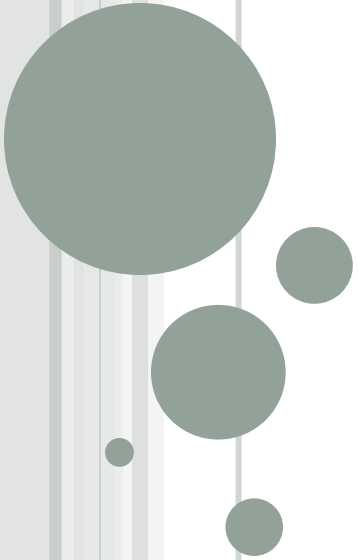


Indexing Methodology: II Discussion Week 7

Database Systems (H)
Dr Chris Anagnostopoulos

B+ Tree on Non-ordering Non-key!

- ***n* = 1000** departments; ***r* = 100,000** employees, e.g., BT Group
- Blocking factor **bfr = 50 records/block**
- DNO is *uniformly* distributed: ***d* = *r*/*n* = 100 employees/department**
- **Leaf node order: *q* = 10 DNO values/data-pointers**
- **Tree node order: *p* = 10 tree-pointers**
- **100 pointers can fit in 1 block.**

Task 1: How many leaf nodes do we need for *all* DNO values?

Task 2: Which is the B+ Tree *leaf* structure, the *whole* B+ Tree structure and finally the *expected cost* searching for the employees of Dept. 3.

B+ Tree on Non-ordering Non-key!

- ***n* = 1000** departments; ***r* = 100,000** tuples (employees)
- Blocking factor **bfr = 50 records/block**
- DNO is *uniformly* distributed: ***d* = *r*/*n* = 100 employees/department**
- **Leaf node order: *q* = 10 DNO values/data-pointers**
- **Tree node order: *p* = 10 tree-pointers**
- **100 pointers can fit in 1 block.**

The file has: ***b* = ceil(100,000/50) = 2000 blocks**

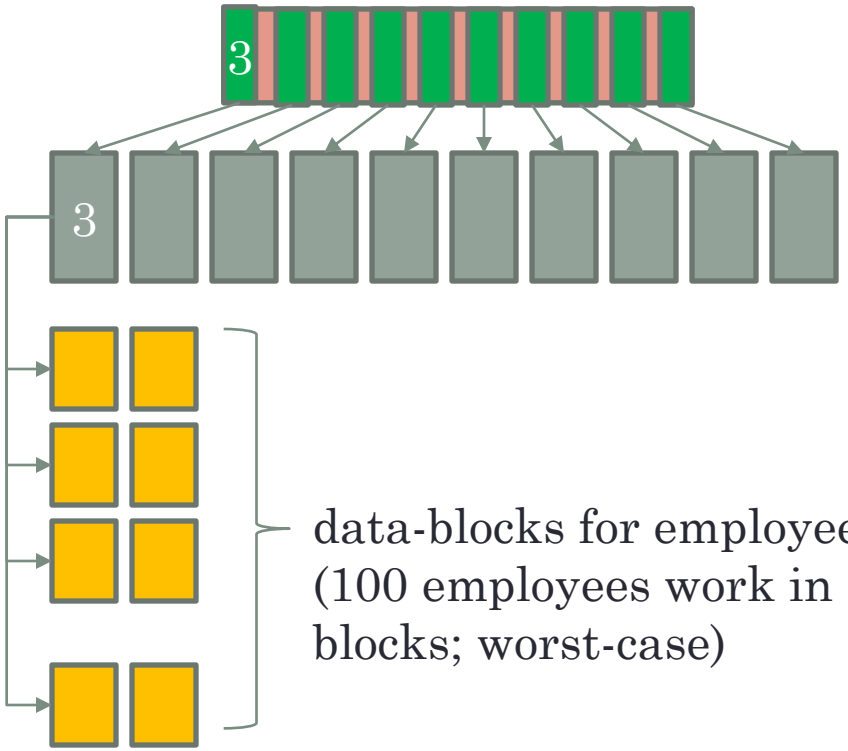
Task 1: How many leaf nodes do we need for *all* DNO values?

- *n* = 1000 departments (DNOs) can fit in ***n*/*q* = 100 leaf nodes;** each leaf node has ***q* = 10** depts.

