Emirhan UZUN
171044019

# CSE 414 DATABASE HOMEWORK #4

1- The advantages of $B^+$ tree for indexing are these:
Since indexing is similar to binary search, even if file grows, the performance does not change so much. It just needs small and little changes. And you don't need change or reorganize the all data. Also all nodes has same distance from root.
The disadvantages of $B^+$ tree for indexing are these:
The leaf and non-leaf nodes are different size. So may be the datas get overflow and it is complicated. And the periodically the reorganization of file is required.

2- The advantages of hashing for indexing are these:
If our file is so big, hashing provides better performance. Because for instance if we search an ID, we don't need search logn times. With hashing it is completed in constant times. Hash used decreased storage overhead since there are no primary index subtables. Also it doesn't need reorganize.
The disadvantages of hashing for indexing are these:
The hash indexes cannot be queried with partial index keys. And the hash index can't be used avoid sorting operations on data. Also for example "select * from A, B where A.C = 58 and A.C < 0,5*B" query can't be used. So $B^+$ tree are better that kind of range queries

3- Because if we create indices on every attribute, it takes much time. Since every index means additional information, it takes CPU time and also disk input - output overhead in insertion and deletion. The other reason is, now non-primary keys may changed. So if we create indices in every attribute, it has to be changed on every updates. Extra informations means extra additional storage.

Emirhan UZUN
171044019

4- Actually in general it's not possible. Because in a relation, the tuples have to be stored in different order to have same value stored together. It could be (two clustering indices on same relation for different search keys) by storing and duplicating but it is not efficient.

5-a) Now, for insertions, if the node holds 2 page, then it may need split and writing to new page. If we have nr entries, it takes nr x 2 times random accesses. And also it takes nr + 2(nr/f) page writes. Because as I said, it may need split and writing again. But we have another situation like this. If the each leaf has half filled, so split number is going to be twice (nr/f)
So the formula is $2 \times (nr/f)/f$

5-b) Takes maximum 2x nr random disk accesses
$= 2 \times 10^7$
And disk access takes 10 ms. Multiply them
$= 2 \times 10^7 \times 10 = 2 \times 10^8$ ms

6-a) Now for worst-case, it needs traverse of height h. So for every record it needs 1. So for just single record cost is h. So for $n_1$ records, the total cost $n_1 \times h$

6-b) Actually the matching tuples are same for $n_1$ and $n_2$. Therefore the answer is again $(n_1 \times h)$

Emirhan UZUN
171044019

7-a)
```
define trigger insertByAccount
after insert on account
referencing new table as newInserted for each statement
insert into branch_cust
select branch_name, customer_name
from newInserted, depositor
where newInserted.account_number = depositor.account_number
```

---

```
define trigger insertByDepositor
after insert on depositor
referencing new table as newInserted for each statement
insert into branch_cust
select branch_name, customer_name
from newInserted, account
where newInserted.account_number = account.account_number
```

7-b)
```
create trigger deleteTrig after delete on account
referencing old row as rowfromold
for each row
delete from depositor
where depositor.customer_name not in (select customer_name from
depositor where account_number <> rowfromold.account_number)
end
```

Emirhan UZUN
171044019

8) NoSQL database systems use low cost commercial processors with seperate RAM and disk. They are cost effective and support flexible schema. They have hipher scalability. That means if you add more processors and you'll get increasing consistent in performance. It runs many different processors

Difference Between SQL and NoSQL

SQL database use structured query language but NoSQL databases have flexible dynamic schemas. NoSQL are non-relational models but SQL are relational. SQL databases are table based models. NoSQL databases are key-value based models. Actually NoSQL also graph or document based models. NoSQL databases are better for data like JSON but SQL databases are better for multi-row transactions.