



Bancos de Dados I

Processamento de Consultas

Prof. Altigran Soares da Silva
IComp/UFAM



1

VISÃO GERAL

Referência: Seções 16.1; 16.1.1 a 16.1.3
GARCIA-MOLINA, Hector; ULLMAN, Jeffrey D.; WIDOM, Jennifer.
Database Systems: The Complete Book (2nd Edition).

2

3

Processamento de Consultas



- Linguagens como a SQL são declarativas
 - O programador especifica o que deve ser feito e não como deve ser feito
 - A idéia é evitar que os programadores tenham de conhecer detalhes do processo de execução
 - As condições para execução da consulta mudam dinamicamente:
 - volume de dados, distribuição dos valores, fragmentação, índices, etc.



Processamento de Consultas

- A execução da consulta é responsabilidade do SGBD
- Existem muitas maneiras de executar uma consulta
- Duas técnicas principais são usadas
 - Aplicação de regras heurísticas para ordenar as operações em uma consulta
 - Comparação de diferentes estratégias com base no seus custos relativos

4

Processamento de Consultas

- Objetivos

- Transformar uma consulta escrita em uma linguagem declarativa de alto nível (SQL) em uma estratégia de execução correta e eficiente expressa em uma linguagem de baixo nível
- Executar corretamente a estratégia para recuperar os dados



Otimização de Consultas

- Atividade de escolher uma estratégia de execução eficiente para processar a consulta
- Como existem muitas estratégias que levam ao mesmo resultados, o objetivo é escolher aquela que minimiza o uso dos recursos.
- Geralmente reduz o tempo de execução



5

Otimização de Consultas

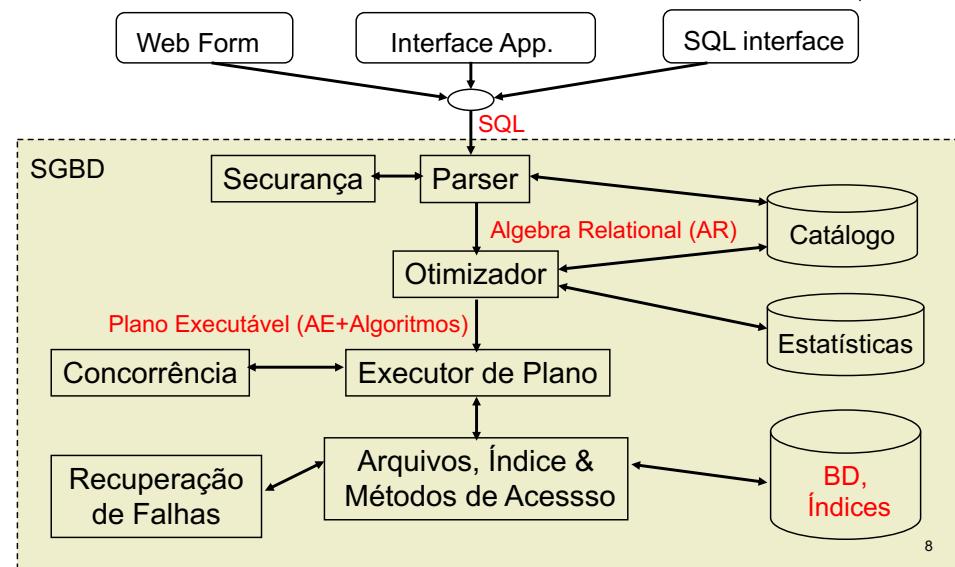


- Dependendo do número de relações e do volume e diversidade dos dados, torna-se um problema computacionalmente intratável.
- Portanto, em geral, os SGBDS produzem estratégias aproximadamente ótimas

7

6

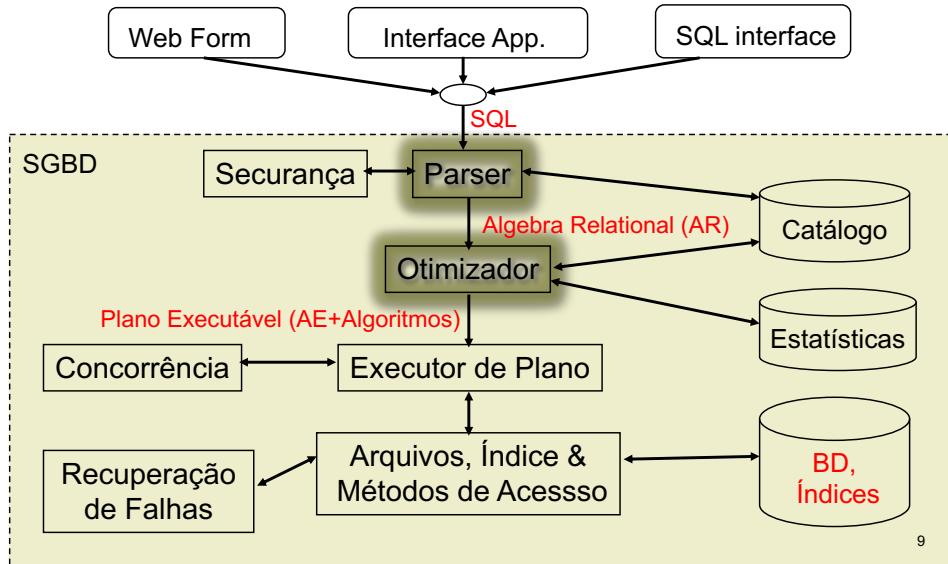
Processamento de Consultas Visão Geral



8

Processamento de Consultas

Nosso Foco

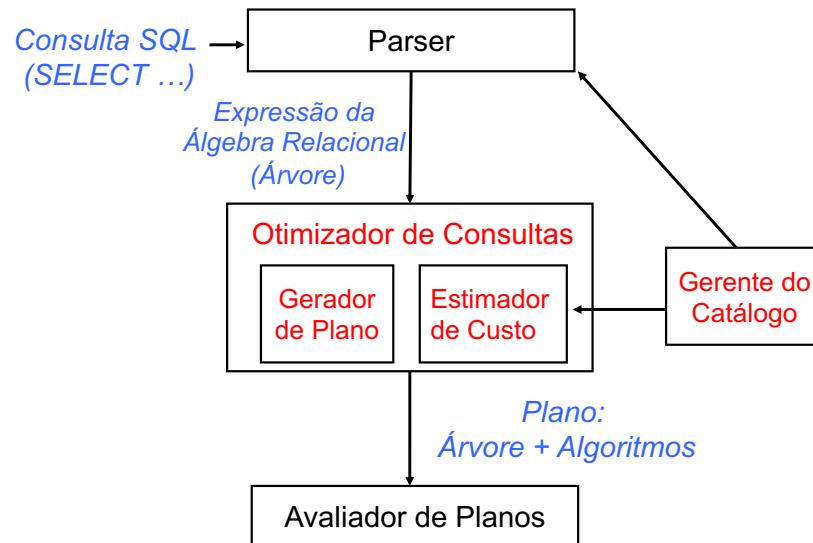


Parser

- Verifica se:
 - A consulta SQL é sintaticamente correta
 - Tabelas, atributos, etc., existem
 - O usuário tem as permissões adequadas
- Transforma a consulta SQL em uma árvore de consulta simples baseada em operadores da Álgebra Relacional



Parsing e Otimização



10



Otimizador

- Gera outras árvores equivalentes
- Para cada árvore gerada
 - Constrói um plano de execução selecionando algoritmos para os operadores
 - Estima o custo do plano
- Escolhe o plano com o menor custo dentre os planos gerados



Plano de Execução da Consulta

- Sequência de passos executados pelo SGBD para obter a resposta a uma consulta
- Seja Q uma consulta, podem haver vários planos de execução de consulta
- O objetivo é gerar e selecionar o melhor PEC considerando o estado atual do BD
 - Volume, distribuição dos dados, índices, etc.
- Nosso foco: Sistemas relacionais

R	A	B	C
a	1	10	
b	1	20	
c	2	10	
d	2	35	
e	3	45	

S	C	D	E
10	x	2	
20	y	2	
30	z	2	
40	x	1	
50	y	3	

Resultado

B	D
2	x

Exemplo

```
SELECT B,D  
FROM R,S  
WHERE R.A="c" AND S.E=2 AND R.C=S.C
```

13

14

Como executar esta consulta?

- Uma possibilidade:
 - Fazer o produto cartesiano das tabelas
 - Selecionar as tuplas que satisfazem as condições
 - Fazer a projeção dos atributos

15

16

R X S

	R.A	R.B	R.C	S.C	S.D	S.E
a	1	10	10	x	2	
a	1	10	20	y	2	
...
b	1	20	10	x	2	
b	1	20	20	y	2	
...
c	2	10	10	x	2	
c	2	10	20	y	2	
c	2	10	30	z	2	
...

