Database Management Systems

- Optimization

Doug Shook

How can we make things faster?

- Indexing
- Query Optimization

Indexing

- Provides a secondary access path
  - Does not alter primary physical representation
- Indexes can take many forms:
  - Single Level
    - Primary, clustered
  - Multi Level
    - Trees

Single Level

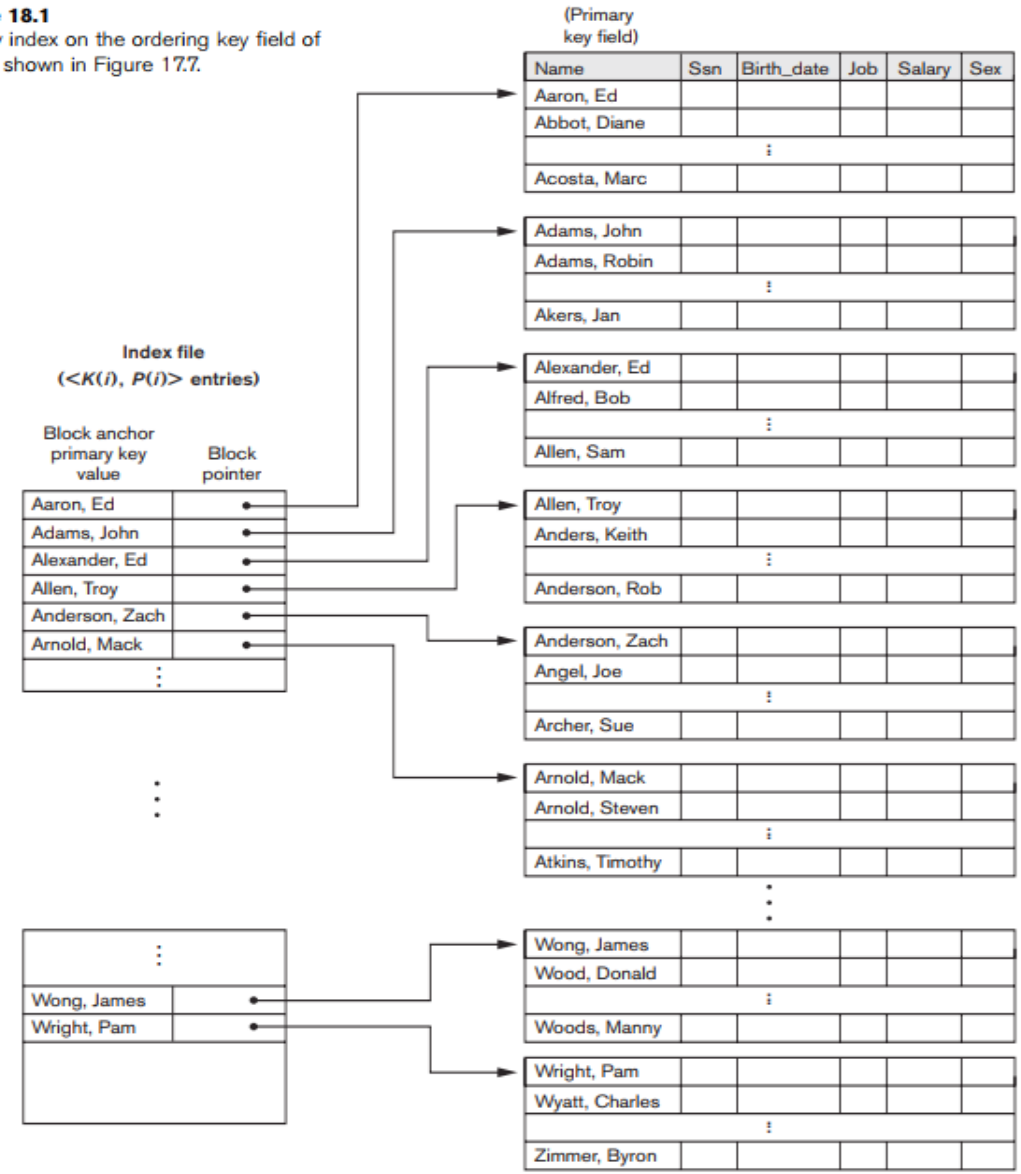
- Much like an index from a book
- Select a column to be indexed
  - Make a list of all values contained within that column
  - Create an index that has a column value as the key and a list of pages as the value
- Indexing values are ordered
  - How does this help?
- Size in comparison to primary storage?

