Database Management Systems

- Optimization

Washington University in St. Louis

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?

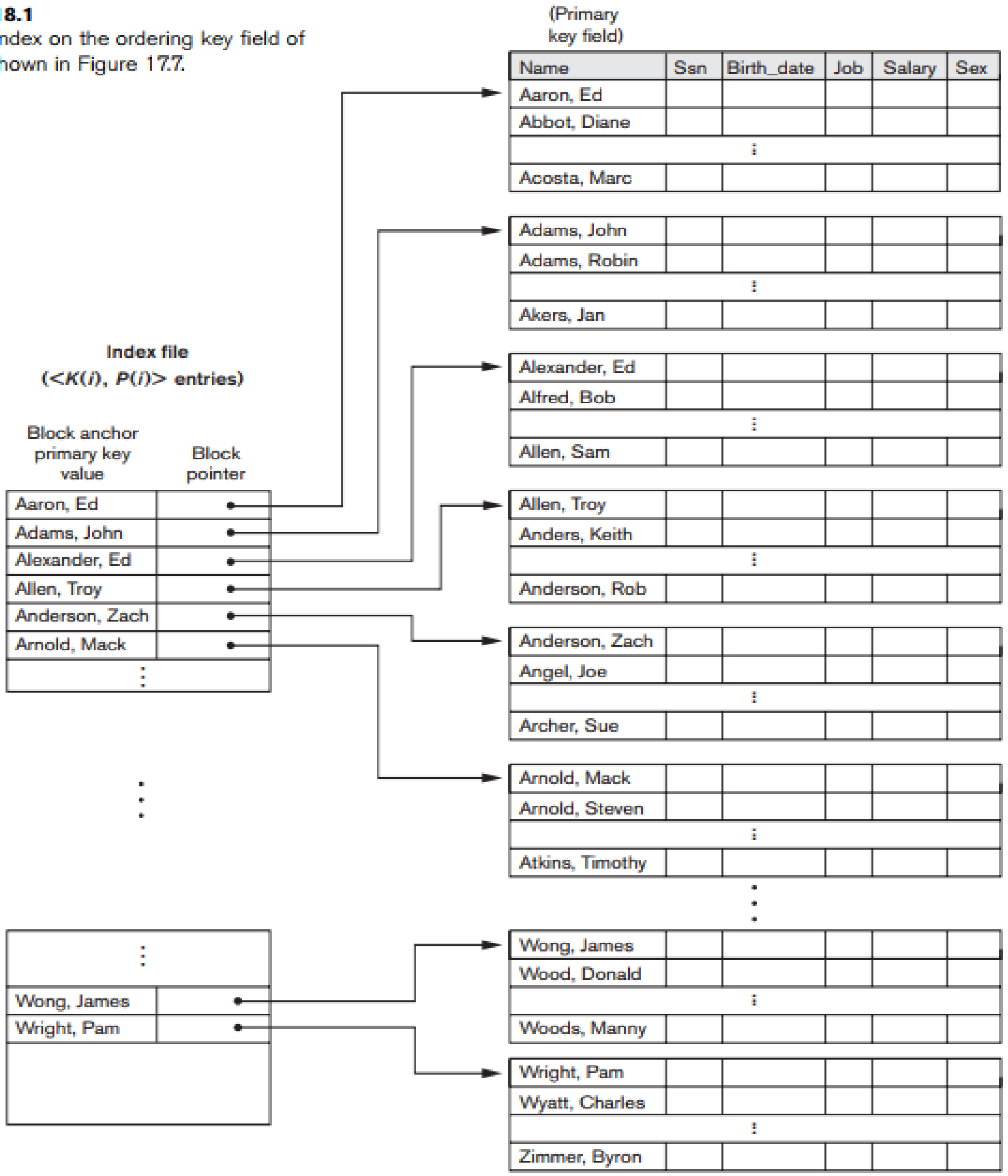


Figure 18.1
Primary index on the ordering key field of the file shown in Figure 17.7.

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?