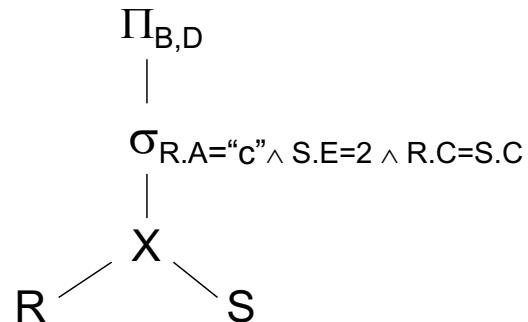
Álgebra Relacional

- Planos de Consultas podem ser descritos em termos da Álgebra Relacional
- Cada passo da execução é descrito em termos de um operador da AR
- Notação de árvore é mais conveniente

17

18

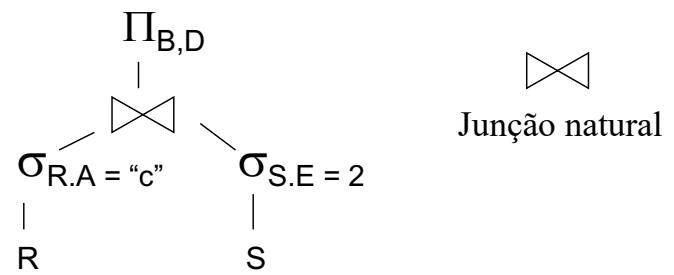
Exemplo – Plano 1



$\Pi_{B,D} [\sigma_{R.A = "c" \wedge S.E = 2 \wedge R.C = S.C} (R \times S)]$

19

Exemplo – Plano 2

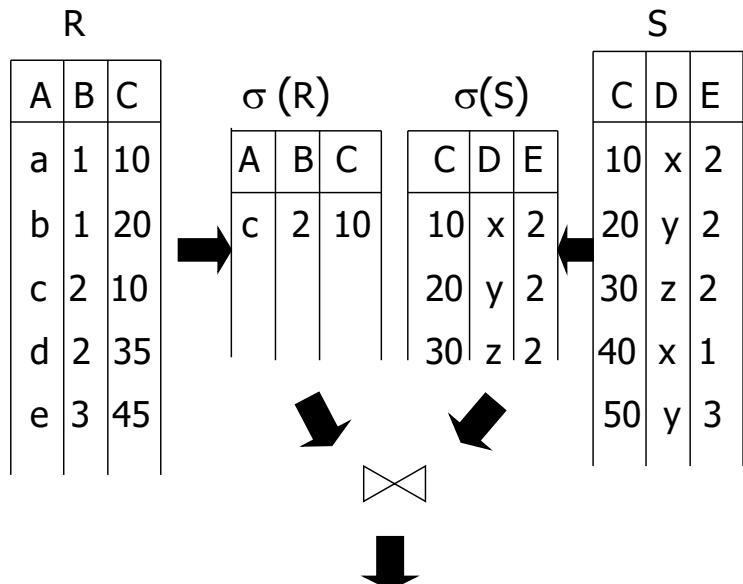


20



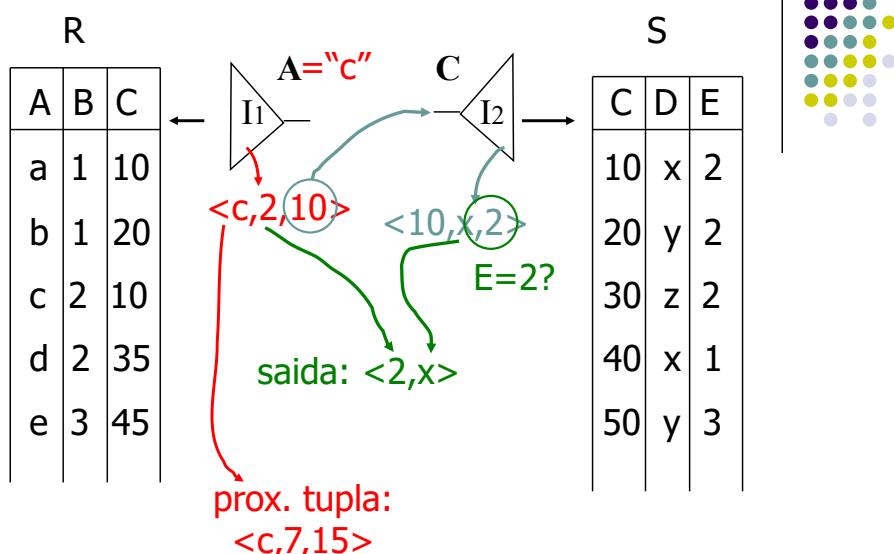
Plano 3

- Usar os índices sobre R.A e S.C
 - (1) Usar o índice sobre R.A para selecionar as tuplas onde R.A = “c”
 - (2) Para cada valor de R.C, usar o índice sobre S.C para achar as tuplas de S
 - (3) Eliminar as tuplas de S onde S.E ≠ 2
 - (4) Aplicar o Join sobre as tuplas de R e S e projetar os atributos B e D



21

22



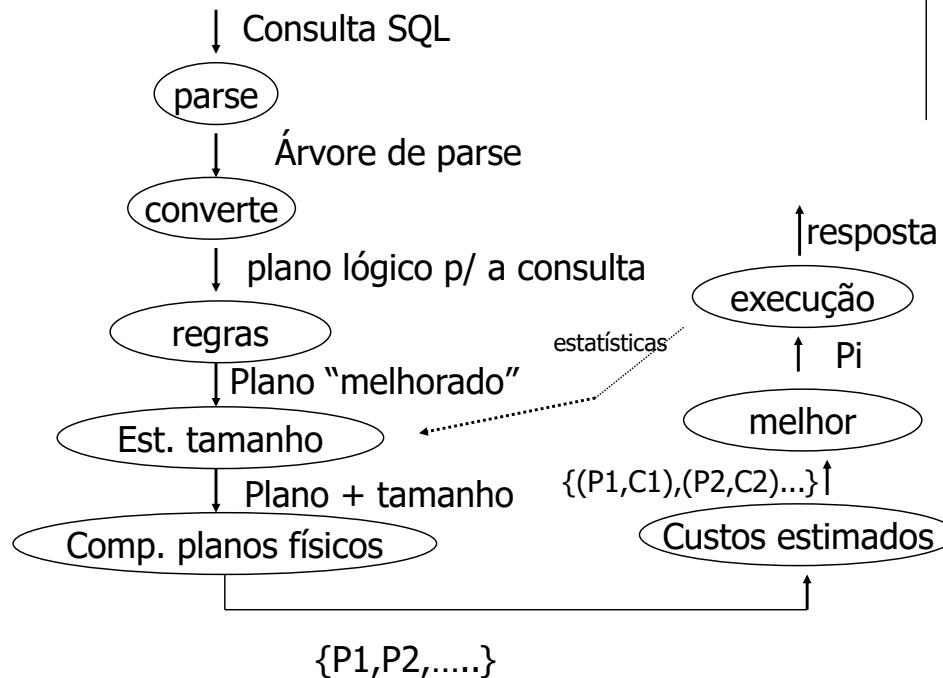
23

24

Otimização de Consulta

- Dada uma consulta, diversos planos são possíveis
- O SGBD precisa determinar qual o “melhor” entre os vários planos de consulta para que seja executado
- A decisão depende do estado atual do BD





25

Exemplo Consulta SQL

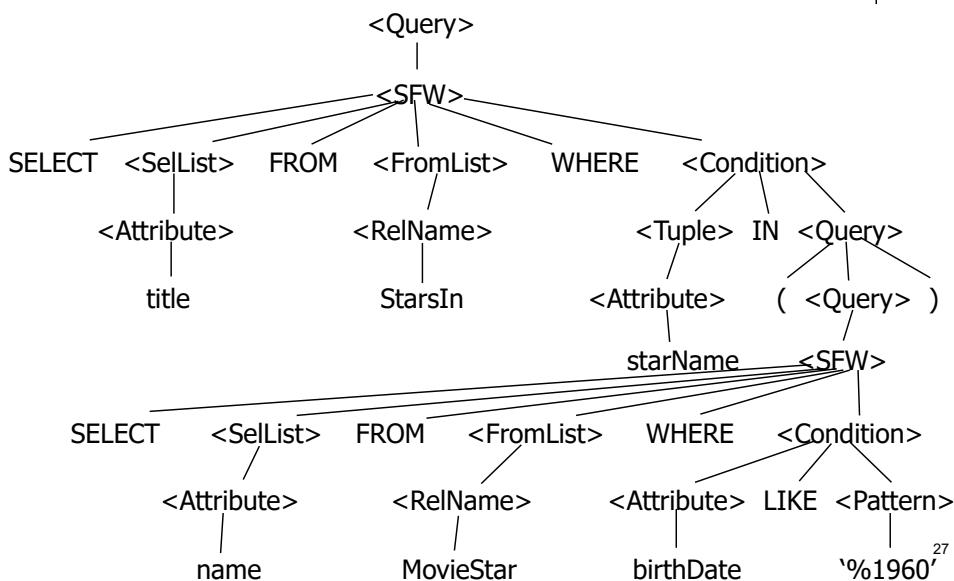
- Filmes cujos atores nasceram em 1960

```

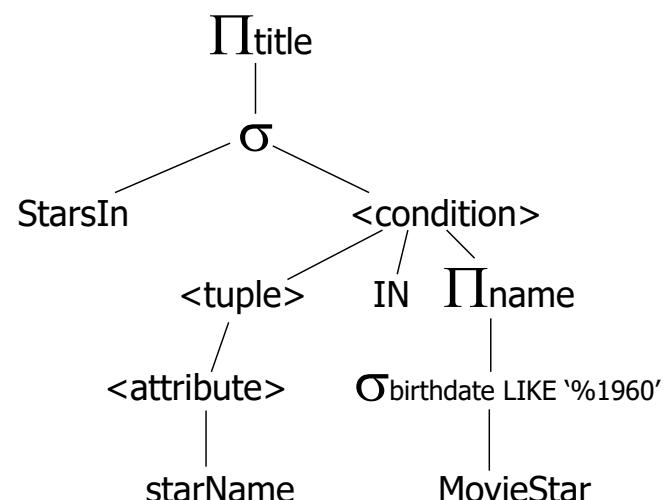
SELECT title
FROM StarsIn
WHERE starName IN (
    SELECT name
    FROM MovieStar
    WHERE birthdate LIKE '%1960'
);
    
```

