Assuming we have **a student file** with

10,000 records (i.e. students)

Blocking factor for the student record = 100, this means we can store 100 student records per block.

Blocking factor for (student-no+pointer) = 1000 records per block

Blocking factor for (major+pointer)= 500 records per block.

Blocking factor for (SSN+pointer) = 200 records per block

We need

10,000/100 = 100 blocks to store the student file (ours actual table)

Any query that searches the student file using the student-no will need a Log2 (100) (binary search cost assuming the student file is sorted using st-no field)

We need for a non-dense (sparse) primary index

One entry for every block 🡪 this means we need to add a record to the index for every block in the data file = 100. These will need one block to store the index.

Any query searching the file based on student-no and using the primary non-dense index will need two reads: one to get the index in RAM and the other to get the right data block from the student file which has the answer.

Assume we have 10 majors for the students to choose from.

If the student file is ordered on major, then we can create a clustering index

The clustering index will have 10 records only (because we keep one record in the index for every distinct value of the major attribute). Thus, we need one block to store the clustering index.

We can create a secondary index on the student file using SSN (note the file is not ordered on SNN). Because the secondary index is defined on a unique attribute then it must be dense. This means the secondary index will have a record for every distinct value for SNN, i.e. 10,000 values.

These will need 10000/200 = 50 blocks to store the secondary index.

The cost of retrieving a record using the secondary index is equal to using binary search on the index plus 1 (to get the data block) = Log2(50)+1. Compare this with using sequential search on the data file itself to get the data record. We need to use sequential search (best=1, average = 50, worst = 100 block accesses).

B/B+trees

To access any data block, we need

A number of disk accesses (reads) = Height of the tree + 1. In the case of B-tree, we can reach the data even earlier.