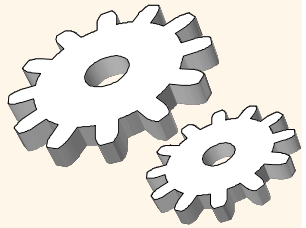
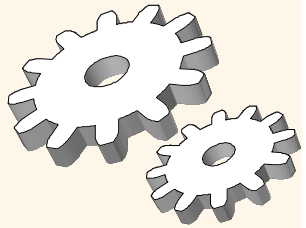


Avaliação das Operações Relacionais

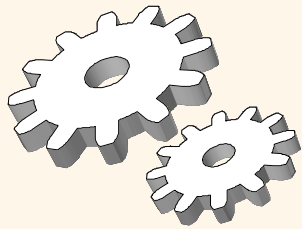
Capítulo 14, Parte A (Junções)

Operações Relacionais



- Vamos considerar como implementar:
  - *Seleção* (σ) Seleciona um subconjunto de linhas da relação.
  - *Projeção* (π) Apaga colunas indesejáveis da relação.
  - *Junção* ($\bowtie$) Permite combinar duas relações.
  - *Diferença entre conjuntos* ($-$) Tuplas na relação 1, mas não na relação 2.
  - *União* ($\cup$) Tuplas na relação 1 ou na relação 2.
  - *Agregação* (SUM, MIN etc.) e GROUP BY
- Como cada operação retorna uma relação, elas podem ser *compostas*! Após abrangermos as operações, discutiremos como *otimizar* consultas formadas pela composição das operações.

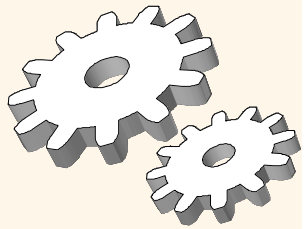
Esquema para Exemplos



Sailors (*sid*: integer, *sname*: string, *rating*: integer, *age*: real)
Reserves (*sid*: integer, *bid*: integer, *day*: dates, *rname*: string)

- Similar ao esquema anterior; *rname* foi acrescentada para variações.
- Reserves:
  - Cada tupla tem 40 bytes de tamanho, 100 tuplas por página, 1000 páginas.
- Sailors:
  - Cada tupla tem 50 bytes de tamanho, 80 tuplas por página, 500 páginas.

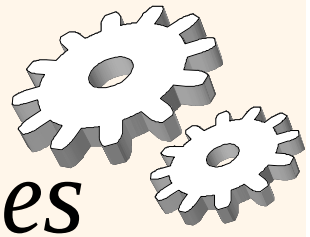
Junções de Igualdade com uma Coluna de Junção



```
SELECT  *
FROM    Reserves R1, Sailors S1
WHERE   R1.sid=S1.sid
```

- Na álgebra: R ⋈ S. Comum! Deve ser otimizada cuidadosamente. R × S é grande; então, R × S seguido por uma seleção não é eficiente.
- Assuma: M páginas em R, p_R tuplas por página, N páginas em S, p_S tuplas por página.
  - Em nossos exemplos, R é Reserves e S é Sailors.
- Consideraremos condições de junções mais complexas posteriormente.
- *Métrica de Custo*: nº de E/Ss. Iremos ignorar custos de saída.

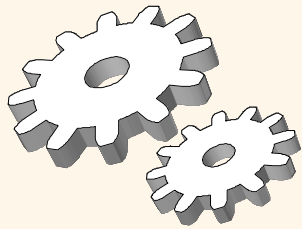
Junção por Laço Aninhado Simples



foreach tuple r in R do
  foreach tuple s in S do
    if r_i == s_j then add <r, s> to result

- Para cada tupla na relação R *mais externa*, nós varremos inteiramente a relação S *mais interna*.
  - Custo: M + p_R * M * N = 1000 + 100*1000*500 E/Ss.
- Junção por Laço Aninhado Orientada a Página: Para cada *página* de R, obtenha cada *página* de S e escreva o par casado de tuplas <r, s>, onde r está na página-R e s está na página-S.
  - Custo: M + M * N = 1000 + 1000*500 E/Ss
  - Se a menor relação (S) é a mais externa, custo = 500 + 500*1000