26

Exemplo Árvore de Parse

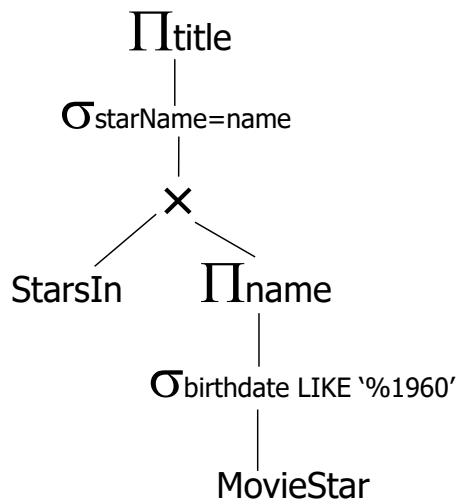


27



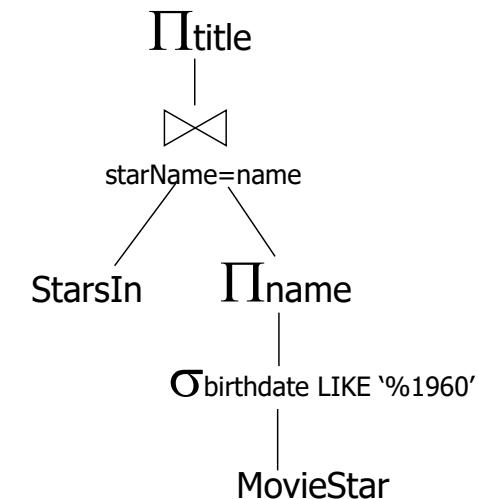
28

Exemplo Plano Lógico



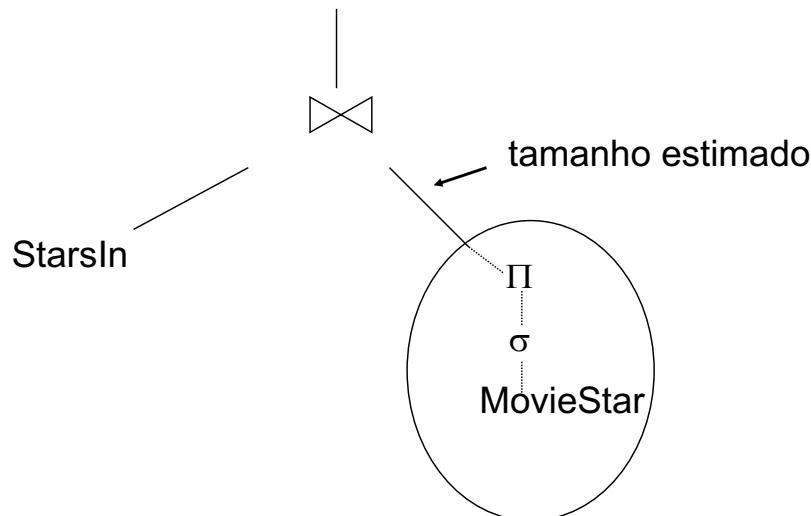
Aplicando regra para condições “IN”

29



30

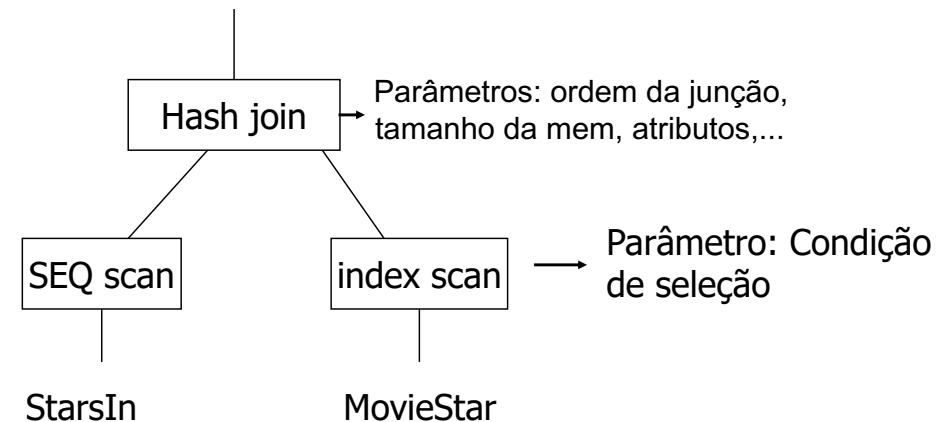
Exemplo Tamanhos de resultados



31

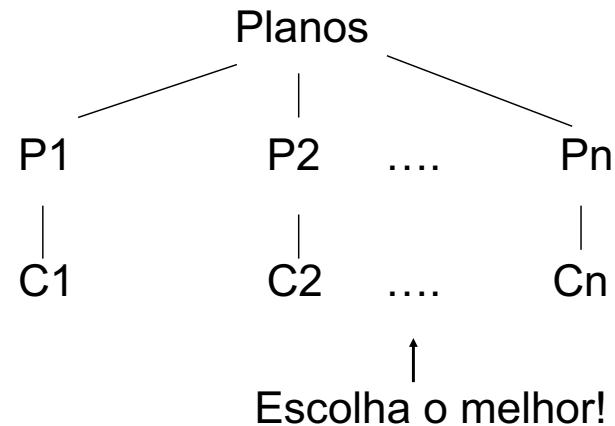
Exemplo Plano Melhorado

Exemplo Possível Plano Físico



32

Exemplo Possíveis Planos



SQL Explain

- Fornece informações sobre o plano de execução usado para uma consulta
- Muito útil quando uma consulta demora a executar mais que o esperado

33

SQL Explain – Exemplo 1



```
Select * from char_name where name='Hercules'
```

id	name	name_pcode_nf	surname_pcode	_search_id
18676	Hercules	H6242		21248471

```
Index Scan using char_name_idx_name on char_name  
(cost=0.42..8.44 rows=1 width=78)  
Index Cond: (name = 'Hercules'::text)
```

34

SQL Explain – Exemplo 2

```
Select * from char_name where name like '%Hercules%'
```

id	name	name_pcode_nf	surname_pcode	search_id
18676	Hercules	H6242		21248471
123658	Young Hercules	Y5262	H6242	21372105
292523	Dr. Hercules	D6242	H6242	21421636
426844	Hercules Jones	H6242	J52	21678900
746653	Hercules Howard	H6242	H63	21979371
748619	Hercules Napolean Cameron	H6242	C565	21940930
796242	Hercules O'Brien	H6242	O165	21979855
944457	The Blacksmith - Hercules	T1425	H6242	22201025
1412943	Hercules 'Herk' Bevans	H6242	B152	22644735
1417337	Hercules Strong	H6242	S3652	22632307
1562657	Hercules Glub	H6242	G41	22764513

35

36

SQL Explain – Exemplo 3



```
EXPLAIN SELECT name.name, title.title  
FROM title, cast_info, name  
WHERE title.id =cast_info.movie_id  
AND cast_info.person_id=name.id  
AND name.name ilike'%denzel%'  
AND name.name ilike'%washington%'  
AND title.title ilike'%gangster%'
```

name	title
Washington, Denzel	American Gangster

SQL Explain – Exemplo 2



```
Select * from char_name where name like '%Hercules%'
```

```
Seq Scan on char_name (cost=0.00..4223.90 rows=21 width=78)  
Filter: (name ~~ '%Hercules%':text)"
```

SQL Explain – Exemplo 3



```
EXPLAIN SELECT name.name, title.title  
FROM title, cast_info, name  
WHERE title.id =cast_info.movie_id  
AND cast_info.person_id=name.id  
AND name.name ilike'%denzel%'  
AND name.name ilike'%washington%'  
AND title.title ilike'%gangster%'
```

```
Nested Loop (cost=4.95..5117.95 rows=1 width=33)  
-> Nested Loop (cost=4.53..5079.86 rows=81 width=22)  
  -> Seq Scan on title (cost=0.00..4089.33 rows=18 width=22)  
    Filter: (title ~~ "%gangster%":text)  
  -> Bitmap Heap Scan on cast_info (cost=4.53..54.90 rows=13 width=8)  
    Recheck Cond: (movie_id = title.id)  
     -> Bitmap Index Scan on cast_info_idx_mid (cost=0.00..4.52 rows=13 width=0)  
        Index Cond: (movie_id = title.id)  
-> Index Scan using name_pkey on name (cost=0.42..0.46 rows=1 width=19)  
  Index Cond: (id = cast_info.person_id)  
  Filter: ((name ~~ '%denzel%':text) AND (name ~~ '%washington%':text))
```

37

38

REGRAS ALGÉBRICAS PARA PLANOS DE CONSULTA



Otimização de Consultas - Níveis



- Nível de Álgebra Relacional
- Nível do Plano de Consulta Detalhado
- Estimativa de custos
 - Sem índices
 - Com índices
- Gerar e comparar planos



Otimização em Álgebra Relacional

- Regras de transformação
 - Preservando equivalência
- O que são boas transformações?

41

Regras Comutativas e Associativas



$$R \bowtie S = S \bowtie R$$

$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

$$R \times S = S \times R$$

$$(R \times S) \times T = R \times (S \times T)$$

$$R \cup S = S \cup R$$

$$R \cup (S \cup T) = (R \cup S) \cup T$$

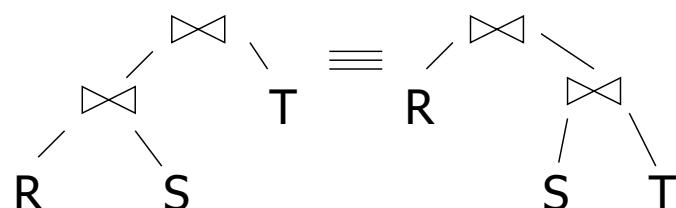
$$R \cap S = S \cap R$$

$$R \cap (S \cap T) = (R \cap S) \cap T$$

42

Observações

- Nomes dos atributos são levados para o resultado. A ordem não é importante
- Expressões podem ser escritas como árvores



43

44

Seleções

$$\sigma_{p_1 \wedge p_2}(R) = \sigma_{p_1} [\sigma_{p_2}(R)]$$

$$\sigma_{p_1 \vee p_2}(R) = [\sigma_{p_1}(R)] \cup [\sigma_{p_2}(R)]$$

$$\sigma_{p_1 \wedge p_2}(R) = \sigma_{p_2 \wedge p_1}(R) = \sigma_{p_2} [\sigma_{p_1}(R)]$$

45

Bags vs. Conjuntos

- $R = \{a, a, b, b, b, c\}$
- $S = \{b, b, c, c, d\}$
- $R \cup S = ?$
 - Opção 1: $R \cup S = \{a, a, b, b, b, c, c, d\}$
 - Opção 2: $R \cup S = \{a, a, b, b, b, b, c, c, c, d\}$