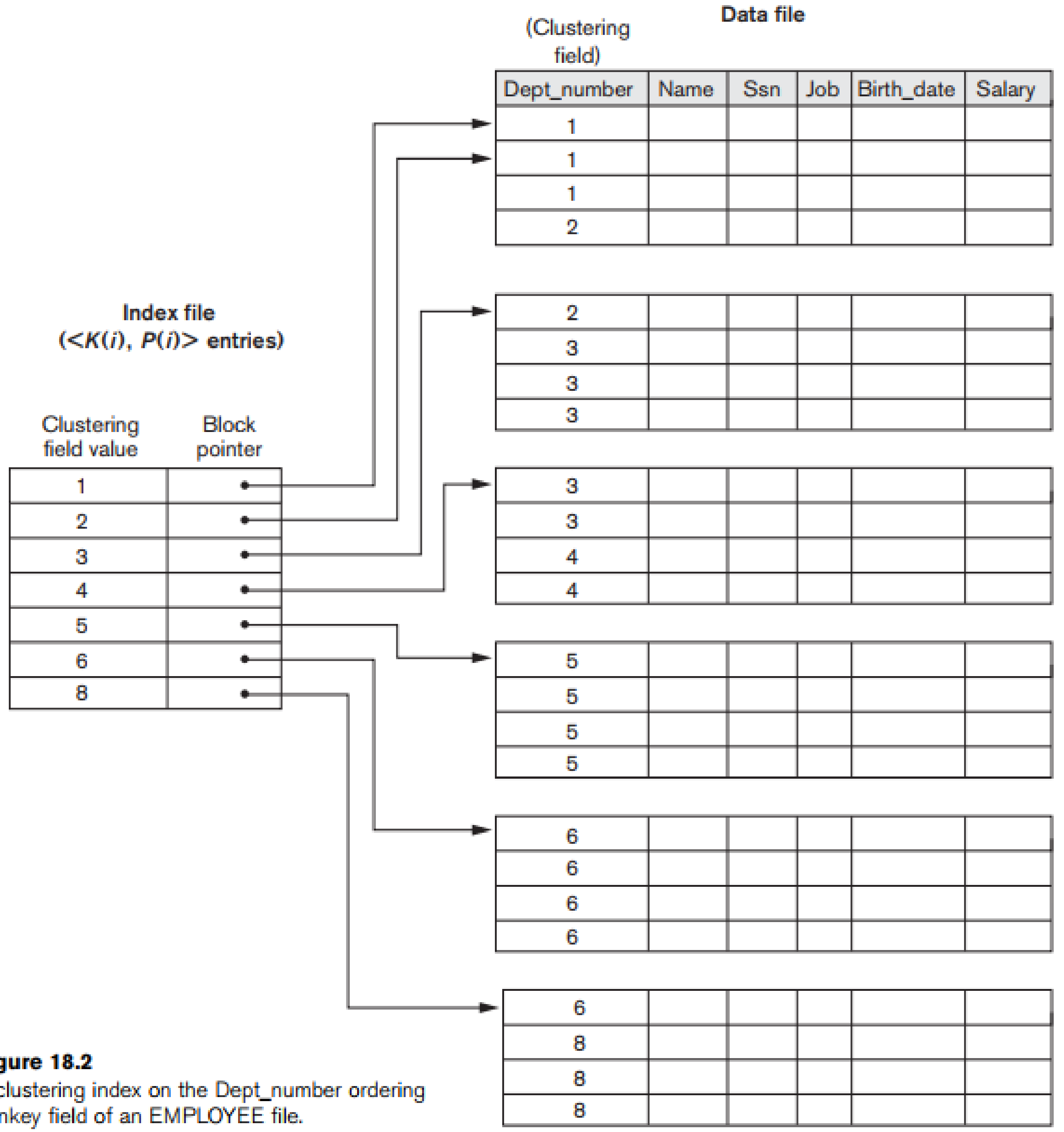


Figure 18.2
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file.

