Fundamentals of Database Systems

Assignment: 4

Due Date: 28th August, 2017

Instructions

This question paper contains 15 questions in 5 pages.

Q1: Assume that a B-tree has been built with data pointers of size 2B, tree pointers of size 3B and key size 5B. The size of each page in the system is 128B. What is the branching factor of the B-tree?

A. 12

B. 14

C. 13

D. 10

Explanation: $p(3) + (p - 1)(5 + 2) \leq 128$

Q2: Assume that a data file contains 1000 records that are ordered by a key attribute A, and a primary index on attribute A is built. Let the size of key be 5B and block pointer be 5B. Each block of the system is of 105B, out of which 100B can be used to store data and 5B is reserved for storing meta data. How many disk accesses will be required to fetch the record using the index in *average* case?

A. 3

B. 4

C. 5

D. 6

Explanation: Number of disk access for binary search the index ($100/10 = 10$blocks) plus one access to fetch data; $\lceil log(100/10) \rceil + 1$

Q3: Consider a relation $R(a, b, c)$ ordered on a non-key attribute b, and an index is maintained on attribute c. To evaluate the following expression, which is a better order of operation?

$$\sigma_{(b=`abc' \wedge c=12)}(R)$$

A. First filter tuples using index on c and then search for b

B. First search tuples for the given value of b and then use index on c

C. The order of operation does not matter

D. Insufficient data to answer

Explanation: The size of block and file size is not mentioned hence we cannot figure out the actual block transfers required to fetch index entry or to binary search the record

Q4: Consider two relations R and S of size 600 and 500 bytes repectively. The size of a tuple in both relations is 20B. The block size of the system is 200B. How many extra block transfers would be required, in the *worst case*, if *nested loop* join is used instead of *block nested loop* join technique to compute $R \bowtie S$?

A. 78

B. 93

C. 66

D. 105

Explanation: Relation R and S require 3 blocks each and have 30 and 25 tuples respectively. Since S is the smaller thus it will be choose as inner relation. In worst case, one block of each relation can fit in memory, so the number of block transfers require for *nested join* is $(25 \times 3) + 3 = 78$ and for *block nested join* is $(3 \times 3) + 3 = 12$. therefore, if we use *nested join* instead of *block nested join* we will require $78 - 12 = 66$ extra block transfers.

Q5: Consider the following operations and the associated list of techniques.

1. Deduplication: Hashing
2. Difference: Hashing, Sorting
3. Full Outer Join: Block Nested join, Index-Nested join
4. Left-Outer Join: Hash join, Index-Nested join

Which of these operations can be implemented using *all* the techniques mentioned in the corresponding list?

A. Only deduplication, full outer join and left-outer join

B. Only deduplication

C. Only deduplication and full outer join

D. Only difference and full outer join

Explanation: Discussed in lecture

Q6: Which of the following relational operators are *commutative*?

A. Difference, Union

B. Full outer join and Left outer join

C. Full outer join and intersection

D. Right outer join and Union

Explanation: From text

Q7: Where are databases generally stored?

A. Cache

B. Main Memory

C. Magnetic Disks (HDD)

D. Optical Disks (CD)

Explanation: Cache and Main Memory are volatile, Optical Disks would not genearally allow update operations (or would make them very slow).

Q8: Search time is of crucial importance for an indexing scheme. What are the other points on which one should evaluate an indexing scheme?

A. Modification overhead

B. Space overhead

C. Both modification and space overheads

D. Neither modification nor space overhead

Explanation: Basic indexing requirements.

Q9: In what kind of index does an index entry appear only for certain keys?

A. Dense

B. Sparse

C. Simple

D. Inner

Explanation: Basics of indexing.

Q10: Consider a B+-tree with the following specifications:
Page size c = 2 kB; Size of key γ = 32 bytes; Size of pointer η = 8 bytes.

What is the order m of the B+-tree? How many records can a tree of height 3 store?

A. $m = 26$, Number of records $\approx 1.90 \times 10^4$

B. $m = 51$, Number of records $\approx 1.38 \times 10^5$

C. $m = 26$, Number of records $\approx 1.75 \times 10^4$

D. $m = 51$, Number of records $\approx 1.25 \times 10^5$

Explanation: $c \geq \gamma \times m + \eta \times (m + 1)$
$2 \times 1024 \geq 32 \times m + 8 \times (m + 1)$
$\therefore m = \lfloor 51 \rfloor = 51$
Number of records $= m \times (m + 1)^2 = 51 \times 52^2 = 137904$.

Q11: Consider a B-tree with the following specifications:
Page size c = 2 kB; Size of key γ = 32 bytes; Size of pointer η = 8 bytes.

What is the order m of the B-tree? How many records can a tree of height 3 store?

A. $m = 21$, Number of records $\approx 9.26 \times 10^3$

B. $m = 22$, Number of records $\approx 1.22 \times 10^4$

C. $m = 42$, Number of records $\approx 7.95 \times 10^4$

D. $m = 43$, Number of records $\approx 8.52 \times 10^4$

Explanation: $c \geq \gamma \times m + \eta \times m + \eta \times (m+1)$
$2 \times 1024 \geq 32 \times m + 8 \times (m+1) + 8 \times m$
$\therefore m = \lfloor 42.5 \rfloor = 42$

Number of records $= m + m \times (m+1) + m \times (m+1)^2 = 42 + 42 \times 43 + 42 \times 43^2 = 79506$.

Q12: Consider a bitmap indexing scheme on the following data

Roll	Year	Department	Program
12111	2012	CS	PhD
13115	2013	CS	MTech
13121	2013	EE	MTech
13125	2013	EE	PhD
14111	2014	CS	MTech
14119	2014	CS	PhD

where the possible values of year, department and program are as follows:
year = {2011, 2012, 2013, 2014}; department = {CS, EE}; program = {MS, MTech, PhD, BTech}.

What will be the bitmaps used and the result for the query: *Find out students from year 2013 that are enrolled in MTech program.*

A. bitmap(2013) = 011100, bitmap(MTech) = 011010, result = 011000

B. bitmap(Year) = 0132, bitmap(Program) = 0330, result = 2

C. bitmap(Year) = 0132, bitmap(Program) = 0330, result = 0130

D. bitmap(2013) = 100011, bitmap(MTech) = 100101, result = 100111

Explanation: Bitmap is on the all possible values of an attribute.
In this case, "bitmap(2013) AND bitmap(MTech)" will be the query to produce the required output.

Q13: In an external merge sort, where number of blocks in the relation is 324, suppose we can put only 4 blocks at a time into the memory. In the worst case, how many total block transfers will happen?

A. 2592

B. 3240

C. 3888

D. 4230

Explanation: $b = 324$, $M = 4$
Number of block transfers $= 2b(\lceil log_{M-1} \lceil b/M \rceil \rceil + 1)$
$= 2 \times 324 \times (\lceil log_3 81 \rceil + 1)$
$= 2 \times 324 \times 5$
$= 3240$

Q14: In an external merge sort, where number of blocks in the relation is 324, suppose we can put only 4 blocks at a time into the memory. In the worst case, how many total seeks will happen?

A. ~2400

B. ~2700

C. ~3000

D. ~3300

Explanation: $b = 324$, $M = 4$
Number of seeks = $2\lceil b/M \rceil + 2b(\lceil log_{M-1}\lceil b/M \rceil \rceil)$
$= 2 \times 81 + 2 \times 324 \times (\lceil log_3 81 \rceil)$
$= 162 + 2 \times 324 \times 4$
$= 2754$

Q15: Consider the simple nested-loop join of the following two relations r and s.

Relation	r	s
Tuples (n)	2400	1500
Blocks (b)	40	50

Assuming the worst case memory availability, i.e., the memory can hold only one block of each relation at a time, what is the number of block transfers and seeks?

A. Transfers = 120040, seeks = 4800

B. Transfers = 4800, seeks = 120040

C. Transfers = 60050, seeks = 3000

D. Transfers = 3000, seeks = 60050

Explanation: Outer relation should be smaller, thus s should be the outer loop and r the inner.
Transfers = $b_s + n_s \times b_r = 50 + 1500 \times 40 = 60050$
Seeks = $2 \times n_s = 3000$.