

Escrita (“W”) do Resultado

- Qualquer alternativa de processamento deve considerar este custo

- b_{res} = número de blocos de resultado a ser “W”

- Exemplo: estimativa de “W” do resultado de um produto

- $b_{res} = \lceil \text{tamanhoProduto} / f_{res} \rceil$

- estimativa do fator de bloco do resultado (f_{res})

- $f_{res} = \lfloor \text{tamanhoBloco} / (t_R + t_S) \rfloor$

arredonda “para baixo” pois uma tupla do resultado não pode estar parcialmente escrita em um bloco

Exemplo

Med(codm, nome, ...) com $n_{Med} = 50$ e $t_{Med} = 50$ b

Cons(codm, codp, ...) com $n_{Cons} = 500$ e

$t_{Cons} = 20$ b e 1 bloco = 2 kb

Dado $Med \times \theta = \sigma_{Med.codm = Cons.codm} Cons$, temos:

- junção por referência ($fk(Cons) = pk(Med)$)

- tamanho resultado = $n_{Cons} = 500$ tuplas

– $f_{res} = \lfloor \text{tamanhoBloco} / (t_R + t_S) \rfloor$

– $f_{res} = \lfloor 2048 / (50 + 20) \rfloor = 29$ tuplas

– $b_{res} = \lceil \text{tamanhoResultado} / f_{res} \rceil$

- $b_{res} = \lceil 500 / 29 \rceil = 18$ blocos

Processamento de Projeções (π)

- Custo (na teoria) de $\pi_{a_1, a_2, \dots, a_n}(R)$
 - custo = (1) varredura de R + (2) eliminação de duplicatas
 - custo de (1) = b_R (gera b_{Res} blocos de resultado)
 - custo de (2) = custo de classificar o resultado pelos atributos da projeção = $2 * b_{Res} (\log n_{buf} (b_{Res} / n_{buf}) + 1)$
 - tuplas iguais estarão adjacentes e apenas uma delas é mantida (deve-se ainda varrer o resultado ordenado)
 - **custo = $b_R + 2 * b_{Res} (\log n_{buf} (b_{Res} / n_{buf}) + 1) + b_{Res}$**
- Custo (na prática) de $\pi_{a_1, a_2, \dots, a_n}(R)$
 - custo = b_R
 - SQL não faz eliminação automática de duplicatas

Processamento de Projeções (π)

- Tamanho de $\pi_{a_1, a_2, \dots, a_n}(R)$ (na prática)
 - tamanho = $n_R * (t_R(a_1) + \dots + t_R(a_n))$
- É difícil determinar exatamente e de forma genérica o tamanho do resultado pois é difícil estimar quantas duplicatas serão eliminadas
 - o que é possível determinar c/ exatidão?
 - se a projeção é apenas da chave primária ($pk(R)$)
 - tamanho = $n_R * t_R(pk(R))$
 - se a projeção é de um único atributo a_i
 - tamanho = $V_R(a_i) * t_R(a_i)$

Processamento de Operações de Conjunto (\cup , $-$ e \cap)

- Aplica-se uma estratégia *merge-junção*
 - (1) classificação de R e S
 - facilita a verificação de tuplas iguais em R e S
 - (2) varredura de R e S para obtenção do resultado
 - custo (pior caso) = $2 * b_R (\log n_{buf} (b_R / n_{buf}) + 1) + 2 * b_S (\log n_{buf} (b_S / n_{buf}) + 1) + b_R + b_S$

Processamento de Operações de Conjunto (\cup , $-$ e \cap)

- Estimativas de tamanho

- pior caso

- tamanho $(R \cup S) = n_R + n_S$
 - tamanho $(R - S) = n_R$
 - tamanho $(R \cap S) = \text{MIN}(n_R, n_S)$ /* R contida em S ou vice-versa */

- melhor caso

- tamanho $(R \cup S) = \text{MAX}(n_R, n_S)$ /* R contida em S ou vice-versa */
 - tamanho $(R - S) = 0$
 - tamanho $(R \cap S) = 0$

- caso médio (considerar este caso na prática!)

