Universidade Federal
de Campina Grande

Banco de Dados I
Unidade 12: Otimização de Consultas

Prof. Cláudio de Souza Baptista, Ph.D.
Laboratório de Sistemas de Informação – LSI
UFCG

Introdução

- Linguagens de alto nível (ex.SQL) podem ter consultas com alto tempo de processamento .
- Existem várias maneiras de escrever uma query.
  - **Solução:** usar um otimizador de consultas para escolher a melhor forma, dentro das suas possibilidades, de executar uma consulta

Introdução

- O sistema gera um código otimizado (sem necessariamente ser o melhor possível)
- O processo de otimização leva em consideração:
  1. A teoria da Álgebra Relacional
  1. Informações sobre:
     - tamanhos de tabelas
     - seletividade de índices
     - agrupamento de dados

Introdução

- O sistema gera um código otimizado (sem necessariamente ser o melhor possível)
- O processo de otimização leva em consideração:
  1. A teoria da Álgebra Relacional
  1. Informações sobre:
     - tamanhos de tabelas
     - seletividade de índices
     - agrupamento de dados

Introdução

- O sistema gera um código otimizado (sem necessariamente ser o melhor possível)
- O processo de otimização leva em consideração:
  1. A teoria da Álgebra Relacional
  1. Informações sobre:
     - tamanhos de tabelas
     - seletividade de índices
     - agrupamento de dados

Introdução

- O sistema gera um código otimizado (sem necessariamente ser o melhor possível)
- O processo de otimização leva em consideração:
  1. A teoria da Álgebra Relacional
  1. Informações sobre:
     - tamanhos de tabelas
     - seletividade de índices
     - agrupamento de dados

