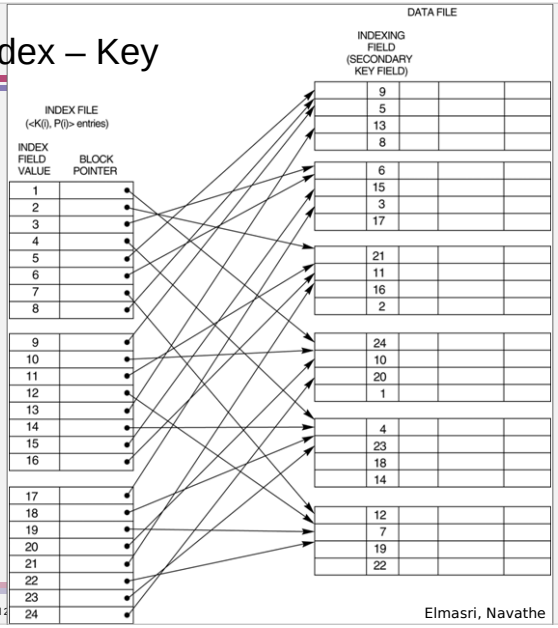
Secondary Index – Key

indexing field is a key –

one index record per data record

index record points to data block containing that record



Elmasri, Navathe

Secondary Index – Key

indexing field is a key
one index record per data record

operation	method	# blocks accessed
search on indexing field	linear search on file	max b = 3000 avg b/2 = 1500
	binary search on index + data block	ceil($\log_2(b_i)$)+1 = 9+1 = 13
range search on indexing field	linear search on file	b = 3000
	using index (binary search on index to find first, then scan index to determine data blocks for the rest of the s matches)	ceil($\log_2(b_i)$) +ceil(s/bfr_i)+s = 9+ceil(s/68)+s note that if s ≥ b (or even close to b), it is better to just scan the file

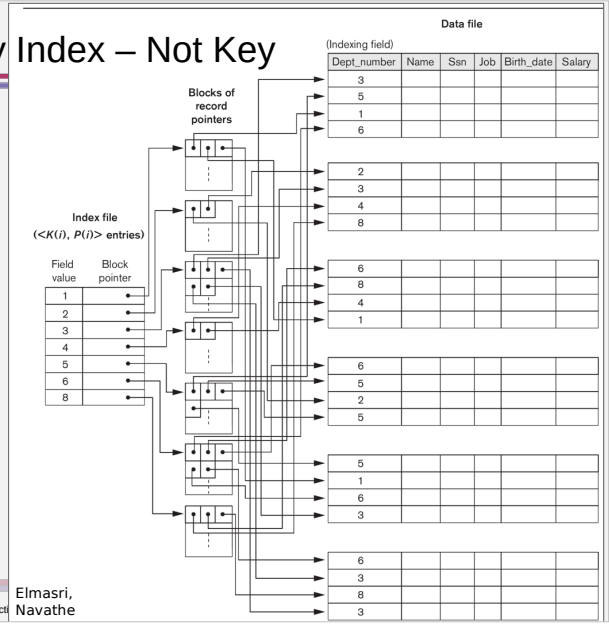
file properties			
# records	r		30,000
block size	B		1024
record length	R		100 bytes (fixed length)
blocking factor (# records per block)	bfr	bfr = floor(B/R)	10
# data blocks	b	b = ceil(r/bfr)	3000
index properties			
ordering key length	V		9 bytes
block pointer	P		6 bytes
# index records	r_i	**one per data record**	30,000
index record length	R_i	R_i = V+P	15 bytes
blocking factor for index	bfr_i	bfr_i = floor(B/R_i)	68
# index blocks	b_i	b_i = ceil(r_i/bfr_i)	442

Secondary Index – Not Key

indexing field is not a key, using indirect blocks –

one index record per distinct value of the indexing field

index record points to *indirect block(s)* containing pointers to the data blocks containing the records with that value



