Name

CWID

Exam
1

March 5th, 2018, 11:25am - 12:40pm

CS525 - Spring 2018 - Midterm Exam Solutions

Please leave this empty! 1 2 3.1 3.2 Sum

Instructions

- Things that you are **not** allowed to use
  - Personal notes
  - Textbook
  - Printed lecture notes
  - Phone
- You are allowed to bring one page (both two sides can be used) of cheat sheet that must be turned in with the exam
- The exam is **75** minutes long
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are `3` parts in this exam (`75` points total)
  1. `Disk Organization` (20)
  2. `SQL` (25)
  3. `Index Structures` (30)

Part 1 Disk Organization (Total: 20 Points)

Question 1.1 Disk Access (20 = 5 + 5 + 5 + 5 Points)

Assume a disk has the following specifications.

- An average seek time of 30ms (milliseconds)
- A 2400 RPM rotational speed
- An average transfer rate of 512B/ms (bytes per millisecond)
- A block size of 512 bytes

1. How much time does it take on average to locate and transfer a single block, given its block address?

   Solution

   Disk Access Time=seek time+rotational delay+transfer time=30ms+$\frac{1}{2} \times \frac{60\times10^3}{2400}+\frac{512B}{512B/ms}$
   =30ms + 12.5 ms+1= 43.5 ms.

2. The drive mentioned above is used to store a database file of `20,000 Student` records with fixed length. Each record has the following fields: `Name` (30 bytes), `SSN` (9 bytes), `Address` (40 bytes), `Phone` (10 bytes), `Birth date` (8 bytes), `Sex` (1 byte), `Major_dept_code` (4 bytes), `Minor_dept_code` (4 bytes), `Class_code` (4 bytes), and `Degree_program` (3 bytes). An additional byte is used as a deletion marker.

   (a) Calculate the record size in bytes

   Solution

   R = 30B + 9B + 40B + 10B + 8B + 1B + 4B + 4B + 4B + 3B + 1B = 114B

   (b) Calculate the average time it takes to find a record by doing a linear search on the file if the blocks are stored contiguously and double buffering is used.

   Solution

   Assume an unspanned organization:
   We need 5000 Blocks to store database file of 20,000 Student records.
   On average half of the 5000 blocks must be examined to find a specific record.
   If the file is contiguous and double−buffered, the seek time and rotational delay happen once, then 2500 contiguous blocks must be read.
   Therefore, seek time+rotational delay+transfer time=30ms+12.5 ms+2500 B/tr
   = 30ms+12ms +2500 $\times\frac{512B}{512B/ms}$= 2.5425 s

   Another solution

   Assume a spanned organization:
   We need around 4454 Blocks to store database file of 20,000 Student records.
   On average half of the 4454 blocks must be examined to find a specific record.
   If the file is contiguous and double−buffered, the seek time and rotational delay happen once, then around 2227 contiguous blocks must be read.
   Therefore, seek time+rotational delay+transfer time=30ms+12.5 ms+2227 B/tr
   = 30ms+12ms +2227 $\times\frac{512B}{512B/ms}$= 2.269 s

(c) Calculate the average time it takes if, instead, the blocks are stored randomly (scattered over the disk).

Solution

```
Assume an unspanned organization:
Once again 2500 blocks must be examined, but the seek and rotational times happen
for each block. That was calculated as 43.5 ms in the first question
Therefore so: 2500 × 43.5 ms = 108.75 s.

Another solution:

Assume a spanned organization:
Once again 2227 blocks must be examined, but the seek and rotational times happen
for each block. That was calculated as 43.5 ms in the first question
Therefore so: 2227 × 43.5 ms = 96.8745 s.
```

Part 2 SQL (Total: 25 Points)

Consider the following relations:

- Student(snum:**integer**, sname:**string**, major:**string**, level:**string**, age:**integer**)
- Class(cname:**string**, meets_at:**string**, room:**string**, fid:**integer**)
- Enrolled(snum:**integer**, cname:**string**)
- Faculty(fid:**integer**, fname:**string**, deptid:**integer**)

The meaning of these relations is straightforward; for example, `Enrolled` has one record per student-class pair such that the student is enrolled in the class. Write the following queries in `SQL`. No duplicates should be printed in any of the answers.