Primary Indexes

- One entry per page
- Key: First index value on the page
- Value: file address of the page
- Requires that data be stored in order (why?)
  - Requires that the index is based on the primary key (why?)
- Primary indexes are an example of sparse indexes
- Search time?

Primary Indexes

Figure 18.1
Primary index on the ordering key field of the file shown in Figure 17.7.



Primary Indexes

- How is this scheme affected by insertions and deletions?

Clustered Index

- Indexing technique for non-key columns
  - What's the major difference?
- Is this sparse or dense
- How does this affect insertion and deletion?

Clustered Index

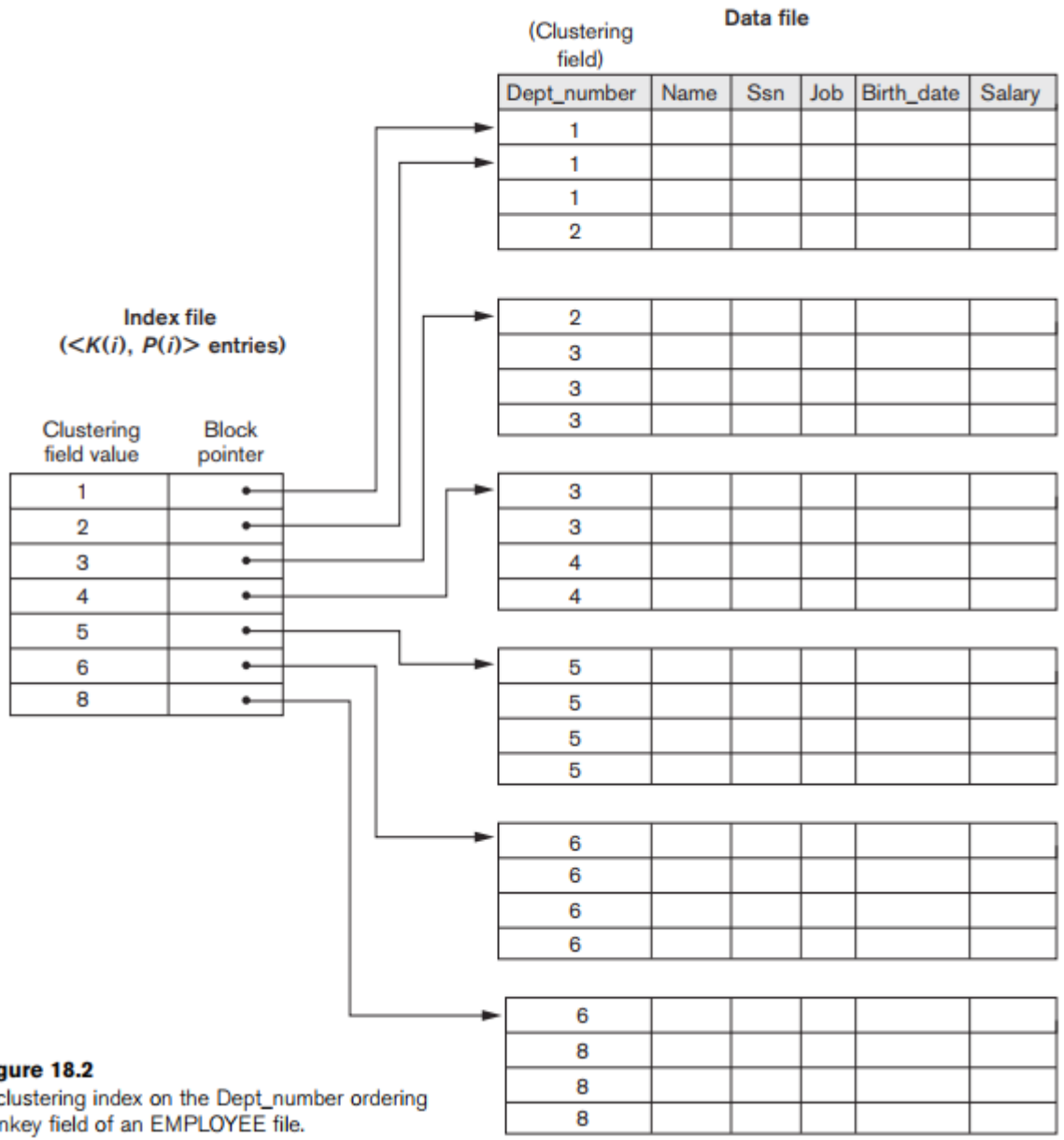
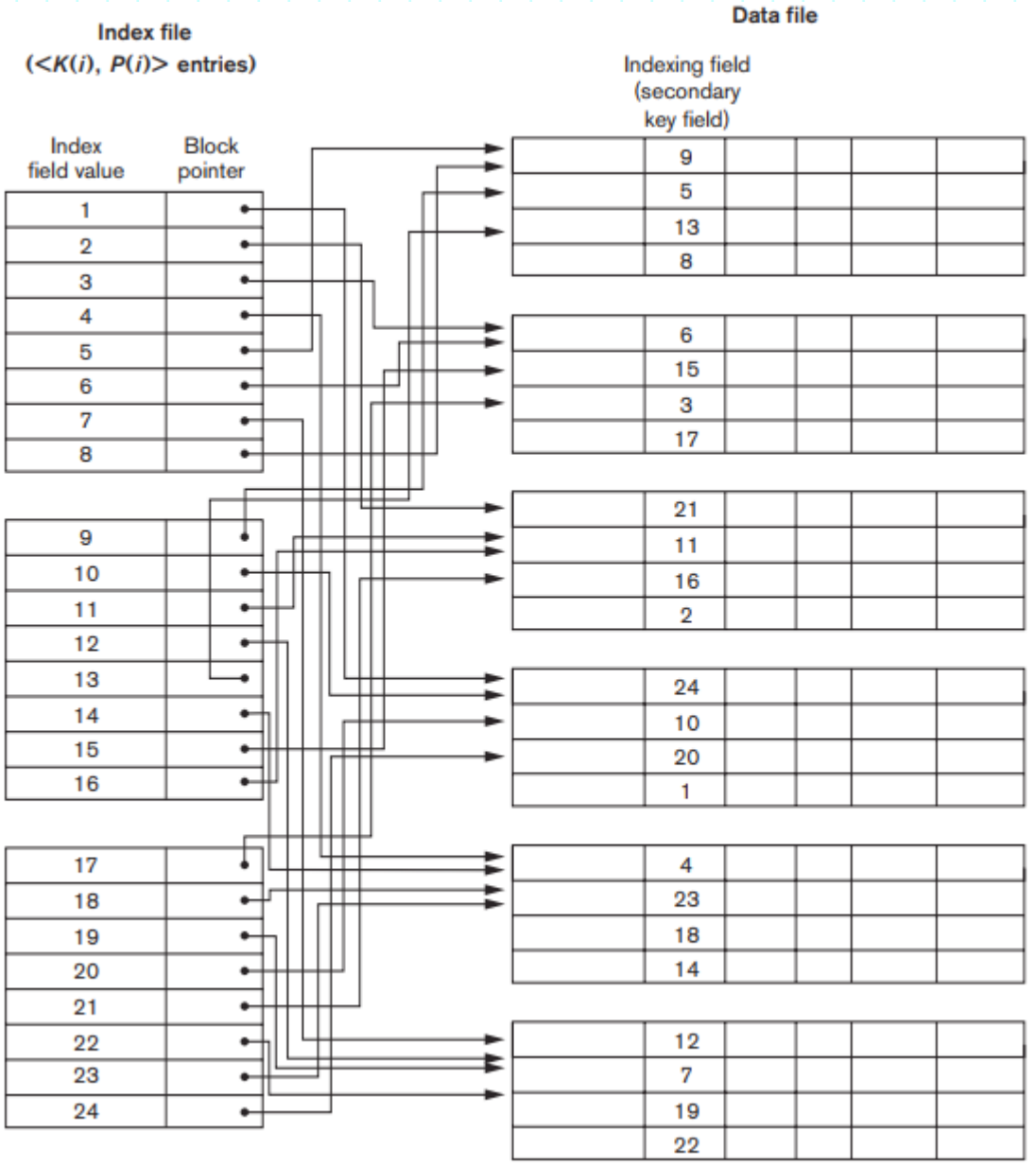


Figure 18.2
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file.

