Fall B561 Assignment 6

Name: Sricharraan Ramaswamy
Username: sriramas
UID: 2000855651

Date: 11th November, 2021

Collabrated with ATHULYA ANAND, SRI VARSHA CHELLAPILLA, SIDDHANT MESHRAM

Problem 3
(a) What are your observations about the query plans for the scanning and sorting of such differently sized bags S?

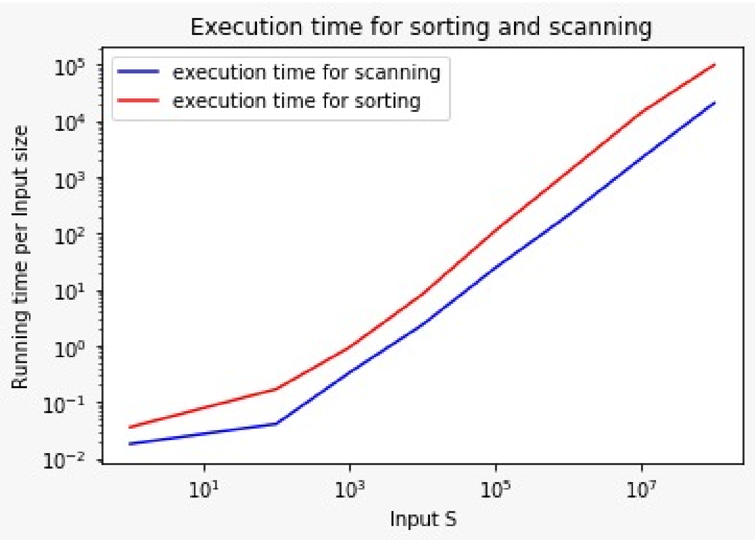
Solution 3(a).

size n of relation S	avg execution time to Scan S	avg execution time to sort S(in ms)
10	0.018	0.036
10^2	0.041	0.170
10^3	0.333	0.935
10^4	2.361	8.224
10^5	24.304	111.439
10^6	209.743	1246.618
10^7	2165.422	14086.454
10^8	20852.735	99516.074

.

Problem 3
(b) What do you observe about the execution time to sort S as a function of n?

Solution 3(b).



Time Complexity: O(n) After running thorough analysis on scanning and sorting from the above graph plotted from the data in question 3(a), we can clearly see that, the time complexity of the sorting function in terms of n is O(n).

.

Problem 3
Does this conform with the formal time complexity of (external) sort- ing? Explain.

Solution Ans. The quicksort algorithm in postgresql takes O(n) time to run. In addition to this, from the above graph and data collected in question 3(a), we can infer that the time complexity of the same is O(n). Thus, we can confirm with the formal time complexity of (external) sorting.

.

Problem 3
(d) It is possible to set the working memory of PostgreSQL using the set work mem command. For example, set work mem = '16MB' sets the working memory to 16MB.8 The smallest possible working memory in postgreSQL is 64kB and the largest depends on you computer memory size. But you can try for example 1GB. Repeat question 3a for memory sizes 64kB and 1GB and report your observations.

Solution 3(d) For Working Memory = 64 KB

size n of relation S	avg execution time to Scan S	avg execution time to sort S(in ms)
10	0.015	0.050
10^2	0.074	0.150
10^3	0.614	0.773
10^4	2.363	9.030
10^5	20.680	125.617
10^6	203.353	1335.010
10^7	2070.674	14261.966
10^8	20817.603	94961.331

.

Solution 3(d) For Working Memory = 1 GB

size n of relation S	avg execution time to Scan S	avg execution time to sort S(in ms)
10	0.015	0.041
10^2	0.047	0.084
10^3	0.245	0.791
10^4	2.601	10.055
10^5	22.741	129.439
10^6	204.97	1240.127
10^7	2058.261	16753.08
10^8	21099.832	94676.565

.

Problem 3
e) What are your observations about the query plans and execution times to create indexedS and the execution times for sorting the differently sized bags indexedS? Compare your answer with those for the above sorting problems.

Solution 3(e)

size n of relation s	avg exe time to create index indexedS	avg exe time-range query(in ms)
10	0.409	0.032
10^2	0.801	0.061
10^3	4.197	1.165
10^4	73.196	3.401
10^5	464.066	33.954
10^6	4000	302.631
10^7	240000	3295

We can clearly see that, the time taken to create an index is more than avg execution time for running the range query and we conclude that for smaller sizes of input s, the postgres sorts using quick sort method whereas for larger size merge sort.