Question 2.1 (4 Points)

Find the names of all Juniors (`level = "JR"`) who are enrolled in a class taught by "I. Teach".

Solution

```
SELECT DISTINCT S.Sname
FROM Student S, Class C, Enrolled E, Faculty F
WHERE S.snum = E.snum AND E.cname = C.cname AND C.fid = F.fid AND
F.fname = 'I.Teach' AND S.level = 'JR'
```

Question 2.2 (4 Points)

Find the `age` of the oldest student who is either a "History" `major` or enrolled in a course taught by "I. Teach".

Solution

```
SELECT MAX(S.age)
FROM Student S
WHERE (S.major = 'History') OR S.snum IN (SELECT E.snum
               FROM Class C, Enrolled E, Faculty F
               WHERE E.cname = C.cname AND C.fid = F.fid AND F.fname='I.Teach')
```

Question 2.3 (4 Points)

For each faculty member that has taught classes only in `room` “R128”, print the faculty member’s name and the total number of classes she or he has taught.

Solution

```
SELECT F.fname, COUNT(*) AS CourseCount
FROM Faculty F, Class C
WHERE F.fid = C.fid
GROUP BY F.fid, F.fname
HAVING EVERY ( C.room = 'R128' )
```

Question 2.4 (4 Points)

Find the names of students enrolled in the maximum number of classes.

Solution

```
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum IN (SELECT E.snum
      FROM Enrolled E
      GROUP BY E.snum
      HAVING COUNT (*) >= ALL (SELECT COUNT (*)
      FROM Enrolled E2
      GROUP BY E2.snum ))
```

Question 2.5 (4 Points)

Find the names of students not enrolled in any class.

Solution

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum NOT IN (SELECT E.snum
           FROM Enrolled E )
```

Question 2.6 (5 Points)

For each `age` value that appears in `Student`, find the `level` value that appears most often. For example, if there are more `FR` level students aged 18 than `SR`, `JR`, or `SO` students aged 18, you should print the pair (18, `FR`). `"FR": Freshman, "SO": Sophomore, "JR": Junior, "SR": Senior.`

Solution

