UnB

Sistemas de Informação

Transações

Edison Ishikawa, D. Sc.

Introdução

- Objetivo
  - Apresentar o conceito de transação e SPT

Sumário

- Introdução
- Conceitos Básicos de Transação
- SPT
- Considerações finais

Transação

- Transação é uma unidade de trabalho que se quer tratar como um todo.
- Isto é, ou se faz toda a operação de transação ou não se faz nada.

Exemplo 1

- **Retirar uma quantia de uma conta bancária e depositar em outra.**
- **Para isso, são realizadas duas operações:**
  - **Debitar R$ X da conta "A"**
  - **Depositar R$ X na conta "B"**
- **Estas duas operações devem ser feitas de forma atômica (como se fosse única)**

Exemplo 1

- Se a operação não for atômica existem situações que podem ocorrer e que levam o sistema a ficar inconsistente
  - Tento debitar R$ X da conta “A”, mas não tem saldo e aborto a operação
  - Deposito R$ X na conta “B”
- Transações existem para garantir que não importa o que aconteça, os dados que você está manipulando estarão em um estado sensível.
- Ela garante que não existirá uma situação em que uma quantia de dinheiro é retirada de uma conta e não é depositada em outra.

Exemplo 1

- Transações concorrentes
  - Estado inicial – contas A e B possuem R$ 100,00 e o banco possui R$ 200,00 em caixa.
  - Transação *
    - Transferir R$ 60,00 da conta A para a conta B
  - Transação #
    - Sacar R$ 50,00 da conta A
  - O que acontece quando realizo:
    - a transação * antes de #?
    - a transação # antes de *?
    - as duas transações ao mesmo tempo?

Exemplo 2

- Pedido de compra de uma mercadoria de um catálogo
  - A transação envolve:
    - Verificar o banco de dados com o inventário do estoque
    - Confirmar que o item está disponível
    - Executar o pedido (tirar do BD e mandar Mensagem ao operador de estoque para remeter o item pedido)
    - Operador confirma que executou o pedido e informa o tempo esperado para a entrega

Exemplo 2

- Se esta sequência de operações for considerada uma única transação atômica, então todos os passos serão realizado antes que a transação termine e o BD seja atualizado para refletir a nova ordem.
- Se alguma coisa acontece que impeça a transação de chegar ao fim, qualquer alteração no BD deve ser rastreado para ser desfeita.

Transações

- Transações são suportadas pela linguagem SQL (structured Query language).
- SQL é a interface padrão de programação de um SGBD.
- Quando uma transação é bem sucedida, se diz que ela foi ***comitted***.
- Quando uma transação não é completada, se diz que as ações já realizadas sofrerão um ***roll-back***

Princípios de uma transação

- ACID
  - Atômica
  - Consistente
  - Isolada
  - Durável

Princípios de uma transação (acid)

- Atômica
  - A transação é toda realizada ou nada é realizado. Evita atualizações parciais que podem causar problemas de integridade nos dados. Significa ser indivisível e irredutível. As transações, embora atômicas, podem fazer atualizações em múltiplos banco de dados operando em múltiplos sistemas.

Princípios de uma transação (acid)

- Consistente
  - Uma transação leva o BD de um estado válido para outro estado válido e que os dados escritos no BD atendem regras definidas.

Princípios de uma transação (acid)

- Isolada
  - Refere-se a manter a transação oculta de outras transações e operações executadas de forma concorrente até que ela termine.

Princípios de uma transação (acid)

- Durável
  - Refere-se ao fato de que qualquer transação após o *commit* no BD não pode mais ser desfeita (não se pode revertê-la, como se ela não tivesse ocorrido, mas pode-se fazer outra transação desfazendo a anterior). Este nível de garantia pode ser implementado copiando-se todas as transações em um arquivo de *log* em um meio de armazenamento secundário. Uma transação é considerada “commited” só após ser copiada para o *log* de transações.