- média aritmética do melhor e pior casos

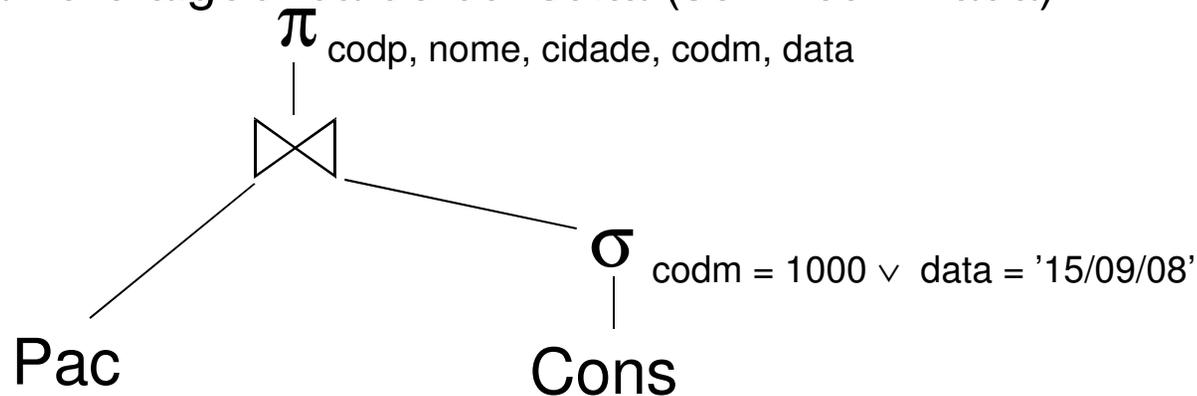
Funções de Agregação e *Group By*

- Função de agregação (*count, max, sum, ...*)
 - custo da varredura da relação $R = b_R$
 - tamanho = *length (int ou float)*
- *Group By* + Função de Agregação
 - processamento: ordenação de R pelos atributos de agrupamento + varredura de R ordenada para definir grupos e aplicar função
 - custo = $2 * b_R (\log n_{buf} (b_R / n_{buf}) + 1) + b_R$
 - tamanho de *group by* a_1, \dots, a_n + função
 - número de grupos * ($t_R(a_1) + \dots + t_R(a_n)$
+ *length (int ou float)*)

Exercício 6

Dado **Pac**(codp, nome, idade, cidade, doença) e **Cons**(codm, codp, data, hora) e as seguintes estimativas: $n_{\text{Pac}} = 500$ tuplas; $t_{\text{Pac}} = 50$ bytes; $t_{\text{Pac}}(\text{codp}) = 5$ bytes; $t_{\text{Pac}}(\text{nome}) = 15$ bytes; $t_{\text{Pac}}(\text{cidade}) = 15$ bytes; $n_{\text{Cons}} = 1000$ tuplas; $t_{\text{Cons}} = 20$ bytes; $t_{\text{Cons}}(\text{codm}) = 5$ bytes; $t_{\text{Cons}}(\text{data}) = 10$ bytes; $V_{\text{Cons}}(\text{data}) = 50$; $V_{\text{Cons}}(\text{codp}) = 500$; $V_{\text{Cons}}(\text{codm}) = 200$; um índice primário árvore-B para *codp* (I1) em **Pac** com $N = 10$ e $f_{I1} = 10$; um índice secundário hash para *codp* (I2) em **Cons**; um índice secundário *hash* para *codm* (I3) em **Cons**; **Pac** está ordenada pelo *codp*; **Cons** está ordenada pela *data*; 1 bloco = 1 kb e $n_{\text{buf}} = 3$

Dada a seguinte árvore algébrica de consulta (semi-otimizada):



Apresente o custo total de acessos (c/ base nos menores custos) e o tamanho do resultado desta consulta.

