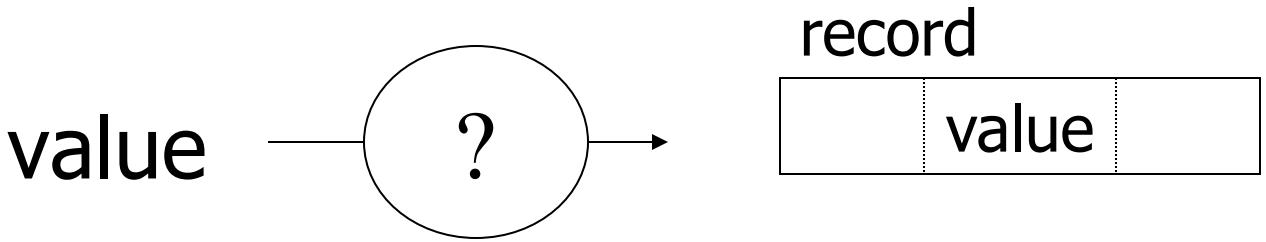
Notes: Indexing General concepts

Based on lectures by Hector Garcia-Molina

Indexing & Hashing



Topics

- Conventional indexes
- B-trees
- Hashing schemes

Sequential File

10	
20	

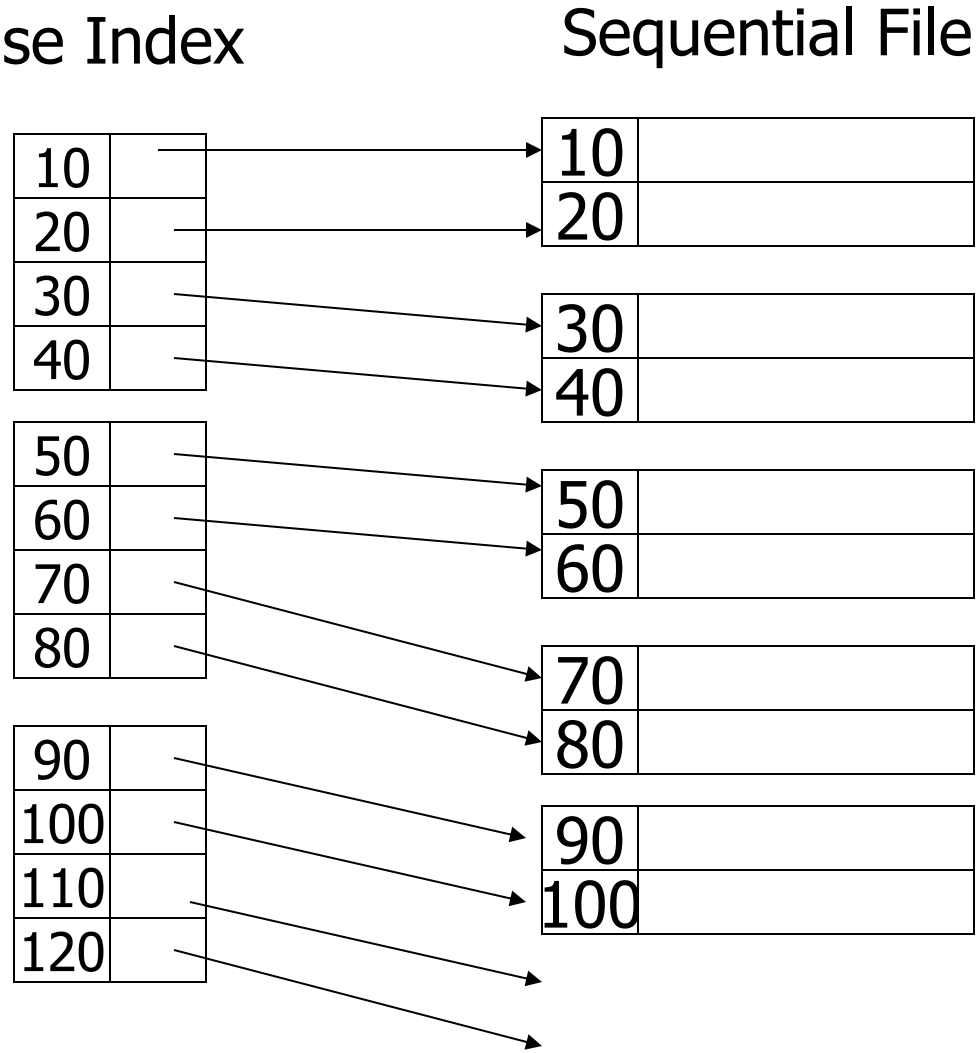
30	
40	

50	
60	

70	
80	

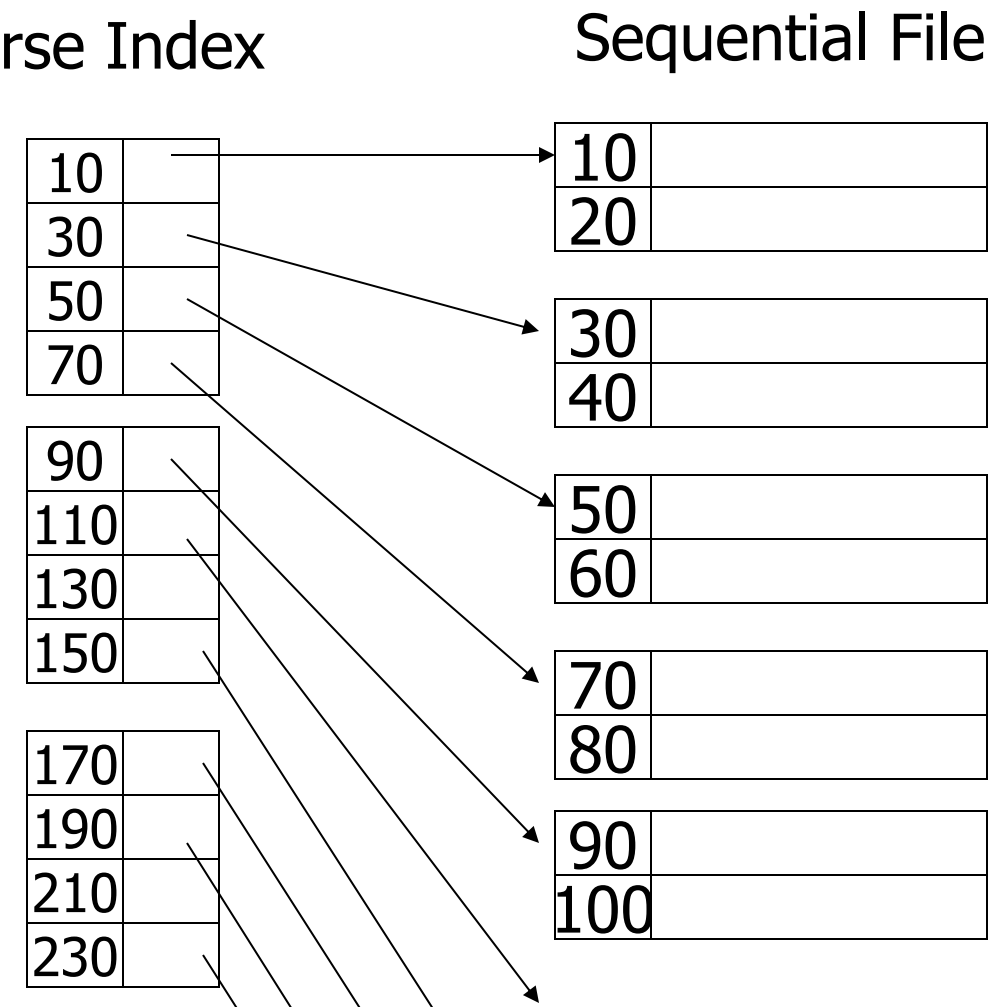
90	
100	

Given a key, a sequential file store records in contiguous block sorted on key values

