# 1 Index - Formulas and Notations

- $B$ is the disk block size given in number of bytes.
- $N$ is the number of rows for a given relation $R$.
- $l$ is the length of the row of the relation (on average) in number of bytes.
- $l_k$ length of key attribute in bytes.
- $l_A$ length of attribute A in bytes.
- $l_{bptr}$ length of disk block pointer.
- $l_{rptr}$ length of relation row/record pointer.
- $l_p$ length of primary index record.
- $l_c$ length of cluster index record.
- $l_{sk}$ length of secondary index on key record.
- $l_{snk}$ length of secondary index on non-key record.
- $S$ size of relation in bytes.
- $bfr$ disk blocking factor for relation - number of rows per disk block.
- $bfr_p$ disk blocking factor for primary index - number of primary index records per disk block.
- $bfr_c$ disk blocking factor for clustered index - number of clustered index records per disk block.
- $bfr_{sk}$ disk blocking factor for secondary index on key - number of secondary index on key records per disk block.
- $bfr_{snk}$ disk blocking factor for secondary index on non-key - number of secondary index on non-key records per disk block.
- $bfr_{rptr}$ number of record pointers that can be stored per disk block.
- $n_R$ number of disk blocks needed to store relation R.
- $n_p$ number of disk blocks needed to store primary index on relation R.
- $n_c$ number of disk blocks needed to store clustered index on attribute A of relation R.
- $n_{sk}$ number of disk blocks needed to store secondary index on key of relation R.
- $n_{snk}$ number of disk blocks needed to store secondary index on attribute A (non-key) of relation R.
- $f_a$ is the number of distinct values attribute A of relation R has.
- $c(A)_i$ is the number of rows of the relation that have $i^{th}$ value for attribute A in the relation. That is, cardinality of the result of query $\sigma_{A=i}(R)$.

- $p$ number of keys in a block for b-tree node, and $B^+$ tree non-leaf node.
- $p_{leaf}$ number of record pointers in the $B^+$ tree leaf node.

Following are some of the questions to be asked.

1. What is the size of the relation in number of disk blocks required to store it?
2. What are the sizes of different types of index and how many disk block accesses are required in worst case to execute $\sigma_{K=val}(R)$ or $\sigma_{A=i}(R)$.

**Size of Relation**

$$S = N \times l \tag{1}$$

**Blocking Factor** gives the maximum number of rows (unspanned allocation) that can fit in a disk block.

$$bfr = \lfloor \frac{B}{l} \rfloor \tag{2}$$

**Minimum number of disk blocks** $n_R$ needed to store relation R.

$$n_R = \lceil \frac{N}{bfr} \rceil \tag{3}$$

$$n_R = \lceil \frac{N}{\lfloor \frac{B}{l} \rfloor} \rceil \tag{4}$$

**Processing Selects on Keys** $\sigma_{K=val}(R)$

**File is unordered** - on average

$$t_k = \frac{n_R}{2} \tag{5}$$

block accesses.

**File is ordered on Key** - in worst case

$$t_k = \lg_2(n_R) \tag{6}$$

and one (1) in best case.

**Processing selects on non-Keys** $\sigma_{A=i}(R)$

**File is unordered** - on average

$$t_k = n_R \tag{7}$$

block accesses. This is because the last record that satisfied $A = i$ could be in the last block. This number can be slightly reduced if we know how many rows satisfy that condition.

**File is ordered on attribute A** - in worst case

$$t_k = \lg_2(n_r) + \lceil \frac{c(A)_i}{bfr} \rceil \tag{8}$$

**Primary Index** **length of primary index** record

$$l_p = l_k + l_{bptr} \tag{9}$$

**blocking factor for primary index**

$$bfr_p = \lfloor \frac{B}{l_p} \rfloor \tag{10}$$

**number of disk blocks to store primary index**

$$n_p = \lceil \frac{n_r}{bfr_p} \rceil \tag{11}$$

**Processing selects on key** Primary index is used to process selects on key attribute number of block accesses required in worst case is

$$\ln(n_p) + 1 \tag{12}$$

**Clustered Index** **length of clustered index** record (index on attribute A)

$$l_c = l_A + l_{bptr} \tag{13}$$

**blocking factor for clustered index**

$$bfr_c = \lfloor \frac{B}{l_c} \rfloor \tag{14}$$

**number of disk blocks to store clustered index**

$$n_c = \lceil \frac{f_a}{bfr_c} \rceil \tag{15}$$

**Processing selects on non-key attribute A** Clustered index is used to process selects on attribute A (retrieve all rows that satisfy $\sigma_{A=i}(R)$. Number of block access required in worst case is

