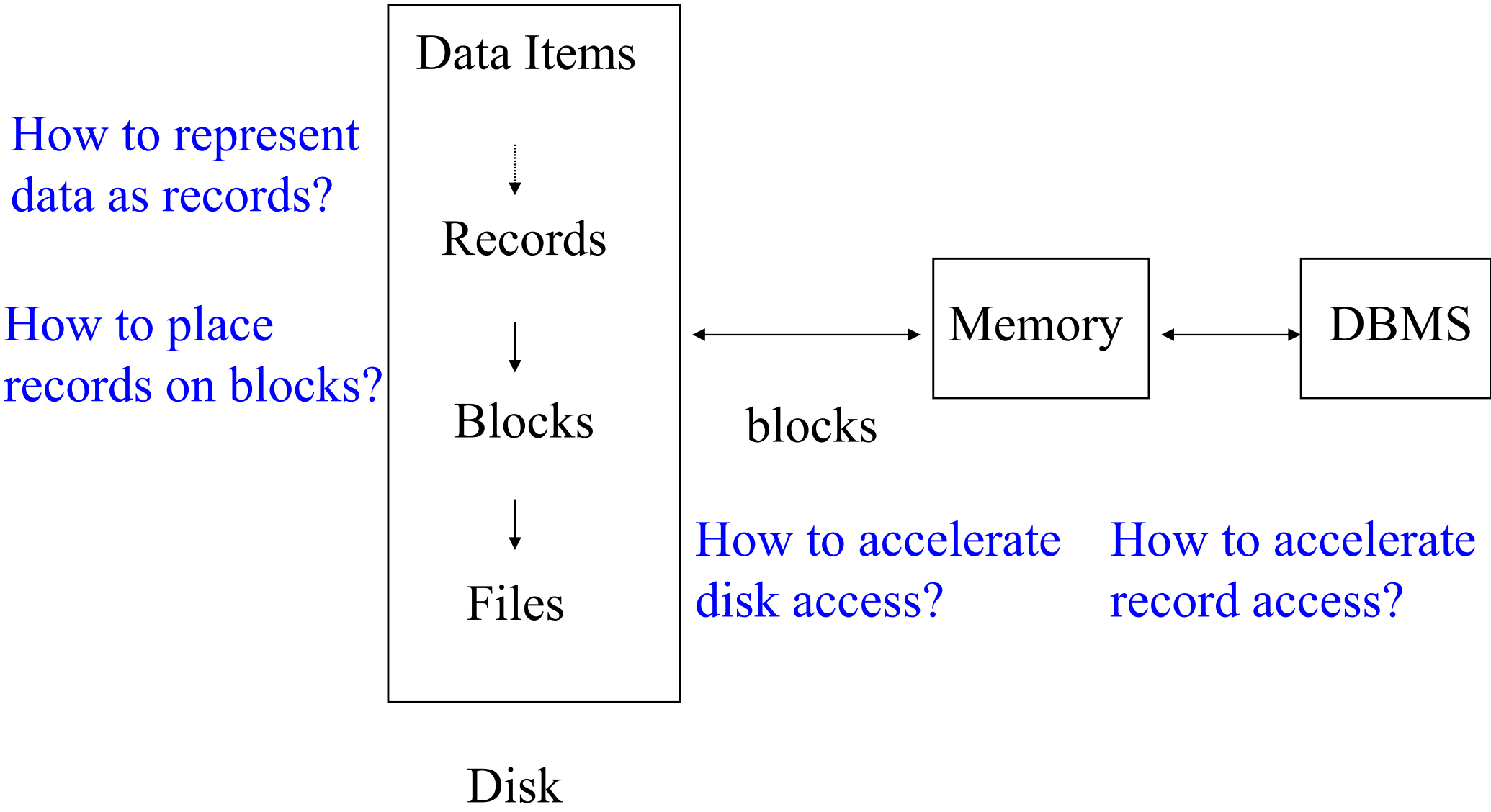
CS 377 Database Systems

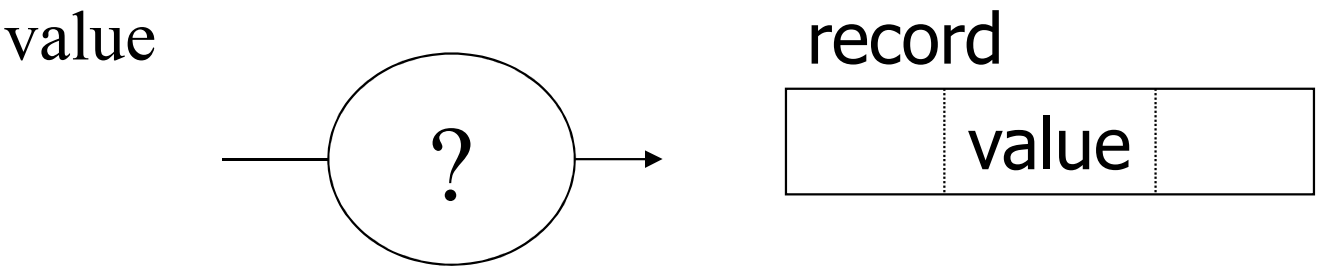
Indexing

Li Xiong
Emory University

Data Storage Preview



