Physical File Organization

There's a big difference in search time, which underlies everything.

- hash file is best for equality search, but worst or close to worst for range search
- ordered file is good for searching (equality and range)

And –

- a file can only be ordered/hashed on one field (or group of fields)

Secondary Access Paths

A *secondary access path* is a way to access data besides just via the physical file organization.

- typically an index

Advantages:

- faster access

Disadvantages:

- take more space
- maintenance has a cost

Questions

Where are indexes stored? Individually for a user or within a database?

- indexes are associated with tables
- defined by DB admin as part of the database implementation (not by users)

Indexes

Indexes are built on a field or group of fields.

The type of index depends on

- whether the indexing field is the same as the file's ordering field
- whether the indexing field is a key

An *index record* contains a value and a pointer to the block containing the record with that value.

Index records are stored in sorted order.

Primary vs Clustering Indexes

Primary index –

- indexing field is the ordering field of the file, and *is a key*

Clustering index –

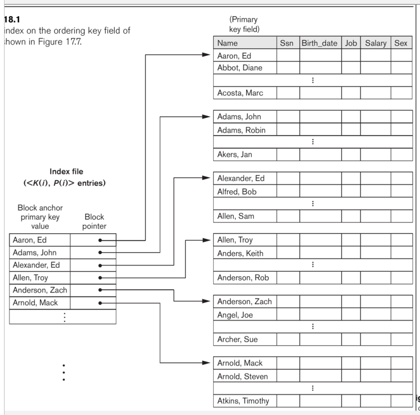
- indexing field is the ordering field of the file, but *isn't a key*

Can there be both primary and clustering indexes for the same file?

- no, at most one primary or clustering index per file

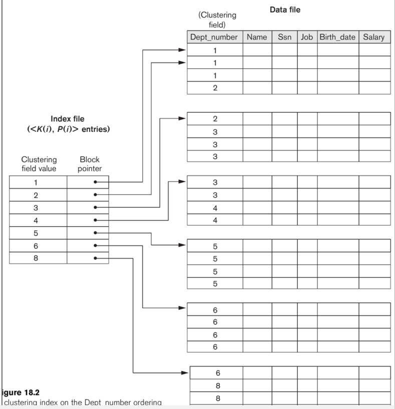
Primary vs Clustering Indexes

primary index



18.1
index on the ordering key field of
hown in Figure 17.7.

clustering index



igure 18.2
clustering index on the Dept_number ordering

one index record *per data block* of the original file

contains value of indexing field for the first record on the block (*anchor record*)

one index record *per distinct value* of the indexing field

contains pointer to first data block containing records with that value

Elmasri, Navathe

Clustering Indexes

a common implementation is to instead allocate separate blocks for each value of the indexing field in the underlying data file

this simplifies insertion and deletion in the data file, as well as updating of the index

for the index, there is still one index record per distinct value of the indexing field

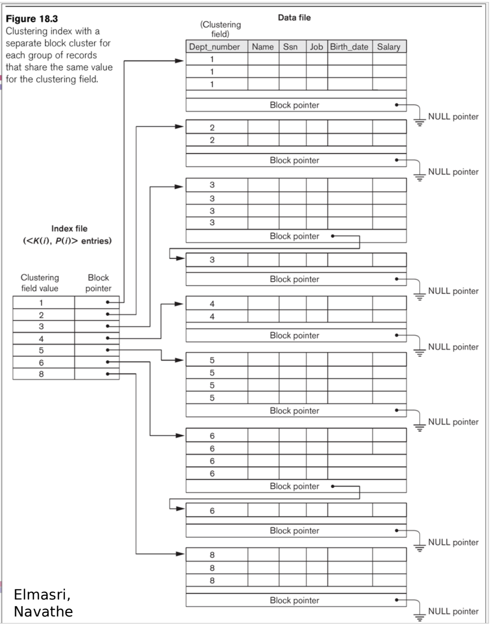


Figure 18.3
Clustering index with a separate block cluster for each group of records that share the same value for the clustering field.

