Databases II
2019-10-24

1. Given a relation R and dense index I1 and sparse index I2 on it with the following parameters:
T(R) = 10000, bf(R) = 20, bf(I1) = 100, bf(I2) = 100
Calculate the following:
B(R) = ?
B(I1) = ?
B(I2) = ?

B(R) = T(R)/bf(R) = 10000/20 = 500
B(I1) = T(I1)/bf(I1) = [I1 is dense, therefore T(I1)=T(R)] = 10000/100 = 100
B(I2) = T(I2)/bf(I2) = [I2 is sparse, therefore T(I2)=B(R)] = 500/100 = 5

2. Using the relation and indexes from the previous exercise let's count B(R), B(I1), and B(I2) if 20% of the blocks should be left free.

Differences:
bf(R) = 0.8 * 20 = 16
bf(I1) = 0.8 * 100 = 80

Therefore:
B(R) = T(R)/bf(R) = 10000/16 = 625
B(I1) = T(I1)/bf(I1) = 10000/80 = 125
B(I2) = T(I2)/bf(I2) = 500/80 = 6.25 ~ 7

3. T(R) = 1000000, and bf(R) = 20, we want to build a B+ tree index on a key column (A) for which bf(I) = 50.
Give the following:
B(I) = ? (help: compute the index blocks level by level starting with the leaf)

What is the cost (in block reads) of an A = c type search (worst case) if
a) the table is stored unordered and we don't use index;
b) the table is stored ordered and we don't use index;
c) we use the above B+ tree index.

Solution: First, it is important to note that on the leaf-node level a B+ tree is a dense index, therefore;

$B(I_{leaves})$=T(R)/bf(I)=20000

The next step is to have a pointer to those 20000 blocks containing pointers to the rows. Realize that since we need pointers to blocks, we are essentially implementing a sparse index from now on! As the blocking factor is 50, to create pointers to 20000 blocks, the next level can be stored on:

$B(I_{leaves-1})=B(I_{leaves})/bf(I)=400$

To create sparse index for these 400 blocks:

$B(I_{leaves-2})=B(I_{leaves-1})/bf(I)=8$

And finally to create sparse index for these 8 blocks:

$B(I_{leaves-3})=B(I_{leaves-2})/bf(I)=1$

Note: Although the blocking factor is 50, we only have to store 8 keys now. Still, to do that, we need that ‘root’ block.

So $B(I) = B(I_{leaves}) + B(I_{leaves-1}) + B(I_{leaves-2}) + B(I_{leaves-3}) = 20000+400+8+1 = 20409$

To answer the questions regarding the worst case scenario for an A=c type search:

a) We have to perform a linear search, in the worst case read every block. That is:

   $B(R) = T(R)/bf(R) = 1000000/20 = 50000$

b) We need to perform a logarithmic search, so the number of block reads:

   $\log_2(B(R)) \sim 16$

c) The number of block reads equals to the height of the B+ tree plus 1:

   $ht(I)+1 = 4 + 1 = 5$

SQL

1. Give the index organized tables of user NIKOVITS.

```
select *
from dba_tables
where owner='NIKOVITS' and iot_name is not null;
```

2. Find the table_name, index_name and overflow name (if exists) of the above tables.

```
select *
from dba_indexes
where table_owner='NIKOVITS' and INDEX_TYPE LIKE '%IOT%';
```

3. Give the names and sizes (in bytes) of the partitions of table NIKOVITS.sells

```
select partition_name, bytes
from dba_extents
where owner='NIKOVITS' and partition_name is not null
and segment_name='sells';
```

4. Create a range-partitioned table Sales which is partitioned according to quarter years based on week-number of a sale.

```
-- Range partitioned table (RANGE):
CREATE TABLE sells (sale_id   NUMBER(5),
                    sell_name    CHAR(30),
                    quantity       NUMBER(6),
                    week           INTEGER )
PARTITION BY RANGE (week)  (
   PARTITION quarter1  VALUES LESS THAN (13) SEGMENT CREATION IMMEDIATE
     STORAGE(INITIAL 8K NEXT 8K) TABLESPACE users,
   PARTITION quarter2  VALUES LESS THAN (26) SEGMENT CREATION IMMEDIATE
     STORAGE(INITIAL 8K NEXT 8K) TABLESPACE example,
   PARTITION quarter3  VALUES LESS THAN (39) SEGMENT CREATION IMMEDIATE
     STORAGE(INITIAL 8K NEXT 8K) TABLESPACE users)
   PARTITION quarter4  VALUES LESS THAN (53) SEGMENT CREATION IMMEDIATE
     STORAGE(INITIAL 8K NEXT 8K) TABLESPACE users
);
```

To select contents of a specific partition:

```
select * from rudas.sells partition(quarter1);
```

Insert the following rows, then check the partitions!

```
insert into sells values(100, 'Sport equipment', 231, 2);
insert into sells values(101, 'Office stuff', 1200, 3);
insert into sells values(102, 'Cutlery', 43, 4);
insert into sells values(103, 'PCs', 21, 6);
insert into sells values(104, 'Furniture', 31, 7);
insert into sells values(105, 'Estate', 3, 8);
insert into sells values(106, 'Services', 200, 9);
insert into sells values(107, 'Food', 300, 54); -- we cannot insert this record, 54 > 52
```