Índice Temporário

- Um índice temporário pode ser criado para o processamento de uma operação algébrica op_x
- Objetivo
 - gerar um custo menor que outras alternativas de processamento de op_x
 - este custo envolve
 - “W” total ou parcial dos blocos do índice no disco
 - acesso a ele durante o processamento de op_x
 - estes custos devem ser estimados antes da criação do índice, para decidir por criá-lo ou não

Índice Temporário - Motivação

- Processamento da junção

- A1: laço aninhado

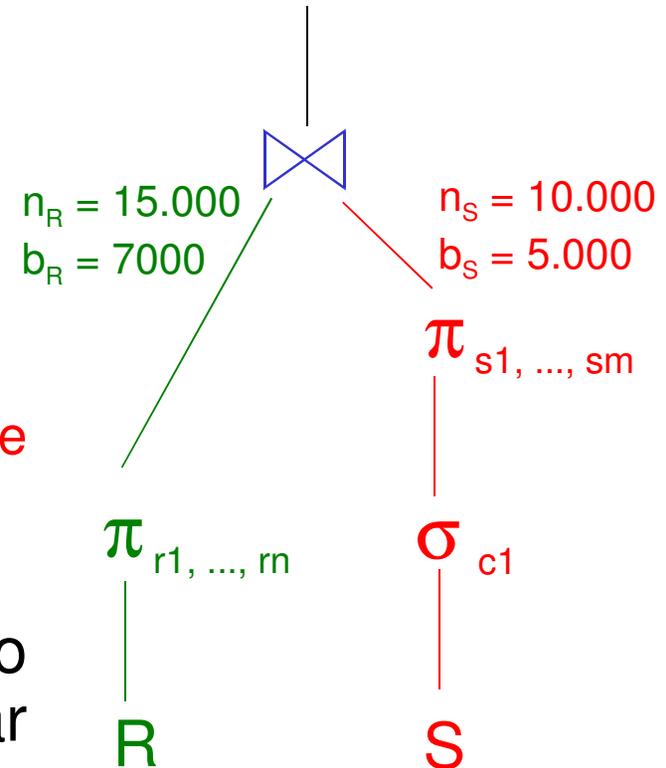
- custo = $b_S + b_S * b_R$
= $5 + 5 * 7 = 40$ mil acessos

- A3: merge-junção ($n_{buf} = 3$)

- custo = ordenação de R + ordenação de S + $b_R + b_S$
= $126 + 80 + 7 + 5 = 218$ mil acessos

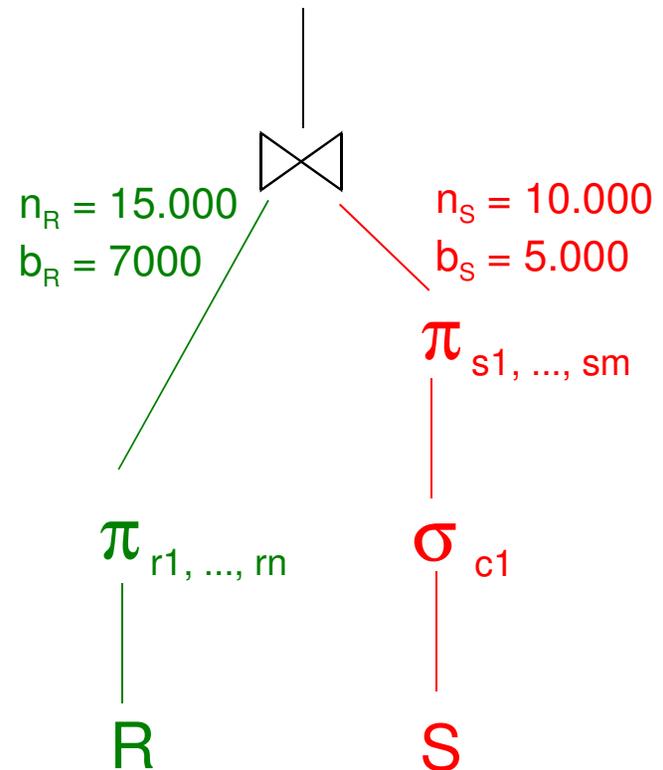
- e se houvesse um índice I_x sobre o resultado de R? Poderíamos estimar A2: laço aninhado indexado