Elmasri, Navathe

Primary vs Clustering Indexes

What's the point in having an index ordered by the same thing the file is already ordered by?

- index records are smaller ($bfr > bfr_i$) and there are fewer of them ($r_i < r$), so there are fewer blocks to load when searching ($b_i < b$)

Secondary Indexes

Organization.

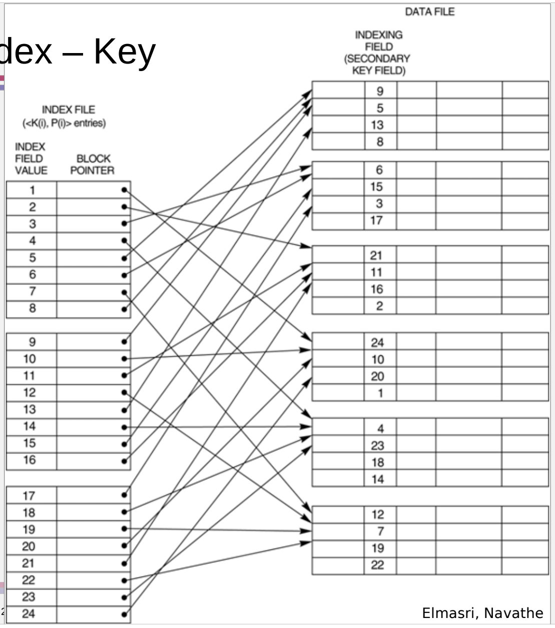
- indexing field is not the ordering field of the file
  - can have any number of secondary indexes for a file, with or without primary or clustering index
- when the indexing field is a key –
  - one index record *for each data record*
  - index record contains pointer to the block containing that record
- when the indexing field is not a key –
  - option #1 – one index record for each data record
  - option #2 – variable-length index records listing all of the data blocks for each indexing field value
  - option #3 (most common) – one index record *for each distinct indexing field value*, containing pointer to an *indirect block* containing pointers to the data records with that field value
    - takes more space but avoids variable-length records

Secondary Index – Key

indexing field is a key –

one index record per data record

index record points to data block containing that record



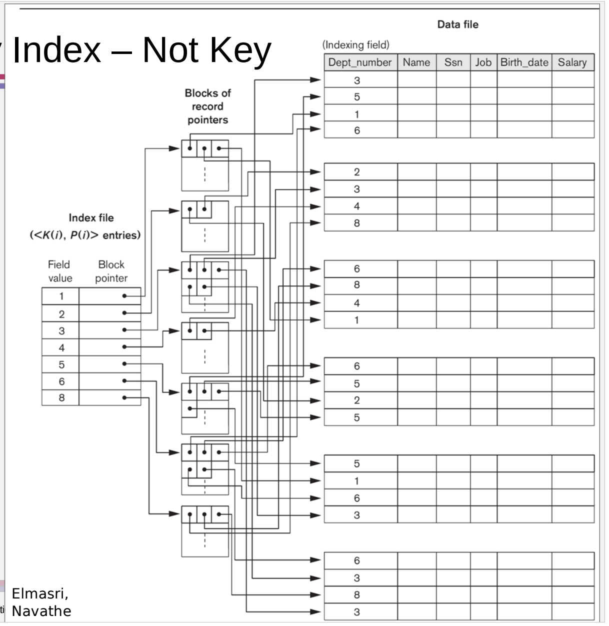
Elmasri, Navathe

Secondary Index – Not Key

indexing field is not a key, option 3 (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 Indexes

Can there be both secondary and primary indexes for the same file?

- yes

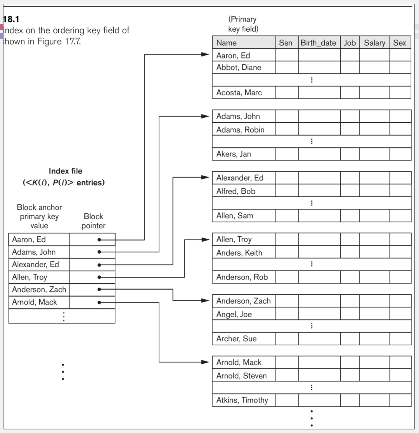
Can there be both secondary and clustering indexes for the same file?