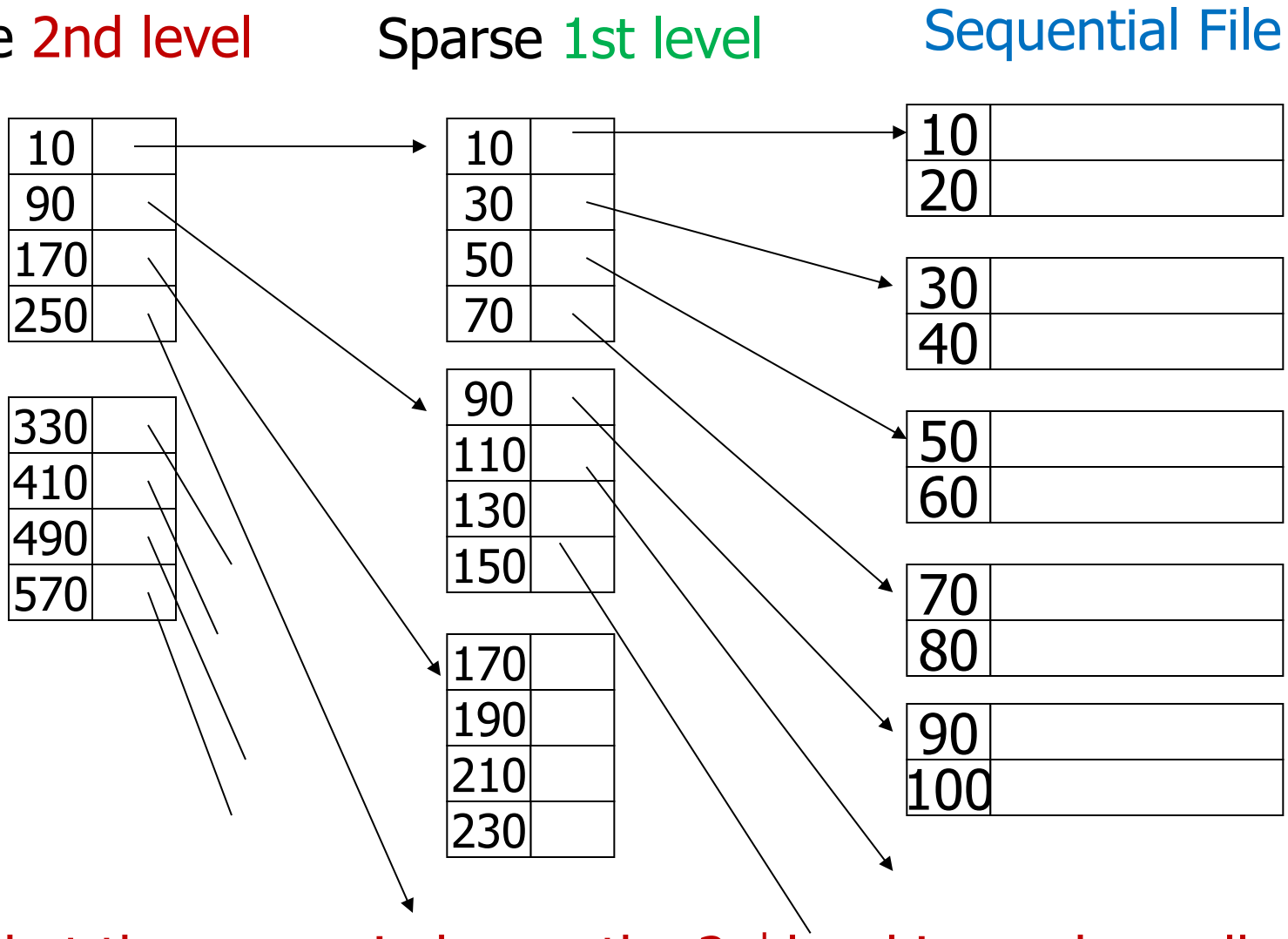


Notice that the dense index is much smaller than the sequential file which hold the data

Each arrow is a block address



In a sparse index, and index pointer points to the block in the sequential file that hold all the records whose key values are in the range of 2 consective index (key) values in the sparse index



Notice that the sparse index on the 2nd level is much smaller than the sparse index on the 1st level

- Comment:

  The blocks in the sequential data file or in at a level of the index may be in contiguous memory or may be in blocks that are in a (chained) list

Inserts in a list is much easier (more efficient) in a list.

Inserts in contiguous memory may require a shift that is linear in the size of the data file or an index level

Question:

- Can we build a dense, 2nd level index for a dense index?

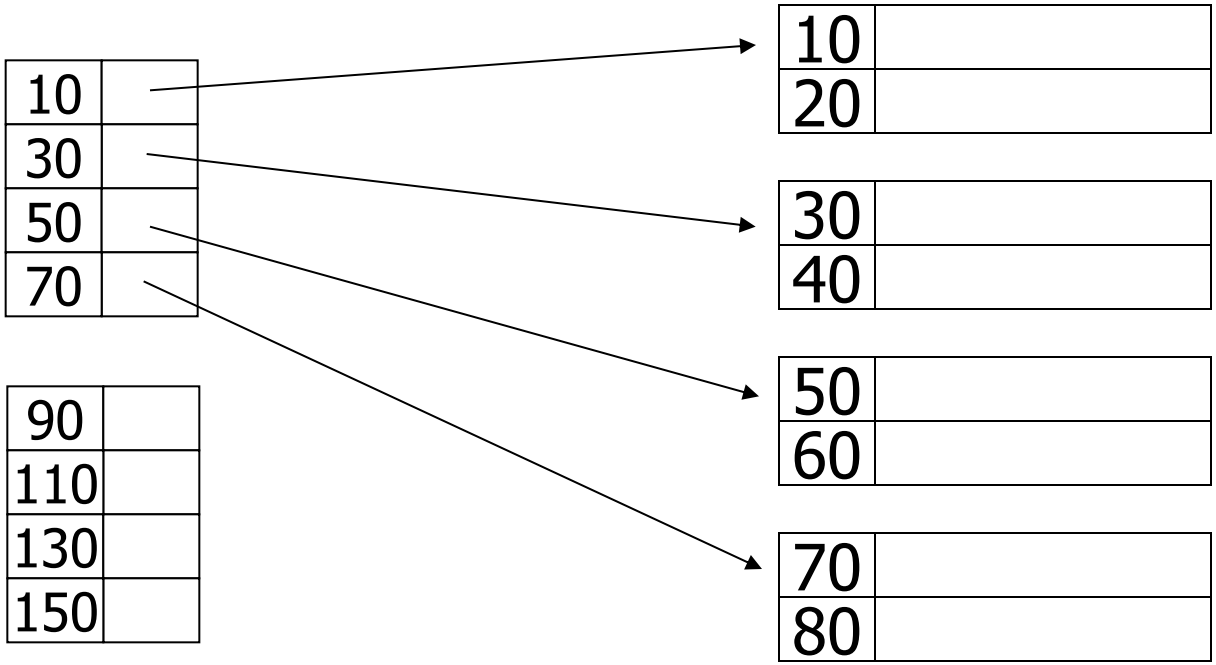
Sparse vs. Dense Tradeoff

- Sparse: Less index space per record
  can keep more of index in memory
- Dense: Can tell if any record exists
  without accessing file

Recap of concepts and terms related to indexing

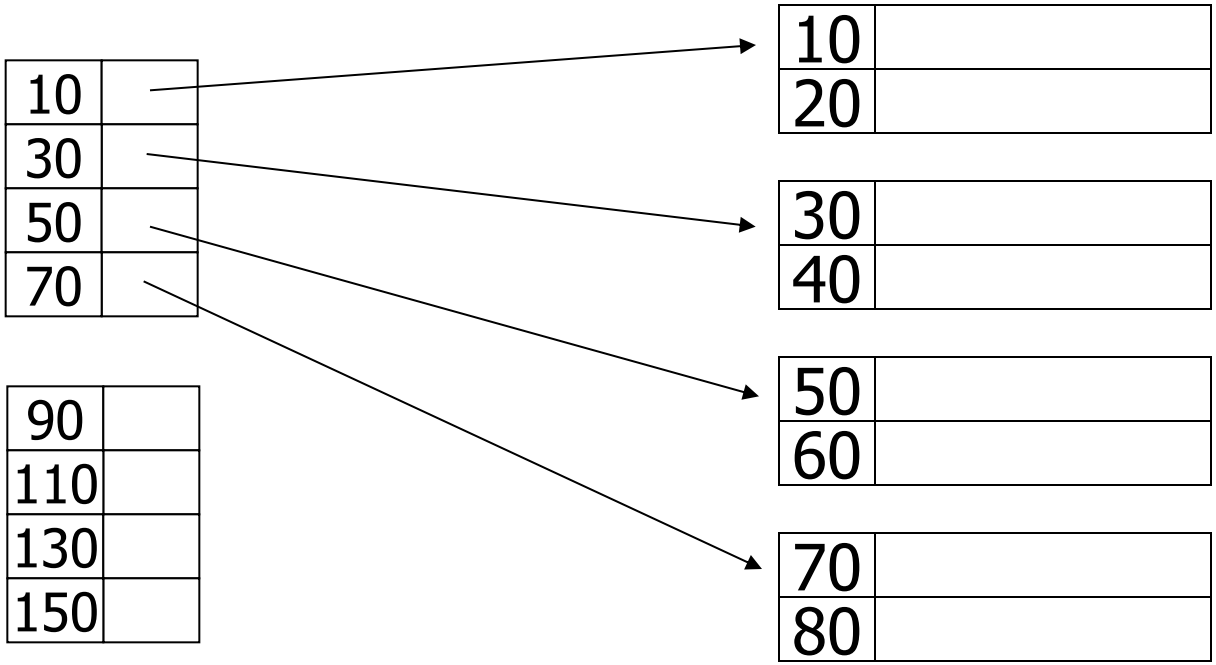
- Sequential file
- Search key ($\neq$ primary key)
- Primary index
- Secondary index
- Dense index
- Sparse index
- Multi-level index

