Implementação de operadores relacionais

R&G - Capítulo 14

Introdução

- Algumas operações são caras!
- Podemos melhorar a performance melhorando a operação
  - Podemos melhorar até 1.000.000x
  - Principais armas são:
    - Boas implementações de operadores
    - Explorar equivalência de operadores
    - Usar estatísticas para estimar o custo de uma operação

Catálogo do Sistema

- Informações armazenadas no catálogo, necessárias no processo de otimização:

- Informações gerais:
  - Tamanho do buffer pool (espaço livre)
  - Tamanho de uma página em disco

- Informações sobre as tabelas

- Informações sobre índices

- Estatísticas sobre tabelas e índices: atualizadas periodicamente

Estatísticas sobre tabelas e índices:

- NTuples (R) = Número de tuplas da tabela R
- NPages(R) = Número de páginas da tabela R
- NKeys(I) = número de chaves distintas do Indice I
- INPages(I) = número de páginas do índice I
- IHeight(I) = Altura do Índice (no caso de B+tree)
- ILow(I) = menor valor de chave do índice I
- IHigh(I) = maior valor de chave do índice I

Esquema

Marinheiros (*sid*: integer, *sname*: char, *rating*: integer, *age*: real)
Reservas (*sid*: integer, *bid*: integer, *day*: dates, *rname*: char)

- Reservas:
  -Tupla com 40 bytes, 100 tuplas por página, 1000 páginas.

- Marinheiros:
  -Tupla 50 bytes, 80 tuplas por página, 500 páginas.

Exemplo

Marinheiros

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

Reservas

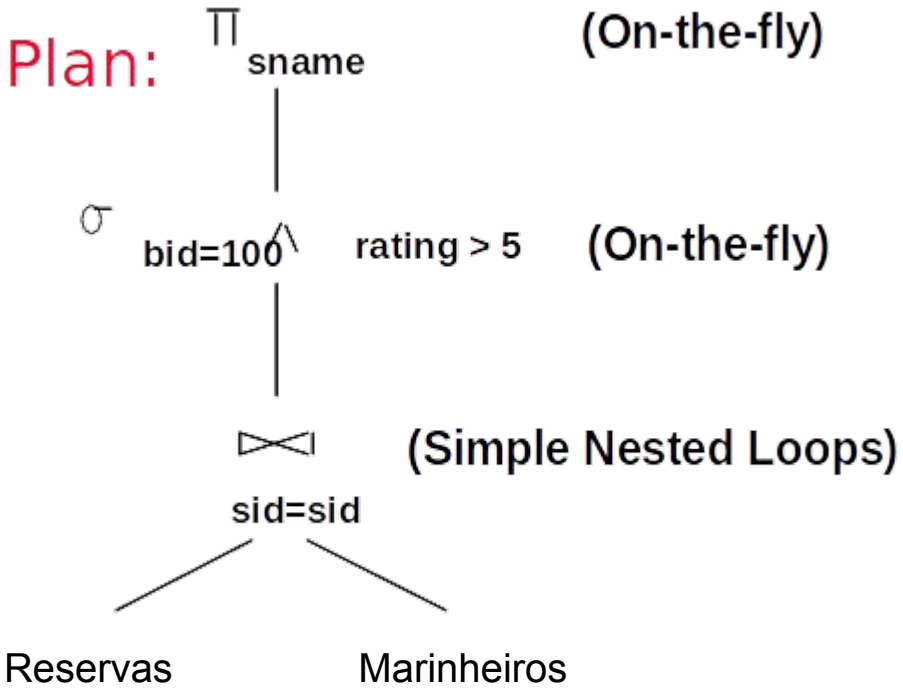
sid	bid	day
28	103	12/4/96
28	103	11/3/96
31	101	10/10/96
31	102	10/12/96
31	101	10/11/96
58	103	11/12/96

Plano de consulta

```
SELECT  S.sname
FROM  Reservas R, Marinheiros S
WHERE  R.sid=S.sid AND
    R.bid=101 AND S.rating>5
```

Plano de consulta

```
SELECT  S.sname
FROM  Reservas R, Marinheiros S
WHERE  R.sid=S.sid AND
    R.bid=100 AND S.rating>5
```



Operadores relacionais

- Vamos considerar os seguintes operadores:
  - Seleção (σ)
  - Projeção (π)
  - *Join* ($\bowtie$)

Joins

- Joins são bastante comuns em BDR
- Joins são bastante custosos
- Muitas abordagens para reduzir o custo

Page (or block)-Oriented Nested Loops Join

foreach page b_R in R do
  foreach page b_m in M do
    foreach tuple r in b_R do
      foreach tuple m in b_m do
        if r_i == m_j then add <r, m> to result