Task 2: We have 100 leaf nodes; each one with 10 depts.

Leaf Node Structure:

- For *each* DNO value, assign a pointer to *block(s)* with *block of pointers* of those blocks with employees working at this DNO (dept.)
- We need ***d* = 100** block pointers per DNO (dept.) stored in 1 block
- We need ***m* = 10 blocks** of block of pointers (one block per DNO value), each block with 100 block pointers (100 pointers fit in one block)



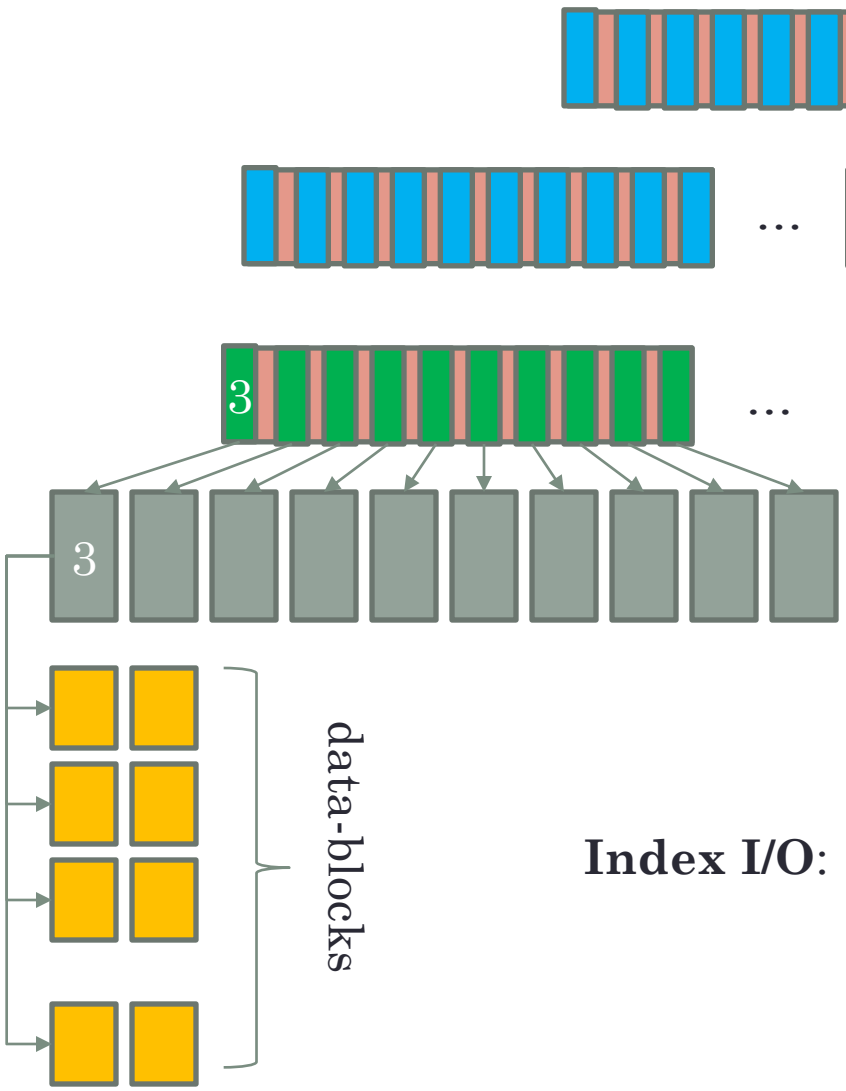
1 Leaf node (10 depts.)

10 blocks of block-pointers

100 data-blocks/dept.

data-blocks for employees in DNO=3
(100 employees work in dept. 3 residing in 100 different blocks; worst-case)

Task 2: Root: 1 node (p = 10); L1: 10 nodes; Lf = 100 nodes (level t = 3)



Root: 1 block

L1: 10 blocks

Lf: 100 blocks

UG: 100*10 = 1000 blocks

Index Size: 1111 blocks.

File Size: 2000 blocks.

Index I/O: 1 + 1 + 1 + 1 + 100 = **104 block accesses**

File I/O: **2000 block accesses**

94.8% speed-up

55% storage