Deletion from sparse index



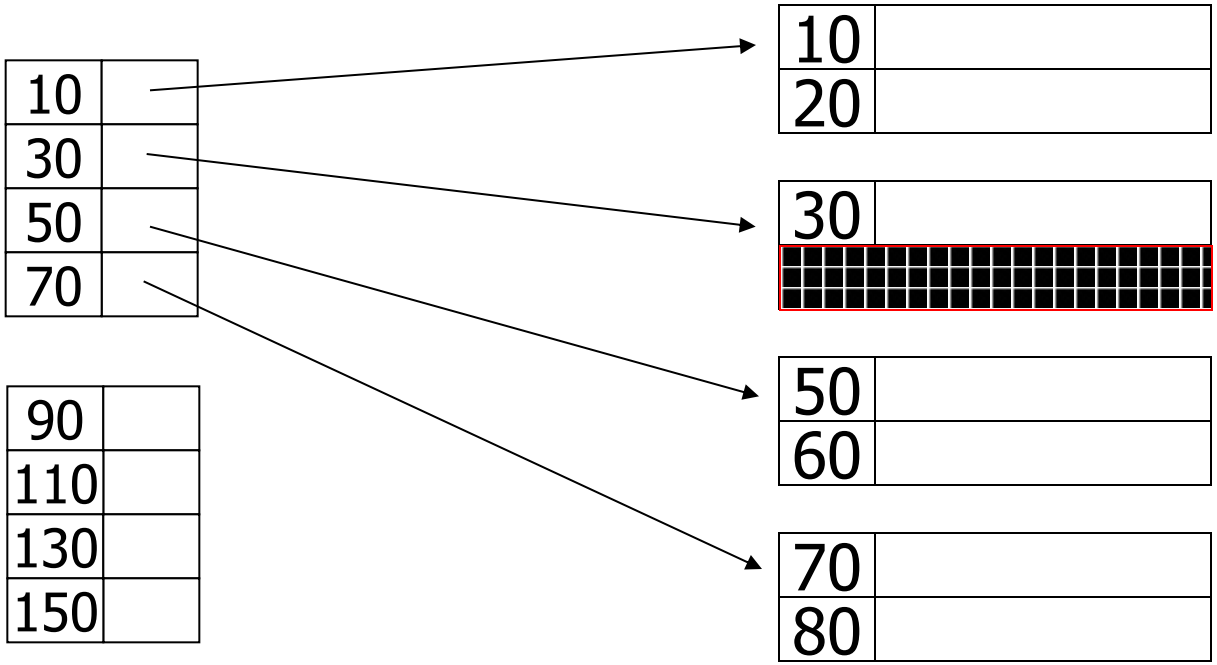
Deletion from sparse index

– delete record 40



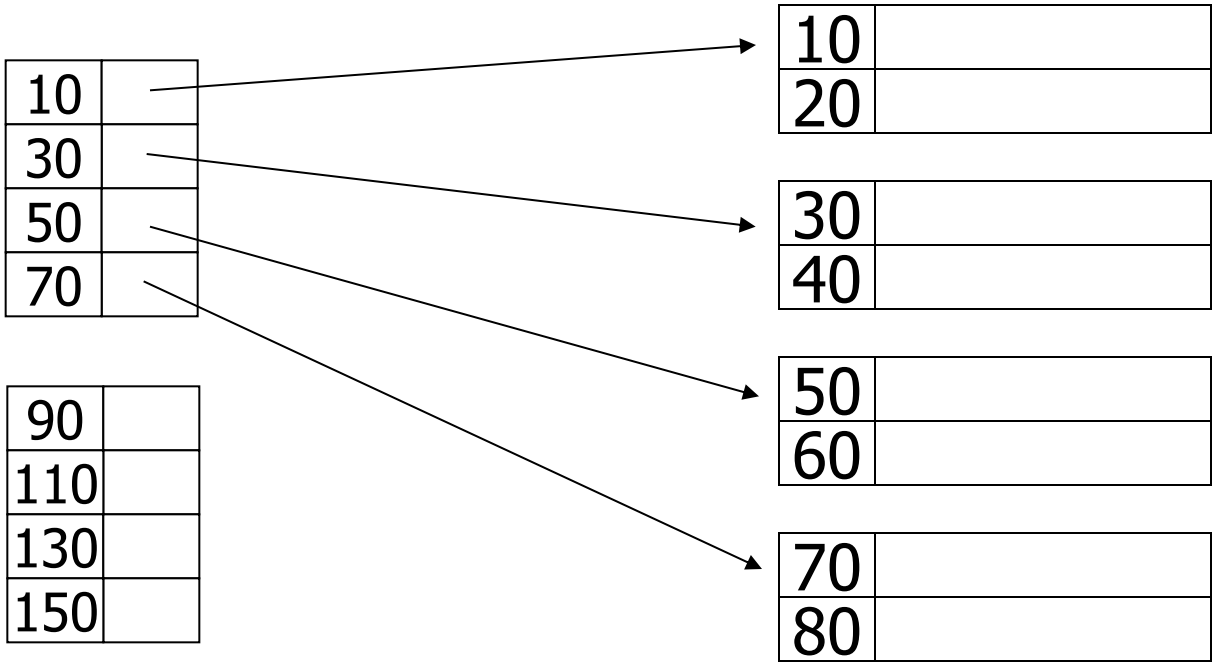
Deletion from sparse index

– delete record 40



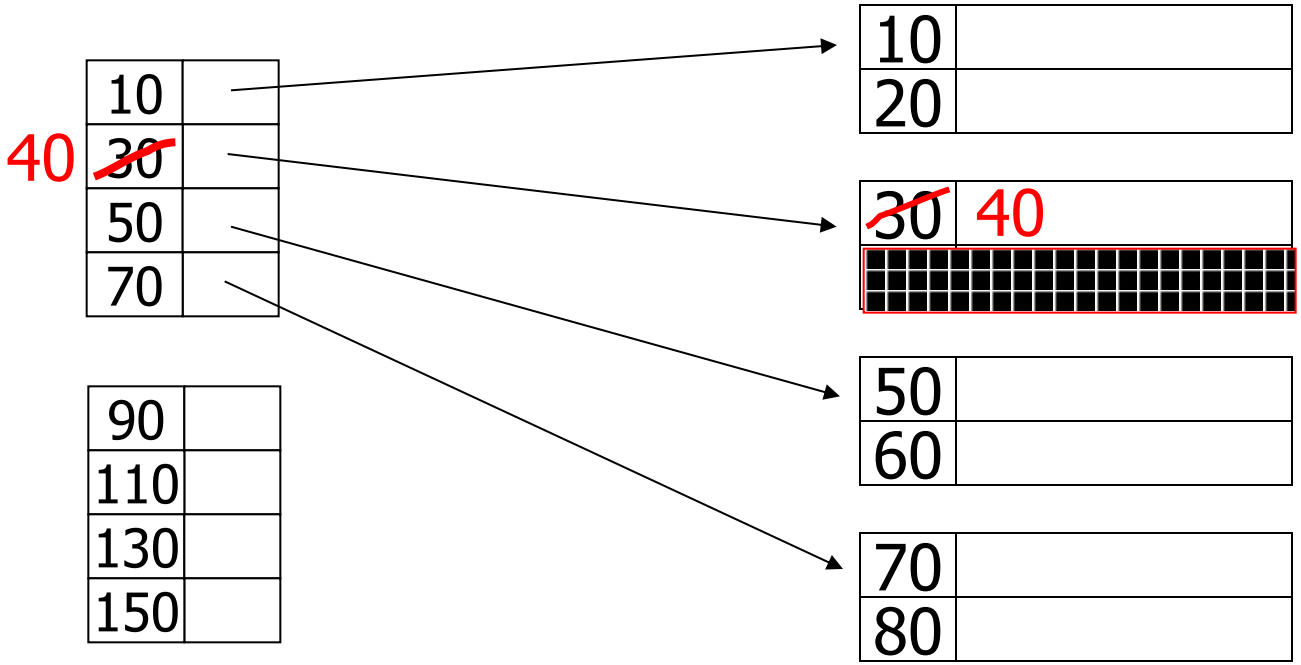
Deletion from sparse index

– delete record 30



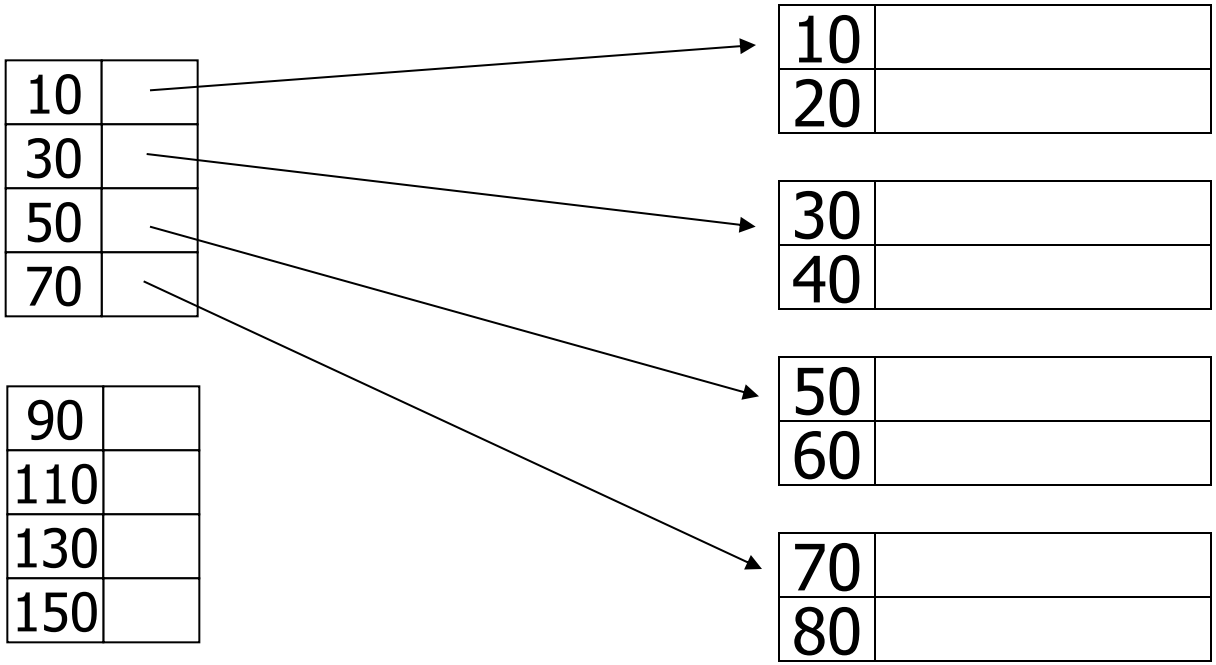