Secondary Index

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

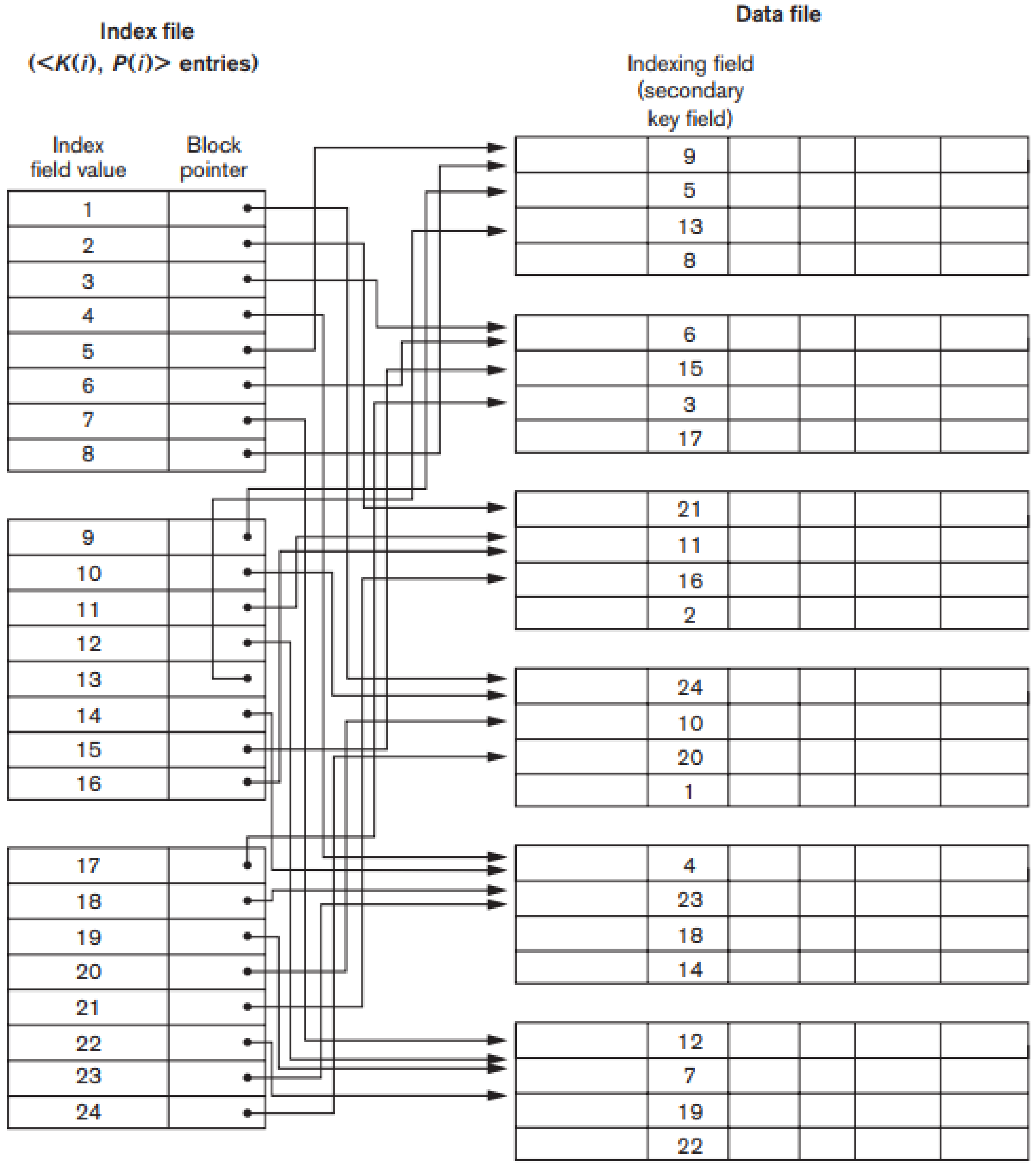
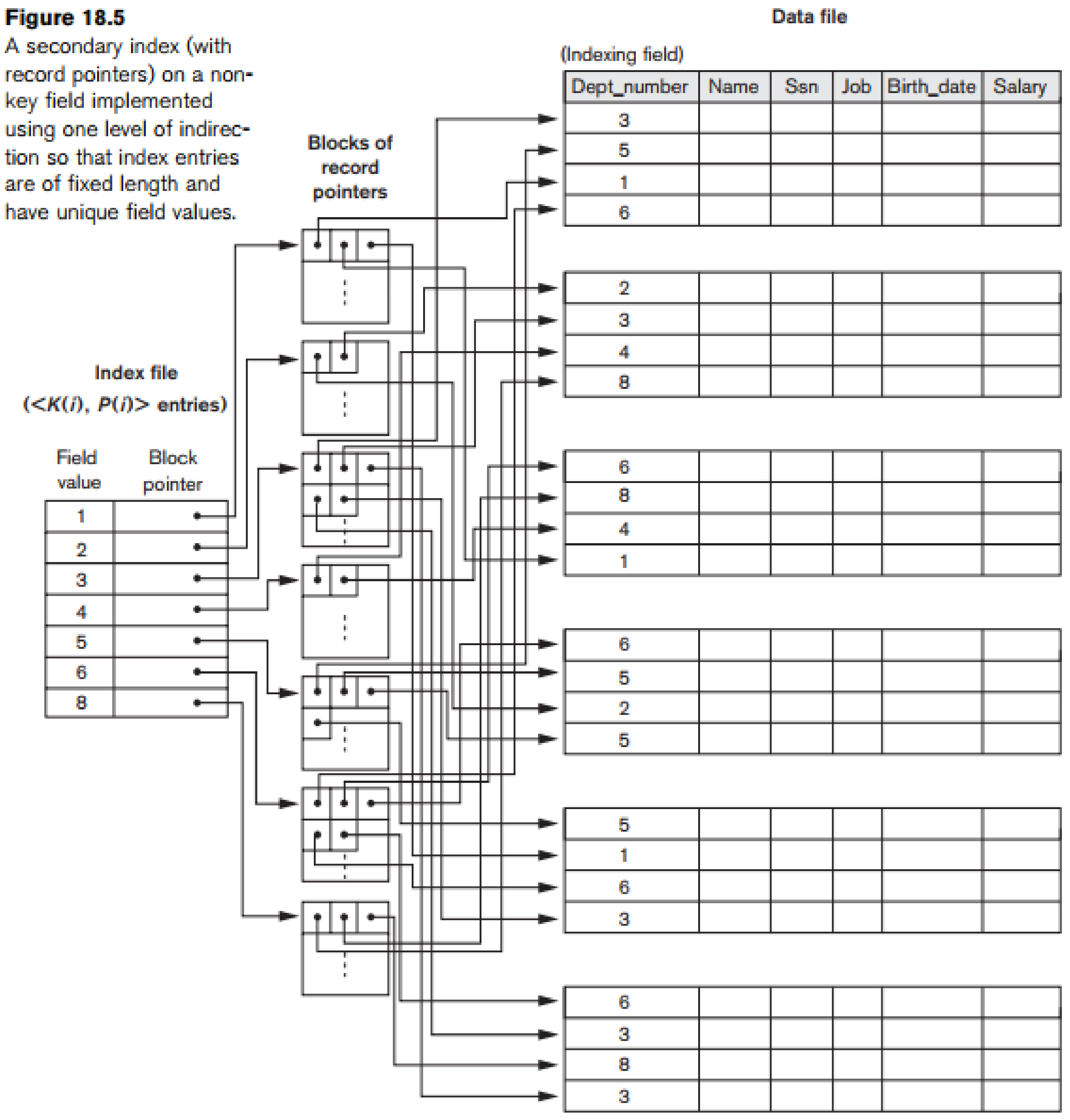