- custo = $b_S + n_s * (h_{I_x} + 1)$ (supondo que o atributo de junção em R é chave)
 - se I_x tiver $h_{I_x} < 3$, A2 será a alternativa de menor custo! Exemplo: $h_{I_x} = 2$:
 - custo = $5 + 10 * (2+1) = 35$ mil acessos



Índice Temporário - Exemplo

- Avaliando custo de criação de índice árvore-B sobre o resultado de R
 - atributo de junção em R é chave: deve-se indexar 15.000 dados
 - supondo $N = 50$ valores, temos:
 - nível 0 \Rightarrow indexa 50 valores
 - nível 1 \Rightarrow indexa $51 \times 50 = 2.550$ valores
 - nível 2 \Rightarrow indexa $51 \times 51 \times 50 = 130.050$ valores (**3 níveis na árvore-B é suficiente!**)
 - supondo que se consegue um $f_i = 55$ nodos
 - se $f_i = 55$, o primeiro nível (1 nodo) e o segundo nível (51 nodos) da árvore podem ficar em um bloco e os restantes em outros blocos. Logo, teremos no máximo 2 acessos (**$h_i = 2$!** **Vale a pena criar o índice!**)
 - custo total de $A2 = 35$ mil + “W” do índice
 - custo “W” do índice = 15.000 valores / $N = 300$ nodos / $f_i =$ “W” **de 6 blocos de índice** (pior caso: o índice não cabe na memória)



Materialização X Pipeline

- Materialização

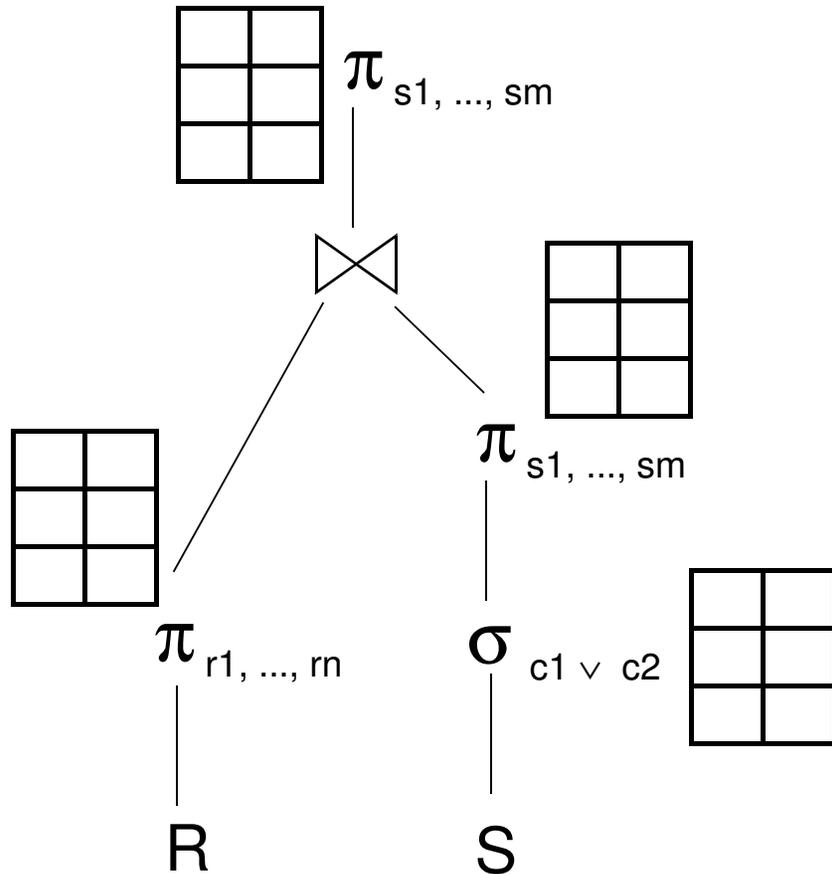
- cada operação da álgebra é materializada em uma relação temporária (se necessário) e utilizada como entrada para a próxima operação
- situação *default* no processamento de consultas

