CS-245 Database System Principles

Midterm Exam – Summer 2001

SOLUTIONS

This exam is open book and notes.

There are a total of 110 points. You have 110 minutes to complete it.

Print your name: ____

The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.
While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.
Signed: ____

Problem 1 (20 points)

State if the following statements are TRUE or FALSE. Write down the answer inside the box on the left

Answer:

$\boxed{F}$ a) On average, repeated random IO's are faster than repeated sequential IO's because random IO's tend to access different cylinders and therefore cause less contention.

$\boxed{F}$ b) The LRU ("least recently used") buffer replacement strategy is a good strategy for repeated sequential scans of the same file.

$\boxed{F}$ c) The LRU ("least recently used") buffer replacement strategy is a good strategy for R-Trees.

$\boxed{T}$ d) B-Trees can be used for multi-key indexing.

$\boxed{T}$ e) When performing a search in a three-level R-Tree, it may be necessary to follow more than one pointer from the root node.

$\boxed{T}$ f) Say it is known that relation R with attributes (x, y, a) can never have more than five tuples (rows) for every unique (x, y) pair. Say we wish to build a grid file indexing attributes x and y. It is possible to do so without using indirection from grid entries to buckets.

$\boxed{F}$ g) For any data file, it is possible to construct two separate sparse indexes on different keys.

$\boxed{F}$ h) It makes sense to construct a two-level index that has a dense first level and a dense second level.

$\boxed{F}$ i) One way to handle duplicate keys in an unclustered index is to maintain a pointer to only the first occurrence of the key in the file.

$\boxed{T}$ j) Linear hashing uses overflow buckets.

$\boxed{F}$ k) Since there is no downside to having indexes, we may as well build an index on every column of every relation to speed up as many queries as possible.

$\boxed{F}$ l) Most commercial database vendors don't implement hashing indexes because they are slower than B-Trees for equality lookups.

T	m) For a given SQL query, there may be several corresponding logical query execution plans.
T	n) For a given logical query plan, there may be several corresponding physical query execution plans.
F	o) If all the tracks on a disk have the same number of sectors, and all sectors are of the same size (in bytes), then the bit density (bits per inch) is uniform across the entire disk.
F	p) Double-buffering speeds up the processing of a single block by parallelizing disk IO with CPU processing.
T	q) A record can be located on disk given the following information: disk number, platter number, platter surface (top or bottom), cylinder number, sector number within track, and offset within sector.
T	r) Say a randomly chosen block is being fetched from disk. The overall access delay per byte decreases as the block size increases.
T	s) Physical data independence means that users are shielded from changes to the physical design of the database such as additions/deletions of indexes or changes to the sequential order of a file.
T	t) A query plan asks for just columns A and B of relation R(A, B, C, D). Relation R is stored in a file using 100 blocks. There is an unclustered B-Tree of height 2 over R with key (A, B) that has 20 leaf nodes. It would be faster (in terms of number of IO's) to use the index than to perform a sequential scan on the file.

Problem 2 (3 points)

a) Name two advantages associated with a declarative query language interface.

Answer: there are more than two ☺ But any two will do for full credit:

- **The application programmer does not need to be aware of the physical design of the database.**
- **The application programmer has less work to do (i.e., writing a SQL query is easier than writing code to traverse an index and perform a join).**
- **The application code does not have to be changed when the physical design of the database changes.**
- **It is possible to have a query optimizer that selects a good execution strategy at run-time.**
- **The query language may be standardized across database systems by different vendors.**

b) What are the three most significant components of the delay incurred while fetching a block from disk?

Answer: Seek time + rotational delay + transfer time

Problem 3 (25 points)

For this problem, consider a disk with the following characteristics:

Rotation speed = 3600 RPM (revolutions per minute)
Average seek time = 15 ms
1024 Sectors per track
Gaps between sectors take up 10% of the space
Sector size = 8KB
Block size = 1 sector = 8KB
(The disk controller does not have to find the "start" of a track to start reading. It can start reading at any sector.)

Relation R is stored on the disk as a contiguous sequential file ordered on attribute A, which is unique (i.e., no two records have the same value for attribute A). Relation R has 2^20 fixed size records. The records in R are 1KB in size, of which attribute A uses only 12 bytes (fixed size). Say we would like to create a static sequential index file for attribute A of R. Assume that the index file will be stored on a distant cylinder from the file for relation R. Consider two design choices for the conventional index:

I. sparse index – one pointer per block of R
II. super-sparse index – one pointer per track

Assume that a pointer to a track requires 2 bytes.

a) How much space (in bytes) is required to represent a pointer to a block (round up to the nearest byte)?