Index file
(<K(i), P(i)> entries)
Index field value
Block pointer
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
Data file
Indexing field (secondary key field)
9
5
13
8
6
15
3
17
21
11
16
2
24
10
20
1
4
23
18
14
12
7
19
22

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?

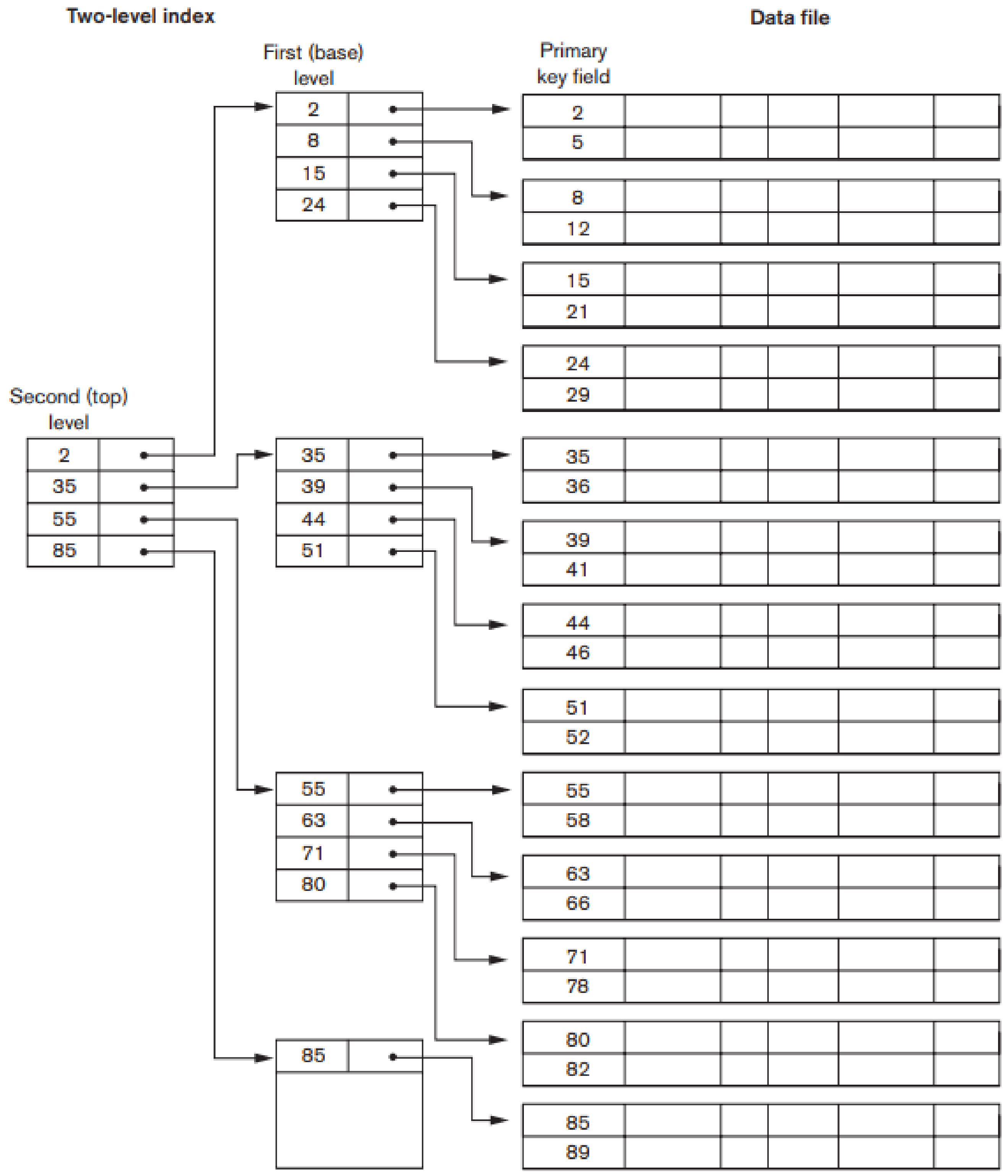
Multilevel Indexes

- 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?