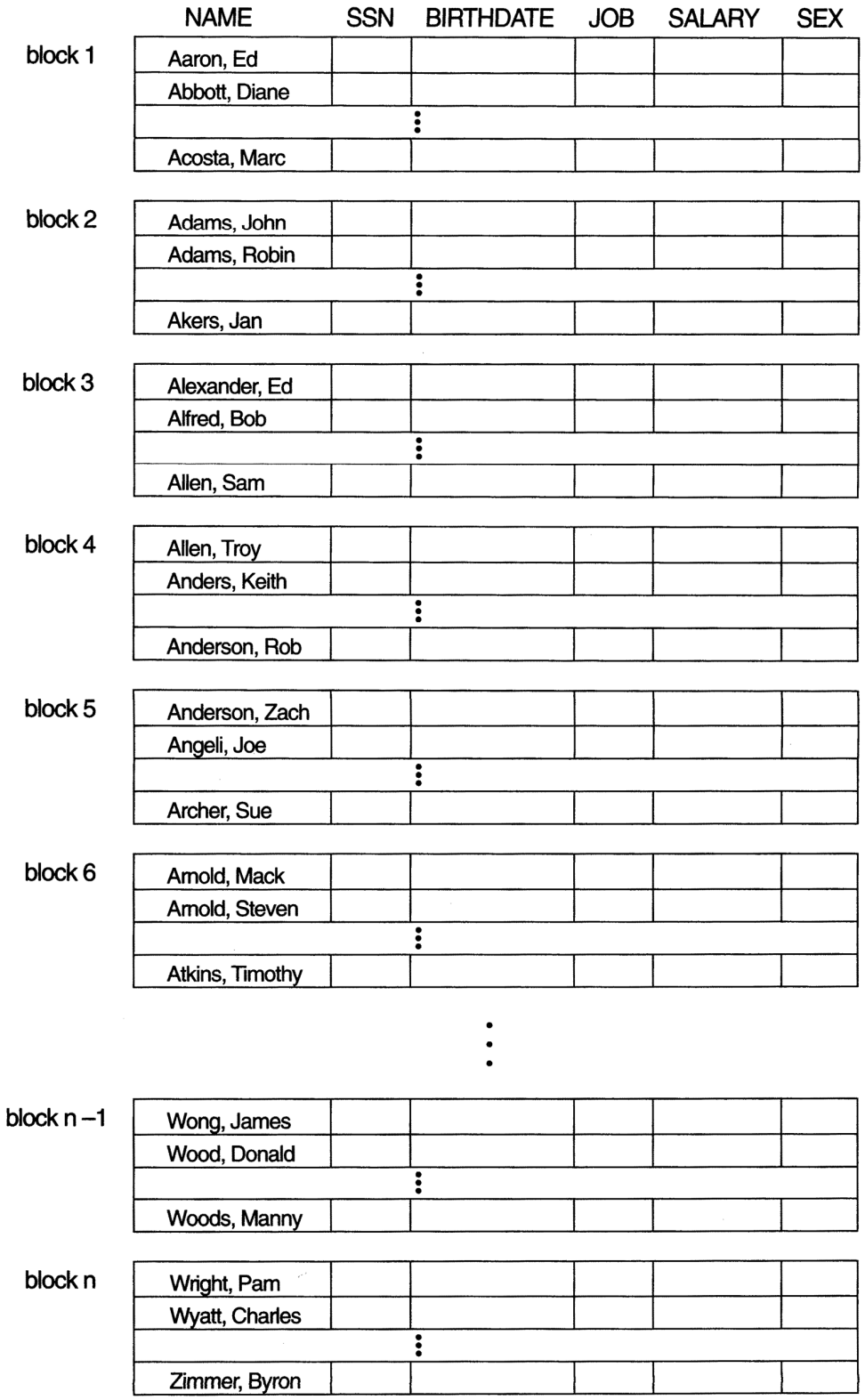
Record Access



Given a value, locate record(s) with this value

SELECT * FROM R WHERE A = value;
SELECT * FROM R, S WHERE R.A = S.B;

