

**CMPT-454 Fall 2009**  
**Instructor: Martin Ester**  
**TA: Yi Liu**

**Solution Midterm Exam**

Total marks: 150 (15 % of the class)

Date: October 28, 2009

**Problem 1** (40 marks)

Suppose a disk with an actual (formatted) capacity of 8 gigabytes ( $2^{33}$  bytes). The disk has 16 surfaces and 1024 tracks. The disk rotates at 7200 rpm (rotations per minute). The average seek time is 9 ms. The block size is 8KB.

a) What is the capacity (in bytes) of a single track?

$$8\text{GB}/(16*1024) = (2^{33})/(2^4)*(2^{10}) = 2^{19} \text{ bytes} = 0.5\text{MB}$$

b) Suppose we are reading a file that occupies exactly one entire track. How long does it take to read the entire file sequentially?

$$\text{Transfer time for one track} = \text{time for one rotation} = 60 \text{ s} / 7200 = 8.3 \text{ ms}$$

$$\text{Read time} = \text{average seek time} + \text{rotational delay} + \text{transfer time for track}$$

$$= 9 + 8.3 / 2 + 8.3 \text{ ms} = 21.5 \text{ ms}$$

**Problem 2** (40 marks)

Suppose that we use a variant of B-trees to maintain sorted tables. In this variant, the leaves contain the records themselves, not pointers to the records. A leaf can contain as many records as will fit in a block (allowing for a sequence pointer). Non-leaf nodes are also one block in size, and contain as many keys (and appropriate pointers) as will fit. Assume the following:

- Blocks size is 4096 bytes.
- Each record is 300 bytes long.
- A block pointer is 10 bytes.
- A record pointer is 12 bytes.
- A search key is 8 bytes long.

- The nodes are 85% occupied. For example, if a leaf can hold 100 records, it will only hold 85. If a non-leaf can hold 100 keys, it will only hold 85. For 85% calculations, round to the nearest integer.
- Note the following exceptions. Of course, the root node can be less than 50% occupied, it needs to have at least one entry. Nodes at the second level (directly under the root) have to be at least 50% occupied but can be less than 85% occupied.
- The indexed file has 1,000,000 records.

How many blocks will this B-tree use?

Leaves can hold  $\text{floor}((4096-10)/300)=13$  records. If leaves are 85% occupied, we can hold 11 records per node. This means the leaves will require  $1000000 / 11 = 90910$  blocks. (10 marks)

For interior nodes we need to determine the value of  $n$  from  $(n+1)*10+n*8 \leq 4096$  which gives us  $n=227$ . (10 marks)

If interior nodes are 85% occupied they actually use 194 pointers. Thus our first level above the leaves will require  $90910 / 194 = 469$  nodes. (10 marks)

The level above that will require  $469 / 194 = 3$  nodes which are all children of the root. (10 marks)

The total is  $90910+469+3+1=91383$  blocks.

### Common errors

- Assuming 100% occupancy instead of 85% occupancy: deduct 4 marks.
- Using record pointers instead of block pointers when computing  $n$ : deduct 2 marks.

### Problem 3 (40 marks)

Consider an extensible hash table with a capacity of 2 entries per block and binary hash keys of length 6. Suppose that we do not create overflow blocks, but perform as many splits as necessary to ensure that all entries fit into their corresponding buckets.

- a) Start with an empty hash table with two directory entries. What is the minimum number of insertions (number of hash keys) that increases the directory size to 64 entries?

Three insertions

- b) Give a sample sequence of insertions that generates this case.

000000, 000000, 000001 or 111111, 111111, 111110

- c) Show the extensible hash table that results from the insertions given in b), including the directory, its global level, all pointers, all buckets and their local levels.

directory		buckets	local level
global level 6			
000000	→	000000, 000000	6
000001	→	000001	6
000010	→	empty	5
000011			
000100	→	empty	4
000101			
000110			
000111			
001000	→	empty	3
001001			
001010			
001011			
001100			
001101			
001110			
001111			
010000	→	empty	2
010001			
...			
011111			
100000	→	empty	1
100001			
...			
111111			

**Problem 4** (30 marks)

Consider the relations R(A, B), S (B, C) and T(A, B, C) and the following relational algebra expression:

$$\pi_C (\sigma_{(A<10) \wedge (C>20)} [(R \bowtie S) - T])$$

Optimize this expression by pushing down selections and projections as much as possible. List all the steps of your optimization and the relational algebra equivalences that you apply in each step.

$$\begin{aligned} \pi_C (\sigma_{(A<10) \wedge (C>20)} [(R \bowtie S) - T]) &= \pi_C [\sigma_{(A<10) \wedge (C>20)} (R \bowtie S) - \sigma_{(A<10) \wedge (C>20)} T] \\ &\text{using } \sigma_{cond} (R - S) = \sigma_{cond} R - \sigma_{cond} S \\ &= \pi_C [((\sigma_{A<10} R) \bowtie (\sigma_{C>20} S)) - \sigma_{(A<10) \wedge (C>20)} T] \end{aligned}$$

using  $\sigma_{cond1 \wedge cond2}(R \bowtie S) = (\sigma_{cond1} R) \bowtie (\sigma_{cond2} S)$

where cond1 references only attributes of R and cond2 references only attributes of S

### Notes

- We cannot push down the projection, i.e. the following expression is not equivalent

$$\pi_C[(\sigma_{A < 10} R) \bowtie (\sigma_{C > 20} S)] \neq \pi_C(\sigma_{(A < 10) \wedge (C > 20)} T)$$

because  $\pi_C(R - S) \neq \pi_C R - \pi_C S$ .

See the following counter example:

$$R = \{(a1, b1, c1), (a1, b1, c2), (a2, b2, c2), (a2, b2, c3)\}, S = \{(a1, b1, c1), (a2, b2, c2)\}$$

$$\pi_C(R - S) = \pi_C\{(a1, b1, c2), (a2, b2, c3)\} = \{(c2), (c3)\}$$

$$\neq \{(c3)\} = \{(c1), (c2), (c3)\} - \{(c1), (c2)\} = \pi_C R - \pi_C S$$

- In the first step of optimization, we can use  $\sigma_{cond}(R - S) = \sigma_{cond} R - S$  instead of  $\sigma_{cond}(R - S) = \sigma_{cond} R - \sigma_{cond} S$ .

### Common errors

- Also pushing the projection: deduct 5 marks.
- Forgetting to list the RA equivalences used: deduct 10 marks.