46

Exemplo – Opção 1

T1	
ano	estado
97	AM
99	AM
98	MG

T2	
ano	estado
99	AM
99	AM
98	AM

T1UT2	
ano	estado
97	AM
98	MG
99	AM
99	AM
98	AM

45

Exemplo – Opção 2

T1	
ano	estado
97	AM
99	AM
98	MG

T2	
ano	estado
99	AM
99	AM
98	AM

T1UT2	
ano	estado
97	AM
99	AM
98	MG
99	AM
99	AM
98	AM

48

Projeções

- Sejam: X e Y conjuntos de atributos
 - Notação: $XY = X \cup Y$

$$\pi_{xy}(R) \neq \pi_x[\pi_y(R)]$$

$\sigma + \bowtie$ combinados

$$\begin{aligned}\sigma_p(R \bowtie S) &= [\sigma_p(R)] \bowtie S \\ \sigma_q(R \bowtie S) &= R \bowtie [\sigma_q(S)]\end{aligned}$$

- Onde
 - p = predicado com atributos de R
 - q = predicado com atributos de S

49

$\sigma + \bowtie$ combinados

$$\begin{aligned}\sigma_{p \wedge q}(R \bowtie S) &= [\sigma_p(R)] \bowtie [\sigma_q(S)] \\ \sigma_{p \wedge q \wedge m}(R \bowtie S) &= \sigma_m[(\sigma_p R) \bowtie (\sigma_q S)] \\ \sigma_{pvq}(R \bowtie S) &= [(\sigma_p R) \bowtie S] \cup [R \bowtie (\sigma_q S)]\end{aligned}$$

Onde

- p = predicado com atributos de R
- q = predicado com atributos de S
- m = predicados com atributos de R,S

π e σ combinados

$$\pi_x[\sigma_p(R)] = \pi_x\{\sigma_p[\pi_{xz}(R)]\}$$

- x = subconjunto dos atributos de R
- z = atributos do predicado P

51

50

$\pi + \bowtie$ combinados

$$\pi_{xy} (R \bowtie S) = \pi_{xy} \{ [\pi_{xz} (R)] \bowtie [\pi_{yz} (S)] \}$$

- x = subconjunto dos atribs de R
- y = subconjunto dos atribs de S
- z = interseção dos atribs de R e S



$\pi + \bowtie$ combinados

$$\pi_{xy} \{ \sigma_p (R \bowtie S) \} = \pi_{xy} \{ \sigma_p [\pi_{xz'} (R) \bowtie \pi_{yz'} (S)] \}$$

- x = subconjunto dos atribs de R
- y = subconjunto dos atribs de S
- z = interseção dos atribs de R e S
- z' = z U {atributos usados em P}



σ com op. de conjuntos



$$\sigma_p (R \cup S) = \sigma_p (R) \cup \sigma_p (S)$$

$$\sigma_p (R - S) = \sigma_p (R) - S = \sigma_p (R) - \sigma_p (S)$$

53

Transformações Úteis

54

$$\sigma_{p1 \wedge p2} (R) \rightarrow \sigma_{p1} [\sigma_{p2} (R)]$$

$$\sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S$$

$$R \bowtie S \rightarrow S \bowtie R$$

$$\pi_x [\sigma_p (R)] \rightarrow \pi_x \{ \sigma_p [\pi_{xz} (R)] \}$$



55

56

Convenção: Projeções ocorrem antes

- Seja $R(A,B,C,D,E)$

$$\pi_E \{ \sigma_{(A=3) \wedge (B="gato")}(R) \}$$

- Pode ser re-escrita como:

$$\pi_E \{ \sigma_{(A=3) \wedge (B="gato")} \{ \pi_{ABE}(R) \} \}$$



Aplicação de regras

- Nenhuma transformação por aplicação da regra é boa em todos os casos
- No entanto, executar seleções mais cedo é geralmente melhor

57

59

Mais Transformações



- Discutidas no livro texto

- Eliminação de sub-expressões comuns
- Eliminação de duplicatas
-

Referência: Seções 16.4 16.4.1 a 16.2.4

GARCIA-MOLINA, Hector; ULLMAN, Jeffrey D.; WIDOM, Jennifer.
Database Systems: The Complete Book (2nd Edition).

58

ESTIMATIVA DE CUSTOS

60

Estimativas de Custos

- A escolha do plano de consulta envolve estimar o custo de sua execução
- O custo está relacionado a:
 - Estimativa do tamanho dos resultados
 - Estimativa do número de operações de E/S

Exemplo

R	A	B	C	D
Gato	1	10	a	
Gato	1	20	b	
Cão	1	30	a	
Cão	1	40	c	
Pato	1	50	d	

A: string 20 bytes
B: inteiro 4 bytes
C: data 8 bytes
D: string 5 bytes



Estimativa de Custo – Estatísticas

- Para estimar o tamanho do resultado o SGBD mantém estatísticas atualizadas para cada relação R:
 - $T(R)$: # tuplas em R
 - $S(R)$: # bytes em cada tupla de R
 - $B(R)$: # blocos p/ armazenar todas as tuplas de R
 - $V(R,A)$: # valores distintos em R p/ o atrib. A

61



Estimativa de Tamanho

- Produto Cartesiano: $W = R_1 \times R_2$
 - $T(W) = T(R_1) \times T(R_2)$
 - $S(W) = S(R_1) + S(R_2)$

$$\begin{aligned}T(R) &= 5 \\S(R) &= 37 \\V(R,A) &= 3 \\V(R,C) &= 5 \\V(R,B) &= 1 \\V(R,D) &= 4\end{aligned}$$

62



62



64

Estimativa de Tamanho

- Seleção: $W = \sigma_{Z=val(R)}$

$$T(W) = T(R)/V(R,Z)$$

- Hipótese:

- Valores na expressão de seleção $Z=val$ são uniformemente distribuídos sobre os possíveis valores de Z

$$W = \sigma_{z \geq val}(R)$$

- Solução 1:
 - Há 50% de chance da tupla satisfazer a condição
 - $T(W) = T(R)/2$
- Solução 2:
 - Intuitivamente, menos que a metade satisfaz
 - $T(W) = T(R)/3$

Exemplo

R	A	B	C	D
Gato	1	10	a	
Gato	1	20	b	
Cão	1	30	a	
Cão	1	40	c	
Pato	1	50	d	

$$T(R) = 5$$

$$V(R,A) = 3 \quad V(R,C) = 5$$

$$V(R,B) = 1 \quad V(R,D) = 4$$

$$W = \sigma_{Z=val(R)}$$

$$T(W) = T(R)/V(R,Z)$$

Z	T(R)	V(R,Z)	T(W)
A	5	3	1,67
B	5	1	5,00
C	5	5	1,00
D	5	4	1,25

65

66

$$W = \sigma_{z \geq val}(R)$$

- Solução 3: Estimar faixa de valores

$$T(W) = f \times T(R)$$

$$f = \text{valores de } z \text{ dentro da faixa}$$

67

68

$W = \sigma_{z \geq \text{val}} (R)$

- Exemplo

- $W = \sigma_{z \geq 15} (R)$

R	z
	min = 1 máx = 20

$$f = \frac{20 - 15 + 1}{20 - 1 + 1} = \frac{6}{20}$$



$W = R1 \bowtie R2$

- Seja $R1(X)$ e $R2(Y)$

- $X \cap Y = A$

- Hipótese 1: $V(R1, A) \leq V(R2, A)$
 - Cada valor de A em R1 está em R2
- Hipótese 2: $V(R2, A) \leq V(R1, A)$
 - Cada valor de A em R2 está em R1

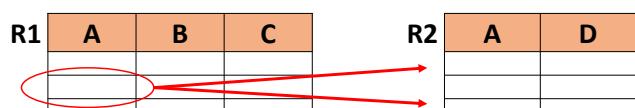
R1	A	B	C
R2	A		D

69

$W = R1 \bowtie R2$

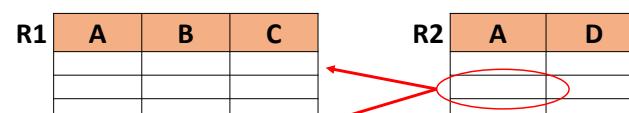
- Hipótese 1: $V(R1, A) \leq V(R2, A)$

- Cada tupla de R1 casa com $T(R2)/V(R2, A)$ tuplas



- Logo:

$$T(W) = T(R1) \times \frac{T(R2)}{V(R2, A)} = \frac{T(R1) T(R2)}{V(R2, A)}$$



$$T(W) = T(R2) \times \frac{T(R1)}{V(R1, A)} = \frac{T(R2) T(R1)}{V(R1, A)}$$

71

70



$W = R1 \bowtie R2$

- Em geral:

$$T(W) = \frac{T(R1) T(R2)}{\max\{ V(R1,A), V(R2,A) \}}$$

$W = R1 \bowtie R2$

- Tamanho das Tuplas

$$S(W) = S(R1) + S(R2) - S(A)$$

75

$W = R1 \bowtie R2$

- Hipótese Alternativa:

- Valores uniformemente distribuídos sobre o domínio
- Cada tupla de R1 casa com T(R2)/DOM(A) tuplas
 - DOM(A) = Tamanho do domínio de A

$$T(W) = \frac{T(R1) T(R2)}{DOM(A)}$$

73

74

Outros Operadores

$\Pi_{AB}(R)$ – Seção 16.4.2

$\sigma_{A=a \wedge B=b}(R)$ – Seção 16.4.3

$R \bowtie S$ c/ vários atributos comuns – Seção 16.4.5

União, Intersecção, Diferença – Seção 16.4.7

76



Expressões Complexas

- Estimar tamanhos dos valores intermediários
- Exemplo:
 - $W = [\sigma_{D=a}(R1)] \bowtie R2$
 - Estimar $U = \sigma_{D=a}(R1)$
 - $T(U) = T(R1)/V(R1,A)$
 - $S(U) = S(R1)$
 - $V(U, *) = ???? ...$