Implementação de Transações

- Como garantir as propriedades ACID?
- Como implementar transações mantendo o máximo de paralelismo possível no processamento de transações concorrentes?

Programação Concorrente

Atualização de uma variável por duas transações ao mesmo tempo

Supor a seguinte estrutura de dados e funções:

```
typedef  struct {
                        char *nome;
                        int Nr_cc;
                        float saldo;
     }CONTA;

void  sacar(CONTA *cc, valor){
                 float saldo = cc->saldo; //lê o saldo da conta e grava valor na variável local
                  saldo = saldo – 60,00;//debita o valor localmente
                  cc->saldo = saldo; //grava o valor de saldo no registro global
};

void depositar(CONTA *cc, valor){
                float saldo = cc->saldo; //lê o saldo da conta e grava valor na variável local
                 saldo = saldo + 60,00;//incrementa o valor localmente
                 cc->saldo = saldo; //grava o valor de saldo no registro global
};

CONTA conta_A={“Fulano”, 777, 100.00};
```

Programação Concorrente

Suponha as seguintes transações:

T1

Sacar R$ 60,00 da conta_A

```
sacar(&conta_A , 60,00);
```

T2

Depositar R$ 60,00 na conta A

```
depositar(& conta_A, 60,00);
```

Programação Concorrente

- Suponha que as transações são feitas “ao mesmo tempo”, qual será o resultado?

```
float saldo = cc->saldo; //lê o saldo da conta
float saldo = cc->saldo; //lê o saldo da conta
saldo = saldo – 60,00;//debita o valor localmente
saldo = saldo + 60,00;//incrementa o valor localmente
cc->saldo = saldo; //grava o valor de saldo R$ 40,00
cc->saldo = saldo; //grava o valor de saldo R$ 160,00
```

Processo de Corrida

Programação Concorrente

- Atualização de uma variável por duas transações ao mesmo tempo
- Processo de corrida
  - como evitar processo de corrida?
    - Protegendo a parte do código que acessa variáveis compartilhadas
    - Esta parte do código se chama **Região Crítica**

```
void  sacar(CONTA *cc, valor){
        float saldo = cc->saldo; //lê o saldo da conta e grava valor na variável local
        saldo = saldo – 60,00;//debita o valor localmente
        cc->saldo = saldo; //grava o valor de saldo no registro global
};

void depositar(CONTA *cc, valor){
        float saldo = cc->saldo; //lê o saldo da conta e grava valor na variável local
        saldo = saldo + 60,00;//incrementa o valor localmente
        cc->saldo = saldo; //grava o valor de saldo no registro global
};
```

Programação Concorrente

- Atualização de uma variável por duas transações ao mesmo tempo
- Processo de corrida
- Região Crítica - RC
  - A RC pode ser protegida por operações como **lock**
  - Por exemplo, usando-se POSIX pthread library

```
#include <pthread.h>
static pthread_mutex_t cs_mutex = PTHREAD_RECURSIVE_MUTEX_INITIALIZER;

void  sacar(CONTA *cc, valor){
        pthread_mutex_lock( &cs_mutex );
            float saldo = cc->saldo; //lê o saldo da conta e grava valor na variável local
             saldo = saldo – 60,00;//debita o valor localmente
             cc->saldo = saldo; //grava o valor de saldo no registro global
        pthread_mutex_unlock( &cs_mutex );
};
```

Programação Concorrente

- Problemas em se usar um lock
  - Pode ocorrer um deadlock
- Considere as seguintes transações:
  - T1 – transferir R$ 60,00 da conta A para a conta B
    - sacar(&conta_A, 60.00);
    - depositar(&conta_B, 60.00);
  - T2 – transferir R$ 60,00 da conta B para a conta A
    - sacar(&conta_B, 60.00);
    - depositar(&conta_A, 60.00);

