Sistemas de Gerência de Banco de Dados

Revisão

Aula 01 - Introdução ao Processamento de Consultas

- Processamento de consultas
- Subetapas da avaliação de consulta
- Estimativa de custo de processamento
- Plano de avaliação de consulta

Plano de avaliação de consulta

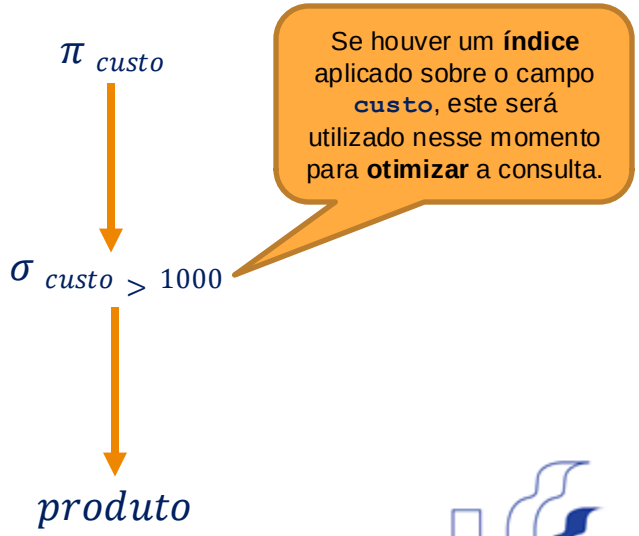
1) Exemplo de *query* SQL:

```
select custo
from produto
where custo > 1000;
```

2) Tradução para álgebra relacional:

$$\pi_{custo}(\sigma_{custo > 1000}(produto))$$

3) Plano de avaliação da *query*:



Aula 2 - Otimização e Estratégias no Processamento de Consultas

- Otimização de consulta
- Avaliação de consultas complexas
- Equivalência nas expressões relacionais
- Estratégias de avaliação das operações
- Estatísticas sobre as relações
- Planos de avaliação alternativos
- Uso de *views* materializadas

Equivalência nas expressões relacionais

- Objetivo: transformar sistematicamente consulta SQL (original) em consulta interna que tenha
  - equivalência à expressão da álgebra relacional (usada internamente)
  - execução com menor custo (para retornar dados)

- Diversas regras que podem ser usadas
- Expressão da álgebra relacional (interna) → sequência distinta de operações

Aula 3 - Transações

- Garantias sobre transações
- Execuções simultâneas
- Transações e propriedades ACID
- Estruturas de armazenamento
- Controle da interação entre as transações
- Serializabilidade
- Componente de gerenciamento de concorrência
- Gerenciamento de controle de concorrência

Serializabilidade

- Garantia de que
  - *schedule* de concorrência = *schedule* de transações executas em série e ordenadas
- Garantida por esquemas de *controle de concorrência*
- *Schedules* precisam ter capacidade de recuperação (desfazimento)
  - se transação X vê efeitos de transação Y e,
  - se transação Y aborta,
  - então transação X também aborta

Aula 4 - Controle de concorrência

- Controle de transações concorrentes
- Protocolo de bloqueio
- Proteção contra impasse (*deadlock*)
- Outros aspectos sobre controle de concorrência

Protocolo de bloqueio

- Conjunto de regras → indica quando transação bloquear e desbloquear dado
- Protocolo de bloqueio em duas fases:
  - permite transação bloquear novo item somente se não tiver desbloqueado nenhum outro
  - assegura serializabilidade
  - não assegura ausência de impasse (*deadlock*)

Proteção contra impasse (*deadlock*)

- Muitos protocolos de bloqueio não protegem contra impasses (*deadlocks*)
- Se *deadlock* não puder ser evitado, sistema deverá:
    - detectar e se recuperar do impasse
    - revertendo tantas transações quanto necessárias para quebrar o impasse

Aula 5 - Recuperação de falhas

- Sistema de recuperação de falhas
- Esquemas baseados em *log*
- Paginação de sombra (*shadow page*)
- Esquema de modificações adiadas
- Algoritmos e técnicas de recuperação modernos
- Sistema de *backup* remoto
- Dump
- Esquema de recuperação ARIES

Esquemas baseados em log

- Atualizações registradas em *log* → em armazenamento estável
- Transação confirmada → último *log* (*commit*) enviado para disco estável
- *Log* contêm valores antigos e novos para todos os registros atualizados
  - Valores novos - caso atualizações necessitem ser **refeitas** após uma falha
  - Valores antigos - para **reverter** atualizações se transação abortar a qualquer momento

Questão para exercitar

- Considerando a utilização do algoritmo de UNDO/REDO para a recuperação de falhas e como técnica de atualização imediata do BD

- E considerando um arquivo de *log*, em que temos as seguintes premissas:
  - um comando de instrução(T_A, start) indica o início de uma transação T_A
  - uma instrução (T_A, commit) indica o registro de conclusão da transação
  - X, Y, Z, W, K e L indicam os itens de dados alterados pelas transações

Questão para exercitar (continua)

- Então, no comando (T_A, X, 20, 50), temos respectivamente que:
  - T_A significa o identificador de transação
  - X significa o item afetado
  - 20 representa o valor antigo
  - 50 representa o novo valor

Questão para exercitar (continua)

Marcação sequencial de data e hora (*timestamping*) de execução das instruções

1. (T_A, start)
2. (T_A, X, 20, 50)
3. (T_B, start)
4. (T_B, Y, 40, 50)
5. (T_A, Z, 56, 34)
6. (T_B, W, 65, 33)
7. (T_B, commit)
8. checkpoint
9. (T_C, start)
10. (T_A, K, 22, 34)
11. (T_C, L, 44, 55)
12. (T_C, commit)
13. crash

Marcação de ponto de verificação (*checkpoint*)

Marcação de ponto de falha (*crash*)

Questão para exercitar (continua)

T_A deverá ser desfeita (**UNDO**), em função de não ter concluído ainda (não há instrução **commit**)

T_B não precisa ser feito **nada**, pois instrução de **commit** está antes do checkpoint

```
1.  (T_A, start)
2.  (T_A, X, 20, 50)
3.  (T_B, start)
4.  (T_B, Y, 40, 50)
5.  (T_A, Z, 56, 34)
6.  (T_B, W, 65, 33)
7.  (T_B, commit)
8.  checkpoint
9.  (T_C, start)
10. (T_A, K, 22, 34)
11. (T_C, L, 44, 55)
12. (T_C, commit)
13. crash
```

T_C deverá ser refeita (**REDO**), pois há a instrução de **commit**, mas ainda não havia passado por um checkpoint

Sistemas de Gerência de Banco de Dados