Deletion from sparse index

– delete record 30



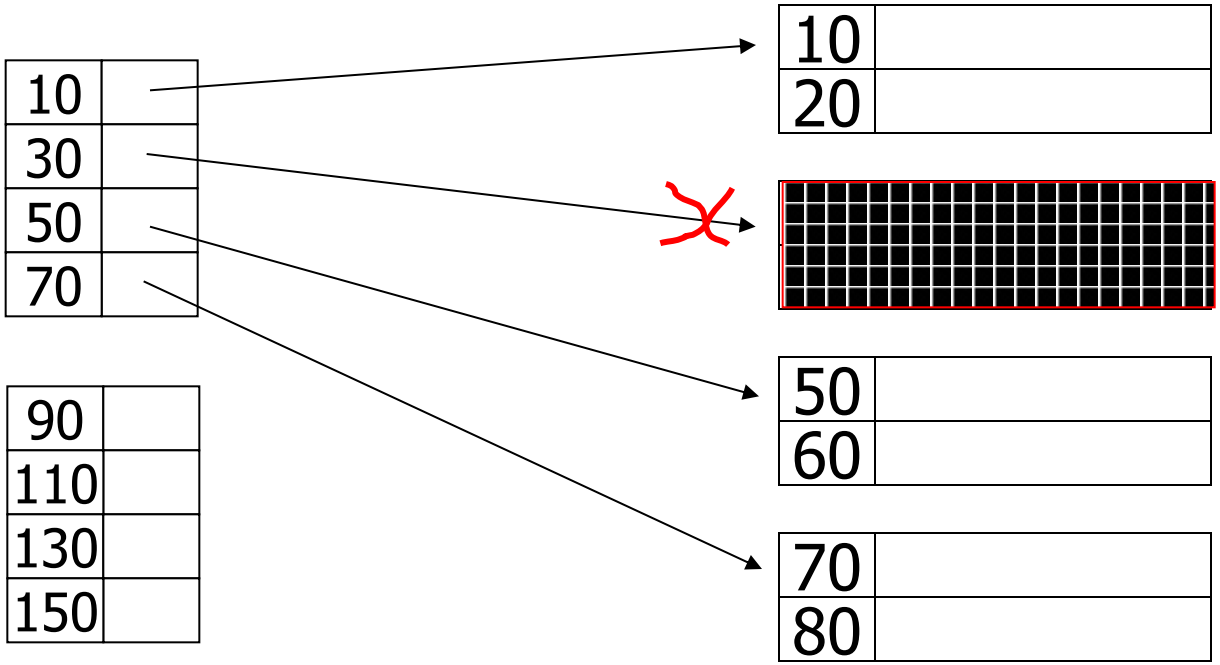
Deletion from sparse index

– delete records 30 & 40



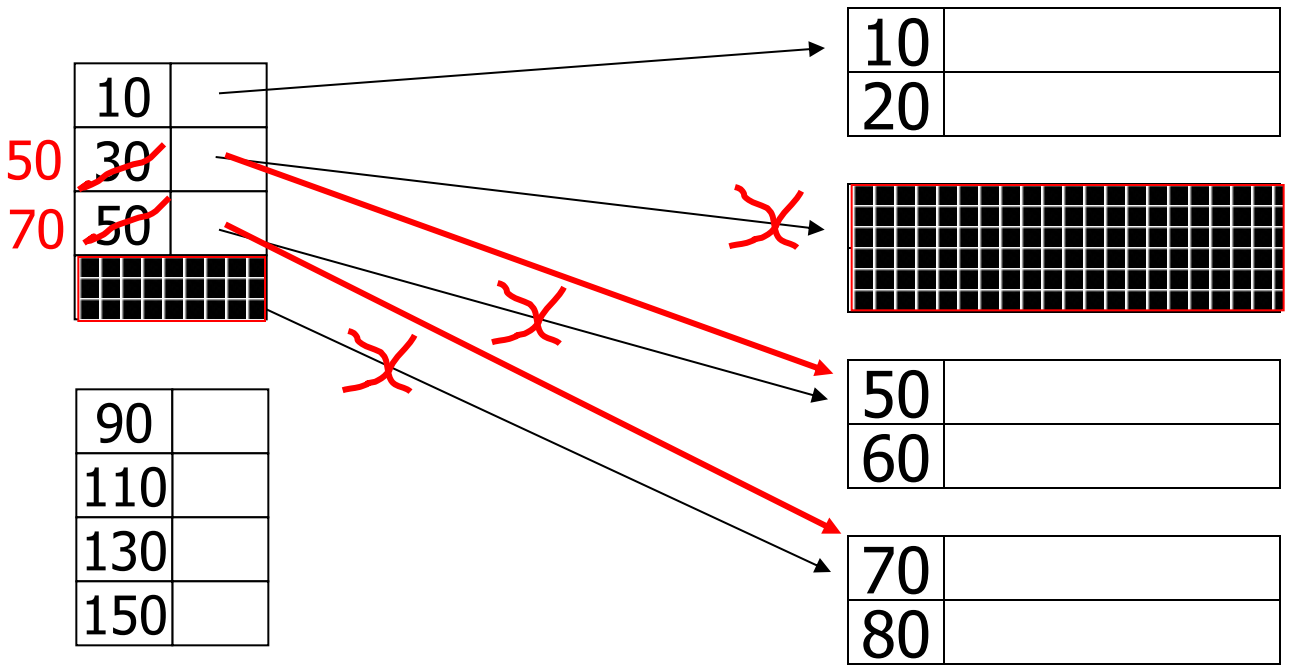
Deletion from sparse index

– delete records 30 & 40

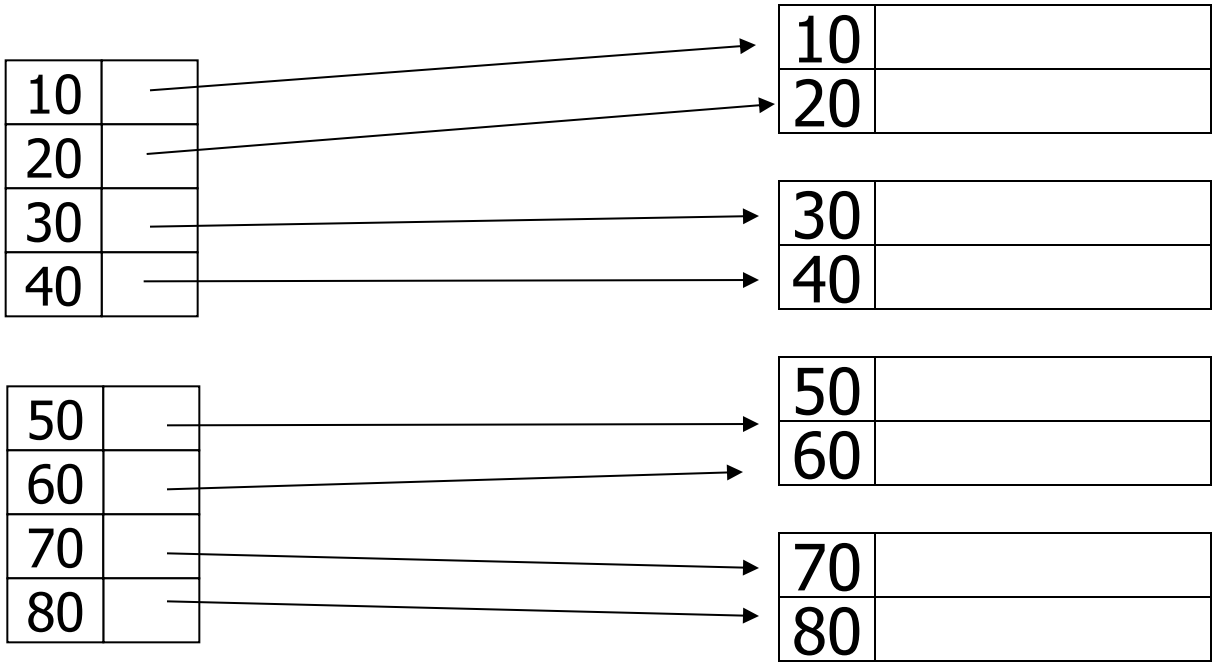


Deletion from sparse index

– delete records 30 & 40