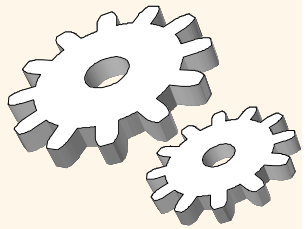
Junção por Laço Aninhado com Índice



foreach tuple r in R do
    foreach tuple s in S where r_i == s_j do
        add <r, s> to result

- Se houver um índice na coluna de junção de uma das relações (por exemplo S), pode-se torná-la a mais interna e explorar o índice.
  - Custo: M + ((M*p_R) * custo de encontrar as tuplas casadas de S)
- Para cada tupla de R, o custo de investigar o índice de S é aproximadamente 1,2 para índice hash, 2-4 para árvore B+. Então, o custo de encontrar tuplas de S (assumindo Alternativa (2) ou (3) para entradas de dados) depende do agrupamento.
  - Índice agrupado: 1 E/S (típica), não agrupado: até 1 E/S por tupla casada de S.

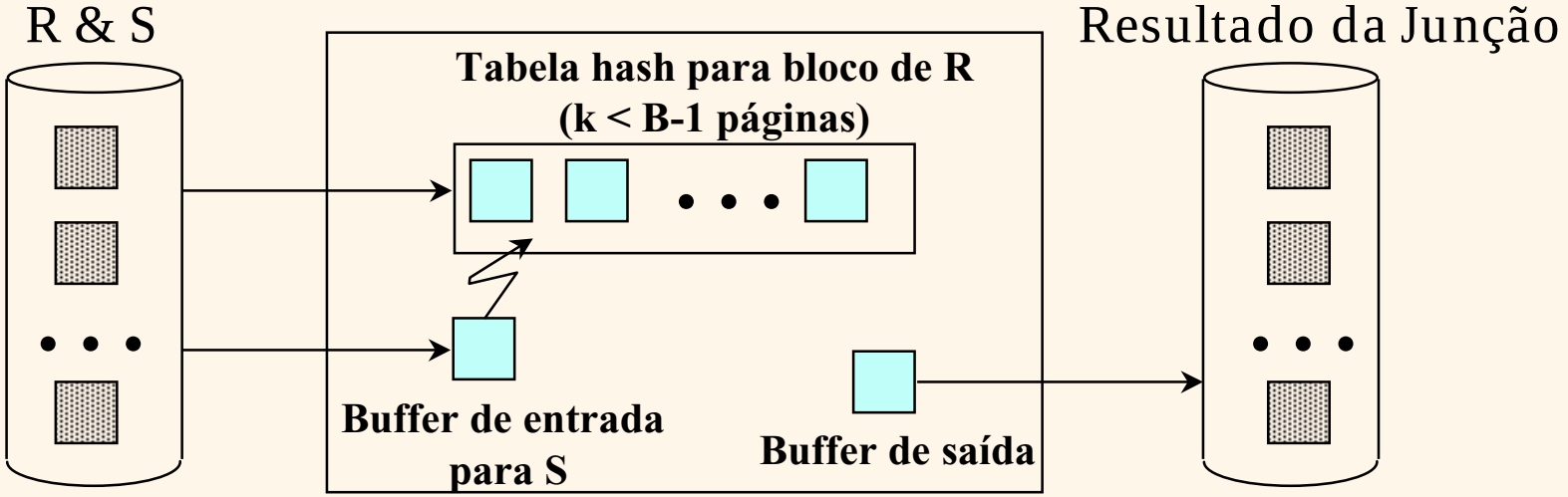
Exemplos de Laços Aninhados com Índice



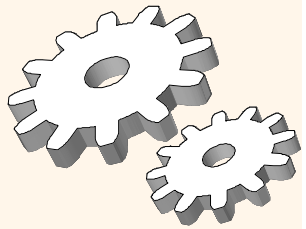
- Índice-Hash (Alt. 2) sobre *sid* de Sailors (como mais interna):
  - Varrer Reserves: 1000 páginas de E/Ss, 100*1000 tuplas.
  - Para cada tupla de Reserves: 1,2 E/Ss para obter a entrada de dados no índice mais 1 E/S para obter (exatamente uma) a tupla casada de Sailors. Total: 220.000 E/Ss.
- Índice-Hash (Alt. 2) sobre *sid* de Reserves (como mais interna):
  - Varrer Sailors: 500 páginas de E/Ss, 80*500 tuplas.
  - Para cada tupla de Sailors: 1,2 E/Ss para encontrar a página de índice para as entradas de dados mais custo de recuperar as tuplas casadas de Reserves. Assumindo uma distribuição uniforme, 2,5 reservas por marinheiro (100.000 / 40.000). Custo de recuperá-los 1 ou 2,5 E/Ss, dependendo se o índice está agrupado.

Junção por Laço Aninhado em Bloco

- Usar uma página como buffer de entrada para varrer a relação S mais interna, uma página como buffer de saída e usar todas as páginas restantes para manter o ``bloco'' da relação R mais externa.
  - Para cada tupla casada r no bloco-R, s na página-S, acrescente <r, s> ao resultado. Então leia o próximo bloco-R, varra S, etc.

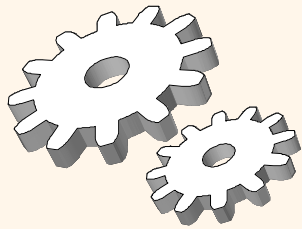


Exemplos de Laços Aninhados em Bloco



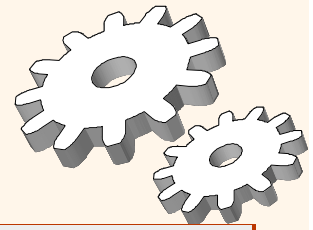