Two-level index
Data file
First (base) level
Primary key field
Second (top) level
2
35
55
85
2
8
15
24
35
39
44
51
55
63
71
80
85
2
5
8
12
15
21
24
29
35
36
39
41
44
46
51
52
55
58
63
66
71
78
80
82
85
89

Multilevel Index

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

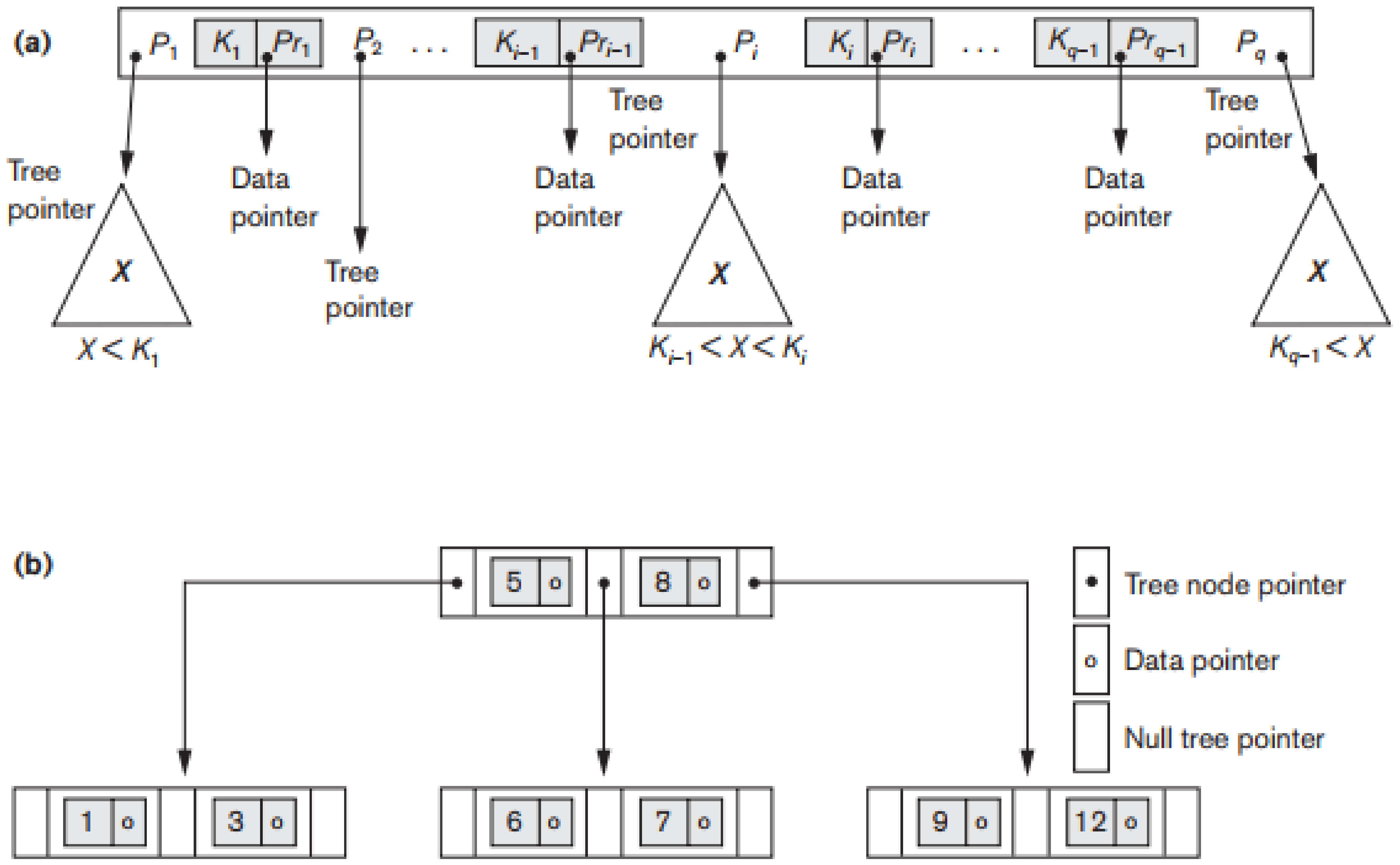
Dynamic Multilevel Index

- 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)


(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
8
1
3
6
7
9
12
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?