

						Data fi	ile		
Secondary	Index – N	Not Kev		(Indexing field)		0		B 1.1	
Cecentaary	index i			Dept_number	Name	Ssn	Job	Birth_date	Salary
		Blocks of	-	5					
		record		1					
indexing field		pointers	┢►	6					
indexing neid	Г								
is not a key,			╟►	2					
using indirect			⊬►	3					
blacks	Index file	→⊡	⊢►	4					
DIOCKS –	$(\langle K(i), P(i) \rangle$ entries)			8					
one index record	Field Block			6					
ner distinct value			⊢►	8					
of the indexing	2 •			4					
field	3 •	┘┍╼┝╹╍╤╤┼╢╫╧║║							
neiu	4 •								
index record	5 •	┐└╧╧┙╢╢┟╫╁	₽►	6					
points to indirect	8	╵╘╸╹╹╹╸╴┼┼╢╵║╵		5					
block(s) containing				5					
pointors to the									
data blacks									
uala DIOCKS		••	-	5					
containing the			-	1					
records with that				3					
value			-						
				6			-		
	Flmasri			8			-		
CPSC 343: Database Theory and Pract	Navathe		-	3			-		

Seco	ndary I	ndex – Ke	indexi ey one in	ng fi dex	eld is a key record per data	record				
enevetien	un ethe et	# blocks	file properties							
operation	method	accessed	# records	r		30,000				
	linear search on	max b = 3000	block size	В		1024				
search on	file	avy 1/2 – 1500	record length	R		100 bytes (fixed lengt				
field	search on	$\operatorname{ceil}(\log_2(b_i))+1$	blocking factor (# records per block)	bfr	bfr = floor(B/R)	10				
da	data block	- 9+1 - 13	# data blocks	b	b = ceil(r/bfr)	3000				
	linear		index properties							
	search on file	b = 3000	ordering key length	V		9 bytes				
range	using index	$ceil(log_2(b_i))$	block pointer	Р		6 bytes				
indexing field index to fin	on index to find first, then scan index to	$+ceil(s/bfr_{i})+s =$ 9+ceil(s/68)+s	# index records	r _i	one per data record	30,000				
	determine data blocks for the rest of the s	note that if $s \ge b$ (or even close to b), it is better to just scan	index record length	R	R _i = V+P	15 bytes				
	matcnes)	the file	blocking factor for index	bfr _i	$bfr_i = floor(B/R_i)$	68				
CPSC 343: Databa	se Theory and Practice •	Fall 2024	# index blocks	b	$b_i = ceil(r_i/bfr_i)$	442				

Secondar	y Index –	Non-Key	/ (Ir	ndirect	Blocks)
----------	-----------	---------	-------	---------	---------

				in oi	idex ne ir	ing field is no idex record p	t a l er d	key istinct v	alu	e + indire	t blocks
file properties											
number of distinct values	d		3000	ave per	erage r disti	tinct value $r_{d} = r/d = r/d$			300 = 1	000/3000 .0	
index properti	ies										
ordering key le	ngtł	ı			V					9 bytes	
block pointer					Р					6 bytes	
# index records					r _i	one per distinct value				3000	
index record length				R,	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			S	$\mathrm{bfr}_{\mathrm{b}}$	$bfr_{b} = floor(B/P)$				170		
average # indirect blocks per index record					ceil(r _d /bfr _b)				ceil(10/170) = 1		
total # indirect blocks				b,	average # indirect blocks per index record × $r_{\!_i}$			er	3000		
CPSC 343: Database Th	eory a	nd Practice	• Fall 2024								

Second		
	indexing field is not a one index record per c	key listinct value + indirect blocks
operation	method	# blocks accessed
	linear search on file	b = 3000
search on indexing field	binary search on index + indirect block(s) + data block(s) (expect r/d records to match value)	$\begin{aligned} & \text{ceil}(\log_2(b_j)) + \text{average } \# \text{ indirect} \\ & \text{blocks per index record } + \text{ceil}(r/d) \\ & = 6+1+10 = 17 \end{aligned}$
	linear search on file	b = 3000
range search on indexing field	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)	$ \begin{array}{l} \mbox{ceil}(\log_2(b_j)) + \mbox{ceil}(s/r_d/bfr_j) + s/r_d \times \\ \mbox{average $\#$ indirect blocks per index} \\ \mbox{record } + s = 6 + s/680 + s/10 + s \\ \mbox{if $s \ge b$ (and even somewhat less), better off} \\ \mbox{just scanning the file} \end{array} $
		d = # distinct values s = # matches
CPSC 343: Database Th	eory and Practice • Fall 2024	47

Secondary Index Non-Key (Indirect Blocks)

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!

CPSC 343: Database Theory and Practice • Fall 2024

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.

n equality search using a nd b _i blocks would requir locks?	single-level ind re reading, on av	lex with r _i index records verage, how many index	An equality search in a file single-level index would re blocks? <i>s</i> is the number of	with r records a equire reading, o matched recor	and <i>b</i> blocks using a on average, how many da ds.
1		0 %	1		0%
log bi	6 respondents	100 %	s/bfr	2 respondents	33 *
bi/2		0 %	s	1 respondent	17 %
bi		0 %	log b		0 %
log ri		0 %	b/2		0 %
ri/2		0 %	b		0 %
ri		0 %	log r		0 %
it depends on the number of		0.2	r/2	1 respondent	17 %
file		U III	r i i i i i i i i i i i i i i i i i i i		0 %
It depends on the kind of index (primary, clustering, or secondary)		0 [%]	it depends on the kind of index (primary, clustering, or secondary)	2 respondents	33 %
none of the above		0 %	none of the above		0 %
he index recor ordered by the → can be searc	rds for all indexing hed with	indexes are field binary search	how the matchi distributed amo depends on the records relative	ing data ongst the ordering to the o	records are data records of the file rdering of the

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(log₂b_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 $O(log_2b_i)$ to $O(log_{{}_{b}fr_i}b_i)$
- indexes are less beneficial for small tables and queries that match most of the rows

— 31