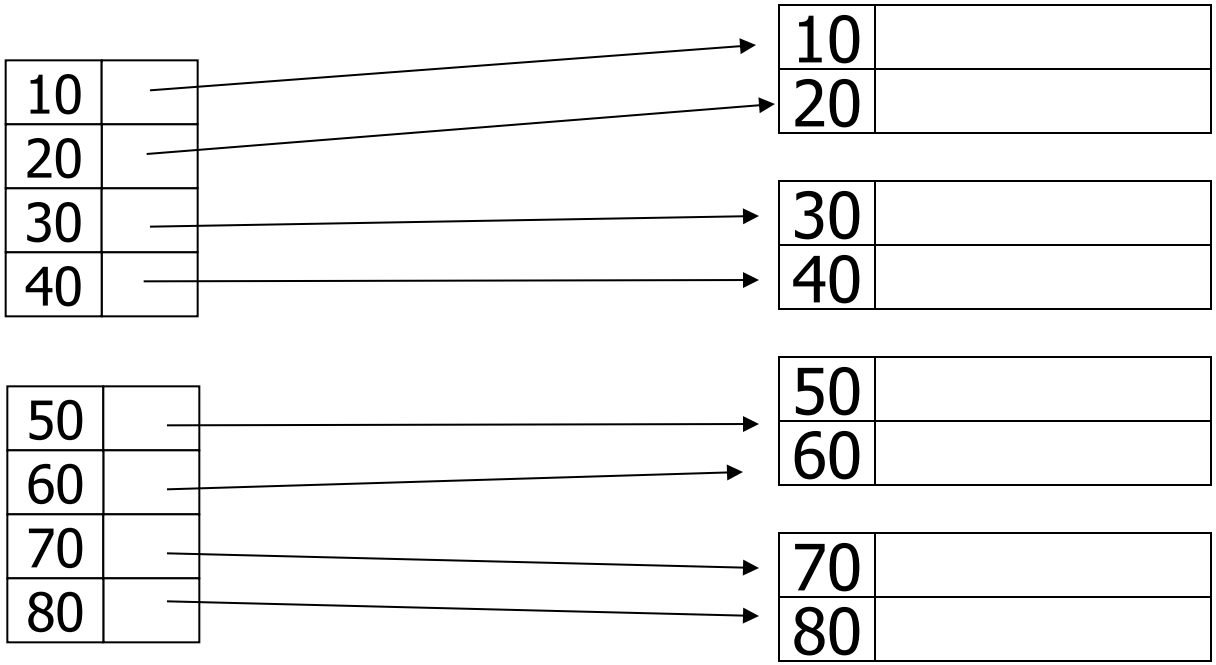


Deletion from dense index



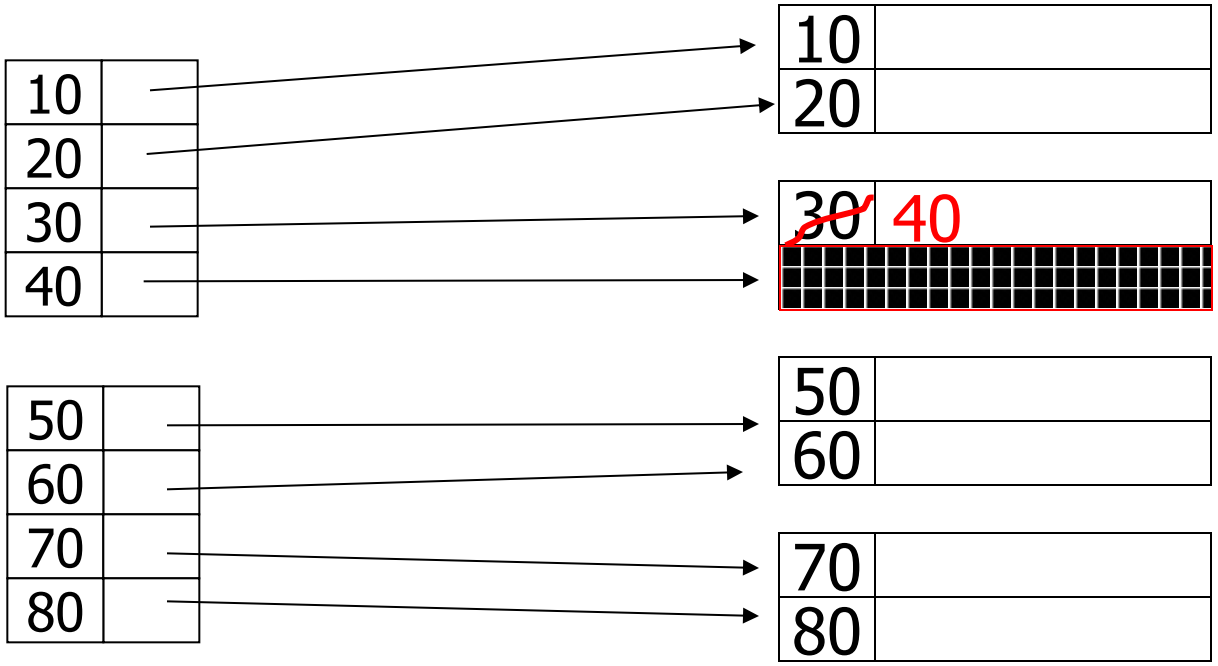
Deletion from dense index

– delete record 30



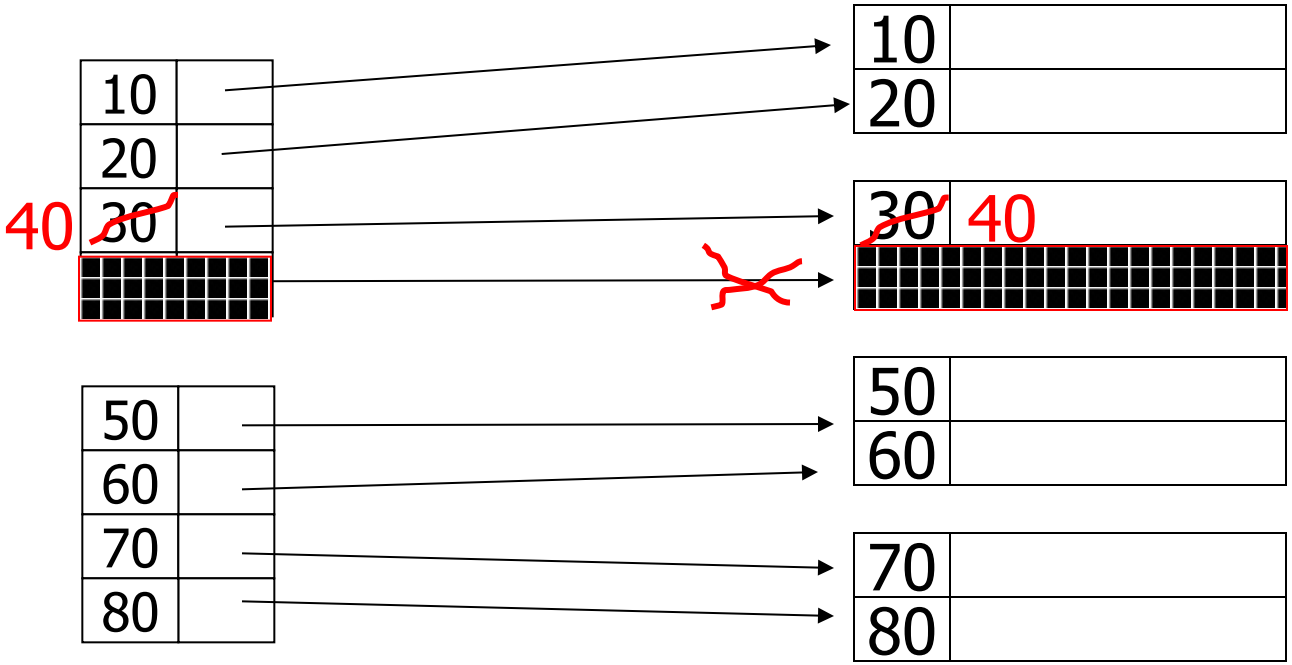
Deletion from dense index

– delete record 30

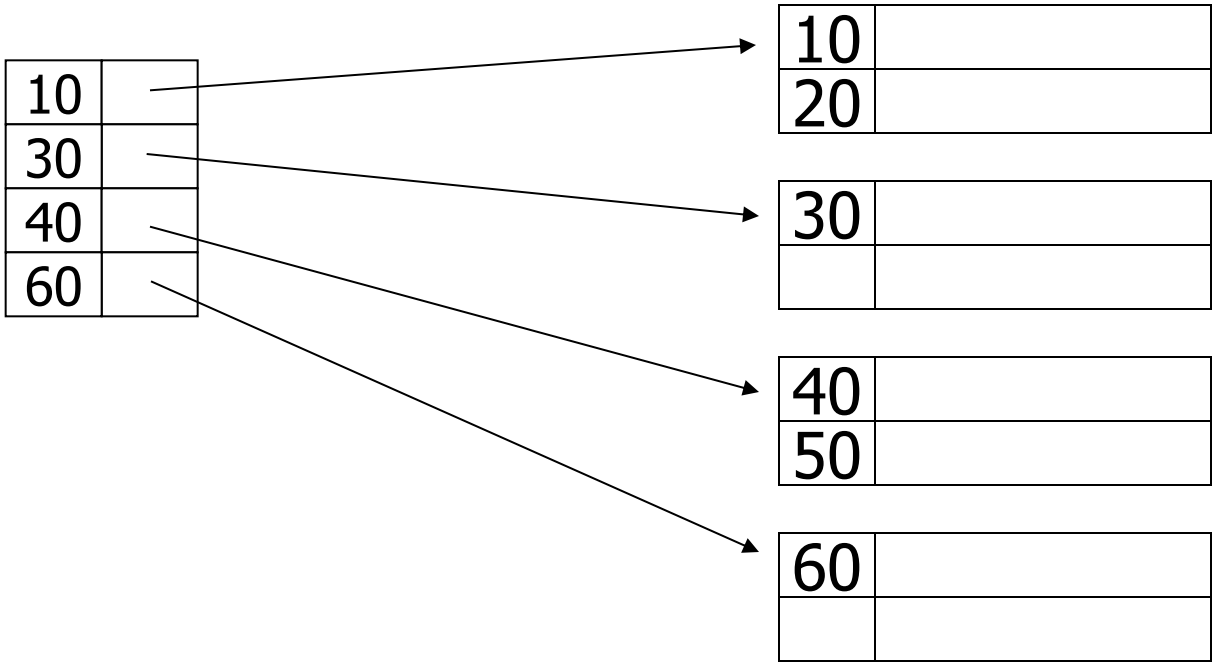


Deletion from dense index

