3. Exercise

Timo Bergerbusch 344408
Thomas Näveke 311392
Shu Xing 381176

05.11.2017

Exercise 3.1

1. a)

$\pi_{A,B}(R) \bowtie \pi_{B,C}(\pi_{A,C}(\sigma_{B=1}(R)) \bowtie \pi_{A,B}(R))$

$S_1 := \{[a,b] \mid \exists c\, R(a,b,c)\}$
$S_2 := \{[a,c] \mid \exists a, c\, R(a,1,c)\}$
$\pi_{B=1}(S_2 \bowtie S_1) := \{[b,c] \mid \exists a, c_1, c_2\, R(a,b,c_1) \wedge R(a,1,c_2)\}$
Insgesamt $:= \{[a,b,c] \mid \exists a_1, c_1, c_2(\, R(a,b,c_1) \wedge R(a_1,1,c) \wedge R(a_1,b,c_2)\}$

1.b)

T_1			
a	b	c_1	R
a_1	1	c	R
a_1	b	c_2	R

2)

$h_1 : T_2 \to T_1 : a \to a, b \to b, a_5 \to a_1, b_5 \to b_1, c_4 \to c_1$
$\Rightarrow T_2 \subseteq T_1 \qquad (1)$

$h_2 : T_1 \to T_2 : a \to a, b \to b, b_1 \to b, c_1 \to c_4, a_1 \to a, b_2 \to b_5, c_2 \to c_4, b_3 \to b_5, c_3 \to c_4$
$\Rightarrow T_1 \subseteq T_2 \qquad (2)$

$(1)\&(2) \Rightarrow T_1 \equiv T_2$

Exercise 3.2

Given 16 buffer pages (B) and :

- Album has a size of 10.000 pages (M), 40 bytes record size (s_1) and 100 tuples/page (p_A)
- Track has a size of 200.000 pages (N), 30 bytes record size (s_2) and 80 tuples/page (p_T)

1)

Since the simple nested loop join is a double iteration over both relations the I/O requirements can be calculated as follows:

$$M + p_A \cdot M \cdot N = 10.000 + 100 \cdot 10.000 \cdot 200.000 = 200.000.010.000 \text{ I/Os}$$

2)

Since the block nested loop join uses 1 input and 1 output buffer the number of I/Os can be calculated with the following formula:

$$M + \lceil \frac{M}{B-2} \rceil \cdot N = 10.000 + \lceil \frac{10.000}{16-2} \rceil \cdot 200.000 = 143.060.000$$

3)

Similarities:

- double loop schema

Differences:

- usage of buffer pages
- number of *outer* elements in the memory
- hashing used in the block nested loop

Explanation:
Lets define the outer loop to be over E and the inner loop to be over T.
The *block nested loop join* loads up to $B-2$ pages of E into the memory. These blocks get stored in a hash table. Now every input of a page from T can be used via one of the two reserved pages. After hash probing the input, if there is any match in a block of Es, we can reduce the number of scanned Es per T and therefore reduce I/Os.

Exercise 3.3

1.

Emp: $4.000 \text{ byte}/20 \text{ byte} = 200 \Rightarrow$ 200 records fit on one page
$20.000 \text{ records}/200\frac{\text{records}}{\text{page}} = 100 \Rightarrow$ need 100 pages

DeptProj: 4.000 byte$/40$ byte $= 100 \Rightarrow 100$ records fit on one page
5.000 records$/100\frac{\text{records}}{\text{page}} = 50 \Rightarrow$ need 50 pages

Proj: 4.000 byte$/2.000$ byte $= 2 \Rightarrow 2$ records fit on one page
1.000 records$/2\frac{\text{records}}{\text{page}} = 500 \Rightarrow$ need 500 pages

2.

Worst case: have to access all pages $\overset{1.}{\Rightarrow} 100$ pages
Index access costs 3 $\Rightarrow$ for $N > 97$ scan is cheaper

3.

(a) Cost of a *block nested loop join*: $M + \lceil \frac{M}{B-2} \rceil \cdot N$
$\overset{1.}{\Rightarrow} M = 50, N = 100, B = 12$, so
$50 + \lceil \frac{50}{12-2} \rceil \cdot 10 = 550$ I/Os

(b) Cost of a *index nested loop join*: $M + ((M \cdot p_T) \cdot \text{cost of finding matching E tuples})$
$\overset{1.}{\Rightarrow} M = 50, p_T = 100, \text{cost of finding matching E tuples} = 3 + 1 = 4$
$50 + ((50 \cdot 100) \cdot 4) = 20.050$ I/Os

(c) The best option would be the *Sort-Merge Join* with costs of: $M \log M + N \log N + (M + N)$
Since they are already sorted we can neglect the sorting cost of $M \log M + N \log N$ and have remaining costs of $M + N$
$\overset{1.}{\Rightarrow} M = 50, N = 100$
$50 + 100 = 150$ I/Os

(d) Identical to 3.3.3.c, since the B+ tree is just a special form of sorting. So analogously we receive the costs $M + N$
$\overset{1.}{\Rightarrow} M = 50, N = 100$
$50 + 100 = 150$ I/Os