Secondary Index

- What if we want to index by a column that is not ordered?
  - Unique vs. not unique

Secondary Index - Unique



Secondary – Not Unique

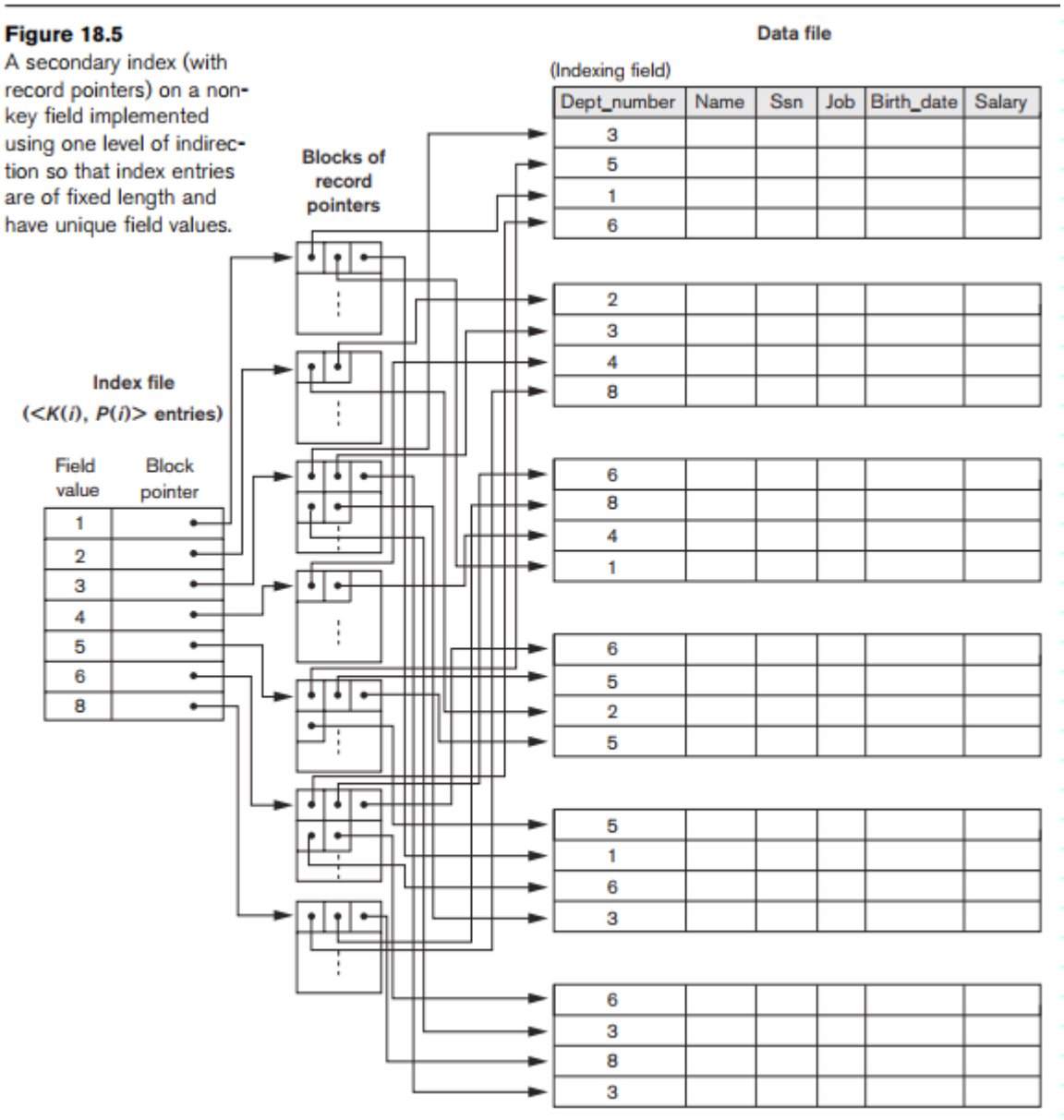


Figure 18.5
A secondary index (with record pointers) on a non-key field implemented using one level of indirection so that index entries are of fixed length and have unique field values.