- Pipeline

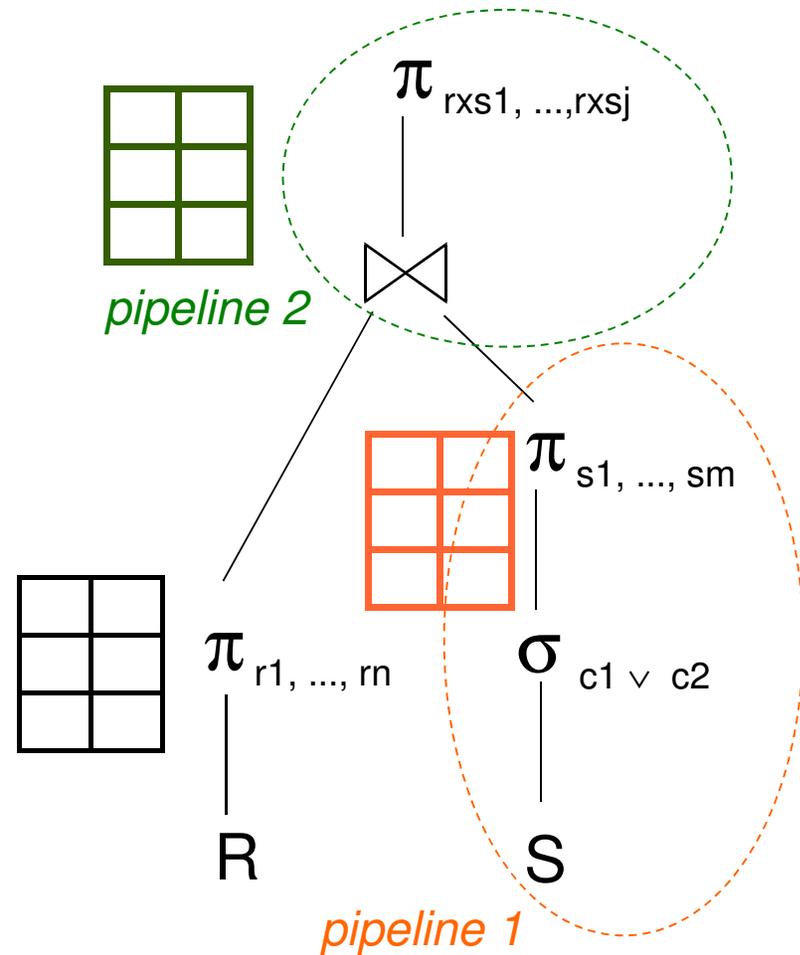
- uma seqüência de operações algébricas é executada em um único passo
 - cada tupla gerada por uma operação é passada para a operação seguinte
 - cada tupla passa por um canal (*pipe*) de operações
 - somente o resultado ao final do *pipeline* é materializado (se necessário)

Materialização X Pipeline

Materialização



Definição de Pipelines



Pipeline de Operações

- Bom: evita a materialização de todos os resultados intermediários no processamento de uma consulta
- Ruim: resultado não é passado de forma completa para uma próxima operação dentro do *pipeline*
 - algoritmos de processamento das operações algébricas devem ser modificados para invocar outras operações para cada tupla gerada
 - algoritmos “dinâmicos”
 - algumas alternativas não podem ser estimadas
 - exemplos: merge-junção; operações de conjunto
 - exigem um resultado completo e ordenado para processar

Exemplo: um Produto sem Pipeline

1) custo σ (pior caso) = 4 acessos

("W" resultado = 2)