```sql
SELECT S.age, S.level
FROM Student S
GROUP BY S.age, S.level,
HAVING S.level IN (SELECT S1.level
         FROM Student S1
         WHERE S1.age = S.age
         GROUP BY S1.level, S1.age
         HAVING COUNT (*) >= ALL (SELECT COUNT (*)
                   FROM Student S2
                   WHERE s1.age = S2.age
                   GROUP BY S2.level, S2.age))
```

Part 3 Index Structures (Total: 30 Points)

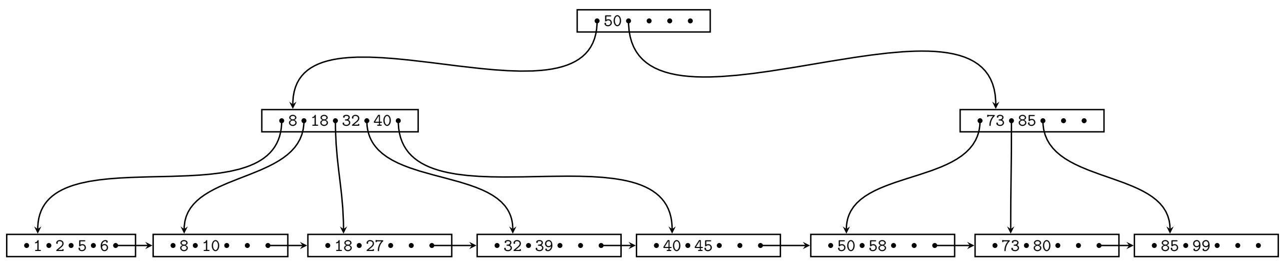
Question 3.1 B$^+$-tree Operations (20 Points)

Given is the `B+-tree` shown below ($n = 4$). Execute the following operations and write down the resulting `B+-tree` after each step:

insert(9), insert(3), delete(8), insert(46), delete(73), delete(85)

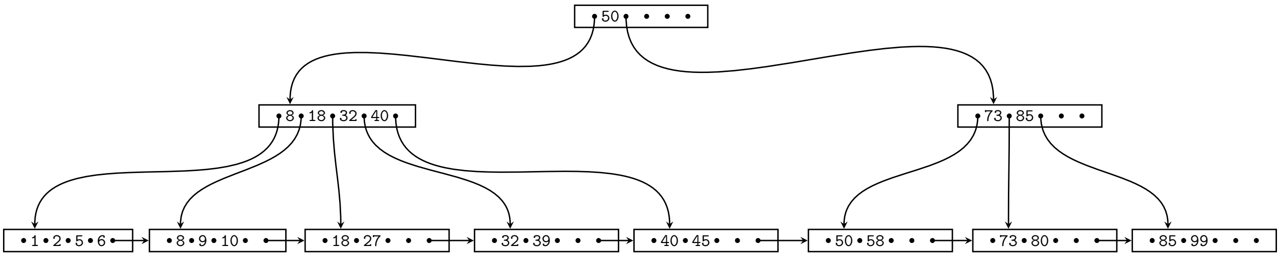
When splitting or merging nodes follow these conventions:

- **Leaf Split**: In case a leaf node needs to be split, the left node should get the extra key if the keys cannot be split evenly.
- **Non-Leaf Split**: In case a non-leaf node is split evenly, the "middle" value should be taken from the right node.
- **Node Underflow**: In case of a node underflow you should first try to redistribute and only if this fails merge. Both approaches should prefer the left sibling.

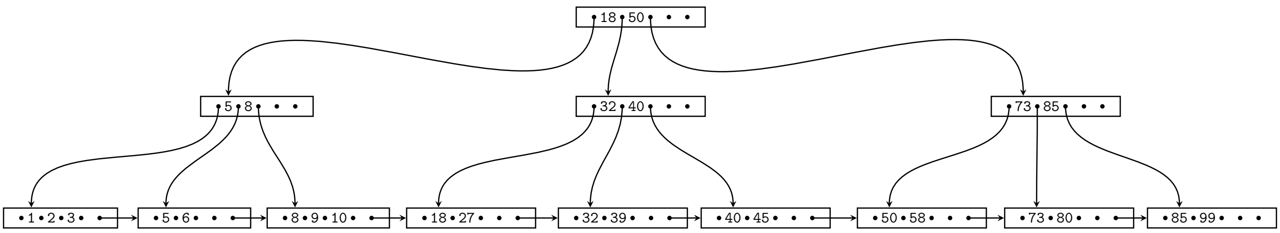


Solution

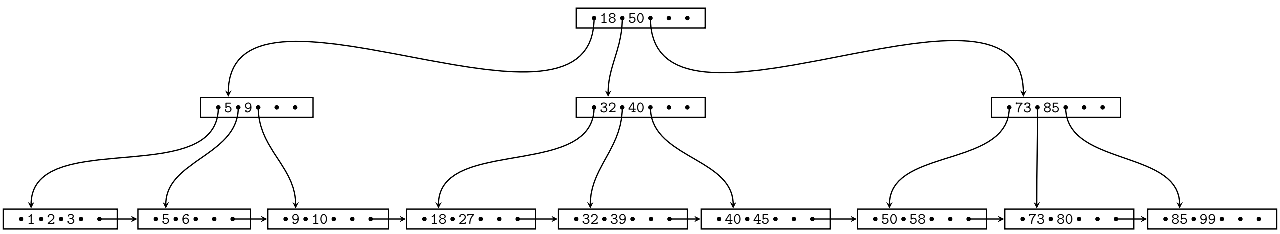