.

Problem 7
(a) Specify (in ms) the minimum time to determine if there is a record with key k in the B+-tree. (In this problem, you can not assume that a key value that appears in an non-leaf node has a corresponding record with that key value.)

Solution 7.(a)
Block size = 4096 bytes
Block address size = 9 bytes
Block access time (I/O operation) = 10 ms
Record size = 150 bytes
Record primary key size = 8 bytes

7(a).
Ans. For calculating the minimum time to determine if there is a record with key k in the $B^+ tree$,
$n \leq 240$
$\therefore n = 241$

$$\text{Minimum Time} = (\lceil log_{241}(10^9) \rceil + 1) * 10 = (4 + 1) * 10 = 50ms$$

.

Problem 7
(b) Specify (in ms) the maximum time to insert a record with key k in the B+-tree assuming that this record was not already in the data file.

Solution 7(b).
Ans. For calculating the maximum time,

The max time will be taken when the branching factor is min i.e., 2.

$\therefore \lceil \frac{240}{2} \rceil + 1 = 121$

Therefore, the height will be, $\lceil log_{121}(\frac{10^8}{2}) \rceil = 4$

In order to access record, we need additional block.

Total Time = (4 + 1 + 1) * 10 = 60 ms

.

Problem 7
(c) How large must main memory be to hold the first two levels of the B+-tree? How about the first three levels?

Solution 7(c).
In a Large Main memory,
Two levels = 1 + 240 = 241 blocks
Each block has 241 pointers 240 keys.

Each block size = 241(block size) + block size

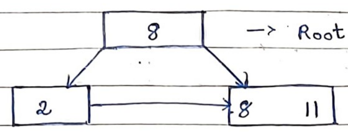
For height 1, 241 * 4096 + 4096 = 991232 bytes $\approx 1MB$
For height 2, 241 * 241 * 4096 + 4096 = 237903872 bytes $\approx 238MB$
.

Problem 8
(a) Show the contents of your B+-tree after inserting records with keys 4, 10, 12, 1, 7, and 5 in that order. Strategy for splitting leaf nodes: when a leaf node n needs to be split into two nodes n1 and n2 (where n1 will point to n2), then use the rule that an even number of keys in n is moved into n1 and an odd number of keys is moved into n2. So if n becomes conceptually k1—k2—k3 then n1 should be k1—k2 and n2 should be k3— and n1 → n2.
(b) Starting from your answer in question 8a, show the contents of your B+-tree after deleting records with keys 12, 2, and 11, in that order.
(c) Starting from your answer in question 8b, show the contents of your B+-tree after deleting records with keys 5, 1, and 4, in that order.

Solution

classmate
Date __/__/__

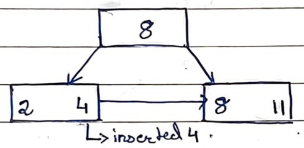
8 In the below given B^+ tree of order 2, with key 2, 8, 11



8a Inserting records in the B^+ tree with keys 4, 10, 12, 1, 7 and 5

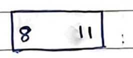
→ Inserting 4

- As 4 is less than 8 (root node)

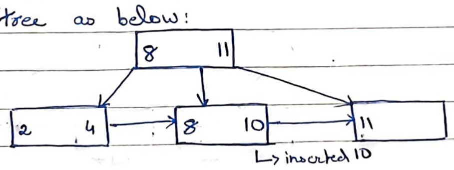


→ Inserting 10

- The root node is smaller than 10 and given the rule that z_1 should be $\boxed{k_1}\boxed{k_2}$ and z_2 should be $\boxed{k_3}\boxed{\ }$.
- So, we conclude that, a new block creation and insertion of new root as 11, so the new root node will be



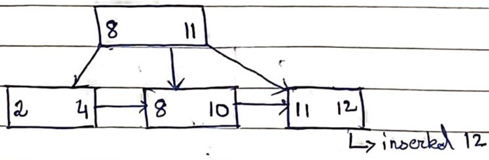
- The B^+ tree as below:



→ Inserting 12

- The number 12 is greater than 11, we insert 12 in the block with empty space.