– delete record 30

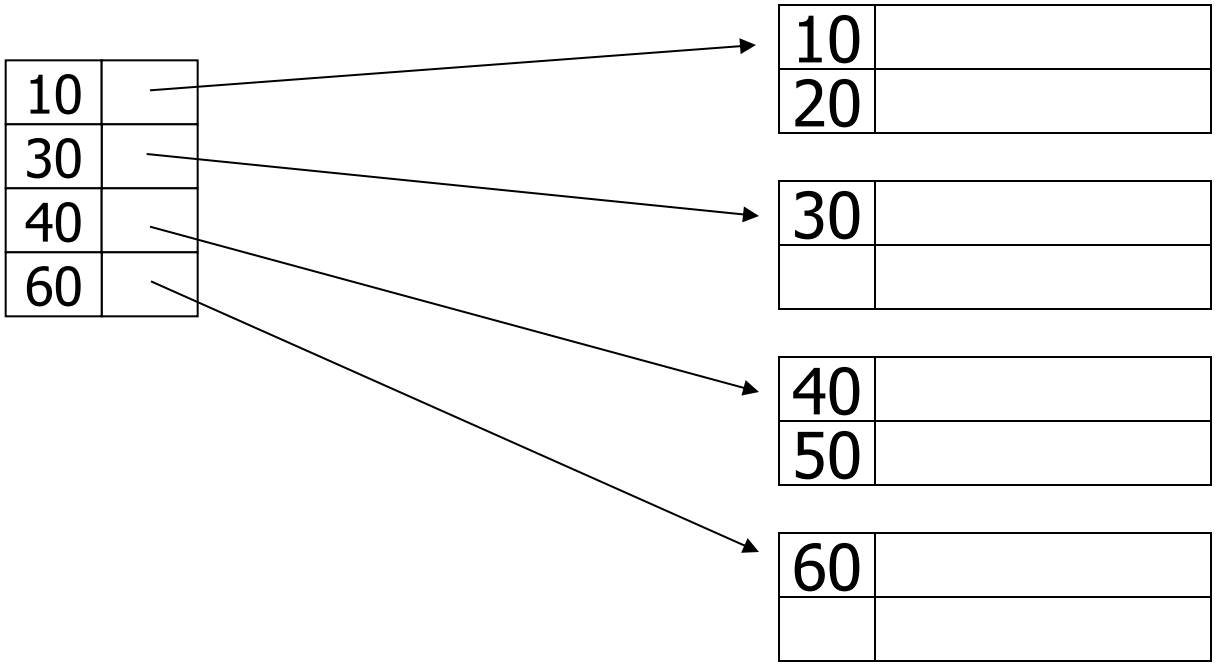


Insertion, sparse index case



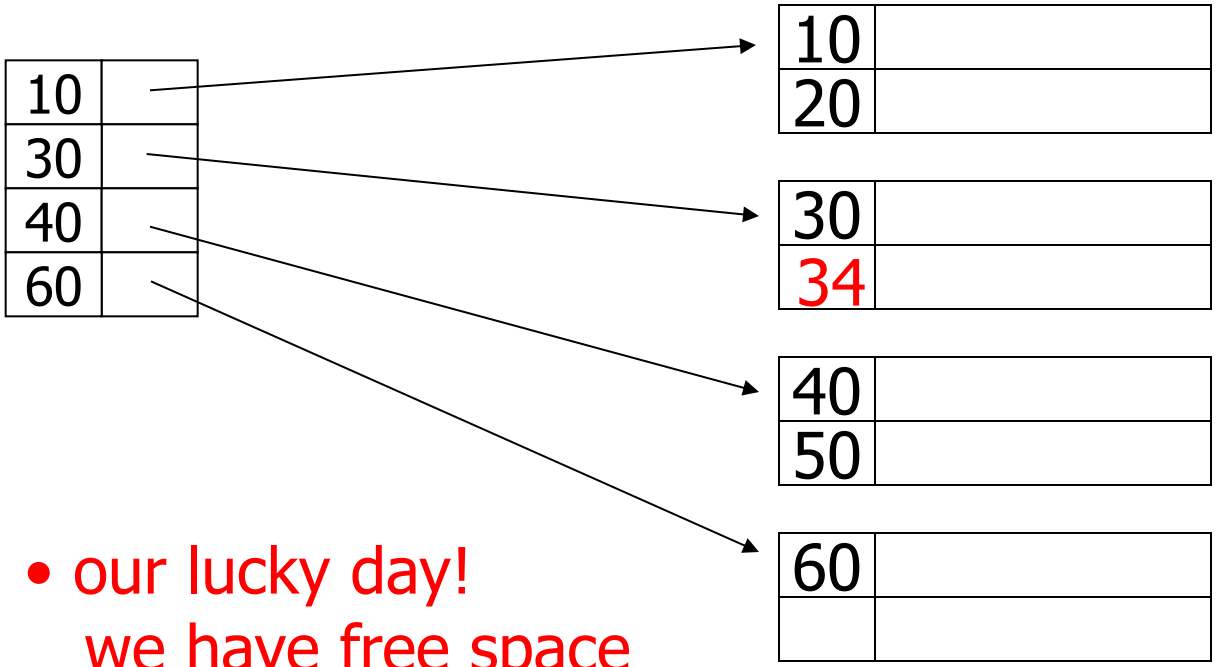
Insertion, sparse index case

– insert record 34



Insertion, sparse index case

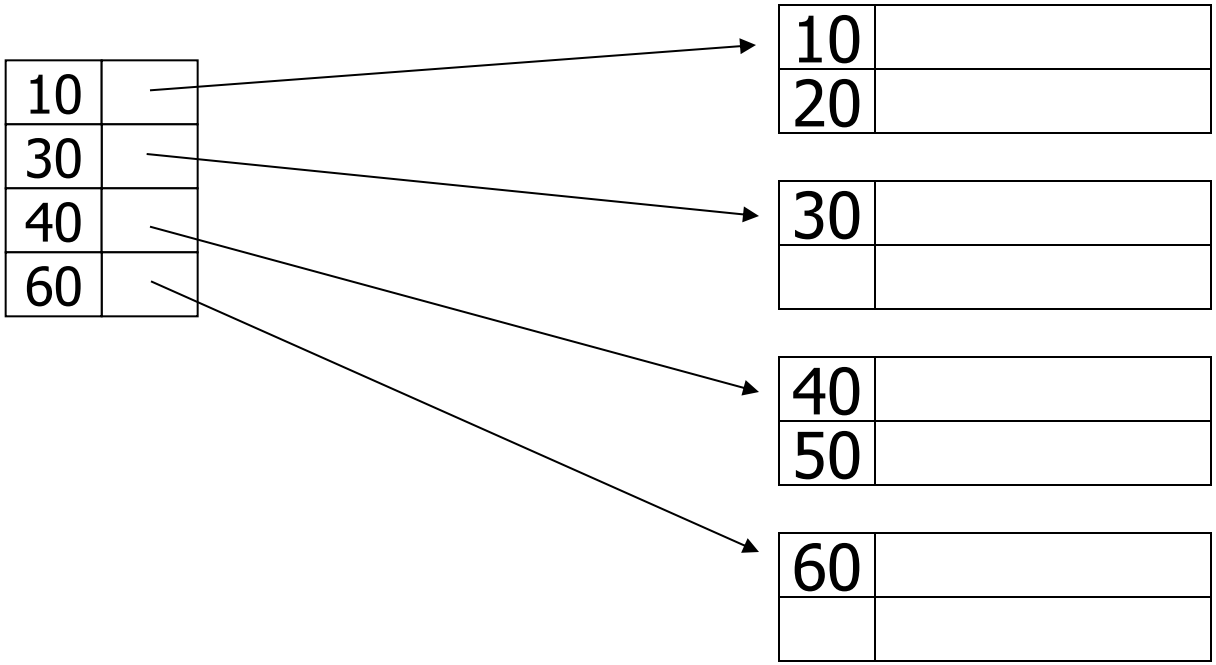
– insert record 34



- our lucky day!
  we have free space
  where we need it!