insert(9)



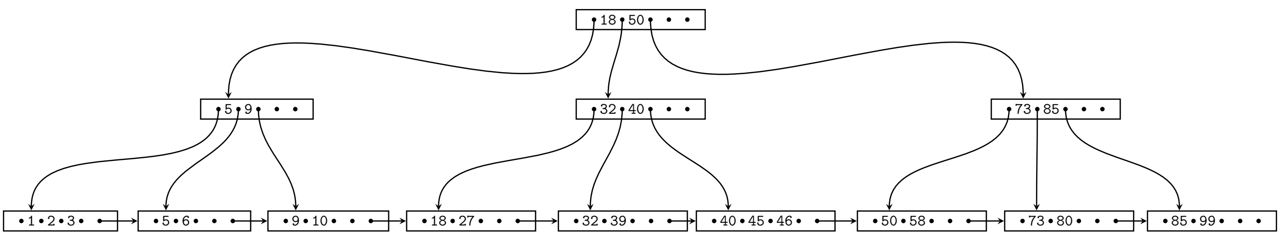
insert(3)



delete(8)

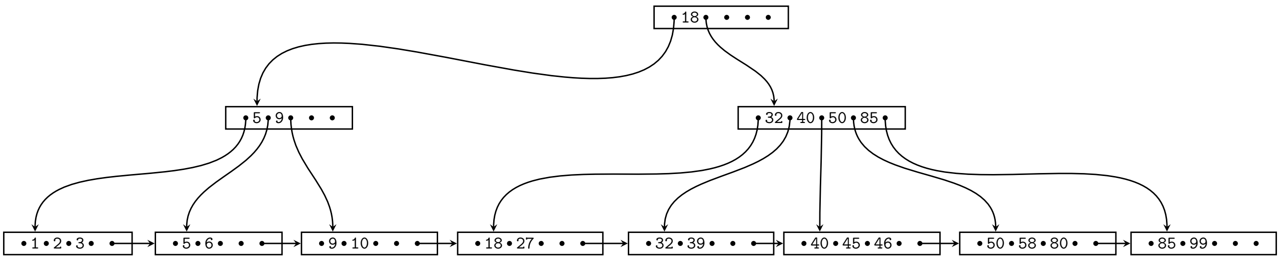


insert(46)

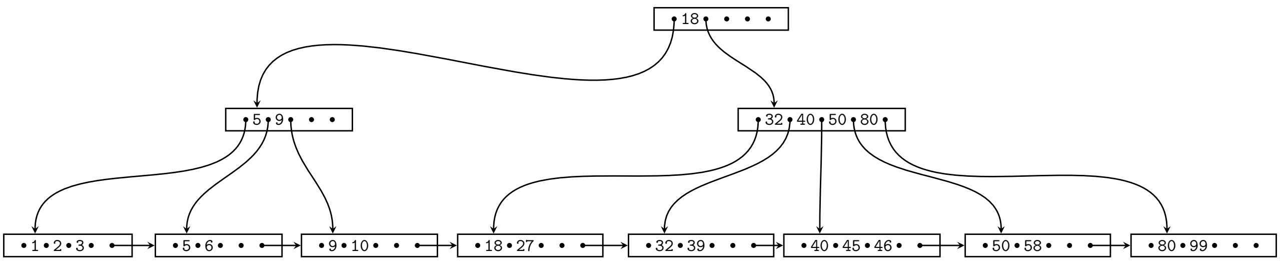


Solution

delete(73)



delete(85)



Question 3.2 B^+-tree Storage structure of records (10 Points)

Consider a `B`$^+$`-tree` with `10` pointers per `block` and depth `5`:

1. If our minimum `node` fill factor is `5` record pointers or `non-leaf` child pointers, what is the minimum number of record pointers the tree can contain?

Solution

If filled to the minimum five pointers:
the first four levels give us $5 \times 5 \times 5 \times 5 = 625$ leaf nodes;
If leaf nodes are half full of record pointers, they contain 5 records each.
Therefore, $625 \times 5 = 3125$ record pointers.

2. What is the maximum number of record pointers the tree can contain? Recall that each `leaf node` has a `next-leaf` pointer, and, for this question, `next-leaf` pointers count towards the `10` pointers per `block` limit.

Solution

If full, the first four levels give us $10 \times 10 \times 10 \times 10 = 10000$ leaf nodes;
To maximize record pointers, we should fill leaf records with the maximum $10 - 1 = 9$ reco
Therefore, $10000 \times 9 = 90000$ record pointers

This page left blank intentionally. There are no more questions.