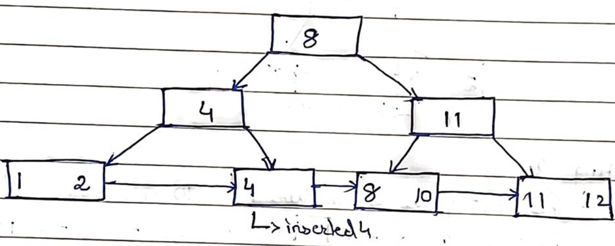
Solution



→ Inserting 1

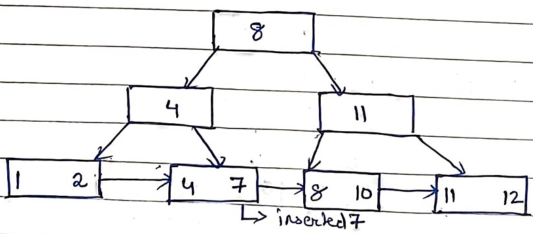
– During the insertion of 1, the creation of 2 non leaf nodes 4 and 11 accompied by the root node 8 as 8 is greater than 4 and smaller than 11

– The tree would look as below.



→ Inserting 7

– As 4 is smaller than 7 and 11 is greater than 11



.

Solution

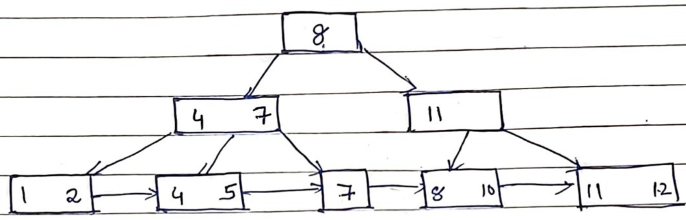
.

Solution

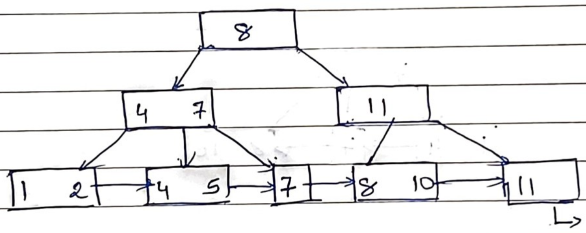
.

→ Inserting 5:

- As 4 is smaller than 5, we create a new block & 7 is added with 4 the non-leaf node.

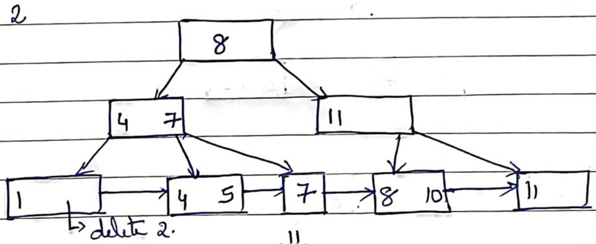


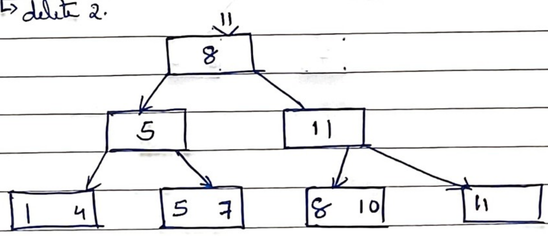
8b) Delete 12



↳ Deleted 12

→ Delete 2

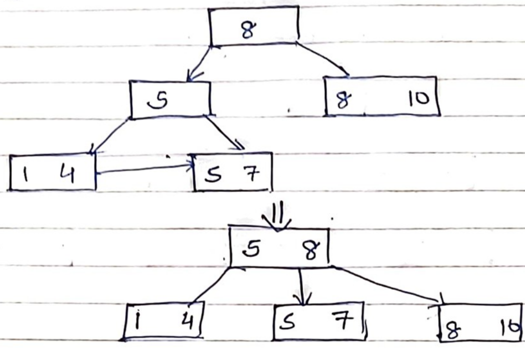




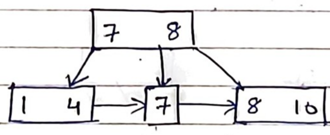
classmate

Date ___/___/___

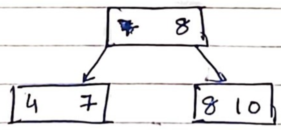
→ Deleting 11



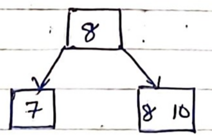
8c Deleting 5:



→ Deleting 1:



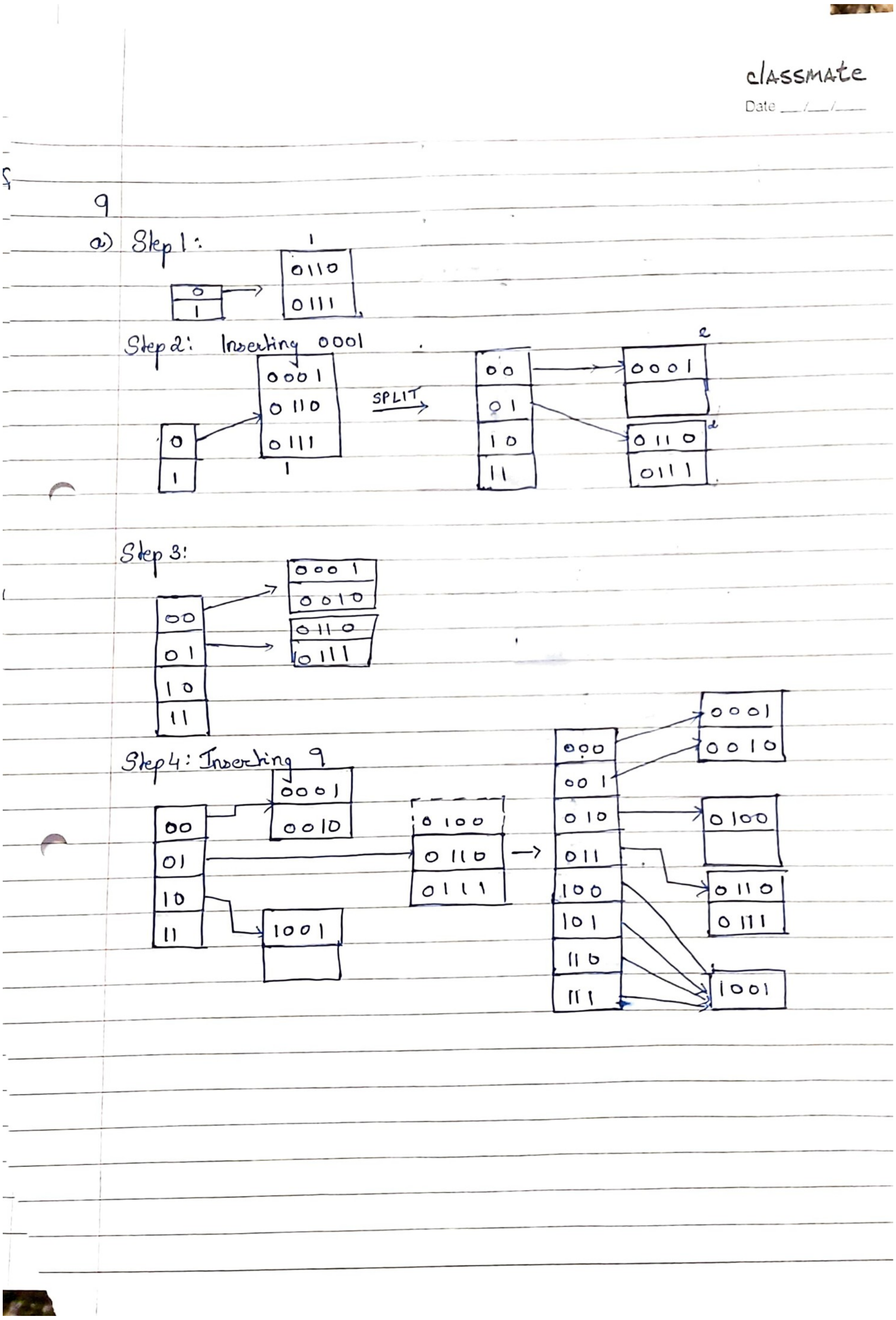
→ Deleting 4:



Problem 9

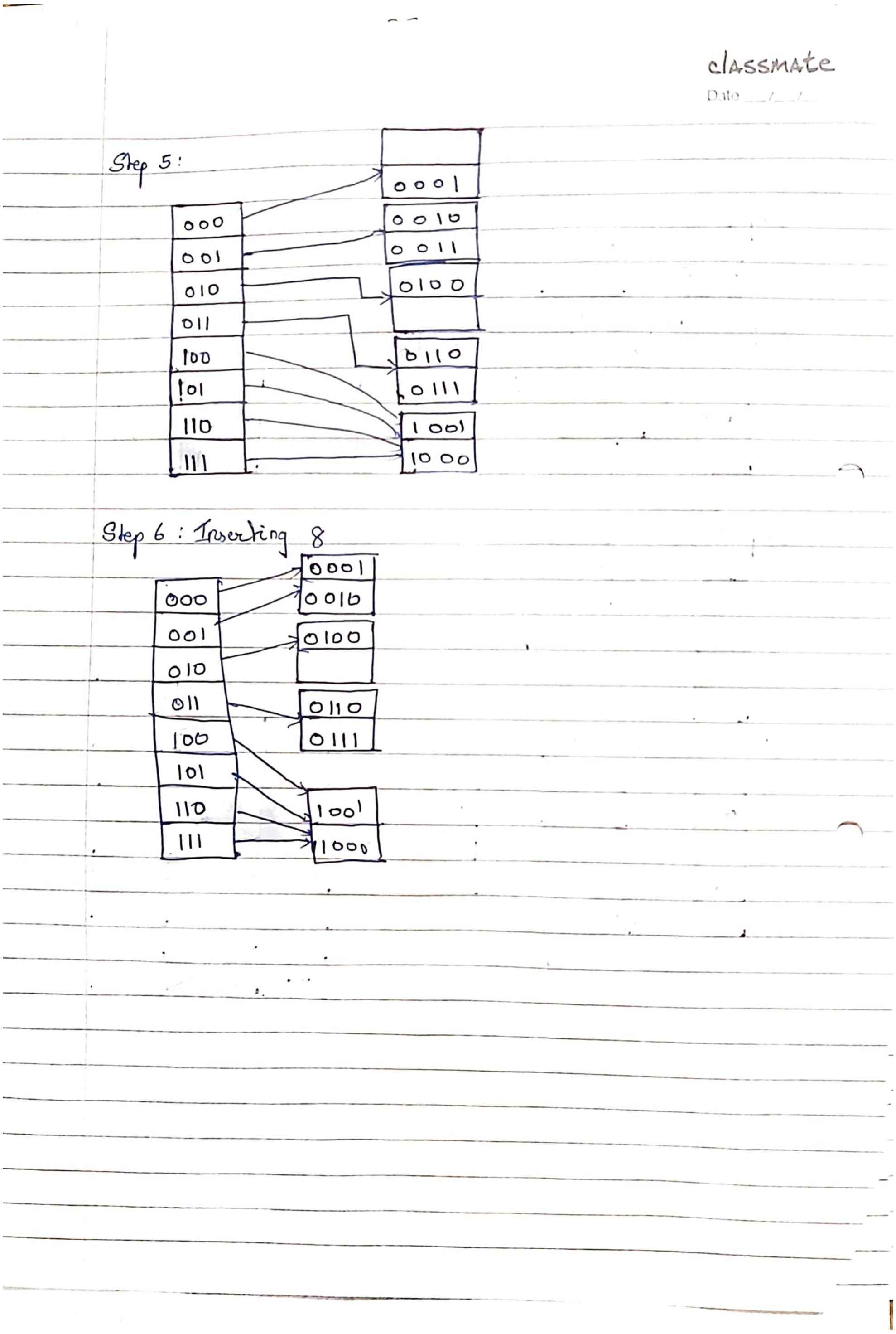
(a) Show the state of the hash data structure after each of the following insert sequences:12 i. records with keys 6 and 7. ii. records with keys 1 and 2. iii. records with keys 9 and 4. iv. records with keys 8 and 3.
(b) Starting from the answer you state of the hash data structure after each of the following delete sequences: i. records with keys 9 and 6. ii. records with keys 4 and 1. iii. records with keys 2 and 8.

Solution



.

Solution

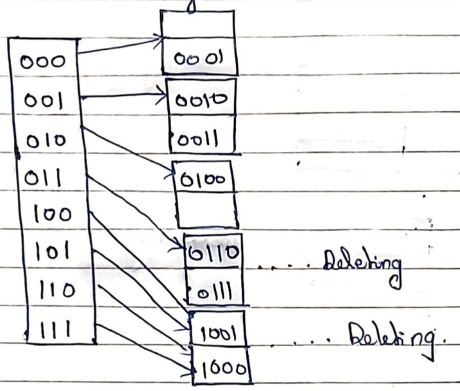


.