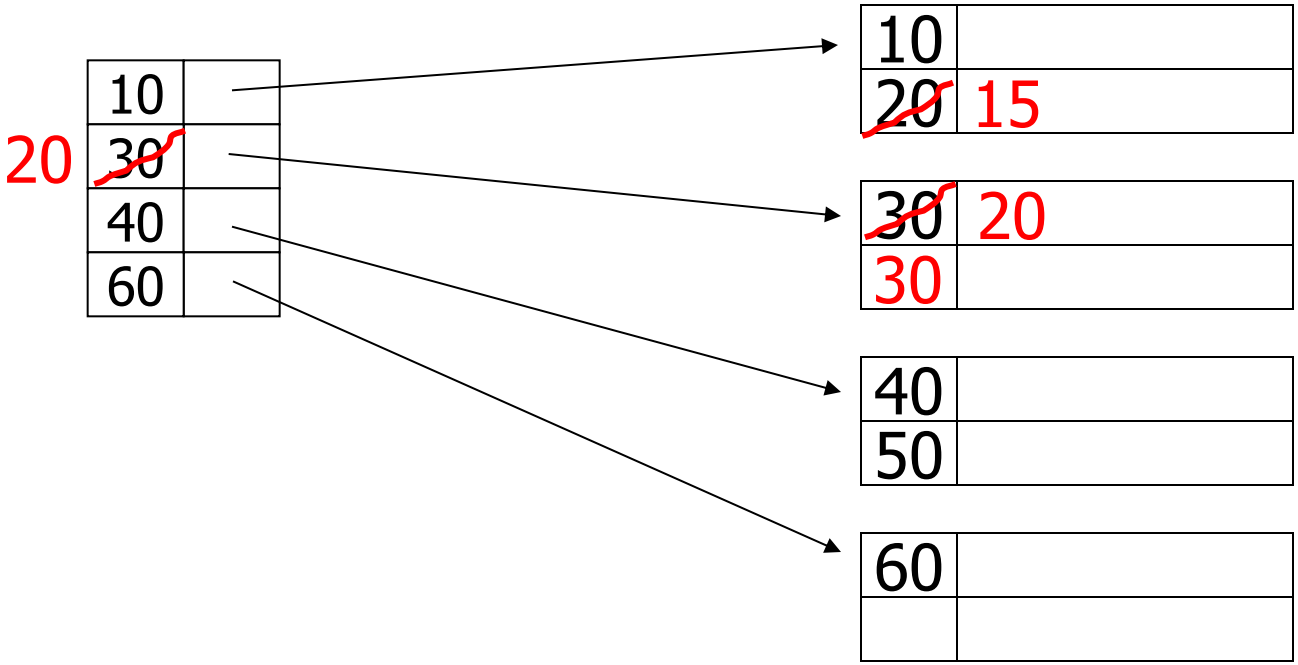
Insertion, sparse index case

– insert record 15



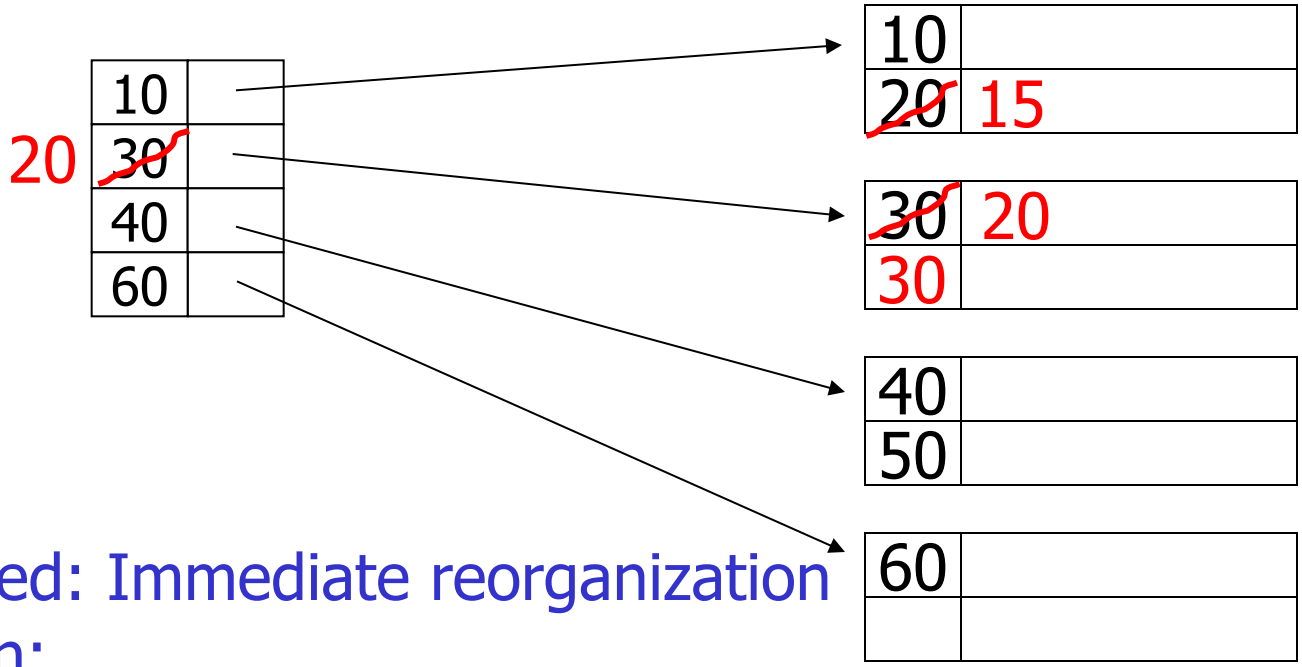
Insertion, sparse index case

– insert record 15



Insertion, sparse index case

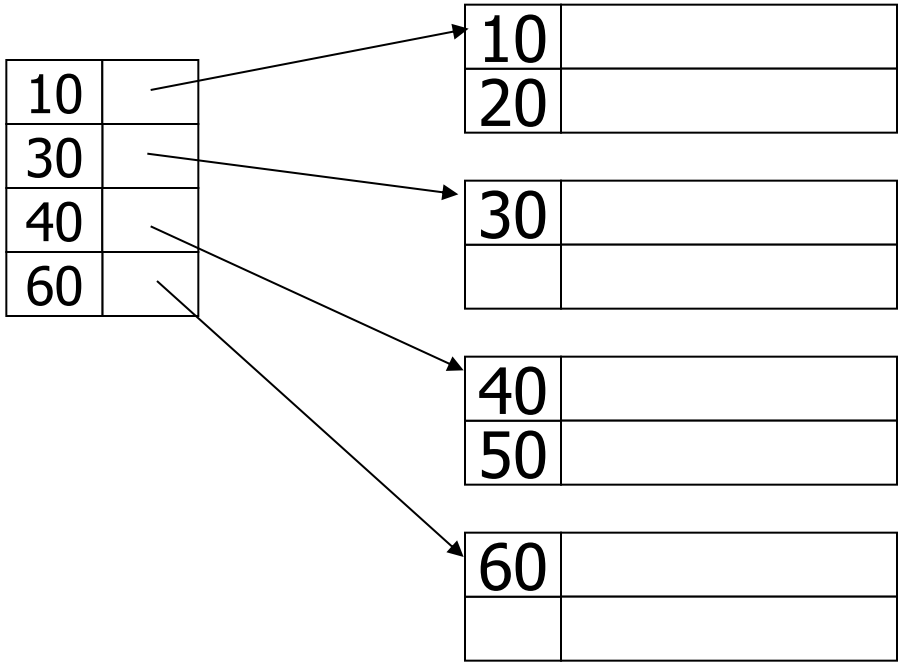
– insert record 15



- Illustrated: Immediate reorganization
- Variation:
  - insert new block (chained file)
  - update index

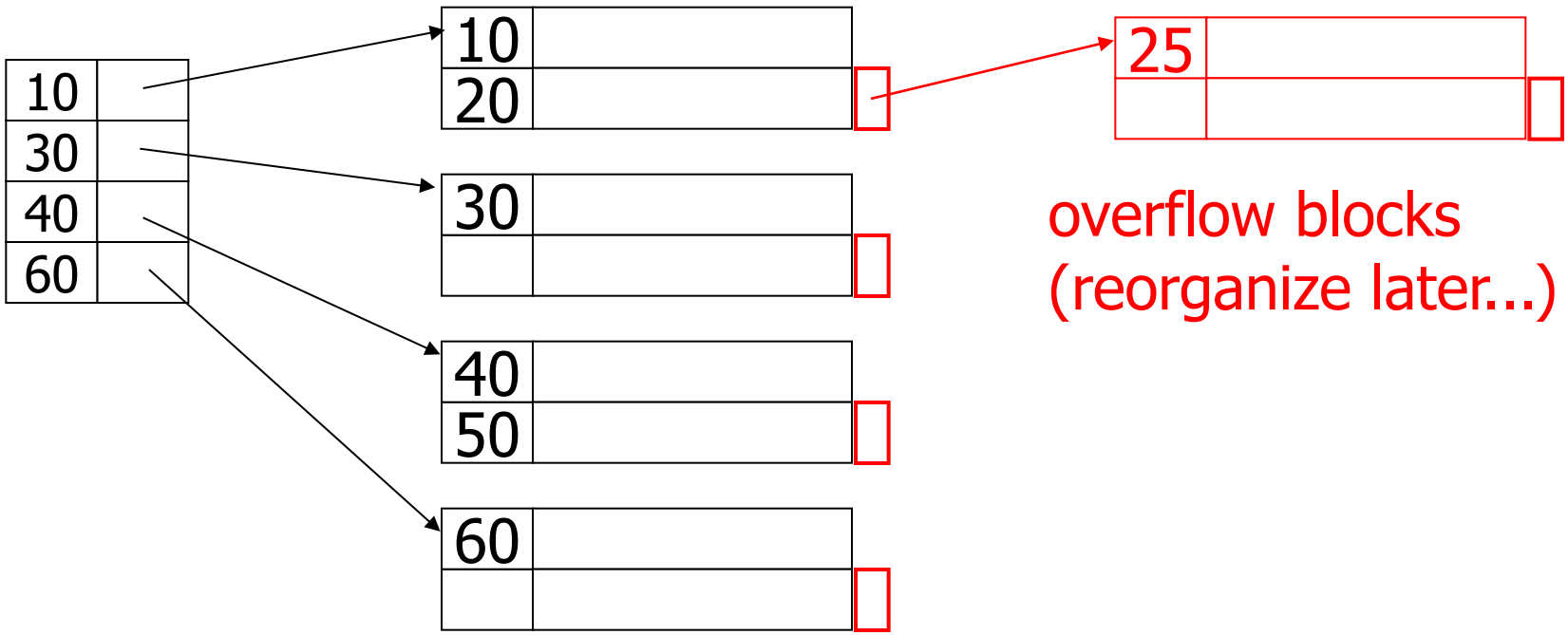
Insertion, sparse index case

– insert record 25



Insertion, sparse index case

– insert record 25



Insertion, dense index case

- Similar
- Often more expensive . . .