- Para cada page of R, buscar uma página de S e escrever tuplas <r,m>

- Custo: R*S + R

M = 1000*500 + 1000=501.000 I/O

Page (or block)-Oriented Nested Loops Join: exemplo

Memória

Marinheiros

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

Reservas

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

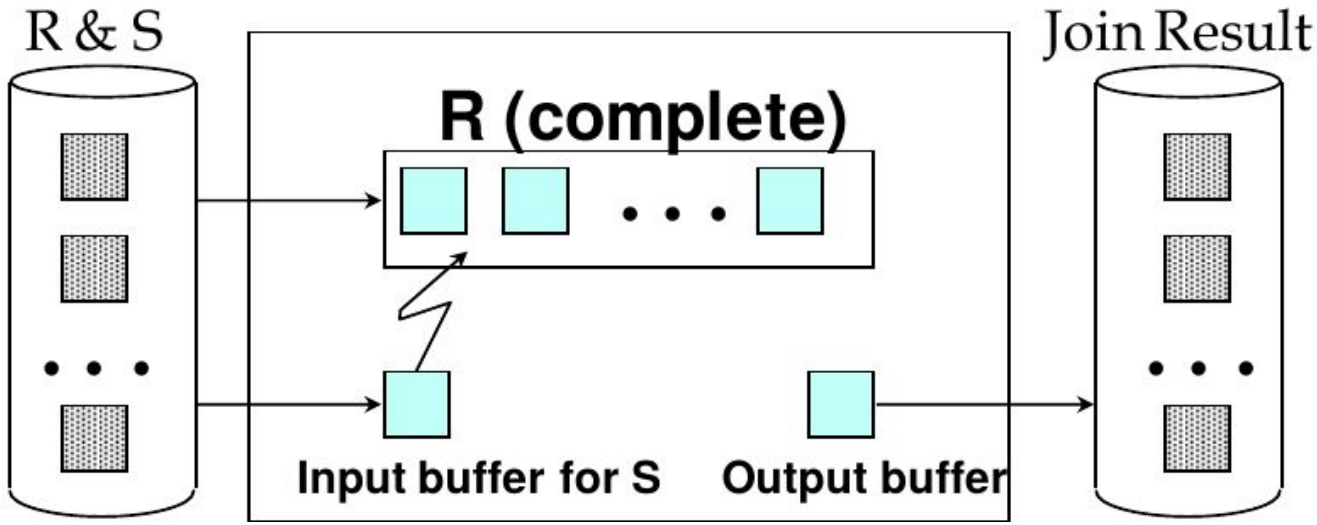
1	6
1	6
1	6
1	6
1	6

Páginas I/O:

***OBS: neste exemplo uma tupla ocupa uma página na memória

Simple nested loops join

Objetivo: Carregar a relação menor inteira na memória



Custo: R+S

Simple nested loops join: exemplo

Marinheiros

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

Reservas

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: 5 +(6)=11

Páginas I/O:

***OBS: neste exemplo uma tupla ocupa uma página na memória

Simple nested loops join

Marinheiros

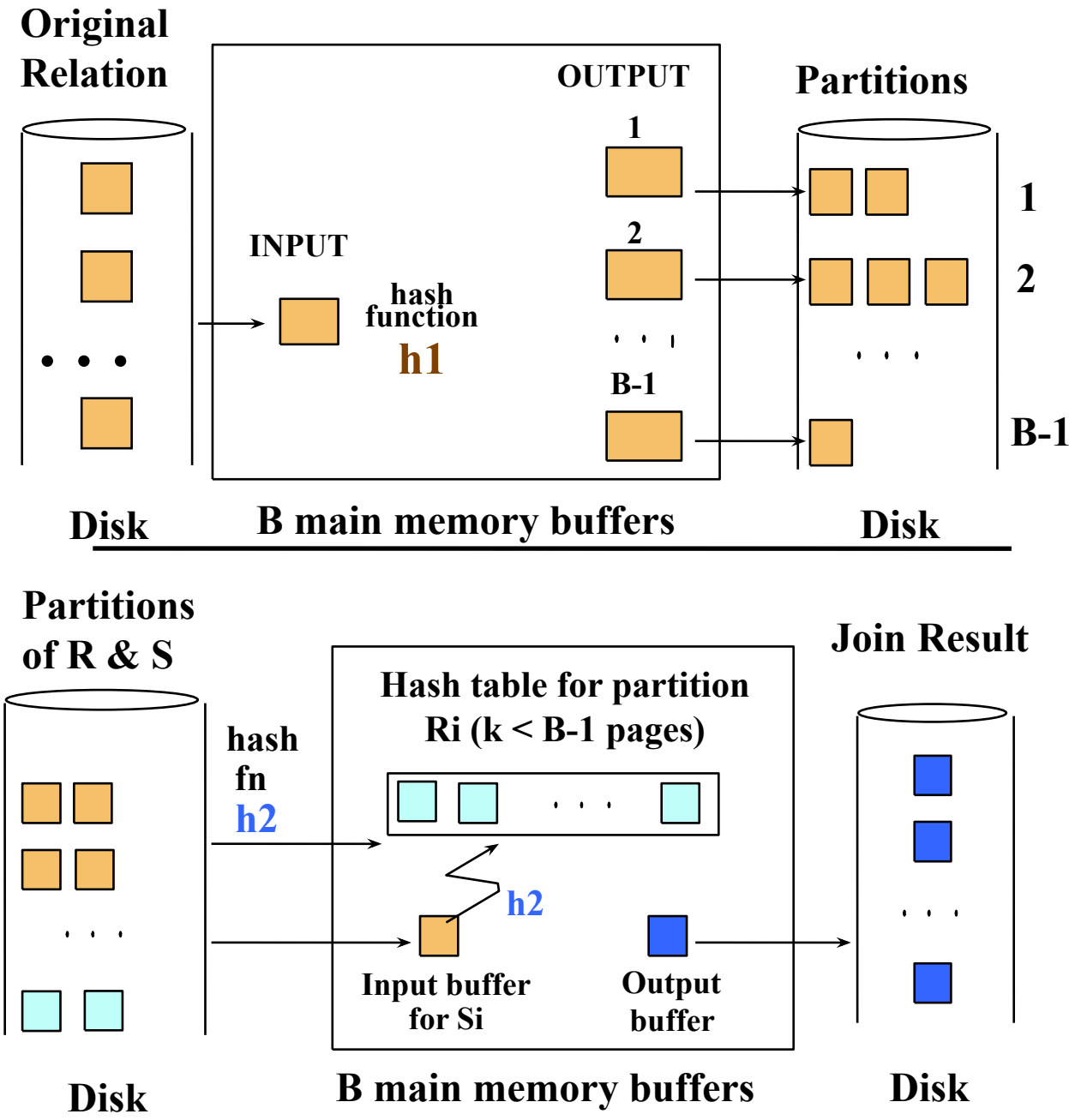
sid	sname	rating	age
22	dustin	7	45.0
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Reservas

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

Qual seria o custo de se aplicar o algoritmo Simple nested loop join nas tabelas acima? Considere que cada registro é armazenado em uma página. Simule a execução do algoritmo. Considere que a memória é capaz de armazenar 6 páginas. Fórmula R+M

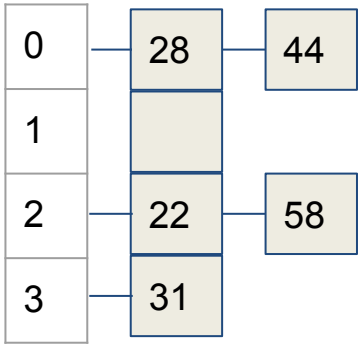
Hash-Join


Original Relation
OUTPUT
Partitions
1
2
B-1
INPUT
hash function
h1
Disk
B main memory buffers
Disk
Partitions of R & S
hash fn
h2
Hash table for partition Ri (k < B-1 pages)
Join Result
h2
Input buffer for Si
Output buffer
Disk
B main memory buffers
Disk

Hash-Join

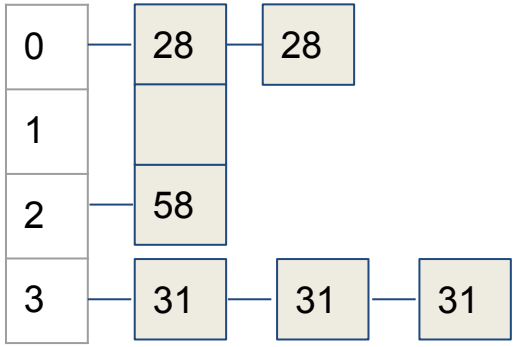
Marinheiros

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



Reservas

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



Hash-Join

Marinheiros

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

Reservas

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 do Hash-Join

Fase de partição: ler e escrever ambas 2(M+N)
Fase de matching: ler ambas (M+N)
Total: 3(M+N)

Index Nested Loops Join

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

- Se existir um índice sobre uma das relações (ex. S):
  Cost: R + ((R*p_R) * custo de encontrar a matching tuple S)

- Para cada tupla de R, o custo de indexar S é de 2-4 IOs para B+

Atividade

Uma relação R (com 150 páginas) consiste de um att a e uma relação S (com 90 pág) com um att a. Determinar o melhor método de join para a query:

select * from R, S where R.a == S.a

Assumir que buffer=10 pag, não existem índices e os seguintes métodos presentes: nested-loop, advanced block nested loop and hash-join. Qual método poderá ser utilizado? Determinar o número de I/Os necessárias em cada método.

-nested-loop: S + (R*S) = 90 + 90*150=13590

-advanced block: carregar uma das relações na memória

R=150 (S=9)*10

R=150*10 +90=1590

- hash - join = (S +R) + (S +R) +(S+R) =3*240 =720