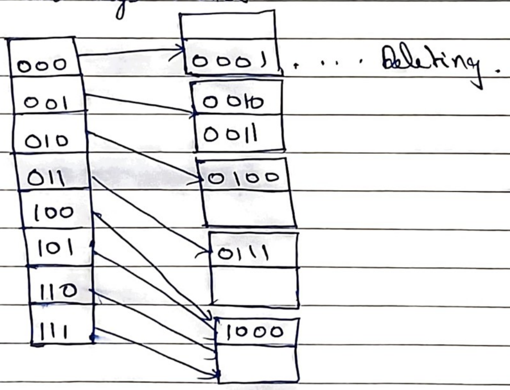
Solution

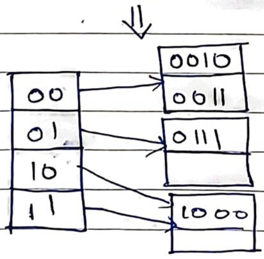
Solution

Q9 b) (i) Records with keys 9 and 6.



(ii) Records with keys 4 and 1

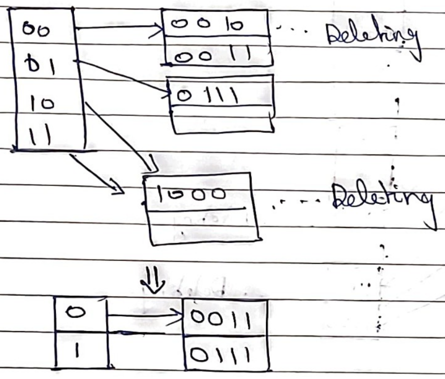




.

classmate
Date ___/___/____

(iii) Records with Keys 2 and 1



Problem 10

Create an appropriate index on the worksFor relation that speedups the lookup query select pid from worksFor where cname = c; Here c is a company name.
Illustrate this speedup by finding the execution times for this query for various sizes of the worksFor relation.

Solution

10.

Size	Without using Indexing(in ms)	Using index with B Tree	Using index with Hash
190	0.075	0.074	0.101
1900	0.794	0.194	0.373
19000	7.098	1.463	1.597

We can clearly see that, the indexing method for larger value of input takes less time as compared to that of the non indexing method. Both binary tree and hash indexing method takes almost same amount of time.

Problem 11

11. • Create an appropriate index on the worksFor relation that speedups the range query select pid, cname from worksFor where s1 ¡= salary and salary ¡= s2;
Here s1 and s2 are two salaries with s1 ¡ s2. Illustrate this speedup by finding the execution times for this query for various sizes of the worksFor relation.

Solution

11.

Size	Without using Indexing(in ms)	Using index with B Tree	Using index with Hash
190	0.110	0.205	0.130
1900	1.080	3.619	3.54
19000	10.396	6.579	8.473

We can clearly see that, the indexing method for larger value of input takes less time compared to the non indexing method. Both binary tree and hash indexing method takes almost same amount of time.

Problem 12

12. • Create indexes on the worksFor relation that speedup the multiple con- ditions query select pid from worksFor where salary = s and cname = c;
Here s is a salary and c is a company name. Illustrate this speedup by finding the execution times for this query for various sizes of the worksFor relation.

Solution

12.

Size	Without using Indexing(in ms)	Using index with B Tree	Using index with Hash
190	0.110	0.102	-
1900	0.580	0.155	-
19000	6.990	1.185	-

We can clearly see that, the indexing method for larger value of input takes less time compared to the non indexing method. Binary tree indexing method takes almost same amount of time. Since there is no multi-columns hashing allowed, hashing cannot be performed.

Problem 13

13. • Create indexes on the appropriate relations that speedup the semi-join [anti semi-join] query select pid, pname from Person where pid [not] in (select pid from worksFor where cname = c);

Solution

13. IN

Size	Without using Indexing(in ms)	Using index with B Tree	Using index with Hash
190	0.359	0.291	0.28
1900	1.6	1.245	1.268
19000	12.949	6.358	6.399

NOT IN

Size	Without using Indexing(in ms)	Using index with B Tree	Using index with Hash
190	0.428	0.2	0.25
1900	2.6	1.743	1.662
19000	13.444	7.256	7.427

We can clearly see that, the indexing method for larger value of input takes less time compared to the non indexing method. Both binary tree and hash indexing method takes almost same amount of time.