$$\ln(n_c) + \lceil \frac{c(A)_i}{bfr} \rceil \tag{16}$$

**Secondary Index on Key** **length of secondary index on key** record

$$l_{sk} = l_k + l_{rptr} \tag{17}$$

**blocking factor for secondary index on key**

$$bfr_{sk} = \lfloor \frac{B}{l_{sk}} \rfloor \tag{18}$$

**number of disk blocks to store secondary index on key**

$$n_{sk} = \lceil \frac{N}{bfr_{sk}} \rceil \tag{19}$$

**Processing selects on key attribute** Secondary index is used to process selects on key. Number of block accesses required in worst case is

$$\lg_2(n_{sk}) + 1 \tag{20}$$

**Secondary Index on non-Key Attribute A** **length of secondary index on non-key Attribute A** record

$$l_{snk} = l_A + l_{rptr} \tag{21}$$

**blocking factor for secondary index on non-key Attribute A**

$$bfr_{snk} = \lfloor \frac{B}{l_{snk}} \rfloor \tag{22}$$

**number of disk blocks to store secondary index on key**

$$n_{snk} = \lceil \frac{f_a}{bfr_{snk}} \rceil \tag{23}$$

**Processing selects on non-key attribute A** Secondary index on non key attribute A is used to process selects on attribute A (retrieve all rows that satisfy $\sigma_{A=i}(R)$. Number of block accesses required in worst case is

$$\lg_2(n_{snk}) + \lceil \frac{c(A)_i}{\lfloor \frac{B}{l_{rptr}} \rfloor} \rceil + c(A)_i \tag{24}$$

**Multi-level index** Let I be any index on a relation. Let $n_I$ be the number of index blocks for I. Let I be on an attribute A (could be key, depending on type of index) of R. Note that A forms the key for this index I and I is ordered on the domain values (only index entry domain values) of A. So, primary index can be built of A for I. Such an index is is second level index on I, the second level index record consists of (A-domain-value, bptr). The blocking factor for second level index is -

$$bfr^2_{PI}(I) = \lfloor fracB(l_A + bptr) \rfloor \tag{25}$$

Note that since length of index entry from second level onwards is the same, we have -

$$bfr^2_{PI}(I) = bfr^3_{PI}(I) = \ldots = bfr^k_{PI}(I) \tag{26}$$

for any k- level index.

The number of blocks needed to store the second level index is -

$$n^2_{PI}(I) = \lceil \frac{n_I}{bfr^2_{PI}(I)} \rceil \tag{27}$$

If $n^2_{PI}(I)$ is greater than 1, we can create another index on it, and so on, till all index entries can fit one block.

The number of levels needed for this determines the level of multi-level index.

Note that number of levels is -

$$k = \lceil \lg_{bfr^2_{PI}(I)}(n_I) \rceil \tag{28}$$

**B-tree Maximum Number of keys in a disk block**

$$p \times (l_{bptr}) + (p-1) \times (l_K + l_{rptr}) \leq B \tag{29}$$

That is,

$$p \leq \frac{(B + l_K + l_{rptr})}{(l_K + l_{bptr} + l_{rptr})} \tag{30}$$

We want to keep each node of B tree at most 66% full. So, number of keys per block for B-tree is

$$p = \lceil \frac{2}{3} \times \frac{(B + l_K + l_{rptr})}{(l_K + l_{bptr} + l_{rptr})} \rceil \tag{31}$$

**Number of blocks** required for B-tree.

1. B-tree primary index

$$n_p^b = lg_p(n_R) \tag{32}$$

2. B-tree clustered index and secondary index on non key

$$n_c^b = lg_p(f_a) \tag{33}$$

3. B-tree secondary index on key

$$n_{sk}^b = lg_p(N) \tag{34}$$

**$B^+$ tree maximum number of keys in a leaf node**

$$p_{leaf} \times (l_K + l_{rptr}) + l_{bptr} \leq B \tag{35}$$

That is,

$$p_{leaf} \leq \lfloor (B - l_{bptr} l_K + l_{rptr} \rfloor \tag{36}$$

We want to keep leaf nodes 2/3 full, so

$$p_{leaf} = \frac{2}{3} \times \lfloor (B - l_{bptr} l_K + l_{rptr} \rfloor \tag{37}$$

Number of leaf nodes needed are for secondary index on key

$$n_p = \lceil \frac{N}{p_{leaf}} \rceil \tag{38}$$

$n_p$ is the number of entries needed to be stored in all internal nodes of $B^+$ tree. The number of key pointer pairs that can be stored in a block of internal nodes of $B^+$ tree is -

$$p \times l_K + (p-1) \times l_{bptr} \leq B \tag{39}$$

$$p \leq \lfloor \frac{B + l_{bptr}}{l_K + l_{bptr}} \rfloor \tag{40}$$

We want to keep internal nodes 2/3 full, so

$$p = \frac{2}{3} \times \lfloor \frac{B + l_{bptr}}{l_K + l_{bptr}} \rfloor \tag{41}$$

The number of levels for B+ tree will be

$$\lg_p n_p + 1 \tag{42}$$

Note that for clustered index and secondary index on non-key attribute A, number of entries in leaf node are $f_a$.