2) custo π = 2 acessos

custo "W" resultado = 2 acessos

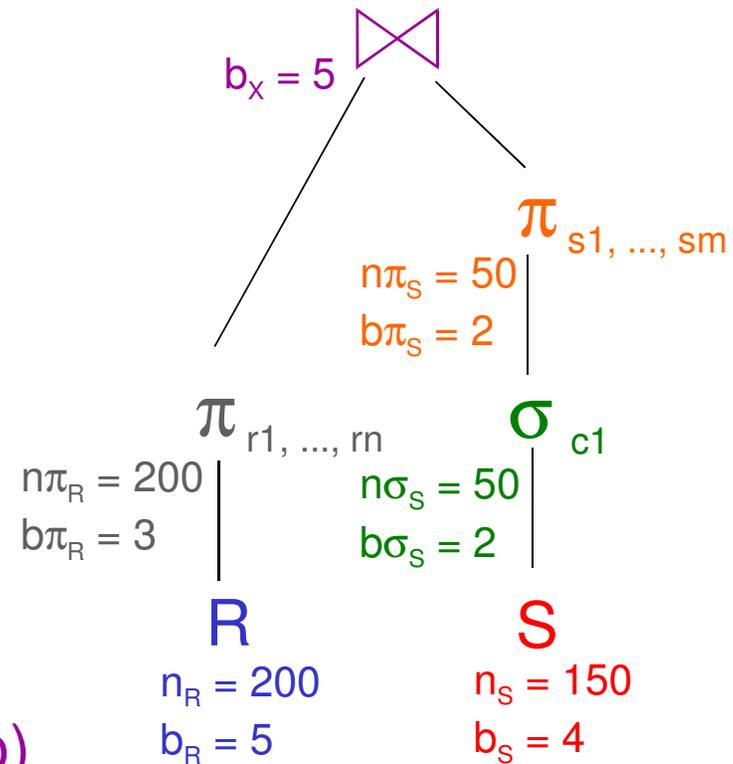
3) custo π = 5 acessos

custo "W" resultado = 3 acessos

4) custo \bowtie (pior caso: laço aninhado)

= $b\pi_s + b\pi_s * b\pi_R = 2 + 2*3 = 8$ acessos

custo "W" resultado = $b_x = 5$ acessos



CUSTO TOTAL = $6+4+8+13 = \underline{31}$ acessos

Produto dentro de um *Pipeline*

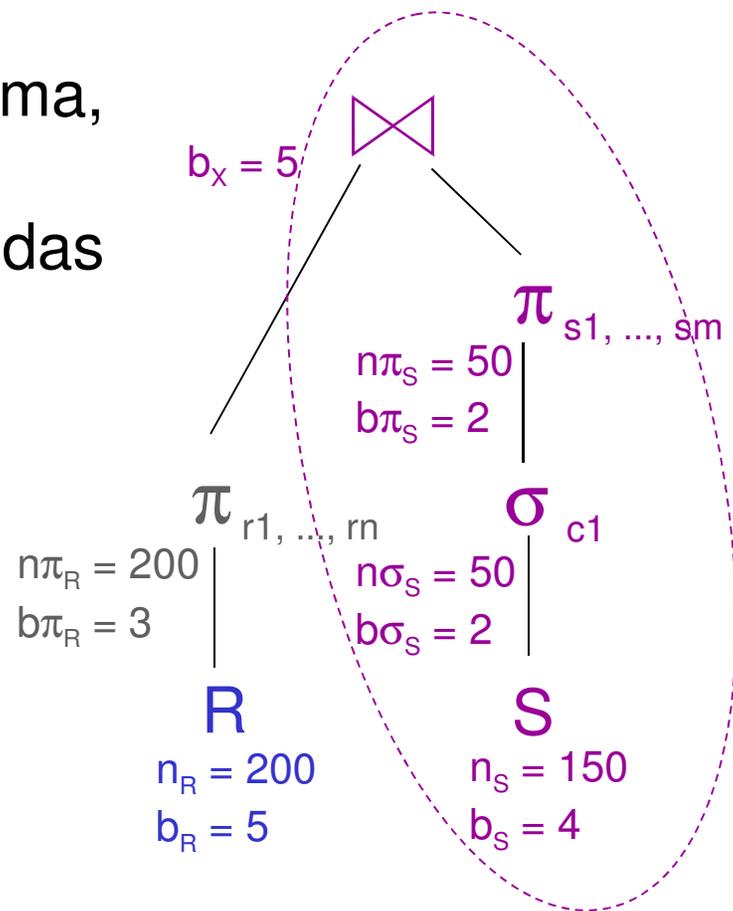
- o produto vai recebendo, uma a uma, as tuplas filtradas de S
- as tuplas de S não são recebidas ordenadas pelo atributo de junção
 - não dá para usar *merge-junção*