Estimando $V(U, *)$

R1	A	B	C	D
Gato	1	10	a	
Gato	1	20	b	
Cão	1	30	a	
Cão	1	40	c	
Pato	1	50	d	

$$\begin{aligned} V(R1,A) &= 3 \\ V(R1,B) &= 1 \\ V(R1,C) &= 5 \\ V(R1,D) &= 4 \end{aligned}$$

$$U = \sigma_{D=a}(R1)$$

$$V(U,D) = 1 \quad V(U,B) = 1 \quad V(U,C) = \frac{T(R1)}{V(R1,D)}$$

77

78



Expressões Complexas – Junções

- Assim:
 - $V(U,D) = 1$
 - $V(U,X) = V(R1,X), X \neq D$
- $U = R1(A,B) \bowtie R2(A,C)$
- $V(U,A) = \min \{V(R1,A), V(R2,A)\}$
- $V(U,B) = V(R1,B)$
- $V(U,C) = V(R2,C)$



79

80



Exemplo – Junções Complexas

- $Z = R1(A,B) \bowtie R2(B,C) \bowtie R3(C,D)$
 - $T(R1) = 1000$ $V(R1,A)=50$ $V(R1,B)=100$
 - $T(R2) = 2000$ $V(R2,B)=200$ $V(R2,C)=300$
 - $T(R3) = 3000$ $V(R3,C)=90$ $V(R3,D)=500$



Exemplo – Junções Complexas

- $U = R1 \bowtie R2$

- $V(U,A) = 50$
- $V(U,B) = 100$
- $V(U,C) = 300$

$$T(U) = \frac{1000 \times 2000}{200}$$

- $Z = U \bowtie R3$

- $V(Z,A) = 50$
- $V(Z,B) = 100$
- $V(Z,C) = 90$
- $V(Z,D) = 500$

$$T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300}$$

81

82



Estatísticas do BD

- A estimativa de custos depende de dados estatísticos mantidos pelo SGBD
- A qualidade da estimativa depende do volume, da granularidade, e da frequência de atualização destas estatísticas



Estatísticas do BD

- A manutenção das estatísticas deve ser problemática
- Se elas forem atualizadas muito frequentemente para cada tupla, pode haver impacto no desempenho
- O SGBD pode atualizar as estatísticas periodicamente quando o sistema está livre

83

84

Programação Dinâmica

- Mesmo para consultas médias, podem haver milhões de planos de execução gerados
- A computação do custo de um plano pode levar alguns mili-segundos, resultando em horas para otimizar uma consulta.

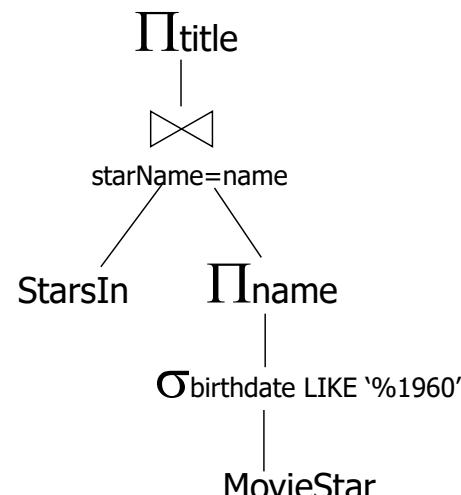


Programação Dinâmica

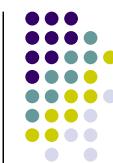
- Em 1979 Selinger propôs uma heurística baseada em Programação Dinâmica
- Estratégia Bottom-Up
 - Otimizar sub-consultas de uma tabela.
 - Utilizar estes planos ótimos para otimizar as sub-consultas de duas tabelas
 - Utilizar estes planos ótimos para otimizar as sub-consultas de três tabelas
 - ...



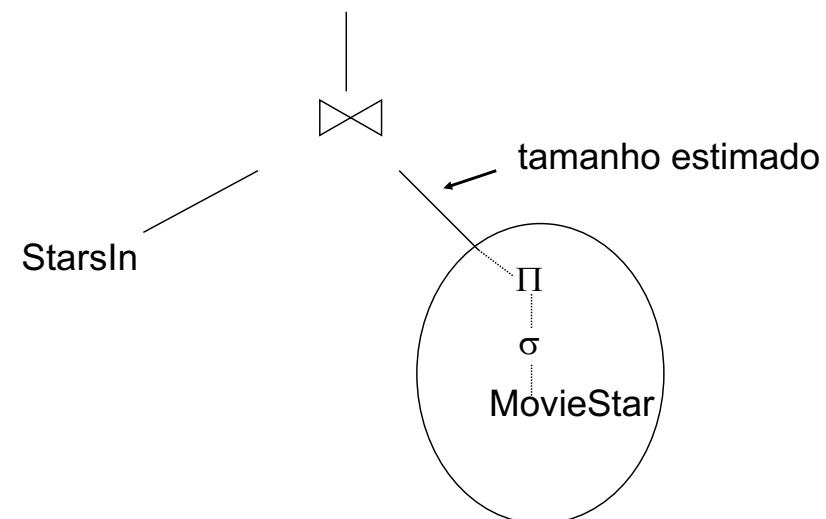
Plano Consulta



85



Tamanhos de resultados

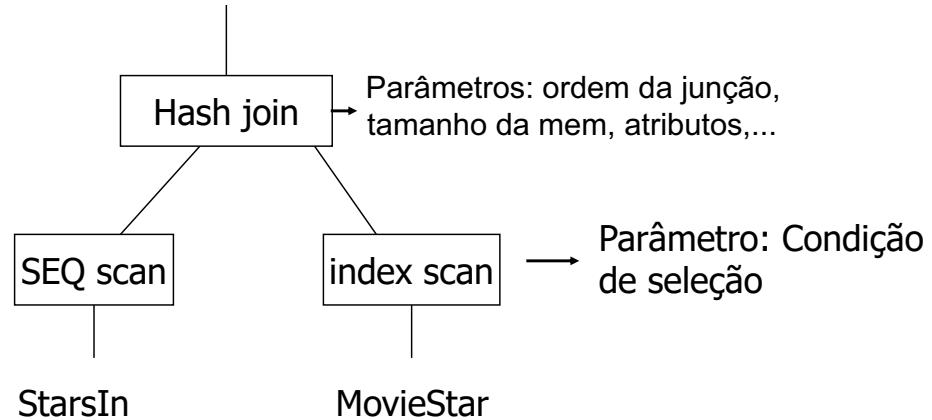


87

86

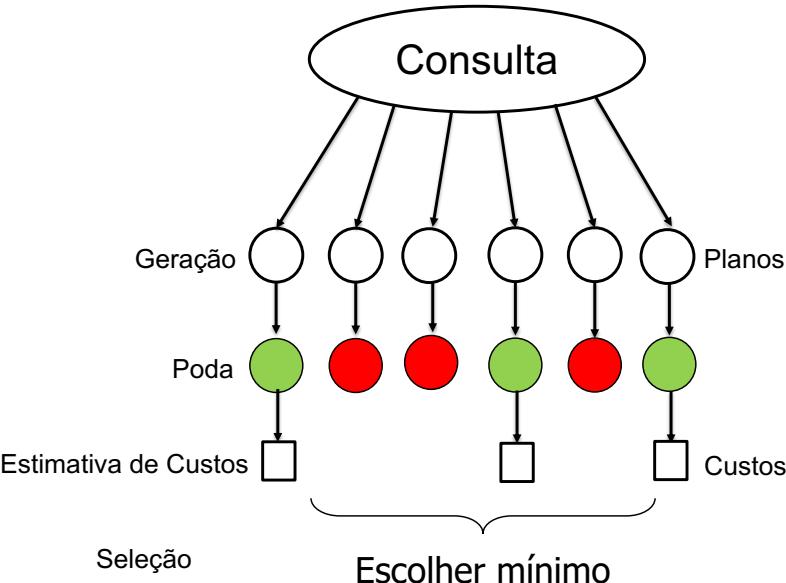


Plano Físico



89

Otimização de Consultas



90

EXECUÇÃO DE CONSULTAS ALGORITMOS E ESTRATÉGIAS

Referência: Seções 12.3 a 12.7

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de Banco de Dados. 6a edição. Editora Campus, 2016. ISBN: 9788535245356.

91

92

Algoritmos para Operadores

Operação de Seleção

- Varredura (File Scan)
- Algoritmo A1 – Busca Linear
 - Recupera cada bloco do arquivo e testa todos os registros para verificar se a condição é satisfeita
 - Custo = N transferências de bloco + 1 busca
 - Se a condição envolve uma chave, a busca termina quando o registro é encontrado record
 - Custo = $(N/2)$ transferências de bloco + 1 busca



Operação de Seleção

- Busca linear independe da condição de seleção, ordenação de registros, índices, etc.
- Nota: busca binária só faz sentido quando os blocos são contíguos



93

Seleção com Índices



- Varredura de Índice
 - Condição de seleção sobre o campo indexado
- A2: Índice Primário, Igualdade sobre chave
 - Recupera um único registro
 - Custo = $(h_i + 1) * (t_T + t_S)$ – onde h_i é altura do índice
- A3: Índice Primário, Igualdade sobre não-chave
 - Recupera múltiplos registros em blocos consecutivos
 - Custo = $h_i * (t_T + t_S) + t_S + t_T * b$
 - b = número de blocos de dados que satisfazem o argumento

t_T =tempo de transferência, t_S =tempo de seek + latência

94

Seleção com Índices



- A4: Índice Secundário com igualdade
- Recupera um único registo se a busca é sobre uma chave alternativa
 - Custo = $(h_i + 1) * (t_T + t_S)$
- Recupera múltiplos regtos se a busca não é sobre uma chave alternativa
 - Cada um dos n registros a recuperar pode estar em um bloco diferente – pode ser caro
 - Custo = $(h_i + n) * (t_T + t_S)$

t_T =tempo de transferência, t_S =tempo de seek + latência

95

Seleções tipo $\sigma_{A \leq v}(r)$ ou $\sigma_{A \geq v}(r)$

- A5: Índice Primário. Arquivo ordenado por A

- $\sigma_{A \geq v}(r)$: usar o índice para encontrar o primeiro registro $A \geq v$. Varrer sequencialmente a partir daí.
- $\sigma_{A \leq v}(r)$ varrer o arquivo sequencialmente até encontrar o primeiro $A > v$; não usar índices.