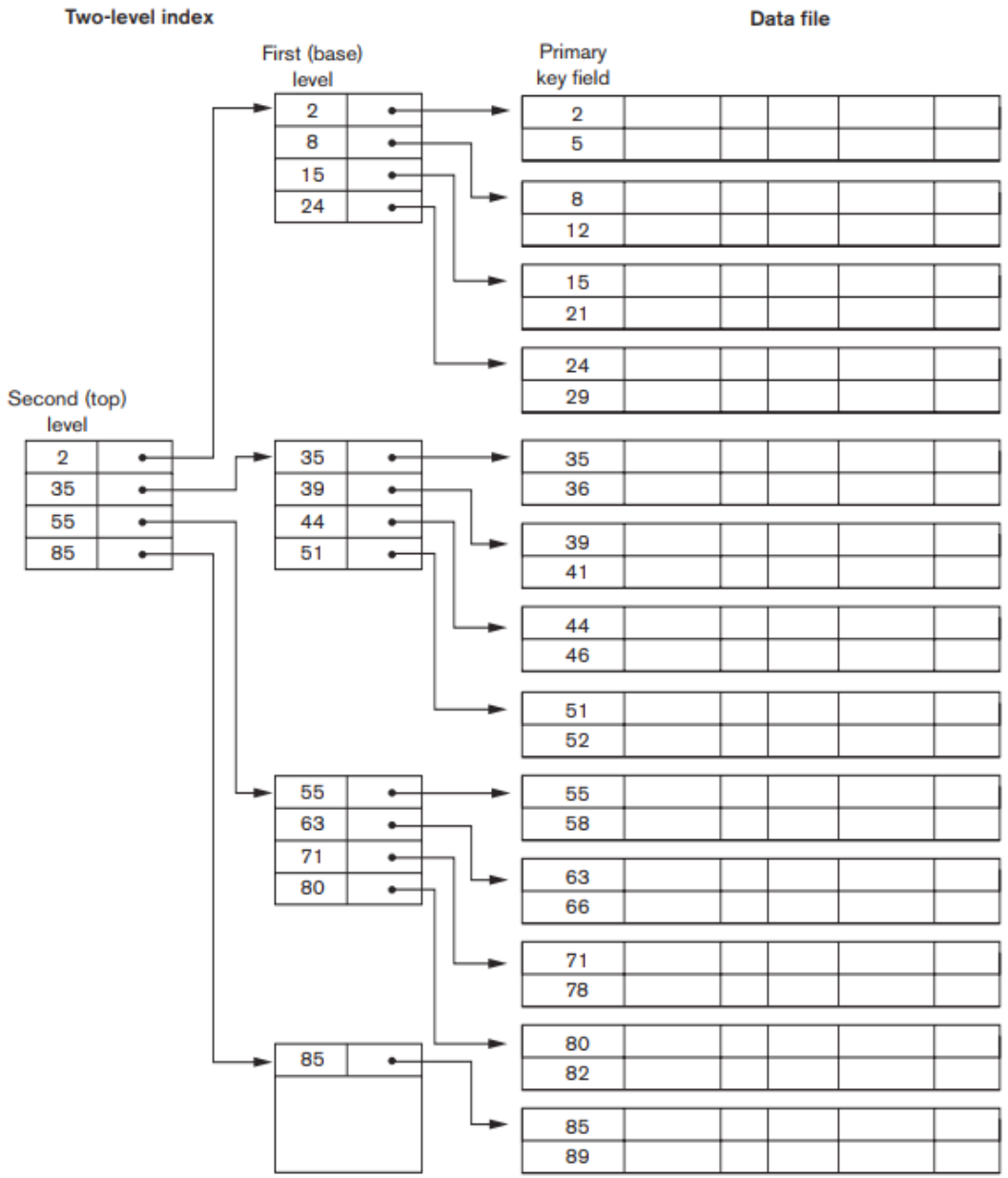
Secondary Index

- Dense or sparse?
- Performance?
  - Performance improvement?

Multilevel Indexes

- Same idea as a single level index
  - Try to reduce the search space even faster
- Idea: create an index (first layer)
  - Then create another index into that index
  - Repeat
- If we keep our indexes ordered, what kind of index can we use for the upper layers?
- Restrictions on index type of first layer?
- Limitations

Multilevel Index



Multilevel Index

- Search time?
- How to deal with insert and delete?