Ordered Files



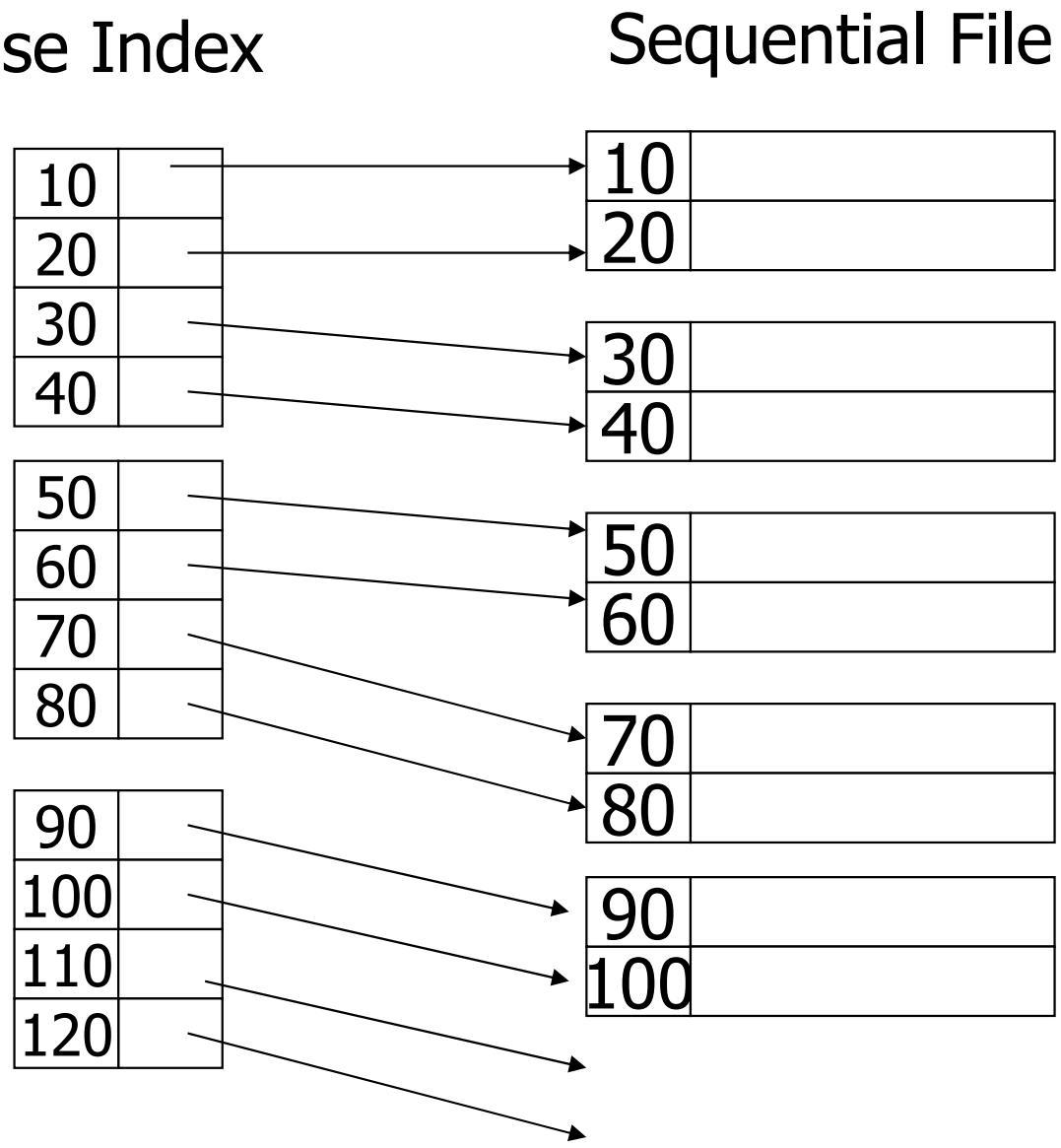
Indexes as Access Paths

- Indexes are auxiliary access structures (files) used to speed up the retrieval of records from data files
- An index is usually specified on one field of the data file, called indexing field
- An index typically contains (field value, block pointer) pairs
  - Field values are typically ordered
- A file may be indexed on different fields