Seleções Complexas $\sigma_{\theta_1 \wedge \dots \wedge \theta_n}(r)$

- A7: Seleção Conjuntiva usando um índice
 - Escolhe uma combinação de θ_i e algoritmos de A1 a A6 que resulta no menor custo para $\sigma_{\theta_i}(r)$
 - Carrega o resultado nos buffers para em seguida testar as outras condições
- A8: Seleção Conjuntiva c/ Índice Composto
 - Usar índice composto por múltiplos atributos, se houver.

Seleções tipo $\sigma_{A \leq v}(r)$ ou $\sigma_{A \geq v}(r)$

- A6: Índice Secundário

- $\sigma_{A \geq v}(r)$: usar o índice para encontrar o primeiro registro $A \geq v$.
- $\sigma_{A \leq v}(r)$: varer as folhas do índice até encontrar o primeiro $A > v$
- Em ambos os casos, recuperar os registros de dados
 - Requer uma operação de E/S para cada registro
 - Varredura linear pode ser mais barata

97

Seleções Complexas $\sigma_{\theta_1 \wedge \dots \wedge \theta_n}(r)$

Seleções Complexas $\sigma_{\theta_1 \wedge \dots \wedge \theta_n}(r)$

- A9: Seleção Conjuntiva com interseção de índices
 - Usar o índice correspondente a cada condição e tomar a interseção dos apontadores de dados dos índices
 - Carregar registros do arquivo de dados
 - Condições adicionais verificadas em memória

99

98

100

Seleções Complexas $\sigma_{\theta_1 \vee \dots \vee \theta_n}(r)$

- A10: Seleção Disjuntiva pela união de índices
 - Aplicável somente se há índices para todas as condições. Se não, usar varredura linear
 - Usar os índices e fazer a união dos apontadores de dados
- Negações: $\sigma_{\neg\theta}(r)$
 - Usar varredura linear nos arquivo de dados
 - Índices só são usados se poucos registros satisfazem $\neg\theta$



Ordenação

- Se a relação cabe na memória, usar técnicas como QuickSort
- Caso contrário, usar ordenação externa
- Alternativa: gerar um índice e varrer a relação de forma ordenada.



Junções

- Vários Algoritmos
 - Nested-loop join
 - Block nested-loop join
 - Indexed nested-loop join
 - Merge-join
 - Hash-join
- Escolha baseada em estimativa de custos

101



Nested-Loop Join: $r \bowtie_{\theta} s$



Para cada tupla t_r em r faça
 Para cada tupla t_s em s faça
 Se(t_r, t_s) satisfaz a condição θ
 Gerar $t_r \cdot t_s$

- r é a *relação externa* e s é a *relação interna*
- Não usa índices
- Pode ser usado por qualquer tipo de junção
- Caro: examina todo os pares de tuplas

102

102

103

104

Nested-Loop Join: $r \bowtie_{\theta} s$

- Pior caso: só há memória suficiente para um bloco de cada relação.
 - Custo: $b_r + n_r * b_s$ blocos e $b_r + n_r$ seeks
 - Cada bloco de s é lido uma vez para cada registro de r
- Melhor caso: se a menor relação cabe na memória, usá-la como relação interna
 - Custo: $b_r + b_s$ blocos e 2 seeks

n_x =nr. de registros de x e b_x =nr. de blocos de x

NLJ – Exemplo

Aluno

Matrícula	Nome	...
21456055	Ana	...
21200462	Bruno	...
21453374	Carlos	...
...

Cursa

Matrícula	Disciplina	...
21456055	BD1	...
21456055	ES	...
21200462	AFC	...
21200462	BD1	...
21200462	CN1	...
21453374	BD2	...
...

Nr. de registros em Aluno: $n_A = 5.000$
Nr. de blocos em Aluno: $b_A = 100$

Nr. de registros em Cursa: $n_C = 10.000$
Nr. de blocos em Cursa: $b_C = 400$

NLJ – Exemplo

- $5.000 \times 10.000 = 50 \times 10^6$ registros
- Custo pior caso: $b_r + n_r * b_s$ blocos e $b_r + n_r$ seeks
 - Externo: Aluno, Interno: Cursa ($A \bowtie C$)
 - $100 + 5.000 * 400 = 2.000.100$ blocos
 - $5.000 + 100 = 5.100$ seeks
 - Externo: Cursa, Interno: Aluno ($C \bowtie A$)
 - $400 + 10.000 * 100 = 1.000.400$ blocos
 - $400 + 10.000 = 10.400$ seeks
- Custo melhor caso: $b_r + b_s$ e 2 seeks
 - $100 + 400 = 500$ blocos

$$\begin{aligned} n_A &= 5.000; \\ b_A &= 100; \\ n_C &= 10.000; \\ b_C &= 400 \end{aligned}$$

105

Block Nested-Loop Join

Para cada bloco B_r em r faça
 Para cada bloco B_s em s faça
 Para cada tupla t_r em B_r faça
 Para cada tupla t_s em B_s faça
 Se (t_r, t_s) satisfaz a condição
 Gerar $t_r \times t_s$

- Variante do nested-loop join não qual cada bloco da relação interna é pareada com cada bloco da relação externa.

107

108

Block Nested-Loop Join

- Pior caso (estimava):
 - $b_r + b_r * b_s$ transferência de blocos + $2 * b_r$ seeks
 - Cada bloco de s é lido uma vez para cada bloco de r
- Melhor caso:
 - $b_r + b_s$ transferência de blocos + 2 seeks.

$b_x = \text{nr. de blocos de } x$

109

BNLJ - Exemplo

- Pior caso: $b_r + b_r * b_s$ blocos e $2 * b_r$ seeks
 - $100 + 100 * 400 = 40.100$ blocos e 200 seeks ($A \bowtie C$)
 - $400 + 400 * 100 = 40.400$ blocos e 800 seeks ($C \bowtie A$)
 - Muito melhor que NLJ (2 mi ou 1 mi de bls)
- Melhor caso: $b_r + b_s$ blocos e 2 seeks
 - $100 + 400 = 500$ blocos

$n_A = 5.000;$
 $b_A = 100;$
 $n_C = 10.000;$
 $b_C = 400$



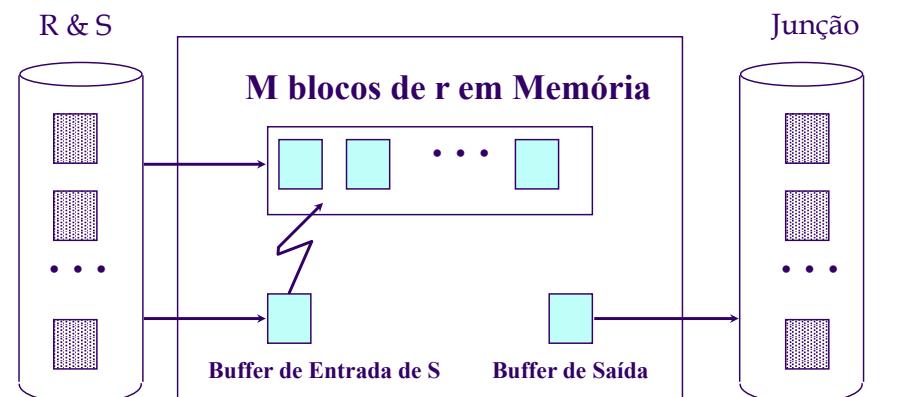
Block Nested-Loop Join

- Otimizações:
 - $M-2$ blocos em memória para a relação externa
 - 1 bloco para a rel. interna e 1 bloco para saída
- Custo:
 - $b_r + \lceil b_r / (M-2) \rceil * b_s$ blocos
 - $2 \lceil b_r / (M-2) \rceil$ seeks

$b_x = \text{nr. de blocos de } x$

111

Block Nested-Loop Join



112

Block Nested-Loop Join

- Otimizações:
 - Se os atributos da equi-junção formam uma chave na relação interna, o loop interno pode parar quando o primeiro valor é encontrado.
 - Zig-Zag join: Para usar melhor o buffer, varrer a relação interna de frente pra trás e de trás para frente alternadamente



Indexed Nested-Loop Join

- Busca no índice ao invés varrer os arquivos
 - Equi-junção + índice na relação interna
- Para cada tupla t_r da relação externa r , buscar no índice as tuplas em s que satisfazem a junção
- Observações:
 - Índice pode ser construído só para a junção
 - Se r e s tem índices no atributo da junção, usar a menor delas como relação interna



113

Indexed Nested-Loop Join



- Pior caso (estimava):
 - Para cada registro de r , procura a chave no índice e lê um bloco de s
 - $b_r + n_r * (h_i + 1)$ blocos, $b_r + n_r * (h_i + 1)$ seeks
- Melhor caso:
 - O índice i cabe todo na memória
 - $b_r + b_i$ transferência de blocos + 2 seeks.

114

INLJ – Exemplo

$$\begin{aligned}n_A &= 5.000; \\b_A &= 100; \\n_C &= 10.000; \\b_C &= 400\end{aligned}$$



- $A \bowtie C$
- Considere uma árvore B^+ sobre Matrícula em com $m=10$ chaves em cada nodo e $h_i = 4$
- Pior caso
 - $b_r + n_r * (h_i + 1) = 100 + 5.000 * (4 + 1) = 25.100$ blocos
 - O mesmo para seeks



Merge-Join

- Se aplica para equi-joins e junções naturais quando os arquivos são ordenados pelo atributo de juncão
- Dependendo do custo, um ou os dois arquivos podem ser antes ordenados
- Varre os dois arquivos simultaneamente procurado por casamentos.