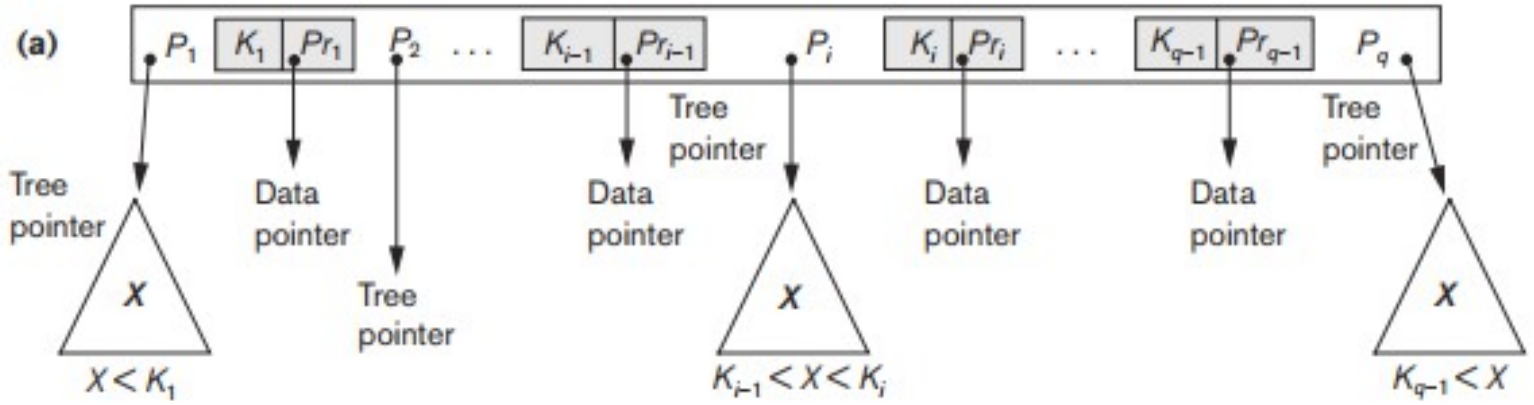
Dynamic Multilevel Index

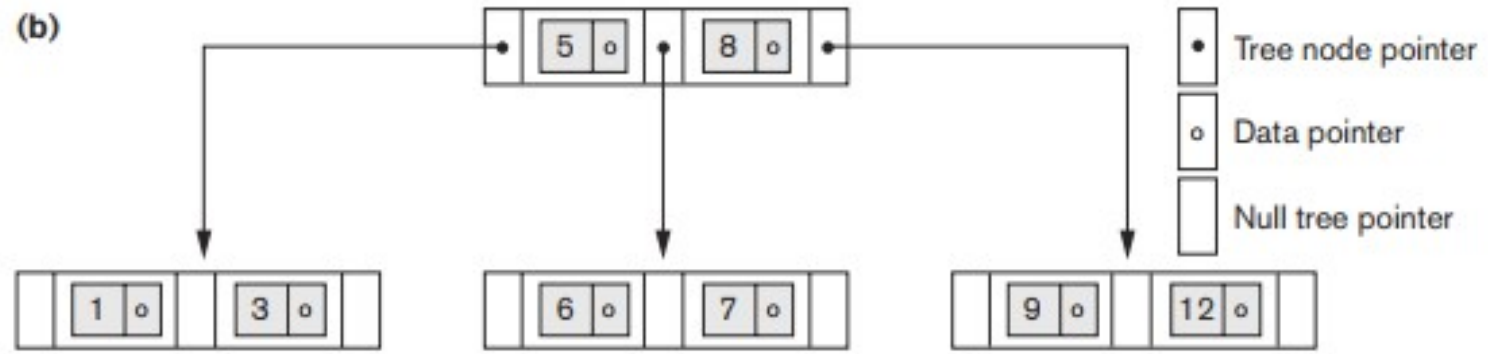
- Attempts to mitigate problems with insert and delete by leaving some empty space in each page of the index
  - Tradeoff?
- Uses search trees
  - B-Trees
  - B+-Trees

B-Trees

- Properties
  - Always balanced
  - Tries to minimize wasted space due to deletions
  - Simplifies insertion and deletion (mostly)

B-Trees


(a)
P_1 K_1 Pr_1 P_2 ... K_{i-1} Pr_{i-1} P_i K_i Pr_i ... K_{q-1} Pr_{q-1} P_q
Tree pointer
Data pointer
Tree pointer
Data pointer
Tree pointer
Data pointer
Data pointer
Tree pointer
X
$X < K_1$
X
$K_{i-1} < X < K_i$
X
$K_{q-1} < X$