Programação Concorrente

- Problemas em se usar um lock
  - Pode ocorrer um deadlock
- Considere a seguinte ordem de execução:
  - sacar(&conta_A, 60.00); // deu um lock na variável conta_A
  - sacar(&conta_B, 60.00); // deu um lock na variável conta_B
  - depositar(&conta_B, 60.00);// fica esperando liberação do lock na variável conta_B
  - depositar(&conta_A, 60.00);// fica esperando liberação do lock na variável conta_A

Programação Concorrente

- Problemas em se usar um lock
  - Pode ocorrer um ***deadlock***
  - Pode ocorre ***starvation***

Exemplo 3

- Venda de passagens aéreas
  - Como a venda de passagens on line é feita?
  - Como permitir que vários usuários façam transações ao mesmo tempo?
  - Como evitar o ***overbooking***?
  - Como fazer isto com alta escalabilidade e baixo tempo de resposta?

Exemplo 4

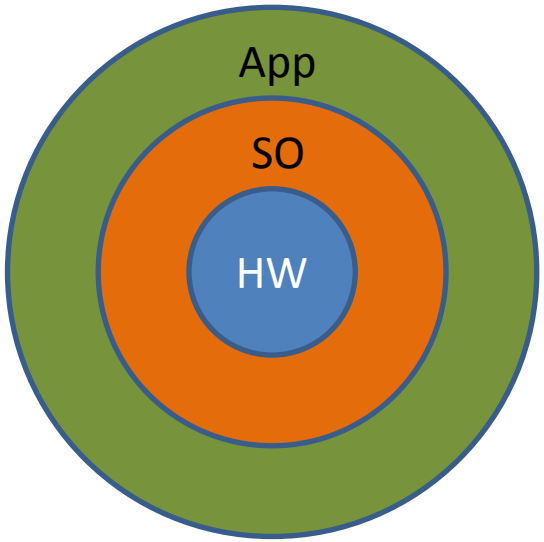
- Banco de Dados Múltiplos
  - Imagine que você é dono de uma agência de turismo e seu cliente quer passar o final de semana em Nova lorque para ver uma peça na Broadway
  - É possível?

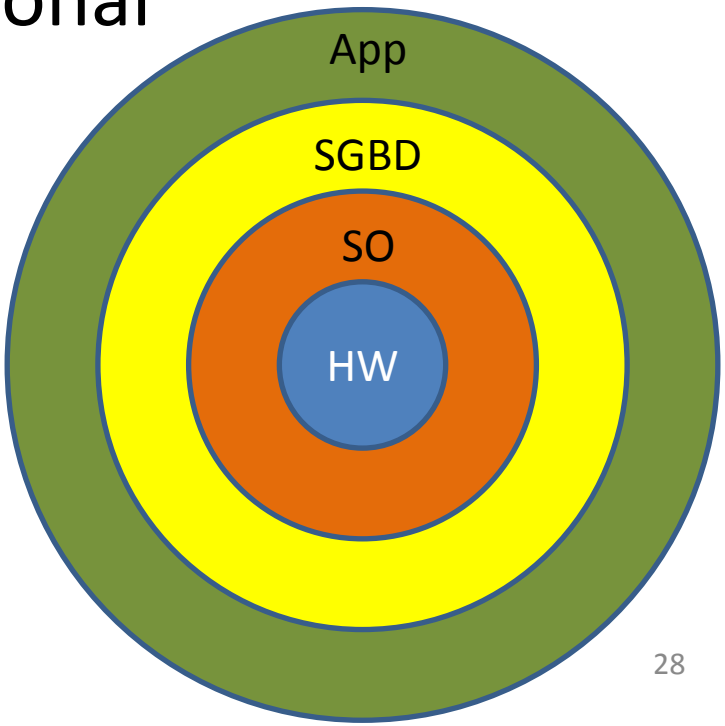
Exemplo 4

- Banco de Dados Múltiplos
  - Imagine que você é dono de uma agência de turismo e seu cliente quer passar o final de semana em Nova lorque para ver uma peça na Broadway
  - Para isso, você quer um SI que faça isso em uma só transação T
    - Reservar a passagem aérea
    - Reservar o hotel
    - Comprar o bilhete da peça na Broadway
  - É possível?