Merge Join

Id	Nome	Ano	Votos	Filme	Gênero
1707262	12 Angry Men	1957	253328	1747853	Action
1710163	2001: A Space Odyssey	1968	262937	1710163	Adventure
1711969	3 Idiots	2009	79396	1747853	Adventure
1715511	8½	1963	52035	1747853	Biography
1716360	A Beautiful Mind	2001	329545	1716360	Biography
1718136	A Clockwork Orange	1971	342375	1711969	Comedy
1730508	A Streetcar Named Desire	1951	57871	1718136	Crime
1747640	Alien	1979	345531	1707262	Drama
1747853	Aliens	1986	312063	1711969	Drama
1748246	All About Eve	1950	54814	1715511	Drama
1749122	All Quiet on the Western Front	1930	36574	1716360	Drama
1752523	Amadeus	1984	175277	1718136	Drama
1754047	American Beauty	1999	534179	1747640	Fantasy

118

117

Merge Join



Id	Nome	Ano	Votos
1707262	12 Angry Men	1957	253328
1710163	2001: A Space Odyssey	1968	262937
1711969	3 Idiots	2009	79396
1715511	8½	1963	52035
1716360	A Beautiful Mind	2001	329545
1718136	A Clockwork Orange	1971	342375
1730508	A Streetcar Named Desire	1951	57871
1747640	Alien	1979	345531
1747853	Aliens	1986	312063
1748246	All About Eve	1950	54814
1749122	All Quiet on the Western Front	1930	36574
1752523	Amadeus	1984	175277
1754047	American Beauty	1999	534179

Id	Nome
1707262	12 Angry Men
1710163	2001: A Space Odyssey
1711969	3 Idiots
1715511	8½
1716360	A Beautiful Mind
1718136	A Clockwork Orange
1730508	A Streetcar Named Desire
1747640	Alien
1747853	Aliens
1748246	All About Eve
1749122	All Quiet on the Western Front
1752523	Amadeus
1754047	American Beauty
Filme	Gênero
1707262	Drama
1710163	Adventure
1710163	Mystery
1710163	Sci-Fi
1711969	Comedy
1711969	Romance
1715511	Drama
1715511	Fantasy
1716360	Biography
1716360	Drama
1718136	Crime
1718136	Drama
1730508	Drama
1747640	Horror
1747640	Sci-Fi
1747853	Action
1747853	Adventure
1747853	Sci-Fi

119

Merge Join

Id	Nome	Ano	Votos	Filme	Gênero
1707262	12 Angry Men	1957	253328	1707262	Drama
1710163	2001: A Space Odyssey	1968	262937	1710163	Adventure
1711969	3 Idiots	2009	79396	1710163	Mystery
1715511	8½	1963	52035	1710163	Sci-Fi
1716360	A Beautiful Mind	2001	329545	1711969	Comedy
1718136	A Clockwork Orange	1971	342375	1711969	Romance
1730508	A Streetcar Named Desire	1951	57871	1715511	Drama
1747640	Alien	1979	345531	1715511	Fantasy
1747853	Aliens	1986	312063	1716360	Biography
1748246	All About Eve	1950	54814	1716360	Drama
1749122	All Quiet on the Western Front	1930	36574	1718136	Crime
1752523	Amadeus	1984	175277	1718136	Drama
1754047	American Beauty	1999	534179	1747640	Horror

120

Merge Join

Id	Nome	Ano	Votos
1707262	12 Angry Men	1957	253328
1710163	2001: A Space Odyssey	1968	262937
1711969	3 Idiots	2009	79396
1715511	8½	1963	52035
1716360	A Beautiful Mind	2001	329545
1718136	A Clockwork Orange	1971	342375
1730508	A Streetcar Named Desire	1951	57871
1747640	Alien	1979	345531
1747853	Aliens	1986	312063
1748246	All About Eve	1950	54814
1749122	All Quiet on the Western Front	1930	36574
1752523	Amadeus	1984	175277
1754047	American Beauty	1999	534179

Filme	Gênero
1707262	Drama
1710163	Adventure
1710163	Mystery
1710163	Sci-Fi
1711969	Comedy
1711969	Drama
1711969	Romance
1715511	Drama
1715511	Fantasy
1716360	Biography
1716360	Drama
1718136	Crime
1718136	Drama
1718136	Sci-Fi
1730508	Drama
1747640	Horror
1747640	Sci-Fi
1747853	Action
1747853	Adventure
1747853	Sci-Fi

121

Merge Join

Id	Nome	Ano	Votos
1707262	12 Angry Men	1957	253328
1710163	2001: A Space Odyssey	1968	262937
1711969	3 Idiots	2009	79396
1715511	8½	1963	52035
1716360	A Beautiful Mind	2001	329545
1718136	A Clockwork Orange	1971	342375
1730508	A Streetcar Named Desire	1951	57871
1747640	Alien	1979	345531
1747853	Aliens	1986	312063
1748246	All About Eve	1950	54814
1749122	All Quiet on the Western Front	1930	36574
1752523	Amadeus	1984	175277
1754047	American Beauty	1999	534179

Filme	Gênero
1707262	Drama
1710163	Adventure
1710163	Mystery
1710163	Sci-Fi
1711969	Comedy
1711969	Drama
1711969	Romance
1715511	Drama
1715511	Fantasy
1716360	Biography
1716360	Drama
1718136	Crime
1718136	Drama
1718136	Sci-Fi
1730508	Drama
1747640	Horror
1747640	Sci-Fi
1747853	Action
1747853	Adventure
1747853	Sci-Fi

122

Merge Join

Id	Nome	Ano	Votos
1707262	12 Angry Men	1957	253328
1710163	2001: A Space Odyssey	1968	262937
1711969	3 Idiots	2009	79396
1715511	8½	1963	52035
1716360	A Beautiful Mind	2001	329545
1718136	A Clockwork Orange	1971	342375
1730508	A Streetcar Named Desire	1951	57871
1747640	Alien	1979	345531
1747853	Aliens	1986	312063
1748246	All About Eve	1950	54814
1749122	All Quiet on the Western Front	1930	36574
1752523	Amadeus	1984	175277
1754047	American Beauty	1999	534179

Filme	Gênero
1707262	Drama
1710163	Adventure
1710163	Mystery
1710163	Sci-Fi
1711969	Comedy
1711969	Drama
1711969	Romance
1715511	Drama
1715511	Fantasy
1716360	Biography
1716360	Drama
1718136	Crime
1718136	Drama
1718136	Sci-Fi
1730508	Drama
1747640	Horror
1747640	Sci-Fi
1747853	Action
1747853	Adventure
1747853	Sci-Fi

123

Merge Join

Id	Nome	Ano	Votos
1707262	12 Angry Men	1957	253328
1710163	2001: A Space Odyssey	1968	262937
1710163	3 Idiots	2009	79396
1710163	Sci-Fi		
1711969	8½	1963	52035
1716360	A Beautiful Mind	2001	329545
1718136	A Clockwork Orange	1971	342375
1730508	A Streetcar Named Desire	1951	57871
1747640	Alien	1979	345531
1747853	Aliens	1986	312063
1748246	All About Eve	1950	54814
1749122	All Quiet on the Western Front	1930	36574
1752523	Amadeus	1984	175277
1754047	American Beauty	1999	534179

Filme	Gênero
1707262	Drama
1710163	Adventure
1710163	Mystery
1710163	Sci-Fi
1711969	Comedy
1711969	Drama
1711969	Romance
1715511	Drama
1715511	Fantasy
1716360	Biography
1716360	Drama
1718136	Crime
1718136	Sci-Fi
1730508	Drama
1747640	Horror
1747640	Sci-Fi
1747853	Action
1747853	Adventure
1747853	Sci-Fi

124

Merge Join

Id	Nome	Ano	Votos
1707262	12 Angry Men	1957	253328
1710163	2001: A Space Odyssey	1968	262937
1711969	3 Idiots	2009	79396
1715511	8½	1963	52035
1716360	A Beautiful Mind	2001	329545
1718136	A Clockwork Orange	1971	342375
1730508	A Streetcar Named Desire	1951	57871
1747640	Alien	1979	345531
1747853	Aliens	1986	312063
1748246	All About Eve	1950	54814
1749122	All Quiet on the Western Front	1930	36574
1752523	Amadeus	1984	175277
1754047	American Beauty	1999	534179

Filme	Gênero
1707262	Drama
1710163	Adventure
1710163	Mystery
1710163	Sci-Fi
1711969	Comedy
1711969	Drama
1711969	Romance
1715511	Drama
1715511	Fantasy
1716360	Biography
1716360	Drama
1718136	Crime
1718136	Drama
1718136	Sci-Fi
1730508	Drama
1747640	Horror
1747640	Sci-Fi
1747853	Action
1747853	Adventure
1747853	Sci-Fi

125

Merge Join

Id	Nome	Ano	Votos
1707262	12 Angry Men	1957	253328
1710163	2001: A Space Odyssey	1968	262937
1711969	3 Idiots	2009	79396
1715511	8½	1963	52035
1716360	A Beautiful Mind	2001	329545
1718136	A Clockwork Orange	1971	342375
1730508	A Streetcar Named Desire	1951	57871
1747640	Alien	1979	345531
1747853	Aliens	1986	312063
1748246	All About Eve	1950	54814
1749122	All Quiet on the Western Front	1930	36574
1752523	Amadeus	1984	175277
1754047	American Beauty	1999	534179

Filme	Gênero
1707262	Drama
1710163	Adventure
1710163	Mystery
1710163	Sci-Fi
1711969	Comedy
1711969	Drama
1711969	Romance
1715511	Drama
1715511	Fantasy
1716360	Biography
1716360	Drama
1718136	Crime
1718136	Drama
1718136	Sci-Fi
1730508	Drama
1747640	Horror
1747640	Sci-Fi
1747853	Action
1747853	Adventure
1747853	Sci-Fi

126

Merge-Join

- Cada bloco precisa ser lido somente uma vez, assumindo que todas os registros de um dado valor do atributo de junção cabe na memória.
- Custo:
 - $b_r + b_s$ blocos
 - $\lceil b_r/B \rceil + \lceil b_s/B \rceil$ seeks; B = blocos no buffer
 - + custo de ordenação, se necessário.

