Name

CWID

Quiz
1

Oct 10, 2018
Due Oct 14, 11:59pm

Quiz 1: CS525 - Advanced Database Organization

Please leave this empty! 1.1 1.2 1.3 Sum

Instructions

- **You have to hand in the assignment using your blackboard**
- **This is an individual and not a group assignment. Fraud will result in 0 points**
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are **3** parts in this quiz
  1. Disk Organization
  2. SQL
  3. Index Structures

BY SUBMITTING THIS EXAM THROUGH THE ONLINE SYSTEM, I AFFIRM ON MY HONOR THAT I AM AWARE OF THE STUDENT DISCIPLINARY CODE, AND (I) HAVE NOT GIVEN NOR RECEIVED ANY UNAUTHORIZED AID TO/FROM ANY PERSON OR PERSONS, AND (II) HAVE NOT USED ANY UNAUTHORIZED MATERIALS IN COMPLETING MY ANSWERS TO THIS TAKE-HOME EXAMINATION.

Part 1.1 Disk Organization (Total: 20 Points)

Question 1.1.1 Disk Access (20 Points)

Consider a disk with a sector size of 512 bytes, 2000 tracks per surface, 50 sectors per track, five double-sided platters, and average seek time of 10 msec. Suppose that a block size of 1024 bytes is chosen. Suppose that a file containing $100,000$ records of 100 bytes each is to be stored on such a disk and that no record is allowed to span two blocks.

1. How many records fit onto a block?

2. How many blocks are required to store the entire file?

3. If the file is arranged sequentially on the disk, how many surfaces are needed?

4. How many records of 100 bytes each can be stored using this disk?

5. If pages are stored sequentially on disk, with page 1 on block 1 of track 1, what page is stored on block 1 of track 1 on the next disk surface?

6. How would your answer change if the disk were capable of reading and writing from all heads in parallel?

Part 1.2 SQL (Total: 35 Points)

Consider the following relations:

- `Suppliers`(`sid`:**integer**, `sname`:**string**, `address`:**string**)
- `Parts`(`pid`:**integer**, `pname`:**string**, `color`:**string**)
- `Catalog`(`sid`:**integer**, `pid`:**integer**, `cost`:**real**)

The key fields are underlined, and the domain of each field is listed after the field name. Therefore `sid` is the key for `Suppliers`, `pid` is the key for `Parts`, and `sid` and `pid` together form the key for `Catalog`. The `Catalog` relation lists the prices charged for parts by `Suppliers`. Write the following queries in SQL statements and in relational algebra expression.

Question 1.2.1 (5 Points)

Find the `name`s of suppliers who supply some red part

Question 1.2.2 (5 Points)

Find the `sid`s of suppliers who supply some red or green part.

Question 1.2.3 (5 Points)

Find the `sid`s of suppliers who supply some red part or are at 10 West 31st Street.

Question 1.2.4 (5 Points)

Find the `sid`s of suppliers who supply some red part and some green part.

Question 1.2.5 (5 Points)

Find pairs of `sid`s such that the supplier with the first `sid` charges more for some part than the supplier with the second `sid`.

Question 1.2.6 (5 Points)

Write only an SQL query that find the `pid`s of parts supplied by at least two different suppliers.

Question 1.2.7 (5 Points)

Write only an SQL query that find the `pid`s of the most expensive parts supplied by suppliers named Yosemite Sham.

Part 1.3 Index Structures (Total: 65 Points)

Question 1.3.1 B$^+$-tree Construction (15 Points)

Assume that you have the following table:

Item

SSN	name	age
2	Pete	13
23	Bob	23
5	John	49
39	Joe	45
29	Alice	77
17	Lily	3
19	Manny	33
7	Gertrud	29
11	Heinz	14
3	Sammy	34

Create a `B+-tree` for table **Item** on key *SSN*. Assume that the tree is initially empty and values are added in ascending order. Construct `B+-tree` for the cases where the number of pointers that will fit in one node is as follows:

a. Three

b. Five

Write down the resulting `B+-tree` after each step and when splitting or merging nodes follow these conventions:

- **Leaf Split**: In case a leaf node needs to be split during insertion and n is even, the left node should get the extra key. E.g, if $n = 2$ and we insert a key 4 into a node [1,5], then the resulting nodes should be [1,4] and [5]. For odd values of n we can always evenly split the keys between the two nodes. In both cases the value inserted into the parent is the smallest value of the right node.
- **Non-Leaf Split**: In case a non-leaf node needs to be split and n is odd, we cannot split the node evenly (one of the new nodes will have one more key). In this case the "middle" value inserted into the parent should be taken from the right node. E.g., if $n = 3$ and we have to split a non-leaf node [1,3,4,5], the resulting nodes would be [1,3] and [5]. The value inserted into the parent would be 4.
- **Node Underflow**: In case of a node underflow you should first try to redistribute values from a sibling and only if this fails merge the node with one of its siblings. Both approaches should prefer the left sibling. E.g., if we can borrow values from both the left and right sibling, you should borrow from the left one.

Question 1.3.2 Extensible Hashing-tree Construction (25 Points)

Suppose that we are using `Extensible Hashing` on a file that contains records with the following search key values: `3, 11, 7, 19, 14, 18, 15, 17, 20, 44, 33, 43`

Show the `Extensible Hash` structure for this file if the hash function is `h(x)=x` mod `7` and buckets can hold

a. 2 records

b. 3 records

Question 1.3.3 Operations (25 Points)

Given is the B$^+$-tree shown below ($n = 3$). Execute the following operations and write down the resulting B$^+$-tree after each operation:

insert(9), delete(23), insert(10), insert(8), delete(19), delete(11)

Use the conventions for splitting and merging introduced in the previous question.