(b)
5 o
8 o
1 o
3 o
6 o
7 o
9 o
12 o
Tree node pointer
Data pointer
Null tree pointer

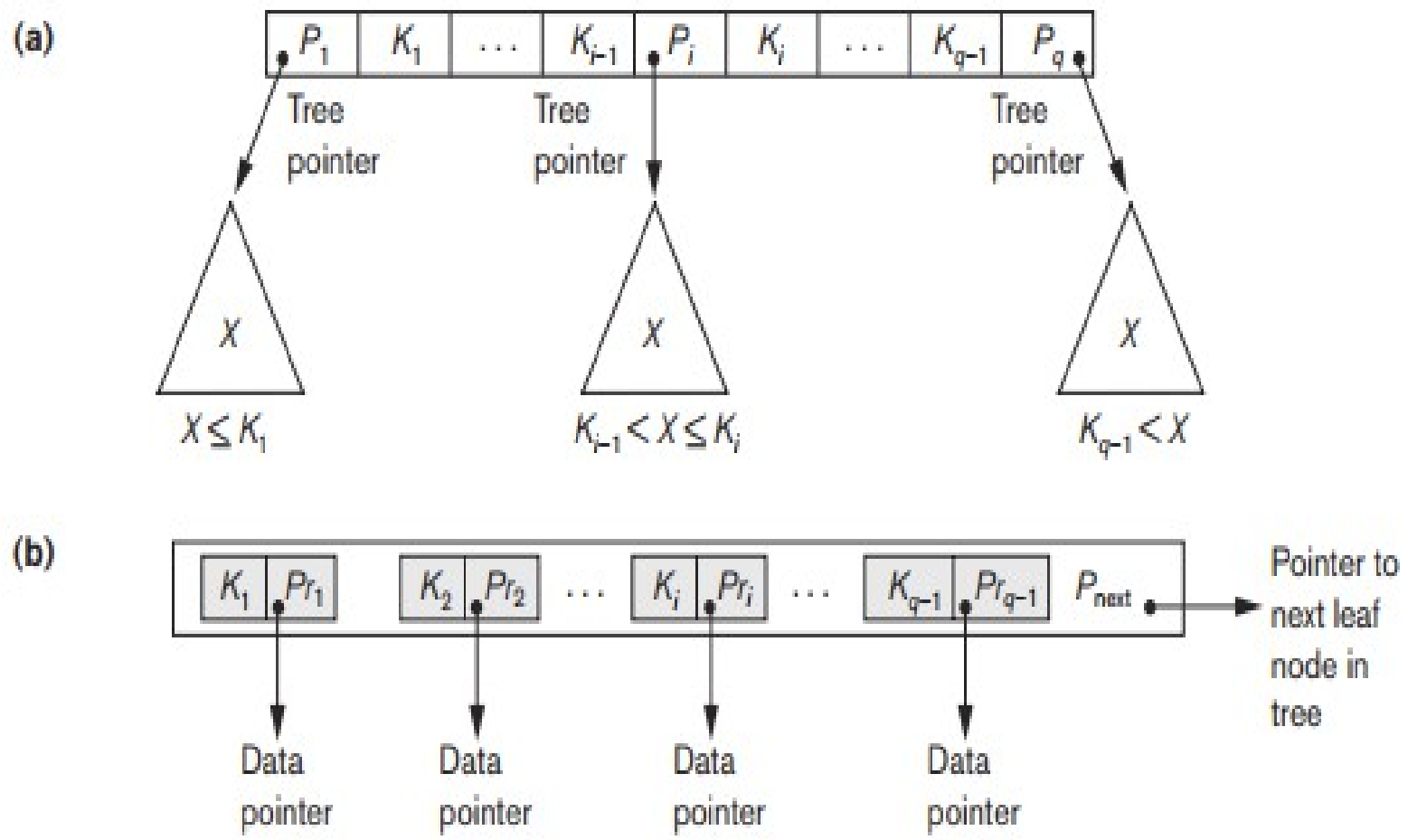
B-Trees

- Previous example assumes we're searching a key
  - What if we're not?
- Insertion and deletion?