- custo \bowtie (pior caso: laço aninhado):
 - $= b_S + n\pi_S * b\pi_R$
 - $= 4 + 50 * 3$
 - $= 154 + 5 \text{ "W"} = 159 \text{ acessos}$

(escrita do resultado da junção (b_x))

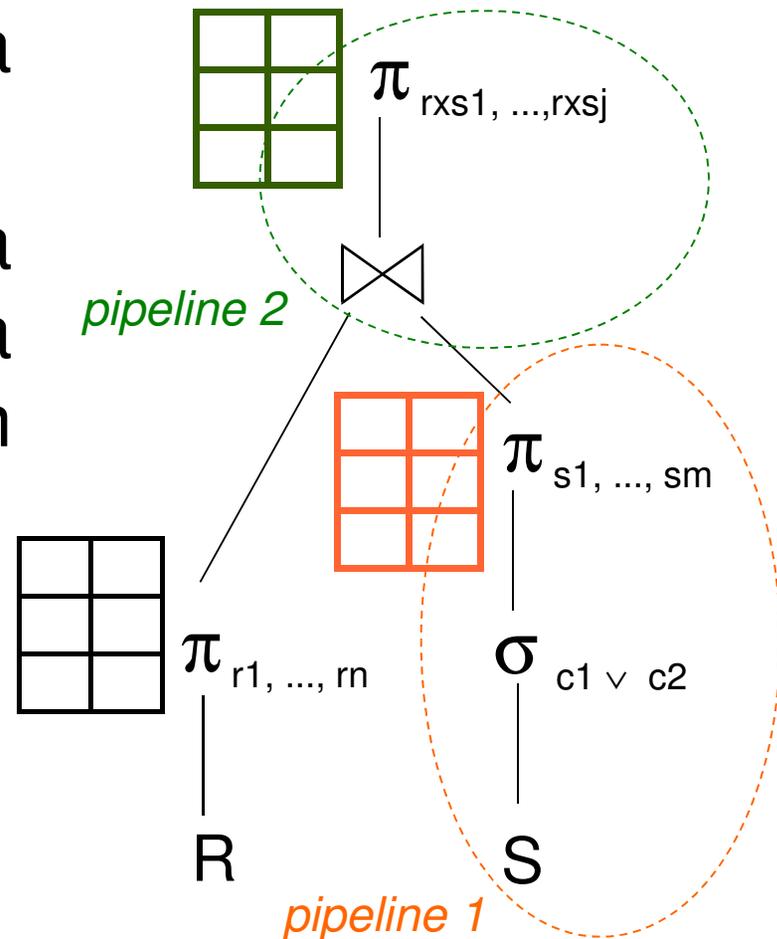
- custo $\pi = 5$ acessos
- custo "W" resultado = 3 acessos

CUSTO TOTAL = $159 + 8 = \underline{167}$ acessos
 (nem todo pipeline é eficiente!)