Answer: # of blocks per track = 1024, so need 2 bytes to represent address of block within a track plus 2 bytes for track address. Total = 4 bytes.

b) What would be the size (in bytes) of the sparse index (option I)?

Answer: # of entries = # of blocks in R = 2^{20}KB/8KB = 2^{17}
size of each entry = 4 bytes for block pointer + 12 bytes for key = 16 bytes

Size of index = (16 bytes)* 2^{17} = 2^{21} bytes = 2MB
Size in blocks = 2MB/8KB = 256 blocks

c) What would be the size of the super-sparse index (option II)?

Answer: # of entries = # of tracks in R = 2^{20}KB/(8KB * 1024) = 2^{7}
size of each entry = 2 bytes for track pointer + 12 bytes for key = 14 bytes

Size of index = (14 bytes)* 2^{7} = 1792 bytes (fits on one block)

Let's say we want to locate a record based on it's A value (i.e., find the record with A = c). We expect that, on average, the key c occurs about half way through the file. Assume that there is a matching record for key c, and that the memory buffer starts out empty. How long will it take to retrieve the record:

d) Using the sparse index (option I)?

Answer: on average, read half of index; then read one block of R.
To read half of index: seek + ½(rotation time) + transfer time for half of index
= $15\text{x}10^{-3}$ + ½(1/60) + (1/60)*(128/1024) = .0254 s
To read one block of R: seek + ½(rotation time) + transfer time for one block
= $15\text{x}10^{-3}$ + ½(1/60) + (1/60)*(1/1024)*0.9 = .0233 s
Total = .0254 + .0233 = .0487 seconds = 48.7 ms

e) Using the super-sparse index (option II)?

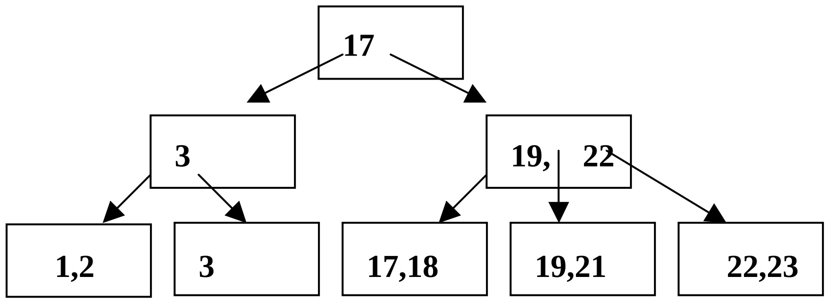
Answer: on average, read half of index; then read one track of R.
To read half of index: seek + ½(rotation time) + transfer time for index (one block)
= $15\text{x}10^{-3}$ + ½(1/60) + (1/60)*(1/1024)*0.9 = .0233 s
On average this one block that we want to read will be half rotation apart.
Time to Read A = seek + ½(rotation time) + transfer time for data (one block)
= $15\text{x}10^{-3}$ + ½(1/60) + (1/60)*(1/1024)*0.9 = .0233 s

Total = .0233 + .0233 = .0466 seconds = 46.6 ms

Problem 4 (10 points)

Draw an example of a valid 3-level B+Tree of order n=2 where inserting an entry with key 20 causes a split to a non-leaf but does not increase the height of the tree.

Answer: (many answers acceptable … here is one example)



Inserting key 20 causes the leaf containing 19,21 to split, and the non-leaf containing 19,22 to split. A new entry is added to the root, but the root does not split.

Problem 5 (16 points)

This problem tackles dynamic hashing.

a) Consider an extensible hash table where 200 buckets are actually allocated at this point. What is the size (in the number of entries) of the smallest possible directory in this case?

Answer: smallest power of 2 larger than 200 (directory size must be a power of 2)

256

b) Consider an extensible hash table whose directory has 1024 entries. What is the largest number of entries that could point to the same bucket?

Answer: could have half of the entries pointing to the same bucket (if that bucket never split)

512

c) Consider a linear hash table with max used bucket m = 110 (in binary). Which bucket should a key with hash value 111 (binary) be sent to?

Answer: since its hash value is greater than m, change leading 1 to 0

011 (binary)

d) Consider a linear hash table whose buckets can hold up to 2 records. Say that the structure currently holds 18 entries, and m = 11. What is the smallest value for m that would eliminate all overflow, in the best case?

Answer: in the best case, all 18 entries evenly distribute themselves into 9 buckets, so m could be as small as 8 (since the buckets start at 0). Note: the initial value for m is irrelevant, as long as it is less than or equal to 8.