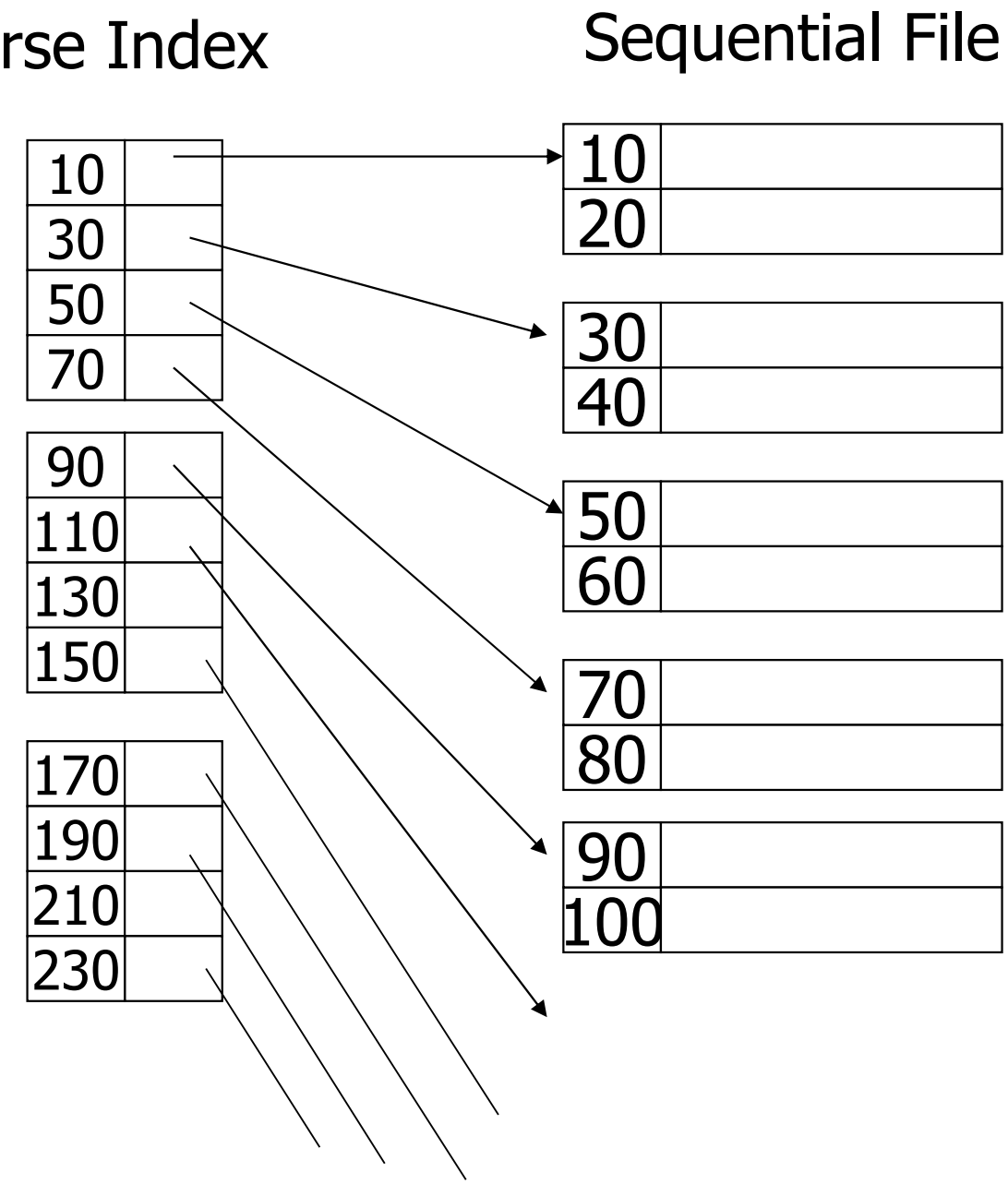
Indexes

- Dense vs. sparse indexes
  - A **dense index** has an index entry for every search key value (and hence every record) in the data file.
  - A **sparse (or nondense) index** has index entries for only some of the search values
- Single-level vs. multi-level indexes
  - A single-level index is one ordered index file
  - A multi-level index consists of indexes for indexes

- One index entry per data key (record)



- Some keys in the data file do not have an index entry
- Normally one index entry per data block



Sparse vs. Dense Index

- Sparse:
  - Less index space
  - Somewhat more and varied time to find a record within a block
- Dense:
  - More index space
  - Can tell if any record exists without accessing file

Single Level Indexes

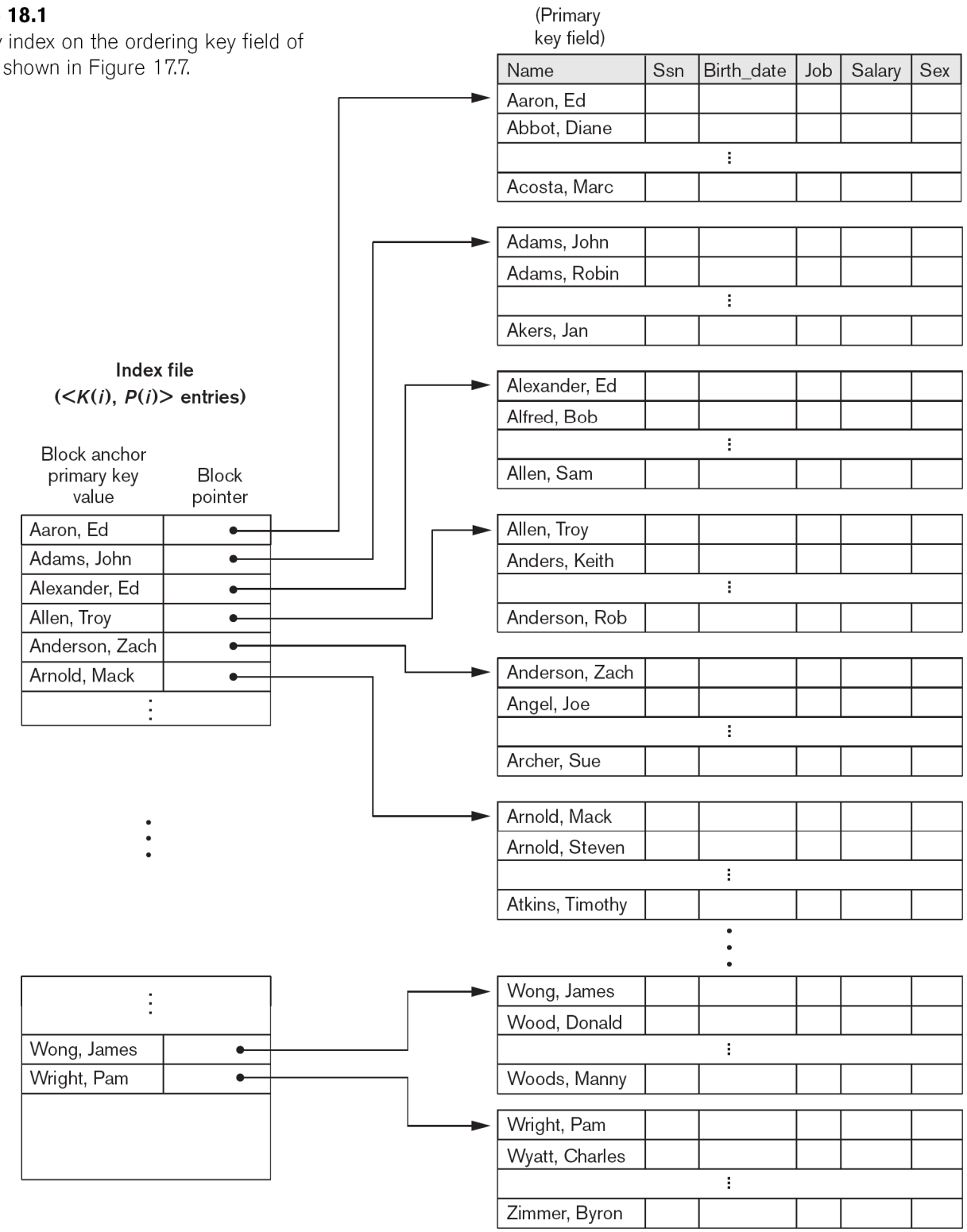
- Single-level ordered indexes on ordered files
  - Primary index – <ordering key field, pointer>
  - Clustering index - <ordering non-key field, pointer>
  - Secondary index - <non-ordering field, pointer>

Primary Index

- Primary Index
  - Defined on an ordering **key field** (unique)
  - Includes one index entry *for each block* in the data file
  - An index entry is *<K(i), P(i)>*
    - → K(i), the key field value for the first record in ith block, which is called block anchor
    - → P(i), pointer to the block
- Characteristics
  - Sparse index - it includes an entry for each block rather than for every search value
  - Few index entries than records
  - Each index entry is smaller in size than a record
  - Binary search on the index file requires fewer block accesses than on the data file

Primary Index

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



Example 1 – Search without Index

- Block size B=1024 bytes; unspanned blocking
- Ordered file for EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ...)
  - record size R=100 bytes
  - r=30000 records
  - blocking factor bfr = B/R= 1024/100= 10 records/block
  - number of file blocks b= (r/Bfr)= (30000/10)= 3000 blocks

- Average linear search cost for non-ordering fields:
  - (b/2)= 3000/2= 1500 block accesses
- Binary search cost for ordering-field:
  - $\log_2 b = \log_2 3000 = 12$ block accesses

Example 1 - Search with Index

- Index on the ordering field Name
  - Name field size V_{Name}=9 bytes
  - record pointer size P_R=6 byte
  - index entry size R_i=(V_{Name}+ P_R)=(9+6)=15 bytes
  - Number of index entries = number of data file blocks = 3000
  - index blocking factor bfr_i= B/R_i= 1024/15= 68 entries/block
  - number of index blocks bi = (3000/68)= 45 blocks

- Search cost
  - Binary search in the index: $\log_2$bi= $\log_2$45= 6 block accesses
  - Data access using the block pointer: 1 block access
  - Total block accesses: 7 blocks

Clustering Index

- Clustering Index
  - Defined on an ordering *non-key* field (not unique), which is called clustering field
  - Includes one index entry *for each distinct value* of the field
  - Index entry points to the first data block that contains records with that field value
- Characteristics
  - *Nondense* index
  - Insertion and Deletion is relatively straightforward

Clustering Index

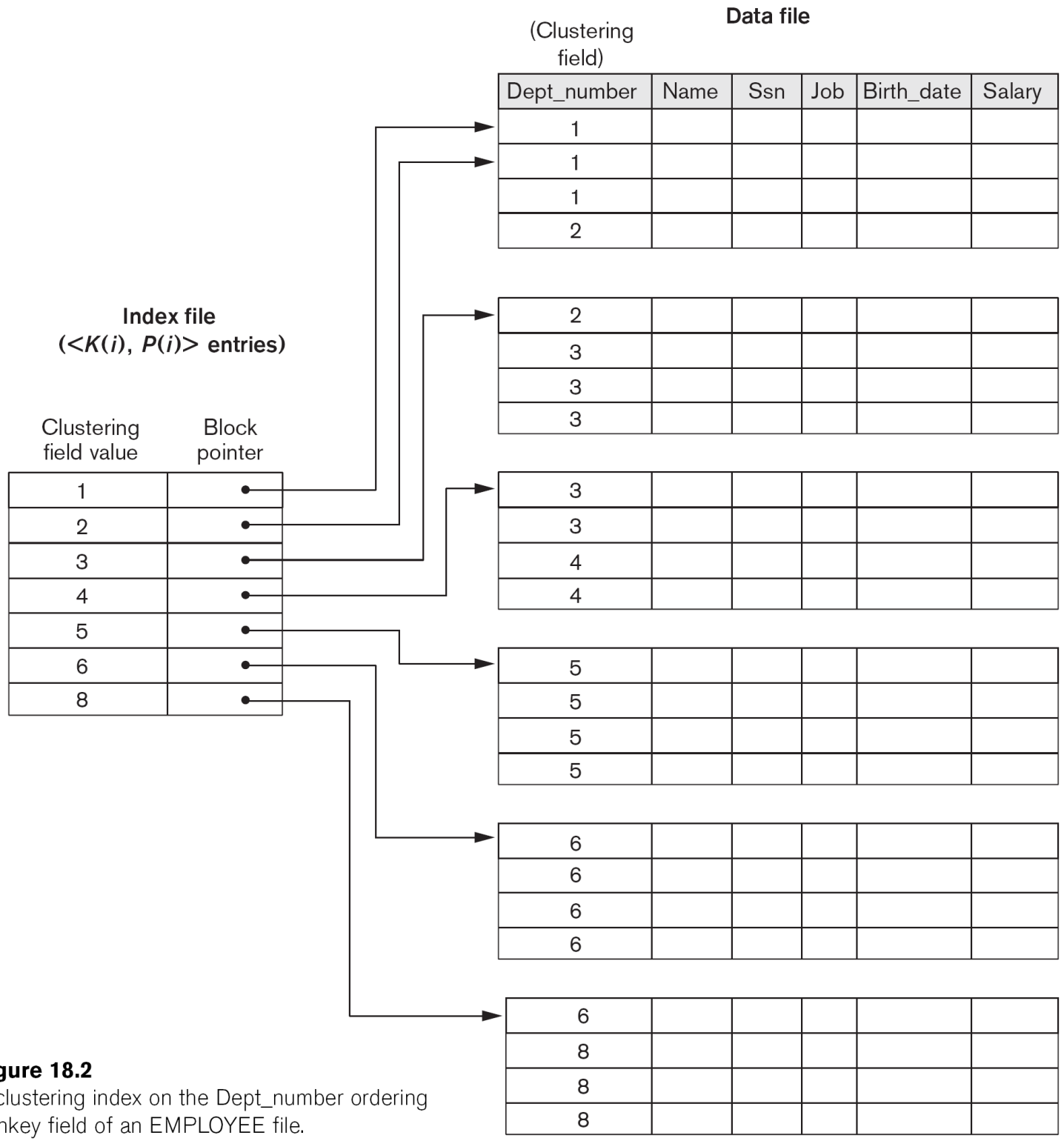
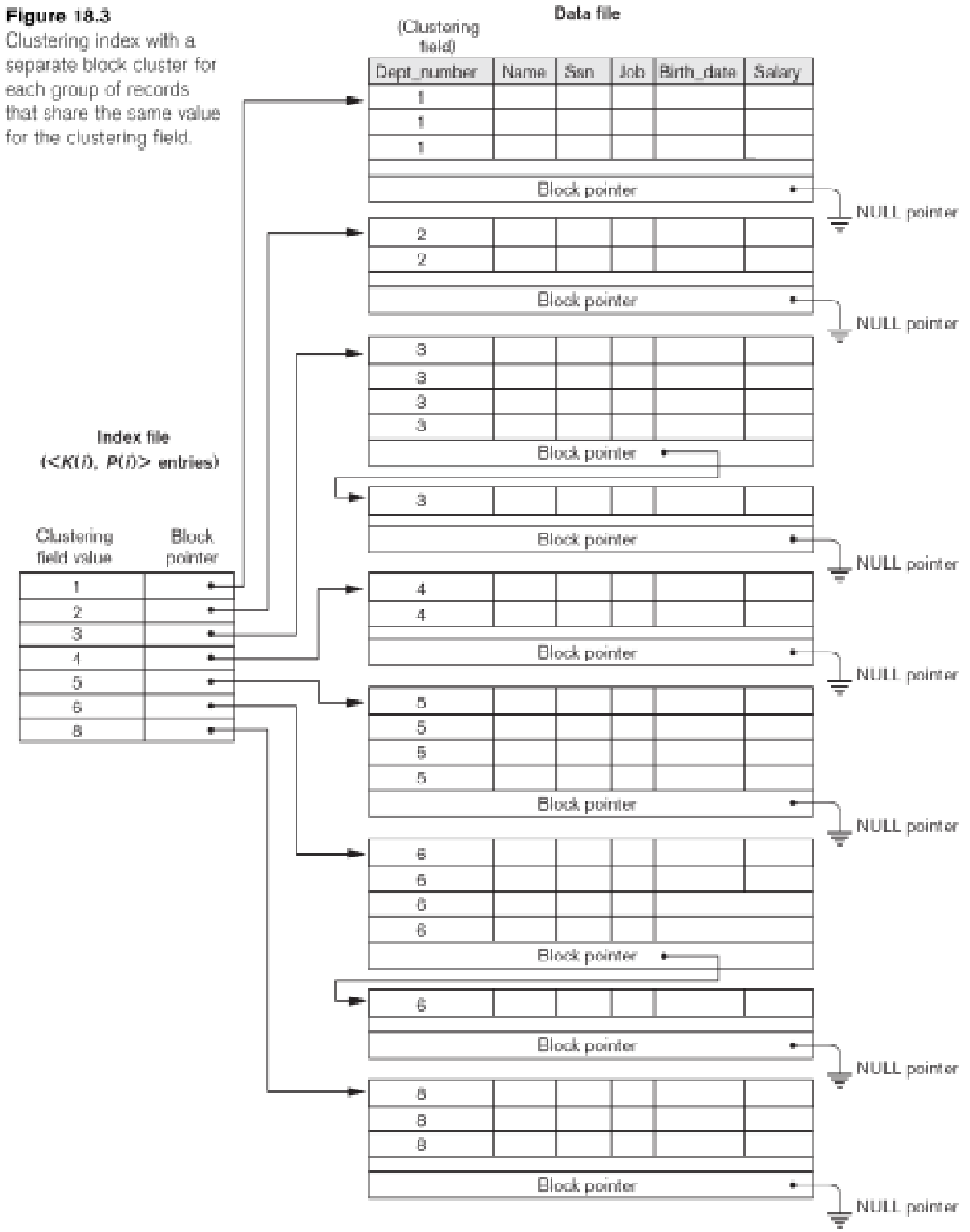


Figure 18.2
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file.

Another Clustering Index Example

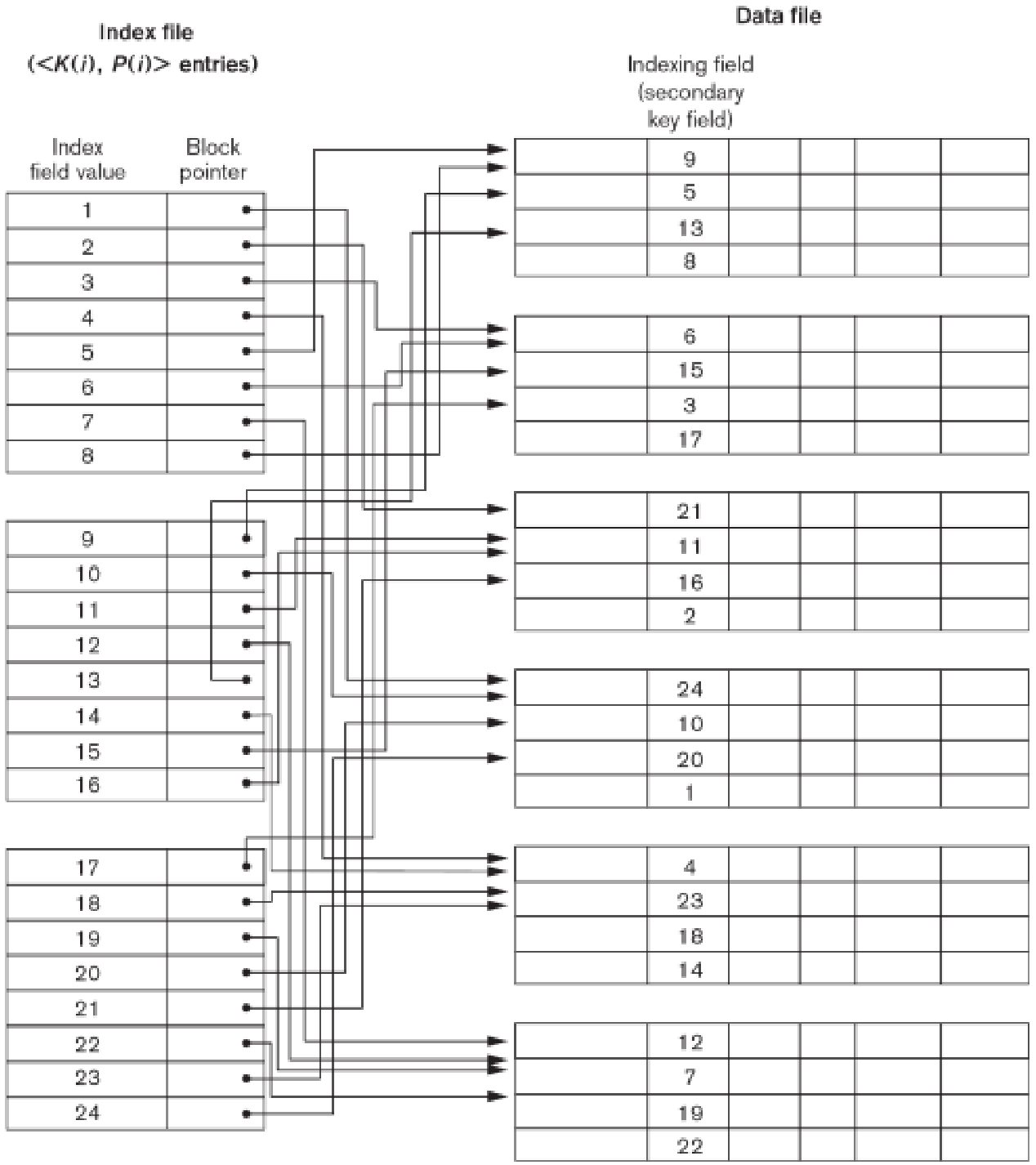


Secondary Index

- Secondary Index
  - Defined on a non-ordering field
  - Non-ordering key field (unique)
    - → An entry for each record (dense index)
  - Non-ordering non-key field (not unique), commonly:
    - → An entry for each distinct value (nondense index)
    - → The pointer points to a block of record pointers; each pointing to one of the records with that value
- Characteristics
  - Can be dense or nondense index
  - There can be *many* secondary indexes for the same file
  - A secondary index usually needs more space and longer search time
  - Greater improvement for an arbitrary record than primary index

Dense Secondary Index (on key field)

Figure 18.4
A dense secondary index (with block pointers) on a nonordering key field of a file.