- Custo: Varredura da mais externa + nº de blocos mais externa * varredura da mais interna
  - nº blocos da mais externa = $\lceil \#\,de\,páginas\,da\,mais\,externa / tamanho\,do\,bloco \rceil$
- Reserves (R) como a mais externa e 100 páginas de R:
  - Custo de varrer R é 1000 E / Ss; um total de 10 *blocos*.
  - Por bloco de R, varre-se Sailors (S); 10*500 E / Ss.
  - Se há espaço para apenas 90 páginas de R, pode-se varrer S 12 vezes.
- Com blocos de 100 páginas de Sailors como mais externa:
  - Custo de varrer S é 500 E / Ss; um total de 5 blocos.
  - Por bloco de S, varre-se Reserves; 5*1000 E / Ss.
- Considerando *leituras seqüenciais*, a análise muda: pode ser melhor dividir os buffers igualmente entre R e S.

Junção por Ordenação-Intercalação ($R \underset{i=j}{\bowtie} S$)



- Ordenar R e S sobre a coluna de junção, então varrê-las para fazer uma ``intercalação'' (uma junção das colunas) e a saída das tuplas resultantes.
  - Prossiga a varredura de R até que tupla-R atual >= tupla-S atual, então prossiga a varredura de S até que tupla-S atual >= tupla-R atual; faça até que tupla-R atual = tupla-S atual.
  - Nesse ponto, todas as tuplas de R com mesmo valor em Ri (*grupo atual de R*) e todas as tuplas de S com mesmo valor em Sj (*grupo atual de S*) *casam*; saída <r, s> para todos os pares de tais tuplas.
  - Então prossiga varrendo R e S.
- R é varrida uma vez; cada grupo de S é varrido uma vez por tupla casada de R. (Múltiplas varreduras de um grupo de S provavelmente encontrarão as páginas necessárias no buffer.)

Exemplo de Junção por Ordenação-Intercalação



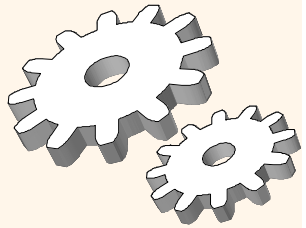
sid	sname	rating	age
22	dustin	7	45.0
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

sid	bid	day	rname
28	103	12/4/96	guppy
28	103	11/3/96	yuppy
31	101	10/10/96	dustin
31	102	10/12/96	lubber
31	101	10/11/96	lubber
58	103	11/12/96	dustin

- Custo: M log M + N log N + (M+N)
  - O custo da varredura, M+N, poderia ser M*N (bem diferente!)
- Com buffers de 35, 100 ou 300 páginas, tanto Reserves quanto Sailors podem ser ordenadas em 2 passadas; custo total da junção: 7500.