1000 (binary) = 8 (decimal)

Problem 6 (20 points)

a) Consider a relation R of size B(R) blocks. Say we want to sort R using an optimized version of external merge sort that uses heap-sort as the in-memory sorting algorithm. Assume that due to heap-sort, we can expect the sorted chunks ("sorted runs" or "sorted sublists") output by the first pass to be twice the size of memory. How much memory (in blocks) would be required to sort R in two passes? Ignore the need for a buffer page dedicated to output in the second pass. Give an answer in terms of B(R).

Answer: after first pass, # of sorted runs = B(R)/2M
To perform second pass, need M ≥ # of sorted runs

$M \geq B(R)/2M$
$M^2 \geq B(R)/2$
$M \geq$ SquareRoot(B(R)/2) ← acceptable solution

More precisely: leave room for one buffer page for input
$M \geq B(R)/2M + 1$ … solve quadratic equation & take (+) root …
$M \geq \frac{1}{2}$ + SquareRoot(1/4 + B(R)/2)

b) Let's say we'd like to compute the join of relations R and S using hash join, where B(R) = 400 blocks and B(S) = 500 blocks. Assume that we have M = 40 blocks of memory available in our buffer. We notice that there is more than enough memory, so we keep one of the hash buckets in memory at all times to avoid writing and then re-reading it. Assuming that all buckets have the same size, how many buckets should be used in order to still be able to perform the join in two passes?

Answer: First of all, since R is the smaller relation, we'll use it as the "load" relation. Therefore, our strategy for phase I should be to first hash S using the standard scheme, and then hash R but keep one bucket of R in memory rather than staging it to disk.

Let k = # of hash buckets.
Bucket size (in blocks) = B(R)/k
# of R buckets written to disk = k – 1. # of R buckets kept in memory = 1.
Memory needed for 1st pass: 1 input buffer + k – 1 output buffers + 1 bucket
$M \geq 1 + (k - 1) + B(R)/k$
$M \geq k + B(R)/k$. This tells us how much memory is required, as a function of k.
Let's find the value of k that results in the smallest possible memory usage:
Derivative of right side w.r.t. k: $1 - B(R)/k^2 = 0$; $k^2 = B(R)$; k = SquareRoot(B(R))
At this point we know that using <u>k = SquareRoot(B(R)) = 20</u> minimizes the memory requirement. Let's make sure we have enough memory to make this work:
$M \geq k + B(R)/k$; Plug in k = 20 → $M \geq 40$. That's exactly how much we have!

c) Let k be the number of hash buckets (i.e., the answer from part (b)). In terms of k, what would be the cost (in number of IO's) of the operation proposed in part (b)?

Answer: normal hash join cost = 3(B(R) + B(S))
By keeping one bucket of R in memory, we avoid writing & re-reading one bucket, so we save 2*bucket_size IO's

Cost = normal hash join cost – 2*bucket_size = 3(B(R) + B(S)) – 2*(B(R)/k)
= 3(400 + 500) – 2*(400/20) = 2700 – 40 = 2660

Problem 7 (15 points)

Say we have two unary relations, R and S:

R	S
7	8
2	4
9	2
8	1
3	3
9	2
1	7
3	3
6	

Show the result of joining R and S using each of the following algorithms. List the results in the order that they would be output by the join algorithm.

a) Naïve (one tuple at a time) nested loops join. Use R for the outer loop and S for the inner loop.

Answer: pretty straightforward – for every tuple in R (in order), find matches in S (in order)

7, 2, 2, 8, 3, 3, 1, 3, 3

b) Sort merge join.

Answer: easy – they come out in sorted order on the join attribute

1, 2, 2, 3, 3, 3, 3, 7, 8

c) Hash join. Assume there are two hash buckets, numbered 0 and 1, and that the hash function sends even values to bucket 0 and odd values to bucket 1. In Phase II, use R as the “load” relation and S as the “stream” relation. Assume that bucket 0 is read first and that the contents of a bucket are read in the same order as they were written.

Answer: after first phase:

R bucket 0: 2, 8, 6
R bucket 1: 7, 9, 3, 9, 1, 3
S bucket 0: 8, 4, 2, 2
S bucket 1: 1, 3, 7, 3

First load R bucket 0 and stream S bucket 0 to find matches: 8, 2, 2
Then load R bucket 1 and stream S bucket 1 to find matches: 1, 3, 3, 7, 3, 3

Final result:
8, 2, 2, 1, 3, 3, 7, 3, 3

Problem 8 (1 point)

What flavor is chocolate cake?

Answer: chocolate (other answers were also accepted)

END OF EXAM. Use these pages for scratch paper: