COSC344
Database Theory and Applications

Lecture 18
Indexing (1)

Overview

- Last Lecture
  - Database files and storage
- This Lecture
  - Database indexing
    - Single-Level Ordered Index
      - Primary Index
      - Clustered index
      - Secondary Index
  - Source: Chapter 17
- Next Lecture
  - Dynamic Multi-Level Index
    - Trees
    - B^+-trees
  - Source: Chapter 17

Question to Ponder

If we have a few million records, how do we find the one(s) we want?

Consider searching on any attribute.

For example

A book index *(try "disks" in Elmasri and Navathe)*

gives a list of pages (addresses)
then search the pages (disk blocks) for what we want

Index

Google has a large **index of keywords** and where those words can be found.

Indexes

- Additional files on disk
- Alternative ways to access the records without affecting the physical placement of records on disk
- Index structure
  - Indexing field
  - A list of pointer(s) to disk blocks
- The index key field is used to physically order the records of the index on disk. Every record has a unique value for that field.
- The index file is much smaller than the data. Binary search on the index is much faster than that on the data file.

Single-Level Ordered Indexes

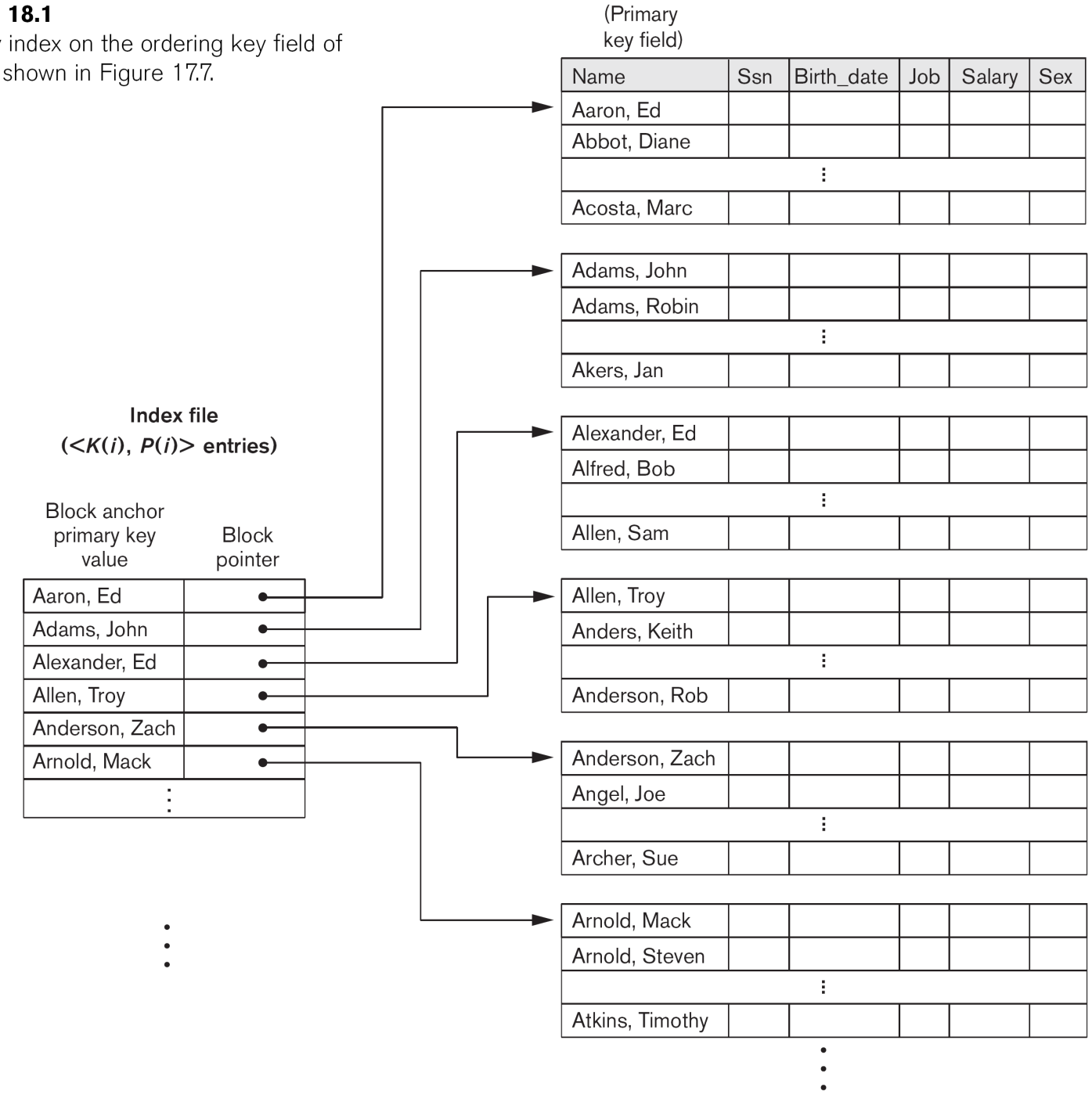
- A file can have, at most, one physical ordering field – related to how the data is physically stored.
- Primary index (key+ordering)
  - specified on the ordering key field of an ordered file
  - unique
- Clustering index (non-key+ordering)
  - ordering field is not a key field
  - more than one record has the same value
  - A file can have at most one primary index or one cluster index, but not both
- Secondary index (non-ordering)
  - specified on any non-ordering field
  - Can have several secondary indexes

Primary Index

- For an ordered file whose ordering field is a key
- A primary index is an ordered file whose records are of fixed length with two fields
  - Data from the ordering key field
  - Pointer to the data block
- Number of index entries = number of disk blocks
  - $< K_i, P_i >$: index entry for block i
  - Anchor record or block anchor – first record of the block
- Dense index has an index entry for every record
- Sparse (nondense) index has index entries for only some of the search values.
- A primary index is a nondense (sparse) index

Primary Index (cont.)

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



Primary Index (cont.)

r = 300,000 records
R = 460 bytes per record
B = 1024 bytes
bfr = floor(1024/460) = 2 (how many records in each block)
blocks = ceiling(300,000/2) = 150,000 (how many blocks for the file)
binary search = $\log_2(150{,}000)$ = 18 accesses (without index)

Ordering field = 10 bytes
pointer = 8 bytes
bfr_i = floor(1024/18) = 56 (how many index records in one block)
$blocks_i$ = ceiling(150,000/56) = 2679 (how many blocks for index file)
binary search on index = $\log_2(2679)$ = 12 accesses
+ 1 access to the datafile block itself
13 accesses in total (with index)

Primary Index (cont.)

- Insertion and deletion problems
  - Moving records will change some index entries as the anchor records of some blocks may change
  - Reorganisation

Clustering Index

- For an ordered file whose ordering field is not a key
  – not unique
- Clustering index has same values as the field (clustering field)
- Same problems as primary index for insertion
- Non-dense index

Clustering Index (cont.)

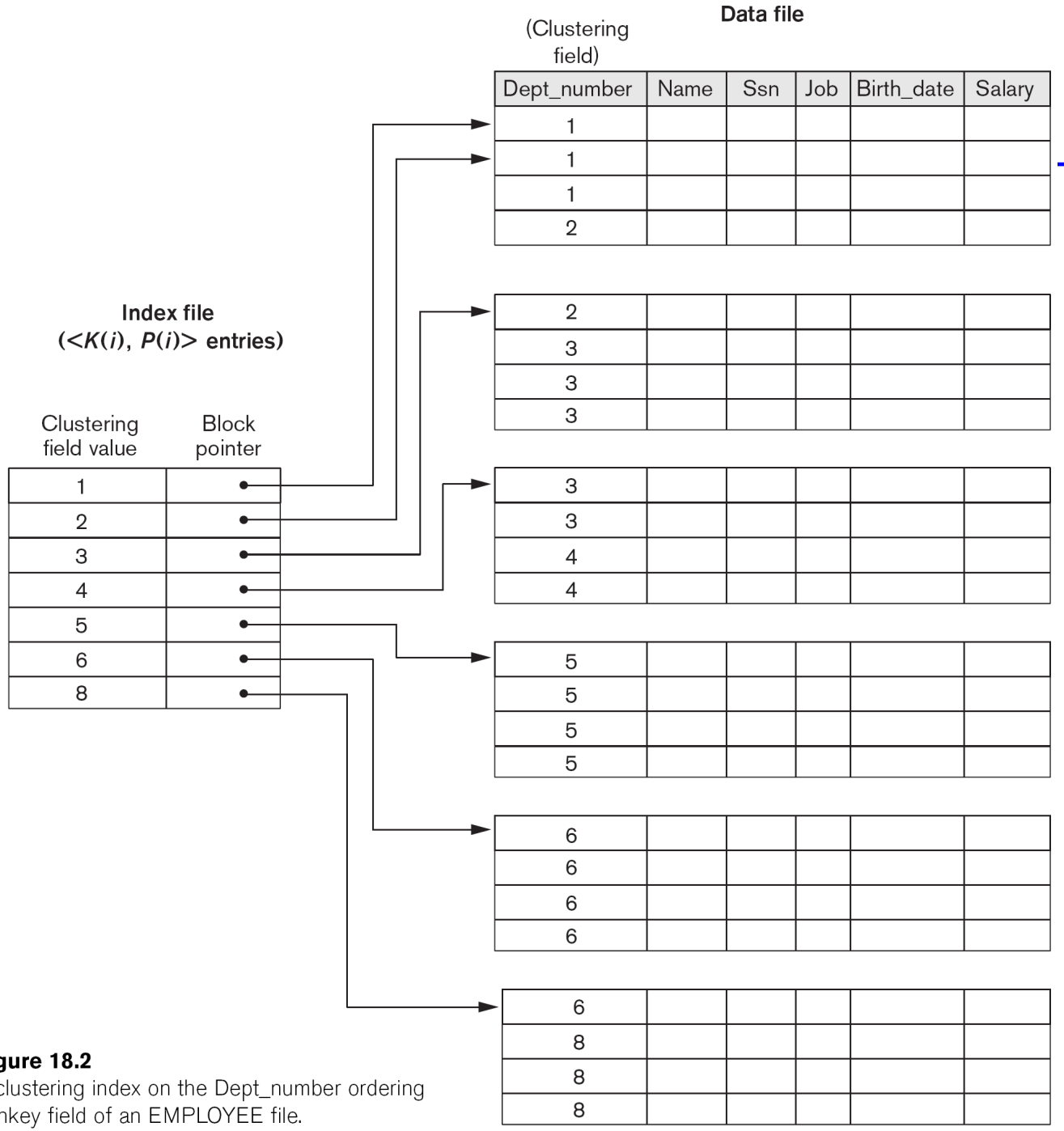
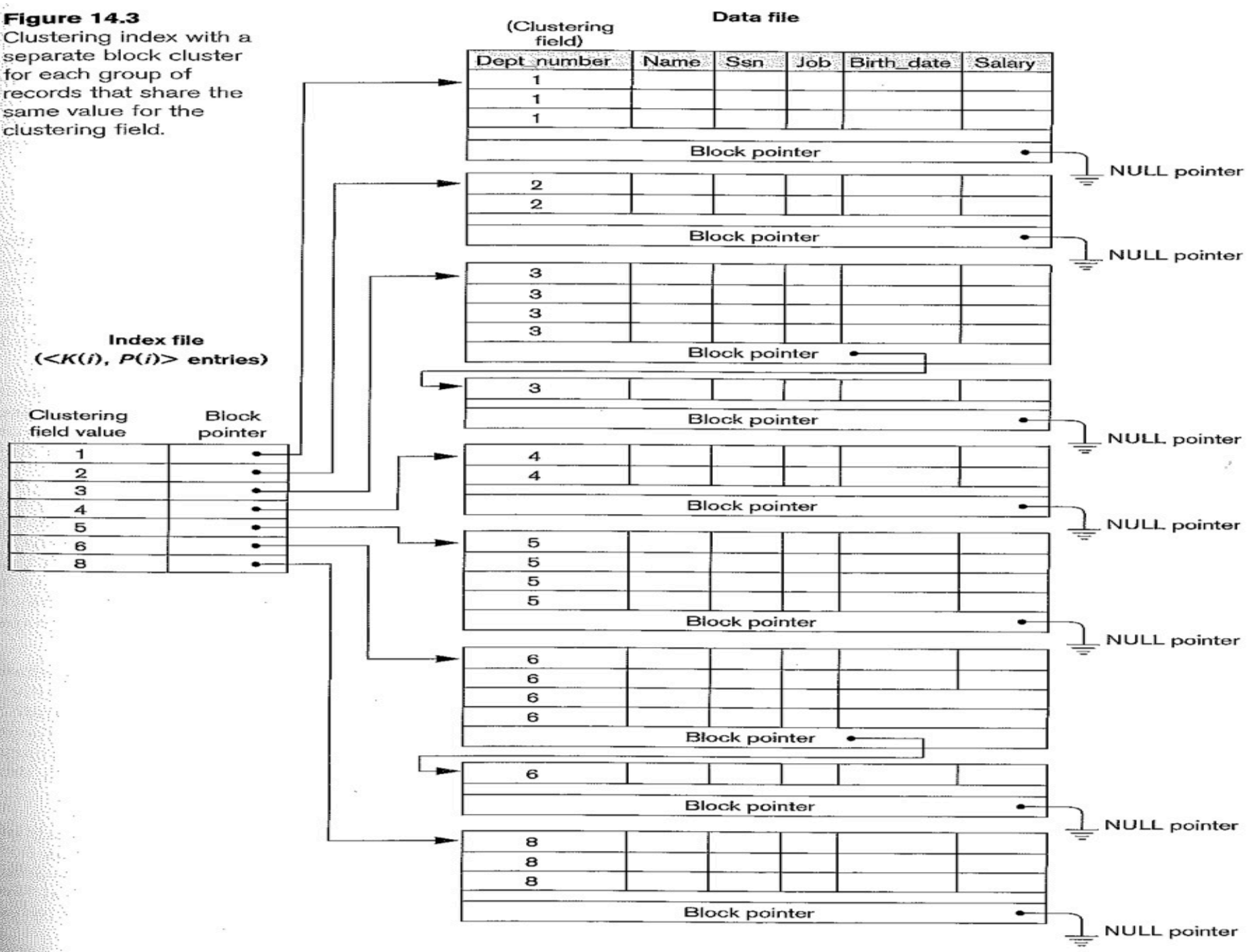


Figure 18.2
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file.

Figure 14.3
Clustering index with a separate block cluster for each group of records that share the same value for the clustering field.



Secondary Indexes

- A secondary means of accessing a file for which a primary access already exists
- Secondary indexes are for unordered files or for attributes which are not the ordering field.
- Two types
  - Secondary index on a key field
  - Secondary index on a non-key field
- Can be many secondary indexes

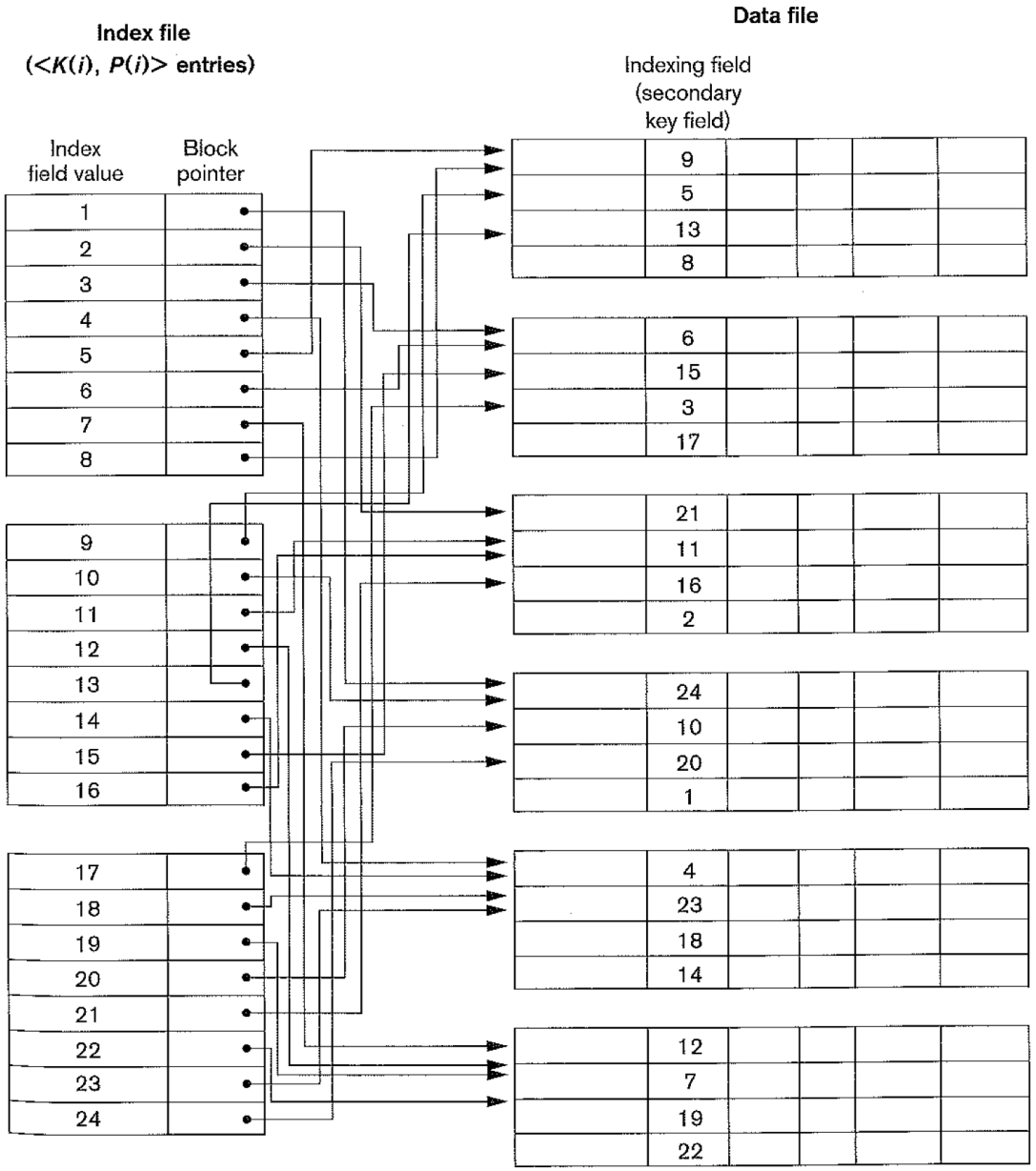
Secondary Index on a Key Field

- Uses a non-ordering field
- Key field -> unique values
- Two entries
  - Data from some non-ordering files
  - Pointer to the data
- Dense index

Secondary Indexes (cont.)

Non-ordering key field

Figure 14.4
A dense secondary index (with block pointers) on a nonordering key field of a file.



Secondary Index on a Key Field (cont.)

r = 300,000 records
R = 460 bytes per record
B = 1024 bytes
bfr = floor(1024/460) = 2
blocks = ceiling(300,000/2) = 150,000
search = b/2 = 75,000 accesses

Ordering field = 10 bytes
pointer = 8 bytes
bfr_i = floor(1024/18) = 56
$blocks_i$ = ceiling(300,000/56) = 5358
binary search on index = $\log_2(5358)$ = 13 accesses
14 accesses

Secondary Indexes on a Non-key Field

- Many records can have the same value for the indexing field
- Options
  - Several index entries for the same field value
    - Dense index
  - Variable-length records for the index entries with a repeating field for the pointers
    - Non-dense index
  - Create an extra level of indirection to handle multiple pointers
    - Non-dense index
    - Linked list of pointer blocks if too many pointers

Secondary Indexes on a Non-key Field (cont.)

Note:
Index file – block pointers,
Extra indirection – record pointers

Note:
If we access
the records in the order
of the entries
in the secondary index,
we get them in order
of the indexing field.

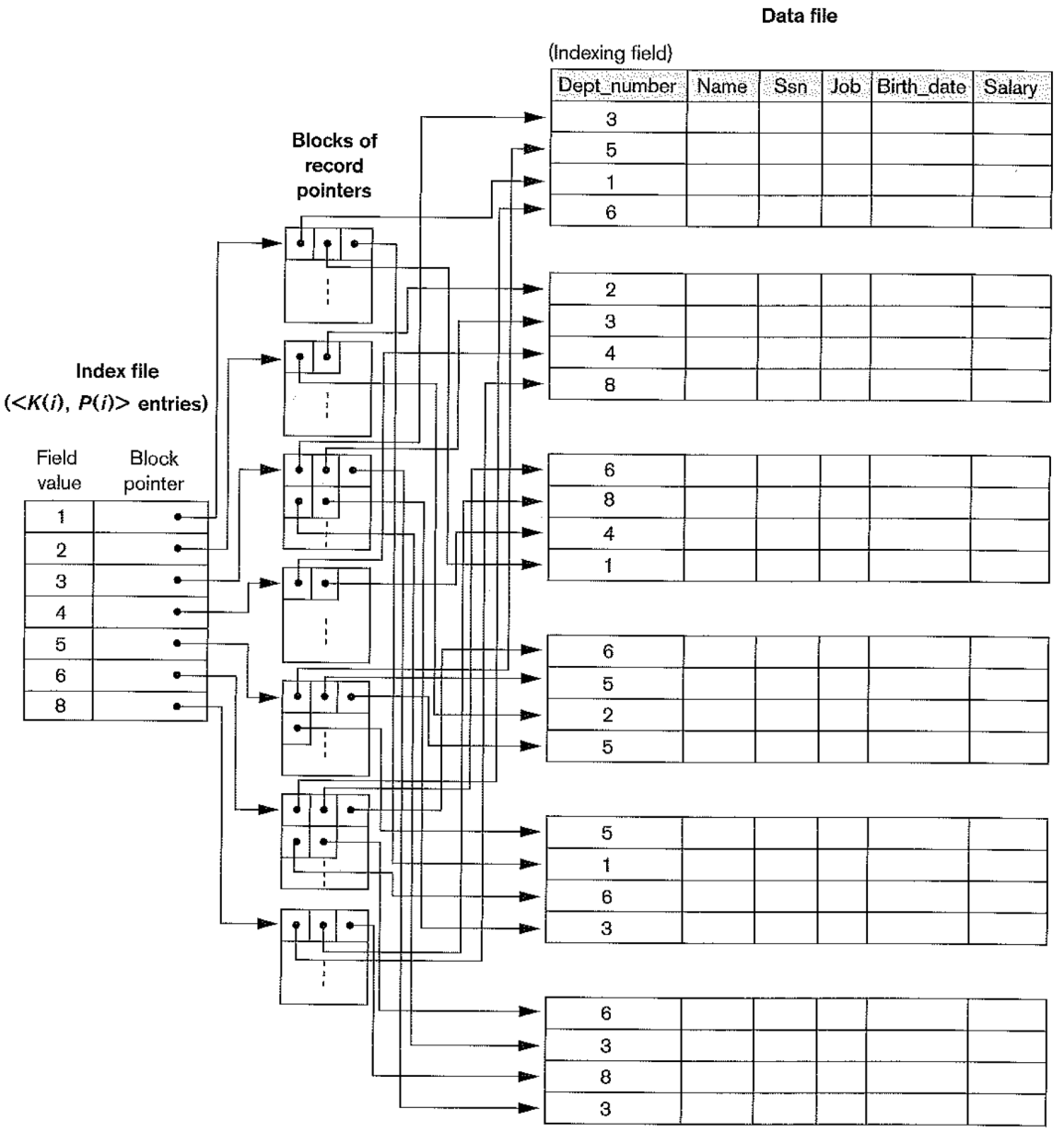


Figure 14.5
A secondary index (with record pointers) on a nonkey field implemented using one level of indirection so that index entries are of fixed length and have unique field values.

Summary: Single-Level Index Types

Table 14.1
Types of Indexes Based on the Properties of the Indexing Field

	Index Field Used for Ordering the File	Index Field Not Used for Ordering the File
Indexing field is key	Primary index	Secondary index (Key)
Indexing field is nonkey	Clustering index	Secondary index (NonKey)

Summary: Single Level Index Properties

Table 14.2
Properties of Index Types

Type of Index	Number of (First-level) Index Entries	Dense or Nondense	Block Anchoring on the Data File
Primary	Number of blocks in data file	Nondense	Yes
Clustering	Number of distinct index field values	Nondense	Yes/no[a]
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	Number of records[b] or number of distinct index field values[c]	Dense or Nondense	No

[a]Yes if every distinct value of the ordering field starts a new block; no otherwise.
[b]For option 1.
[c]For options 2 and 3.

Next Lecture

- More complex index
  - Multi-level index
  - Dynamic multi-level index
    - B-Tree
    - B^+-Tree