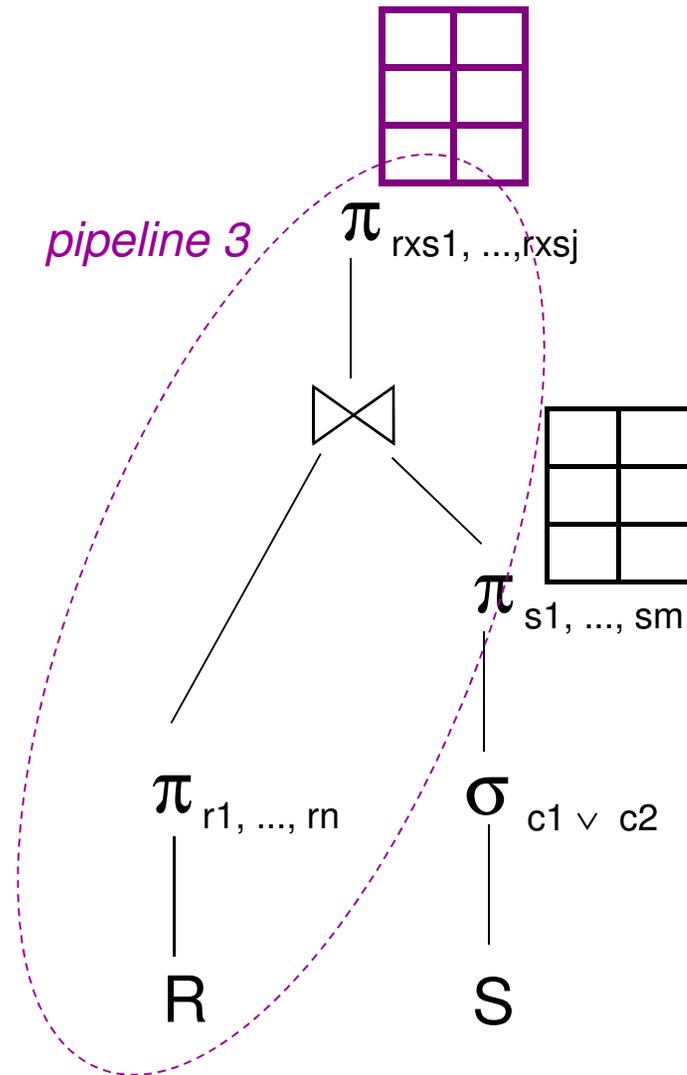
Uso mais Comum de *Pipelines*

- Em uma seqüência de operações que
 - **inicia** em um nodo folha ou uma operação binária
 - **termina** ou no resultado da consulta ou em uma operação binária ob_x , sem incluir ob_x



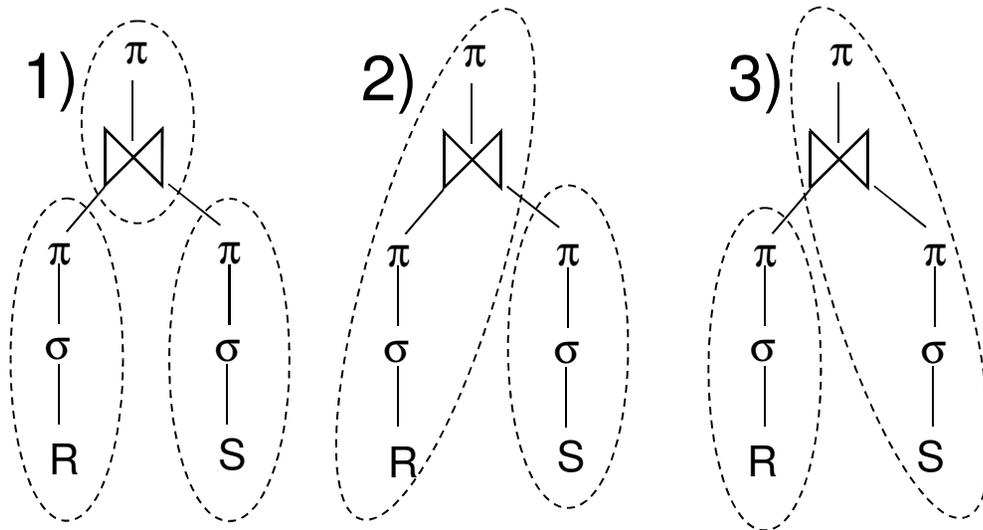
Uso mais Comum de *Pipelines*

- Em uma seqüência composta apenas por operações π e operações produtórias, a partir de um nodo folha ou uma operação binária ob_x , incluindo ob_x
 - considera que o tamanho dos resultados intermediários das operações π é muito grande para ser materializado
 - mesmo assim, avaliar se o custo das operações produtórias não aumenta com o *pipeline*...



Exercício 7

a) qual alternativa de *pipeline* para a árvore ao lado possui o menor custo de pior caso?



b) se $b\pi_R = 1$, a resposta do item anterior é diferente?