Elmasri, Navathe

Secondary Index – Non-Key (Indirect Blocks)

indexing field is not a key
one index record per distinct value + indirect blocks

file properties							
number of distinct values	d		3000	average # of records per distinct value	r_d	r_d = r/d	30000/3000 = 10

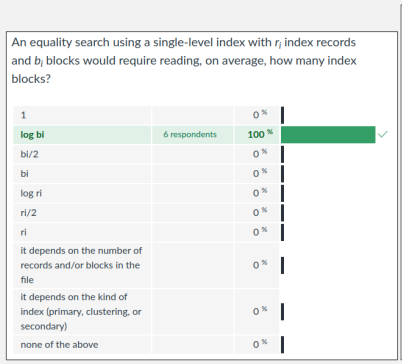
index properties			
ordering key length	V		9 bytes
block pointer	P		6 bytes
# index records	r_i	**one per distinct value**	3000
index record length	R_i	R_i = V+P	15 bytes
blocking factor for index	bfr_i	bfr_i = floor(B/R_i)	68
# index blocks	b_i	b_i = ceil(r_i/bfr_i)	45
blocking factor for indirect blocks	bfr_b	bfr_b = floor(B/P)	170
average # indirect blocks per index record		ceil(r_d/bfr_b)	ceil(10/170) = 1
total # indirect blocks	b_b	average # indirect blocks per index record × r_i	3000

Secondary Index – Non-Key (Indirect Blocks)

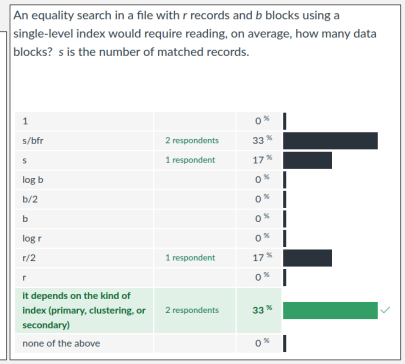
indexing field is not a key
one index record per distinct value + indirect blocks

operation	method	# blocks accessed
search on indexing field	linear search on file	b = 3000
	binary search on index + indirect block(s) + data block(s) (expect r/d records to match value)	ceil($\log_2(b_i)$) + average # indirect blocks per index record + ceil(r/d) = 6+1+10 = 17
range search on indexing field	linear search on file	b = 3000
	using index (find first using binary search on index, then scan index to get rest of the s matches, loading indirect block(s) for each, then retrieving the data block as each record is identified)	ceil($\log_2(b_i)$) + ceil($s/r_d/bfr_i$) + s/r_d × average # indirect blocks per index record + s = 6+s/680+s/10+s if s ≥ b (and even somewhat less), better off just scanning the file

d = # distinct values
s = # matches



the index records for all indexes are ordered by the indexing field
→ can be searched with binary search



how the matching data records are distributed amongst the data records depends on the ordering of the file records relative to the ordering of the index records

Multilevel Index

If there are lots of records, there could be a lot of index blocks and searching them all could take some time.

- index the index!

The first level index is one of the single-level indexes.

The second level index is a primary index on the first-level index.

- first level index is an ordered file with records containing just an ordering key value and block pointer(s) – and no duplicates

The third level index is a primary index on the second-level index...

...until the index at a particular level requires only one block.

Multilevel Index

Searching –

- read one block at each level
- read the final data block(s)

Range searching –

- search for the first value down to the first-level index
  - read one block at each level
- scan the first-level index to find all the matching index records
- read the data blocks indicated by the first-level index records

Indexing Recap

Three possible index organizations:

- primary – ordered by same field(s) as file, and field(s) are a key
- clustering – ordered by same field(s) as file, but field(s) not a key
- secondary – ordered by different field(s) than file

Performance:

- $O(\log_2 b_i + s)$ for searching on indexing field(s)
  - s = number of matching records
- primary is better than clustering is better than secondary
  - but there can be at most one primary or clustering index per file
- multilevel index reduces search time from $O(\log_2 b_i)$ to $O(\log_{bfr_i} b_i)$
- indexes are less beneficial for small tables and queries that match most of the rows