5. Exercise

Timo Bergerbusch 344408
Thomas Näveke 311392
Shu Xing 381176

11.12.2017

Exercise 5.1

Important numbers:

B: $\frac{20\,000\,000 \text{ Records}}{40 \frac{\text{Records}}{\text{Page}}} = 500\,000$ Pages

R: $40 \frac{\text{Records}}{\text{Page}}$

1.a) $\sigma_{ID=500}$

Answer: Use path A3.
Explanation: The unclustered hash index needs only $2D$ I/Os. All other produce higher costs. The clustered B^+ tree index would need a G like the following

$$D \cdot (log_G 0.15B) \leq 2D$$
$$log_G 75000 \leq 1$$
$$4.261159991 \cdot 10^{-411955} \leq G < 1$$

Such a G is not possible as a tree-branch-factor. So the unlustered hash index is most efficient.

1.b) $\sigma_{ID \neq 500}$

Answer: Use path A2
Explanation:

type	formula	cost
sorted	$B \cdot D$	$500\,000D$
B^+ tree	$0.15BD + 1.5BD$	$825\,000D$
hash index	$BD(R + 0.125)$	$20\,062\,500D$

1.c) $\sigma_{ID>500 \wedge ID<1500}$

Answer:
Explanation:
For the sorted file and the clustered has index the cost would be $B \cdot D = 500\,000D$. For a clustered tree index to be cheaper we have to consider the two variables G as the branching factor of the B^+ tree. The number of matching pages is given by $1500 - 500 = 1000$. The cost can be computed as $D \cdot (\log_G 0.15B + \#matchingPages) \Rightarrow D \cdot (\log_G 75\,000 + 1000)$. Now for the tree index to be cheaper this has to be $< 500\,000D$.

$$\begin{aligned} D \cdot (\log_G 75\,000 + 1000) &< 500\,000D \\ \log_G 75\,000 + 1000 &< 500\,000 \\ \log_G 75\,000 &< 499\,000 \\ G &< \sqrt[499\,000]{75\,000} \approx 1.00002 \end{aligned}$$

2.a)

2.b) i.

2.b) ii.

2.b) iii.

Exercise 5.2

1.

Split the computing into two parts:

- Finding the first suitable records

  1. through the stated average size and load we retrieve $0.15 \cdot B$ data entries
  2. the height of such a tree then is given by $log_G 0.15 \cdot B$
  3. Since every one has to be read we multiply it by D

  $\Rightarrow$ $D \cdot (log_G 0.15 \cdot B)$

- find first following non-suitable record We then continue reading the index structure until we encounter a record that is not suitable. Through the clustering we know that there are no suitable records following. So we read the number of suitable records = $\#matchingPages$ many records
  $\Rightarrow D \cdot \#matchingPages$

$\Rightarrow$ $D \cdot (log_G 0.15 \cdot B) + D \cdot \#matchingPages = D \cdot (log_G 0.15 \cdot B + \#matchingPages)$

2.

- read the page of the hash-group, which contains all entries that satisfy the equality $\Rightarrow D$
- read the data entry satisfying the equality $\Rightarrow D$

$\Rightarrow D + D = 2D$

3.

- the costs of retrieving the file containing the data computes as in Exercise 5.2.1 to $D \cdot (\log_G 0.15 \cdot B)$
- the costs of finding the data entry is constant D
- rewriting the index page and datafile takes $2D$

$\Rightarrow D \cdot (\log_G 0.15 \cdot B) + D + 2D = D \cdot (3 + \log_G 0.15 \cdot B)$

Exercise 5.3

- Regarding `Q1`:
  Equality Selection based on its primary key sno:

type	formula	cost
heap	$\frac{1}{2}BD$	$500D$
Sorted	$D \cdot \log_2 B$	$9.97D$
Clustered Tree Index	$D \cdot (1 + \log_G 0.15B)$	$2.09D$
Unclustered Tree Index	$D \cdot (1 + \log_G 0.15B)$	$2.09D$
Unclustered Hash Index	$2D$	$2D$

  $\Rightarrow$ for `Q1` a unclustered hash index on sno would be the best option. A(n) (un)clustered tree index would only be very slightly less efficient.

- Regarding `Q2`:
  Range Selection based on the salary:
  Obviously a tree index is the most efficient for such queries. Also having a clustered tree index would be more efficient than an unclustered tree index, since otherwise each suitable records would cost 1 page I/O. A clustered tree index can read the records continuously which results in less page I/Os.
  So a clustered tree index would be the best choice.

- Regarding `Q3`:
  maybe indexing sno?