sara executa com nível de isolamento “READ COMMITTED”, então ela pode ver somente dados permanentes (= *committed)*, mas não necessariamente os mesmos dados a cada vez
- Exemplo: Sob “READ COMMITTED”, o entrelaçamento (max)(del)(ins)(min) é permitido, desde que o gerente faça um COMMIT depois de (ins)
  - Nesse caso, Sara verá MAX < MIN

Transações
“READ COMMITED”

- Para indicar que uma transação é “READ COMMITED”:

SET TRANSACTION READ ONLY
ISOLATION LEVEL READ COMMITED;

Ou

SET TRANSACTION READ WRITE
ISOLATION LEVEL READ COMMITED;

Transações “REPEATABLE READ”

- Os requisitos são semelhantes aos do “READ COMMITED”, com a adição de: se o dado é lido novamente, então tudo visto na primeira vez será visto na segunda
  - Mas a segunda vez (e as leituras subsequentes) poderão ver mais tuplas também

Exemplo: "REPEATABLE READ"

- Suponha que Sara execute sob REPEATABLE READ, e a ordem da execução é (max)(del)(ins)(min)
  - (max) vê os preços 2.50 e 3.00
  - (min) pode ver 3.50, mas precisa também ver 2.50 e 3.00, porque eles foram vistos na leitura anterior feita por (max)

Transações “REPEATABLE READ”

- Para indicar que uma transação é “REPEATABLE READ”:

SET TRANSACTION READ ONLY
ISOLATION LEVEL REPEATABLE READ;

Ou

SET TRANSACTION READ WRITE
ISOLATION LEVEL REPEATABLE READ;

“READ UNCOMMITED”

- Uma transação executando sob “READ-UNCOMMITTED” pode ver dados no BD mesmo se eles tiverem sido escritos por uma transação que ainda não sofreu COMMIT (e que pode nunca sofrer!)
- Exemplo: Se Sara executar sob “READ UNCOMMITTED”, ela poderá ver o preço 3.50 mesmo se o gerente aborte sua transação depois

Transações
“READ UNCOMMITED”

- Para indicar que uma transação é “READ UNCOMMITED”:

SET TRANSACTION READ ONLY

ISOLATION LEVEL READ UNCOMMITED;

Ou

SET TRANSACTION READ WRITE

ISOLATION LEVEL READ UNCOMMITED;

- Para “READ UNCOMMITED”, o padrão é “READ ONLY”, e não “READ WRITE” como nos outros casos

Violações que Podem Ocorrer em Isolamentos “Inferiores” a SERIALIZABLE

- **Leitura de dados sujos**
  - T1 pode ler uma atualização de T2 que ainda não foi confirmada. Se T2 for abortada, T1 terá lido um valor sujo (incorreto).
- **Leitura não repetitiva**
  - T1 pode ler um dado valor de uma tabela. Se depois T2 atualizar esse valor e T1 lê-lo novamente, T1 verá um valor diferente.
- **Fantasmas**
  - T1 pode ler um conj. de linhas de uma tabela a partir de uma dada consulta. Suponha que depois T2 inseriu novas tuplas na tabela que também entrariam na resposta da consulta executada por T1. Se T1 for repetida, então ela verá “fantasmas”, ou seja, tuplas que não existiam anteriormente.

Obs.: T1 e T2 usadas nos exemplos acima são transações.

Possíveis Violações por Nível de Isolamento

	Tipo de violação		
Nível	**Leitura "suja"**	**Leitura não repetitiva**	**Fantasma**
READ UNCOMMITED	Sim	Sim	Sim
READ COMMITED	Não	Sim	Sim
REPEATABLE READ	Não	Não	Sim
SERIALIZABLE	Não	Não	Não

Um Parênteses sobre a Leitura de “Dados Sujos”...

- Dados sujos → dados escritos por uma transação que não foi confirmada (“commited”) ainda
- A leitura de dados sujos pode:
  - Não ser um problema
  - Ser um problema grave
  - Ser um problema leve, que justifica o risco de se ter dados ocasionalmente sujos em troca de melhor desempenho no BD
    - Evitar a leitura de dados sujos custa bastante tempo a um SGBD (e diminui as possibilidades de paralelismo na execução das transações)

Exemplo: Problema Causado pela Leitura de Dados Sujos

- Considere um programa P que faz uma transferência entre duas contas bancárias:
  - Passo 1: Adiciona o dinheiro na conta 2
  - Passo 2: Verifica se a conta 1 tem saldo suficiente
    - Se não tiver, remove o dinheiro da conta 2 e termina
    - Se tiver, subtrai o dinheiro da conta 1 e termina

Exemplo: Problema Causado pela Leitura de Dados Sujos

- Se P é executado de forma seriada, não há problema algum em colocarmos dinheiro temporariamente na conta 2 (ninguém verá esse dinheiro caso a transferência não seja concluída)
- Mas se a leitura de dados sujos for possível durante a execução de P, então o resultado de P pode ser incorreto.
  - Suponha que temos três contas:
    A1 ($100), A2 ($200) e A3 ($300)

Exemplo: Problema Causado pela Leitura de Dados Sujos

- Suponha que as seguintes transações são executadas +/- no mesmo tempo:
  - T1 → executa P para transferir $150 de A1 para A2
  - T2 → executa P para transferir $250 de A2 para A3

Exemplo: Problema Causado pela Leitura de Dados Sujos

Possível sequência de eventos:

1. T2 executa passo 1 e soma $250 ao saldo de A3, que fica com $550
2. T1 executa passo 1 e soma $150 ao saldo de A3, que fica com $350
3. T2 executa o teste do passo 2 (e conclui que A2 tem saldo suficiente → $350)
4. T1 executa o teste do passo 2 (e descobre que A1 não tem saldo suficiente → $100)
5. T2 executa passo 2b e subtrai $250 de A2, que fica com $100
6. T1 executa passo 2ª e subtrai $150 de A2, que fica com -$50 (!)

A leitura de dados sujos em P nesse exemplo fez com que a conta de um cliente ficasse com o saldo negativo, apesar de P implementar uma verificação para proteger os clientes disso.

Referências Bibliográficas

- ***Database Systems – A Complete Book*** (2ª edição), Garcia-Molina, Ullman e Widom, 2002. Capítulo 8.
- **Sistemas de Bancos de Dados** (6ª edição), Elmasri e Navathe. Capítulo 21.