- yes

Can there be more than one secondary index per file?

- yes

Primary Index


18.1
ndex on the ordering key field of
hown in Figure 17.7.

Block anchor primary key value	Block pointer
Aaron, Ed	●
Adams, John	●
Alexander, Ed	●
Allen, Troy	●
Anderson, Zach	●
Arnold, Mack	●
⋮	

Name	Ssn	Birth_date	Job	Salary	Sex
Aaron, Ed					
Abbot, Diane					
⋮					
Acosta, Marc					
Adams, John					
Adams, Robin					
⋮					
Akers, Jan					
Alexander, Ed					
Alfred, Bob					
⋮					
Allen, Sam					
Allen, Troy					
Anders, Keith					
⋮					
Anderson, Rob					
Anderson, Zach					
Angel, Joe					
⋮					
Archer, Sue					
Arnold, Mack					
Arnold, Steven					
⋮					
Atkins, Timothy					
⋮					

one index record *per data block* of the original file

contains value of indexing field for the first record on the block (*anchor record*)

Elmasri, Navathe

Primary Index

indexing field is a key
one index record per data block

operation	method	# blocks accessed
search on indexing field	binary search on file	ceil($\log_2$(b)) = 12
	binary search on index + data block	ceil($\log_2$(b_i))+1 = 7

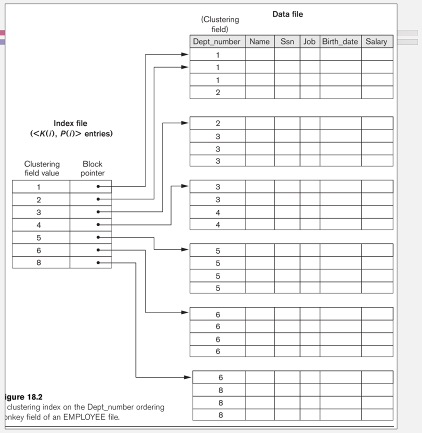
range search on indexing field	on file
	(binary search to find first, then scan to get rest of the s matches)
		ceil($\log_2$(b))
		+ceil(s/bfr)
		= 12+ceil(s/10)

	using index
	(find first using index, then scan file to get rest of the s matches)
		ceil($\log_2$(b_i))
		+ceil(s/bfr)
		= 6+ceil(s/10)

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 block = b	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

Clustering Index


igure 18.2
clustering index on the Dept_number ordering
onkey field of an EMPLOYEE file.

Clustering field value	Block pointer
1	●
2	●
3	●
4	●
5	●
6	●
8	●

Dept_number	Name	Ssn	Job	Birth_date	Salary
1					
1					
1					
2					
2					
3					
3					
3					
3					
3					
4					
4					
5					
5					
5					
5					
6					
6					
6					
6					
6					
8					
8					
8					

one index record *per distinct value* of the indexing field

contains pointer to first data block containing records with that value

Elmasri, Navathe

Clustering Index

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

operation	method	# blocks accessed

search on indexing field	binary search + scan file
	(expect r/d matches)
		ceil($\log_2$(b))
		+ceil(r/d/bfr)
		= 12+1 = 13

	binary search on index + data block(s)
	(expect r/d matches)
		ceil($\log_2$(b_i))
		+ceil(r/d/bfr)
		= 6+1 = 7

range search on indexing field	binary search + scan file
	(expect s matches)
		ceil($\log_2$(b))
		+ceil(s/bfr)
		= 12+ceil(s/10)

	binary search on index + data block(s)
	(expect s matches)
		ceil($\log_2$(b_i))
		+ceil(s/bfr)
		= 6+ceil(s/10)

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
# distinct values for index field	d		3000

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