$b_x = \text{nr. de blocos de } x$

Merge-Join Híbrido

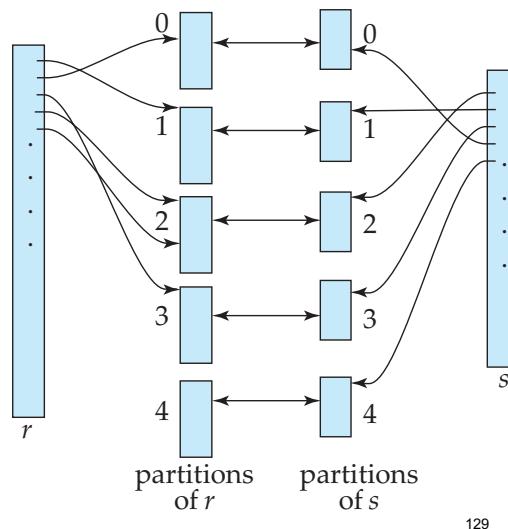
- Útil se um arquivo R está ordenado e outro tem uma Árvore-B⁺ como índice secundário no atributo de junção.
- Faz merge de R com as folhas da Árvore

127

128

Hash-Join

- Aplicável para equijoins e natural joins
- Um função hash h é usada para distribuir os registros dos dois arquivos
- h mapeia os valores do atributo da junção para $\{0, 1, \dots, n\}$



129

Hash-Join

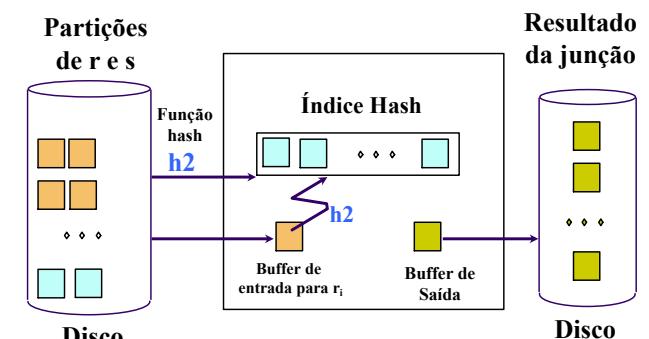
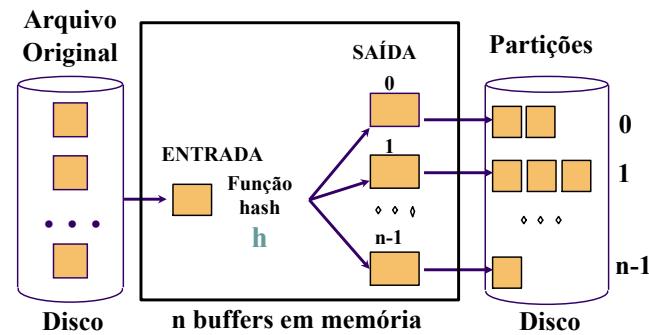
- r_0, r_1, \dots, r_n partições dos registros de r
 - Cada registro $t_r \in r$ é colocado na partição r_i , $i = h(\dots)$
- s_0, s_1, \dots, s_n partições dos registros de s
 - Cada registro $t_s \in s$ é colocado na partição s_i , $i = h(\dots)$
- Os registros de r em r_i só precisam ser comparados com os registros de s em s_i

130

Algoritmo Hash-Join

- Particionar s usando a função h (gera s_1, \dots, s_n)
- Particionar r usando a função h (gera r_1, \dots, r_n)
- Para $i = 1$ até n
 - Carrega partição s_i na memória e constrói um índice de hash em memória usando o atributo de junção.
 - Esse índice hash usa uma função diferente da função h usada no particionamento.
 - Carrega os registros de r_i um a um do disco. Para cada registro t_r , localiza o registro t_s de s_i correspondente usando o índice hash em memória. Retorna a concatenação dos campos

131



Algoritmo Hash-Join

- O valor de n e a função h devem ser escolhidas de tal forma que cada s_i caiba na memória
- Tipicamente, n é escolhido como $\lceil b_s/M \rceil * f$
 - b_s : número de blocos de s
 - M : blocos de memória disponíveis
 - f : fator de folga (ex., 1,2)



Particionamento Recursivo

- Necessário se $n > M$ páginas de memória
 - Ao invés de n partições, usar $M-1$ partições de s
 - Reparticiona estas partições usando outra função
 - O mesmo para r
- Raramente é necessário para tamanhos de memória em servidores típicos



133

Junções Complexas

- Condição conjuntiva tipo $r \bowtie_{\theta_1 \wedge \theta_2 \wedge \dots \wedge \theta_n} s$
 - Opção 1: NLJ/BNLJ
 - Opção 2: Computar junções simples $r \bowtie_{\theta_i} s$ e aplicar as outras condições sobre o resultado
- Condição disjuntiva tipo $r \bowtie_{\theta_1 \vee \theta_2 \vee \dots \vee \theta_n} s$
 - Opção 1: NLJ/BNLJ
 - Opção 2: União de joins individuais:
 - $(r \bowtie_{\theta_1} s) \cup (r \bowtie_{\theta_2} s) \cup \dots \cup (r \bowtie_{\theta_n} s)$



Hash-Join – Custo

- $4(b_r + b_s) + 4 * n_h$ blocos
 - Particionamento: $b_r + b_s$ leituras + $b_r + b_s$ escritas
 - Comparação: $b_r + b_s$ leituras
 - Se resultado for para o disco: $b_r + b_s$ escritas
 - n_h : partições c/ blocos parcialmente preenchidos
 - 1 para cada arquivo para cada fase: $4 * n_h$
- $2(\lceil b_r/B \rceil + \lceil b_s/B \rceil)$ seeks
- Assumindo que não será necessário particionamento recursivo.



134

Outras Operações: Remoção de Duplicatas



- Implementada com hashing ou ordenação
- Ordenação:
 - duplicatas se tornaram contíguas. Remover todas cópias, exceto uma.
 - Otimização: remover durante os passos de merge no merge-sort.
- Hashing: similar – duplicatas ficam no mesmo bucket.

Outras Operações: Agregação



- Similar à eliminação de duplicatas
- Ordenação ou Hashing podem ser usadas para reunir os registros de um mesmo grupo
- Em seguida, a função de agregação é aplicada em cada grupo
- Otimização: fazer as combinações durante os merges do merge-sort

137

Outras Operações: Conjuntos



- \cup , \cap e —
- Implementadas usando algoritmos similares ao merge-join ou hash-join.

139

138

Avaliação de Árvores



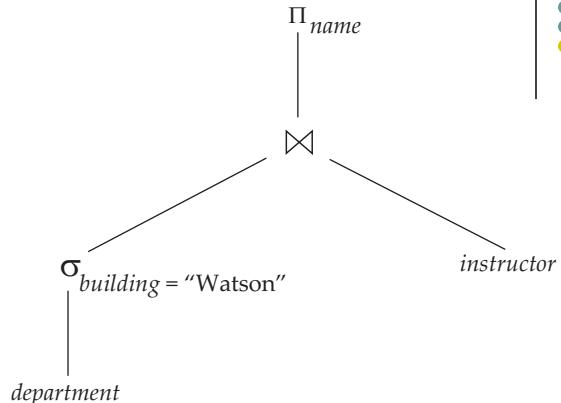
- Materialização:
 - Resultados das operações são armazenados em disco com se fosse relações reais
- Pipelining:
 - Registros gerados para a operação "pai" na árvore assim que eles são gerados.

140

Materialização

- Avalia um operador de cada vez, começando dos níveis mais baixos.
- Usa os resultados intermediários materializados em relações temporárias para avaliar as operações dos níveis acima.

Materialização Exemplo



- Computa e armazena $\sigma_{building = "Watson"}(department)$
- Computa e armazena a junção
- Computa a projeção

141

Materialização

- É sempre aplicável
- O custo de escrita e leitura das relações temporárias pode ser muito alto
- Custos das operações individuais aumentam em uma escrita
- Double buffering:
 - Usar dois buffers de saída para cada operação
 - Quando um está cheio, faz a sua escrita no disco e passa a escrever no outro buffer.
 - Permite a sobreposição de escritas com a computação e reduz o tempo de execução

Pipelining

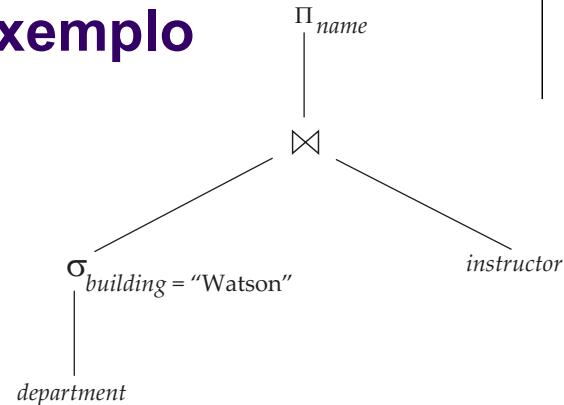
- Várias operações são avaliadas simultaneamente
- Resultados de uma operação são passados para a seguinte assim que possível
- Menos custo: não é necessário armazenar relações temporárias
- Nem sempre é possível.
 - Ex: hash-join, sort

143

142

144

Pipeline - Exemplo



- Não armazena $\sigma_{building = "Watson"}(department)$
 - Ao invés disso, envia cada tupla para a junção

Pipelining sob Demanda

• "Lazy Evaluation" (Pull)

- O sistema repetidamente requisita a próxima tupla da operação de nível mais alto
- Cada operação requisita a próxima tupla das operações "filhas" para poder produzir a sua próxima tupla
- Entre chamadas, cada operação deve manter controle do seu estado, de forma a saber o que retornar na próxima chamada.

145

Pipelining Orientada à Produção

- "Eager" (Push)
 - Os operadores geração suas tuplas continuamente e passam para os operadores pai.
 - Buffers são mantidos entre os operadores. Filhos escrevem e pais lêem as tuplas do buffer
 - Se o buffer enche, o filho esperar o pai esvaziá-lo
 - O sistema pode escalar primeiro operações que tem já têm espaço no buffer e podem processar novas tuplas

146

147