(custo de *LAB: 2500 a 15000 E/Ss*)

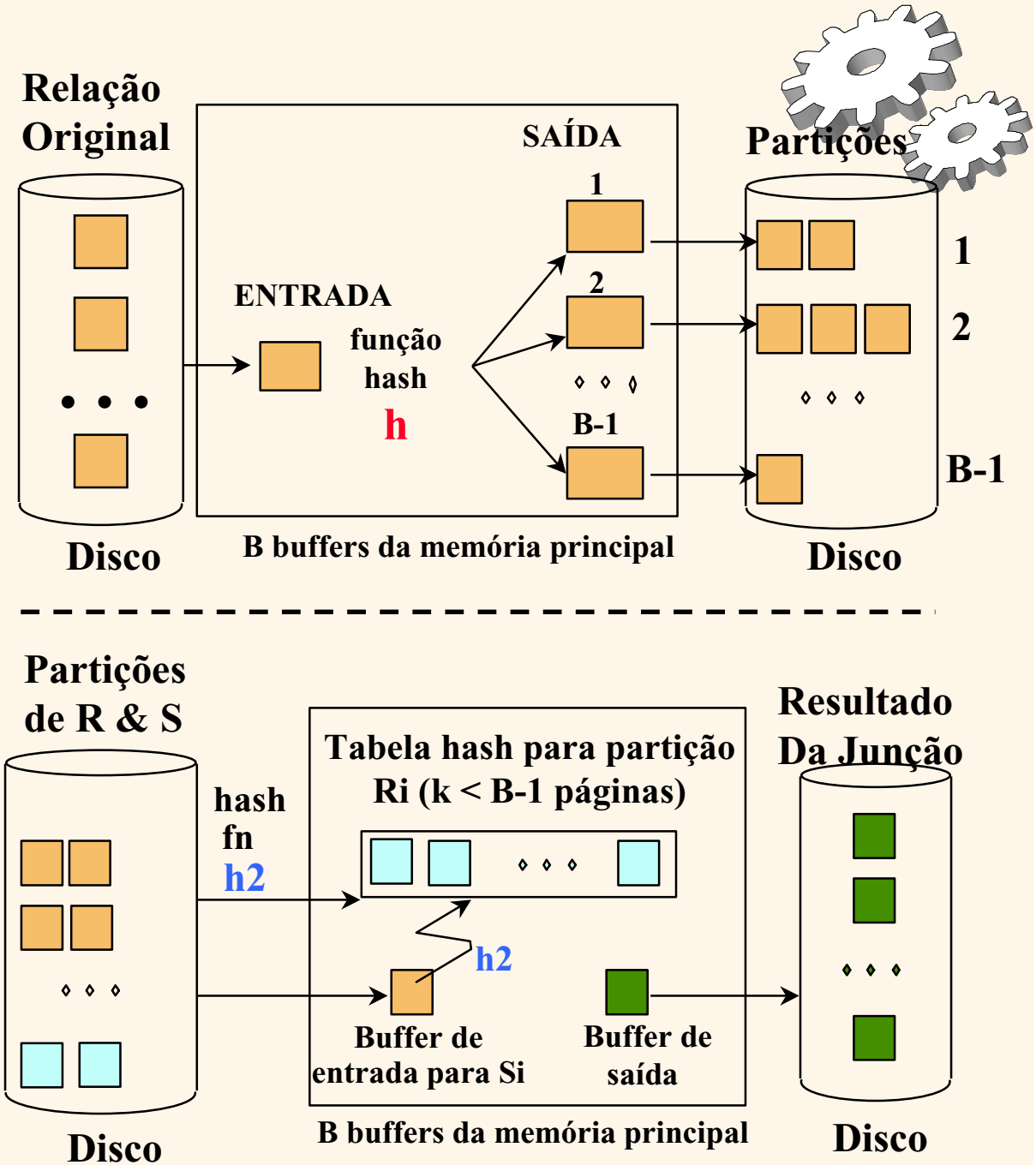
Refinamento da Junção por Ordenação-Intercalação