B+-Trees

- Similar to a B-Tree
  - Data only stored at the leaf nodes

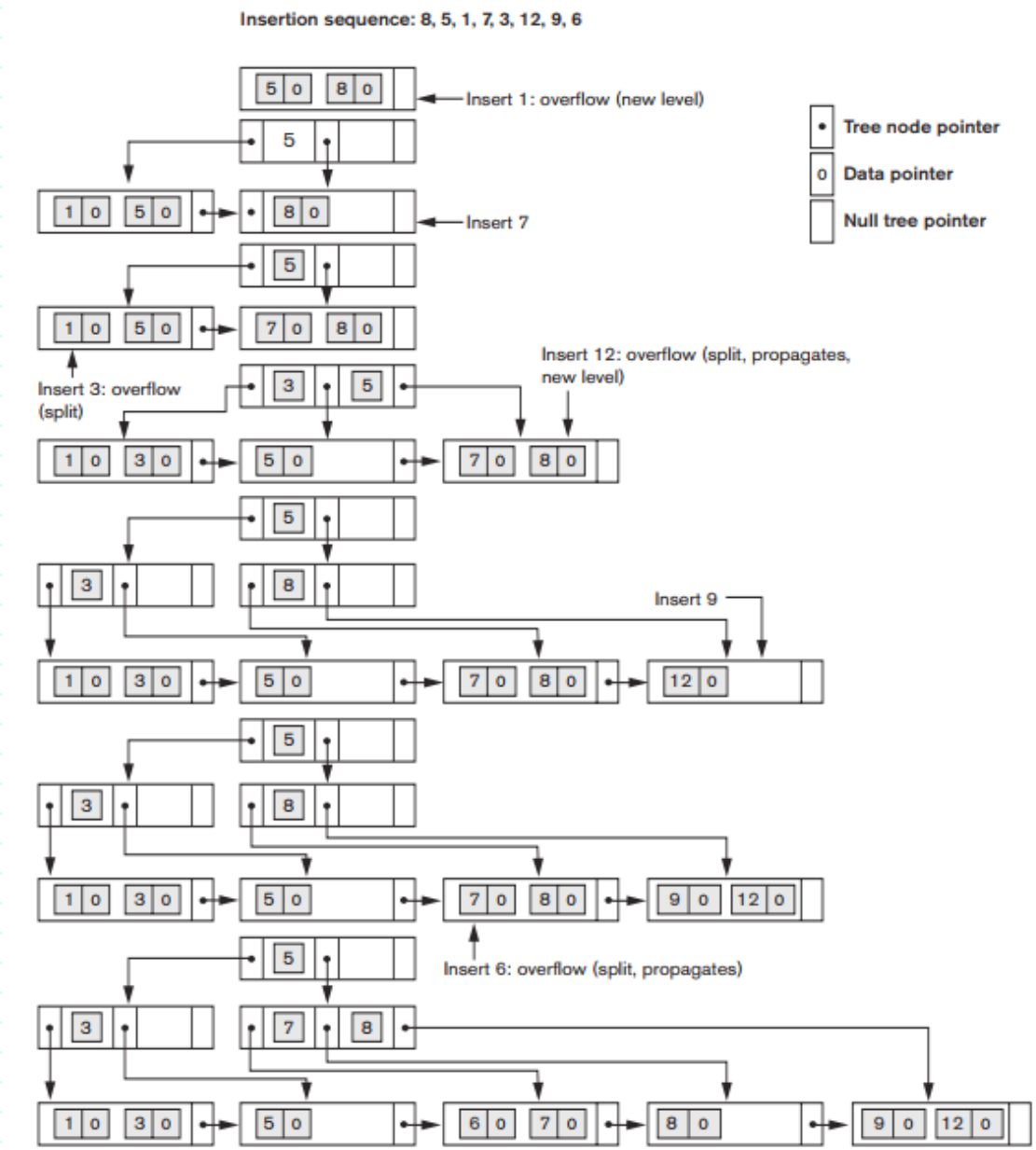
B+-Trees


(a)
P_1 K_1 ... K_{i-1} P_i K_i ... K_{q-1} P_q
Tree pointer
Tree pointer
Tree pointer
X
X
X
$X \leq K_1$
$K_{i-1} < X \leq K_i$
$K_{q-1} < X$
(b)
K_1 Pr_1 K_2 Pr_2 ... K_i Pr_i ... K_{q-1} Pr_{q-1} P_{next}
Pointer to next leaf node in tree
Data pointer
Data pointer
Data pointer
Data pointer

B+-Trees

- Advantages of two different node types?
- Insertion and deletion?

Insertion Example



Deletion Example