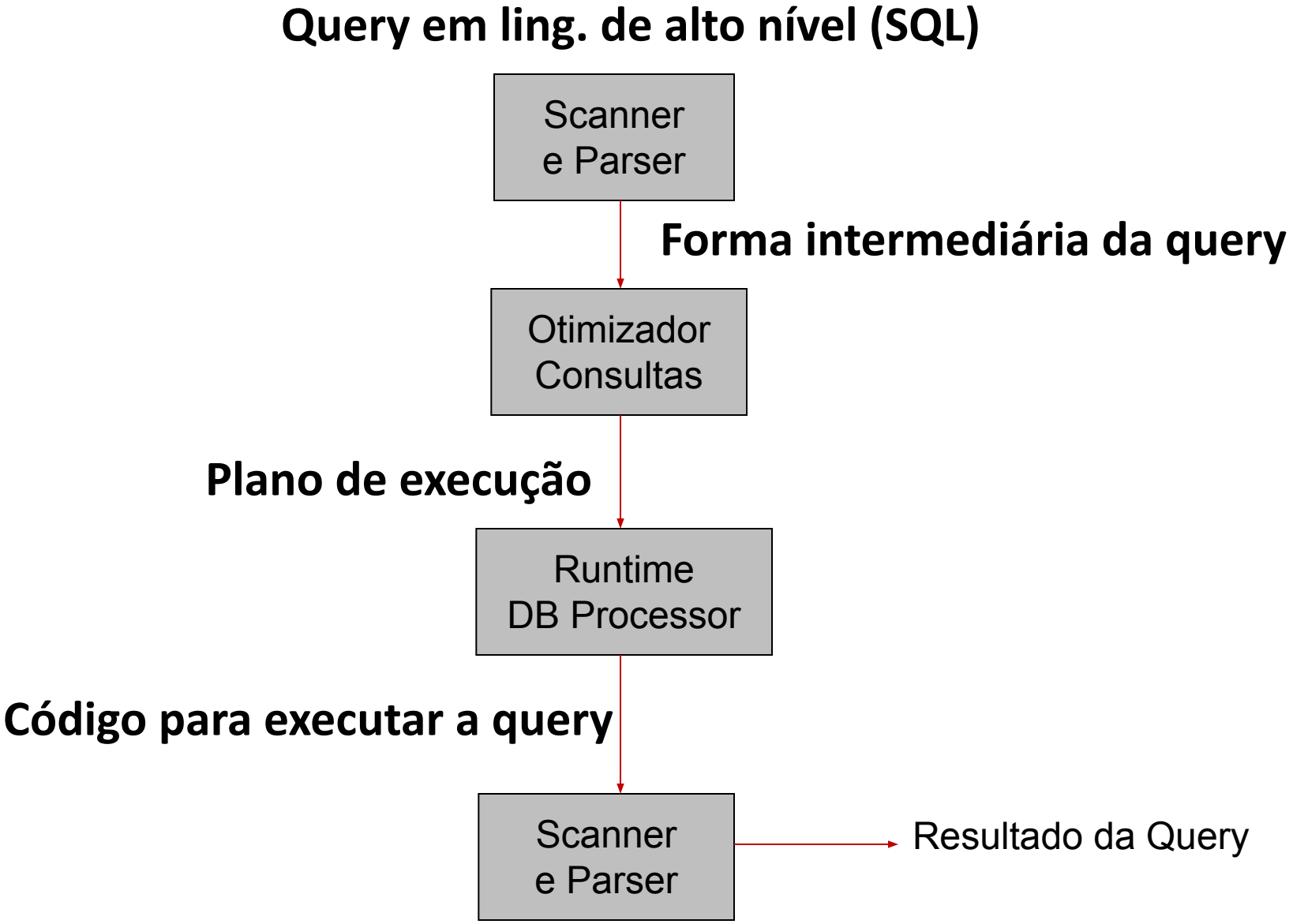
Introdução

- O sistema gera um código otimizado (sem necessariamente ser o melhor possível)
- O processo de otimização leva em consideração:
  1. A teoria da Álgebra Relacional
  1. Informações sobre:
     - tamanhos de tabelas
     - seletividade de índices
     - agrupamento de dados

Introdução

- O sistema gera um código otimizado (sem necessariamente ser o melhor possível)
- O processo de otimização leva em consideração:
  1. A teoria da Álgebra Relacional
  1. Informações sobre:
     - tamanhos de tabelas
     - seletividade de índices
     - agrupamento de dados

Introdução


Query em ling. de alto nível (SQL)
Scanner e Parser
Forma intermediária da query
Otimizador Consultas
Plano de execução
Runtime DB Processor
Código para executar a query
Scanner e Parser
Resultado da Query

Introdução

- Quais operações levam mais tempo para executar?

Produto Cartesiano e Junção

Introdução

- Quais operações levam mais tempo para executar?

Produto Cartesiano e Junção

Introdução

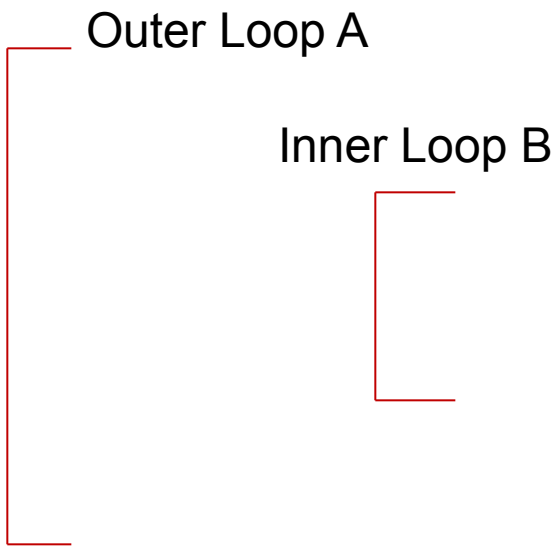
- Quais operações levam mais tempo para executar?

Produto Cartesiano e Junção

Ex.: A influência do tamanho de uma tabela no tempo de resposta a uma consulta. Sejam A e B duas relações, como podemos implementar A X B? Qual o custo de cada solução?

Introdução

Solução 1: Associar cada tupla de A com cada tupla de B (cada tupla de B é lida tamanho de A vezes)



Introdução

Solução 2: Colocar o máximo de blocos de A na memória e processar estas tuplas para cada tupla de B

Redução de leituras de cada bloco B = número de tuplas de A na memória

Introdução

Estratégia X

Outer Loop A

Inner Loop B

Estratégia Y

Outer Loop B

Inner Loop A

Introdução

- Número de acessos (leituras) a blocos:
  - Estratégia X:

$$nA/bA \left(1 + \frac{nB}{(m-1)*bB}\right) \quad <=> \quad nA/bA$$

  - Estratégia Y:

$$nB/bB \left(1 + \frac{nA}{(m-1)*bA}\right) \quad <=> \quad nB/bB$$

n_i = tamanho da relação i
b_i = fator de bloco de i
m = tamanho da memória principal (em blocos), geralmente um número muito grande

Introdução

- Otimização:

Outer Loop: **Se nA/bA < nb/bB**
então A
senão B

Inner loop: **Se A ∈ Outer loop**
então B
senão A

Ex.: **nA = 5.000**, **bA = 5**; **nB = 10.000**, **bB = 5** => a extratégia **X** pode responder à query na metade do tempo levado pela estratégia **Y**.

Introdução

Ex2. A importância de se reduzir o tamanho das tabelas intermediárias

Sejam A(a1,a2) e B(b1,b2) esquemas relacionais

```
SELECT a.a1
FROM A a, B b
WHERE a.a1 = 'valor' AND a.a2 = b.b1
```

Introdução

Estratégia 1:

1) Produto Cartesiano
1) Seleção
1) Projeção

⇒ **Completamente inviável!** Só o produto gera um resultado intermediário de tamanho 5.000 x 10.000 = 50.000.000!

