Implementation of Databases Exercise 3

Ilya Kulikov 351063, Alina Shigabutdinova 351017, Oleg Chernikov 351016

November 30, 2015

Exercise 3.1

1. a. $\{A, B, C | \exists A_1, B_1, C_1, C_2$
   $R(A_1, B, C) \wedge$
   $R(A, B, C_1) \wedge$
   $C_1 = 1 \wedge R(A, B, C_2) \wedge (R(A, B, C_2) \wedge C_2 = 1 \wedge R(A_2, B, C_2))\}$

   b.

2. This is the minimal tableau. The rows, containing a_1 and a_2 are mapped to themselves, while the rest mapped to the last row.

ans	a_1	a_2	
	a_1	b_2	R
	a_1	b_4	R
	b_1	a_2	R
	b_2	b_3	R

Exercise 3.2

1. $\{p.name, p \in Prof | \exists c_1 \in Course, \exists c_2 \in Course \exists d_1 \in Dept, \exists d_2 \in Depth$
   $c_1.ctitle \neq c_2.ctitle \wedge c_1.ssn = c_2.ssn \wedge d_1.dnr = c_1.dnr \wedge d_2.dnr = c_2.dnr \wedge d_1.dname \neq d_2.dname\}$

2. $\{p.name, p \in Prof | \exists c \in Course, \exists d \in Dept$
   $c.ssn = p.ssn \wedge p.city = d.city \wedge c.dnr = d.dnr\}$ Interpretation is: Names of the professors, that give lectures at departments, which located at the same city, as they are.

3. Only the example in 3.2.2 has a cycle in it (3.2.1 has a strict tree-like structure). That means, we can't use semi-joins, as there is no combination of them, computing the correct result. Also, there are problems with computation of the range queries for them, as we can't reduce the amount of computation, as in other cases. Finally, we spend more memory, as we have to store intermediate results of our query.

Exercise 3.3

1. The costs are represented in the following table:

	Sorted	Clustered B+ tree	Clustered Hash
$\sigma_{a=50000}(R)$	$D \cdot log_2 B$	$D \cdot (1 + log_G 0.15B)$	2D
$\sigma_{a \neq 50000}(R)$			
$\sigma_{a>50000 \wedge a<50010}(R)$	$D \cdot (log_2 B + \#matchin\_records)$	$D \cdot (log_G 0.15B + \#matching\_records)$	$B \cdot D$

Thus, for the first query, most probable cheapest result will be given by *Clustered Hash*, for third: *B+ tree*.

2. a. Number of runs $n = \frac{N}{B} = \frac{500000}{2 \cdot 1000} = 250$. We put 2 in denominator, as from the optimization we can place 2 times as much records on page. Length of each run will be equal to 2000 in this case, while I/O Cost $= 2N = 10^7$

   b. The number of passes is $\lceil 1 + log_{B-1}\lceil N/B \rceil \rceil = 2$ and cost for each pass is $2N = 10^7$

3. Block-nested loop join is a variation of naive approach and therefore it will give the poorest performance $O(n^2)$. It is not recommended to use it on such a large table. Sort-merge is more advanced approach, which has complexity $O(nlog_2 n)$ and may be used to get better performance in some cases. Hash join has $O(n)$ complexity and is used prior to Sort-merge in some modern DBMS, like Oracle, for instance. However, if the memory buffer is big enough it is close to performance of sort-merge, which isn't probably the case this time. Thus, for this task it is better to use Hash-join.