Implementação de Transações

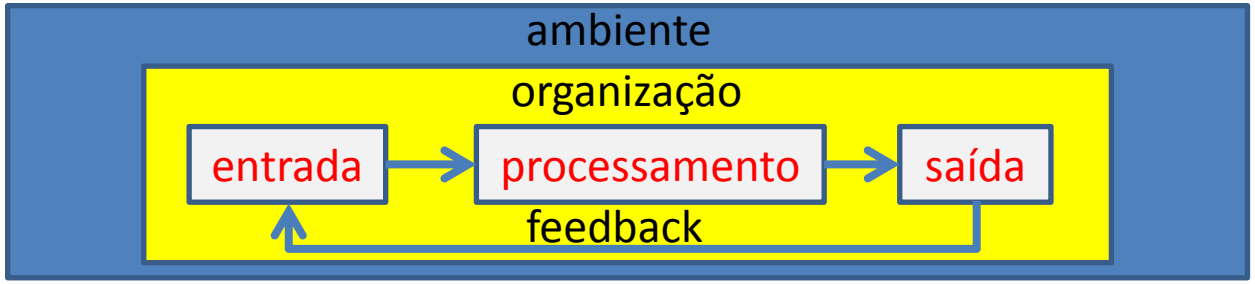
- A nível do SGBD
- A nível de programação da aplicação
- A nível de Sistema Operacional
- Ao nível do hardware





Sistemas de Processamento de Transações (SPT) x Sistemas de Informação (SI)

- Sistema de Informação
  - Conjunto de componentes interrelacionados, desenvolvidos para coletar, processar, armazenar e distribuir informação para facilitar a coordenação, o controle, a análise, a visualização e o processo decisório.



- SPT
  - Um sistema transacional é o principal sistema da maior parte das organizações empresariais, que dá apoio à monitoração e à realização das negociações de uma organização, gerando e armazenando dados sobre estas negociações.
  - Os sistemas transacionais são os sistemas operacionais de uma empresa.

SPT

- também conhecidos como OLTP – Online Transactional Processing
- Exemplos de tipos de SPT:
  - Sistemas Contábeis;
  - Aplicações de Cadastro;
  - Sistemas de Compra, Estoque, Inventário;
  - ERPs (Enterprise Resource Planning);
  - CRMs (Customer Relationship Management);
  - Sistemas de Folha de Pagamento;
  - Faturamento;
  - Etc...

SPT

- Características
  - Coleta de dados através de mecanismos de entrada;
  - Realização de cálculos;
  - Armazenamento de informações e ordenação dos dados, facilitando o acesso a eles;
  - Permite consultas, *on line* ou em *batch* dos dados, em diversas formatações;
  - Produção de documentos e relatórios operacionais.

SPT

- Objetivos
  - Processar dados gerados, por e sobre as transações;
  - Manter um alto grau de precisão e confiabilidade;
  - Assegurar a integridade dos dados da informação;
  - Produzir documentos e relatórios em tempo real;
  - Aumentar a eficiência do trabalho;
  - Auxiliar no fornecimento de mais e melhores serviços;
  - Fornecer informações para apoio à decisão.

EDP

- Eletronic Data Processing
  - São SI para o controle operacional das organizações.
  - Os dados por ele gerados são a entrada de outros sistemas de gestão. Embora esses sistemas só controlem o fluxo de informações operacionais, eles também disponibilizam informações para a tomada de decisão.
  - Exemplo:
    - sistema de controle de estoques
      - fornece informações sobre a movimentação do estoque para o departamento de compras.
      - este departamento poderá, através dessas informações, tomar decisões sobre quais produtos deverão ser comprados e em que quantidade.
  - Desta forma umUm **EDP** pode fornecer informações para apoio à decisão. Mas isto não o torna um SAD (Sistema de Apoio à Decisão).