Secondary indexes

Sequence field

30	
50	

20	
70	

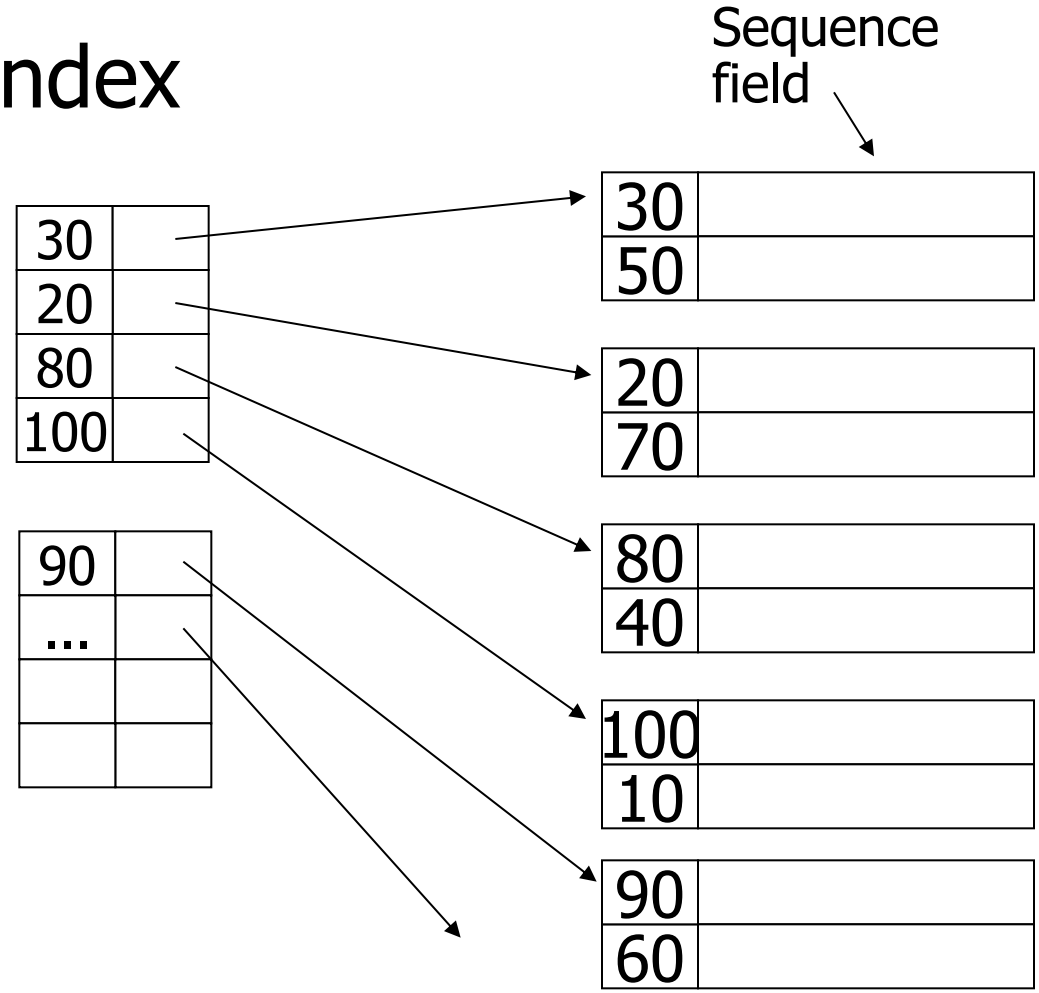
80	
40	

100	
10	

90	
60	

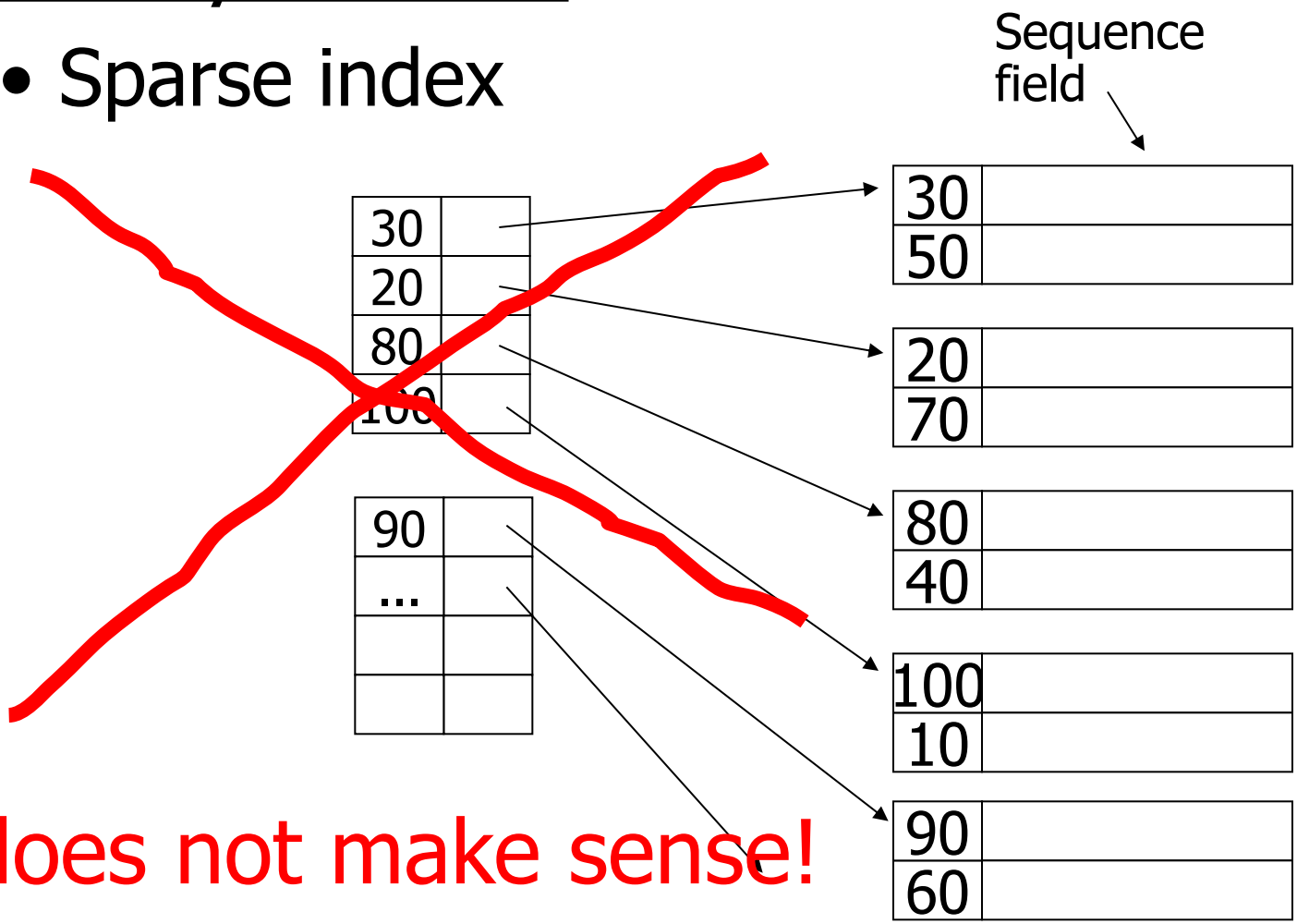
Secondary indexes

- Sparse index



Secondary indexes

- Sparse index



does not make sense!

Secondary indexes

- Dense index

Sequence field

30	
50	

20	
70	

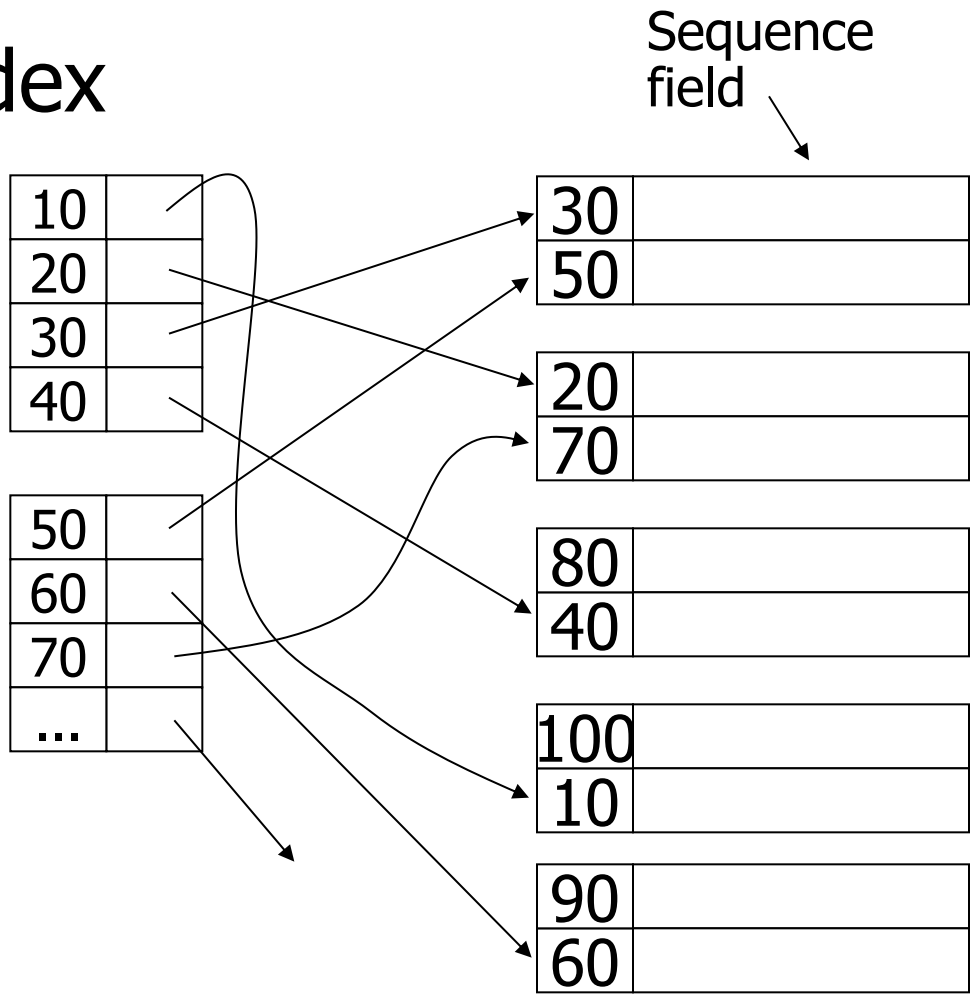
80	
40	

100	
10	

90	
60	

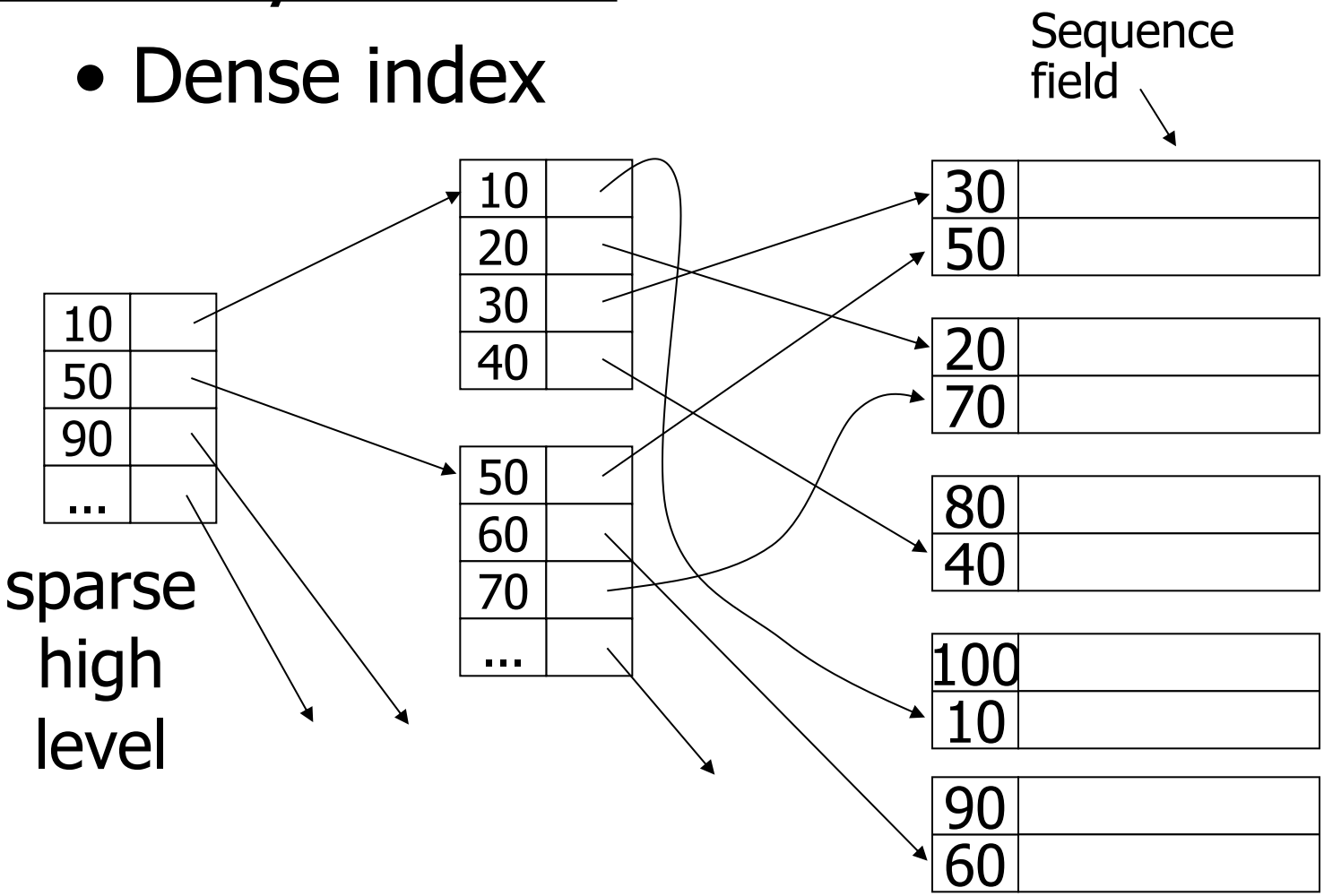
Secondary indexes

- Dense index



Secondary indexes

- Dense index



With secondary indexes:

- Lowest level is dense
- Other levels are sparse

Also: Pointers are record pointers

(not block pointers; not computed)

Conventional indexes

Advantage:

- Simple
- Index is sequential file
  good for scans

Disadvantage:

- Inserts expensive, and/or
- Lose sequentiality & balance