Introdução

Estratégia 2:

1) Seleção de a1 = ‘valor’
   - ⇒ nB é pequeno, normalmente cabe na memória

   Número de acessos = 2.000 = nB/bB

2) (A x B) a.a2 = b.b1
   - ⇒ número de acessos = 1.000 = nA/bA

2) Projeções são feitas em cada passo
   - ⇒ Número total de acessos = 3.000 (razoável)

Introdução

Conclusão:

- Operações que reduzem tabelas:
  - **Seleção**
  - **Projeção**

Introdução

Estratégias gerais de Otimização:

1. Execute seleções o mais rápido possível
   ⇒ Reduz o tamanho das tabelas (resultados intermediários)
1. Combine, quando possível, uma seleção com o produto cartesiano anterior formando uma junção
1. Combine sequências de operações unárias (seleção e projeção)
1. Procure subexpressões comuns e guarde-as caso seja mais eficiente lê-las do que reprocessá-las

Introdução

Estratégias gerais de Otimização:

5. Pré-processar arquivos apropriadamente (SORT e INDEX)

⇒ Por exemplo, quando não for viável manter um índice permanente pode-se criar índices temporários para processar a query

6. Avaliar opções antes de executar:
   - Tamanho das tabelas;
   - Tamanho dos blocos;
   - Existência de índices;
   - Existência de tabela ordenada, etc.

Técnicas de Otimização

Tipos de Otimização:

- Lógica
- Física

Otimização Lógica

- Também conhecida como otimização algébrica e otimização heurística
- Faz manipulações algébricas, usando as leis da Álgebra Relacional, sem considerar o modo como as relações estão armazenadas, visando reduzir os tamanhos dos resultados intermediários.

Leis Algébricas

1) Produto Cartesiano e Junção são comutativos: se **E1** e **E2** são expressões relacionais e **F** uma condição sobre atributos de **E1** ou **E2**, então:

$$E1 \times E2 = E2 \times E1$$

$$E1 \mid X \mid_F E2 = E2 \mid X \mid_F E1$$

2) Produto cartesiano e junção são associativos

$$(E1 \times E2) \times E3 = E1 \times (E2 \times E3)$$

$$(E1 \mid X \mid E2) \mid X \mid_F E3 = E1 \mid X \mid_F (E2 \mid x \mid E3)$$

Leis Algébricas

3) Cascata de Projeções

$$\prod A1,\ldots,An(\prod B1,\ldots,Bm(E)) = \prod A1,\ldots An(E) \text{ com } A1, \ldots, An \subset B1, \ldots, Bm$$

4) Cascata de Seleções

$$\sigma F1\ (\sigma F2\ (E)) = \sigma F1\ \wedge\ F2\ (E)$$

e como **F1 ^ F2 = F2 ^ F1**, então

$$\sigma F1\ (\sigma F2\ (E)) = \sigma F2\ (F1\ (E))$$

Leis Algébricas

5) Comutatividade de seleções e projeções

Se a condição **F** envolve somente atributos **A1, …, An** então:

$$\prod_{A1,\ldots,An}(\sigma_F(E)) = \sigma_F(\prod_{A1,\ldots,An}(E))$$

Generalizando, se **F** também envolve atributos **B1, …, Bm**, que não estão entre **A1, …, An**, então:

$$\prod_{A1,\ldots,An}(\sigma_F(E)) = \prod_{A1,\ldots,An}(\sigma_F(\prod_{A1,\ldots,An,B1,\ldots,Bm}(E)))$$

Leis Algébricas

6) Comutatividade de seleção e produto cartesiano:

se todos os atributos em **F** são só atributos de **E1**, então:

$$\sigma_F (E1 \times E2) = \sigma_F (E1) \times E2$$

Se **F** é da forma **F1 ^ F2**, onde **F1** só tem atributos em **E1** e **F2**

só tem atributos em **E2**, usando as regras (1), (4) e (6) temos:

$$\sigma_F (E1 \times E2) = \sigma_{F1} (E1) \times \sigma_{F2} (E2)$$

Se F1 tem somente atributos de E1 mas F2 tem atributos de E1 e E2, temos:

$$\sigma_F (E1 \times E2) = \sigma_{F2} (\sigma_{F1} (E1) \times E2)$$

Leis Algébricas

7) Comutatividade de seleção e união

$$\sigma F\ (E1\ U\ E2) = \sigma F\ (E1)\ U\ \ \sigma F\ (E2)$$

8) Comutatividade de seleção e diferença

$$\sigma F\ (E1 - E2) = \sigma F\ (E1) -\ \ \sigma F\ (E2)$$