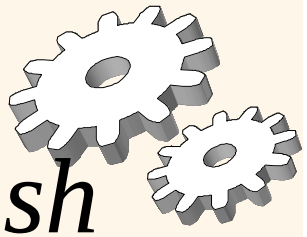
- Pode-se combinar as fases de intercalação da *ordenação* de R e S com a intercalação necessária para a junção.
  - Para $B > \sqrt{L}$, onde L é o tamanho da maior relação, usando o refinamento da ordenação que produz *runs* de tamanho 2B no Passo 0, nº de *runs* de cada relação < B/2.
  - Alocar 1 página por *run* de cada relação e `intercalar' enquanto se verifica a condição de junção.
  - Custo: ler+escrever cada relação no Passo 0 + ler cada relação no (e apenas no) passo de intercalação (+ escrita das tuplas resultantes).
  - No exemplo, custo diminui de 7500 para 4500 E/Ss.
- Na prática, o custo da junção por ordenação-intercalação, assim como o custo da ordenação externa, é *linear*.

Junção por Hash

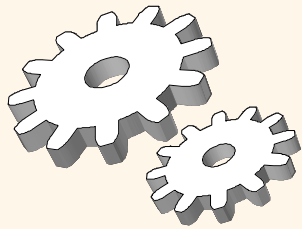
- Dividir ambas as relações usando a função hash **h**: tuplas de R na partição i só casarão com tuplas de S na partição i.
- Ler uma partição de R, aplicando uma função hash **h2 (<> h!)**. Varrer a partição casada de S, procurando combinações.



Observações sobre Junção por Hash



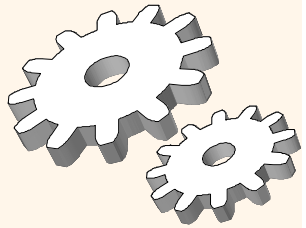
- nº de partições k < B-1 (por quê?) e B-2 > tamanho da maior partição para ser mantida na memória. Assumindo partições de tamanhos uniformes e maximizando k, temos:
  - k= B-1, e M / (B-1) < B-2, isto é, B deve ser > $\sqrt{M}$
- Se construirmos uma tabela hash em memória para acelerar o casamento de tuplas, será necessário um pouco mais de memória.
- Se a função hash não divide uniformemente, uma ou mais partições de R podem não caber na memória. Pode-se aplicar a técnica de junção por hash recursivamente para se fazer a junção desta partição-R com a correspondente partição-S.



Custo da Junção por Hash

- Na fase de divisão, ler+escrever ambas as relações; 2(M+N). Na fase de casamento, ler ambas as relações; M+N E/Ss.
- No nosso exemplo de rodadas, um total de 4500 E/Ss.
- Junção por Ordenação-Intercalação vs Junção por Hash:
  - Dada uma quantia mínima de memória (*o que é isto, pra cada?*) ambas têm o custo de 3(M+N) E/Ss. Junção por Hash é superior nesta contagem se os tamanhos da relação diferem muito. Além disso, Junção por Hash aparenta ser altamente passível de paralelização.
  - A Ordenação-Intercalação é menos sensível a uma distribuição irregular dos dados; o resultado está ordenado.

Condições Gerais para Junções



- Igualdades sobre vários atributos (e.g., *R.sid=S.sid* AND *R.rname=S.sname*):
  - Para Laços Aninhados com Índice, construir o índice sobre *<sid, sname>* (se S é mais interna); ou usar índices existentes sobre *sid* ou *sname*.
  - Para Junção por Ordenação-Intercalação e Junção por Hash, ordenar / dividir sobre a combinação de duas colunas de junção.
- Condições de desigualdade (e.g., *R.rname < S.sname*):
  - Para Laços Aninhados com Índice, é necessário um índice de árvore B+ (agrupado!) .
    - Pesquisas por intervalo na mais interna; nº de combinações provavelmente será muito maior do que para junções de igualdade.
  - Junção por Hash, Junção por Ordenação-Intercalação não aplicáveis.
  - LA em Bloco é bem provável ser o melhor método de junção neste caso.