Example 2 - Search with Dense Secondary Index

- Employee File with ordering field name (as in Example 1)
- Secondary index on the non-ordering field SSN
  - Name field size V_{SSN}=9 bytes
  - record pointer size P_R=6 byte
  - index entry size R_i=(V_{SSN}+ P_R)=(9+6)=15 bytes
  - Number of index entries = number of records = 30000
  - index blocking factor bfr_i= B/R_i= 1024/15= 68 entries/block
  - number of index blocks bi = (30000/68)= 442 blocks
- Search cost on non-ordering field SSN
  - Binary search in the index: $\log_2 bi = \log_2 442$= 9 block accesses
  - Data access using the block pointer: 1 block access
  - Total block accesses: 10 blocks
- Search cost on SSN without secondary index (linear search): 1500 blocks
- Search cost on ordering field with primary index: 7 blocks

secondary indexes on non-ordering non-key field

20	
10	

20	
40	

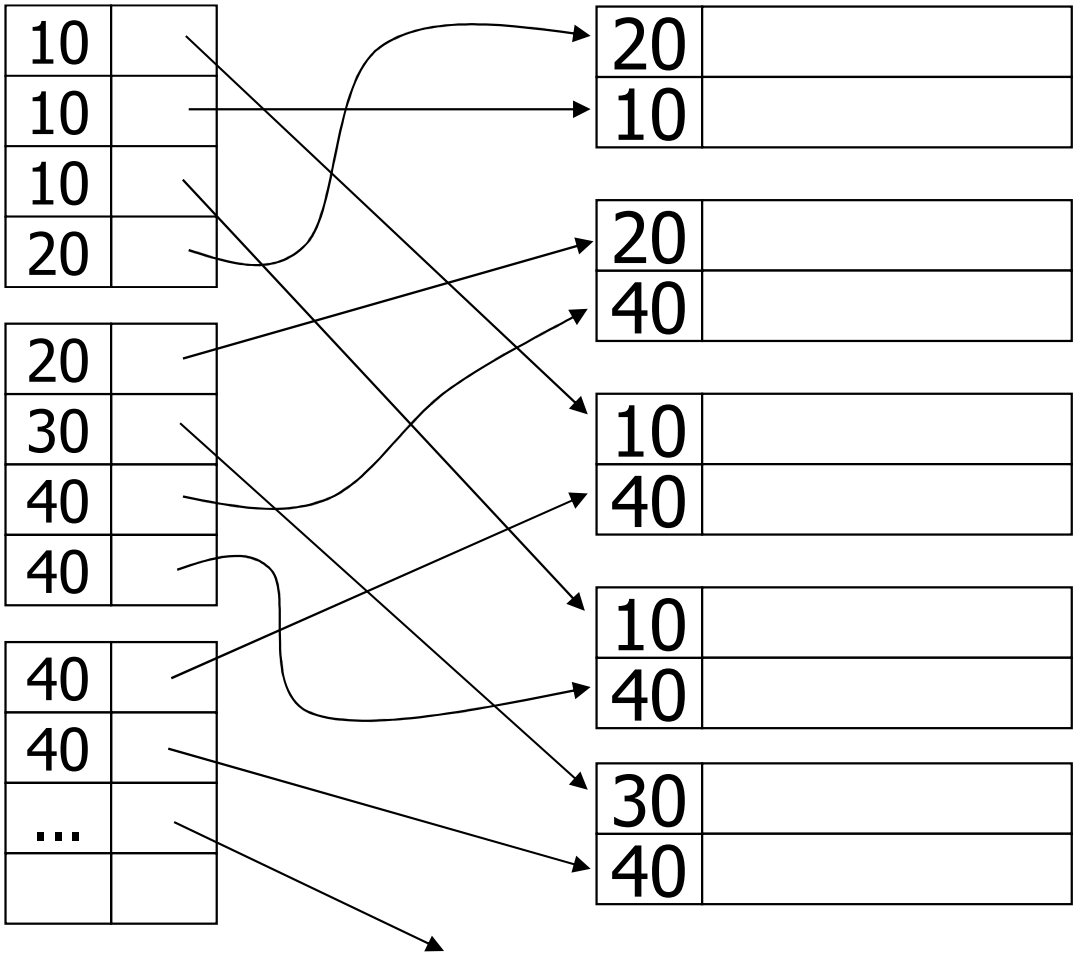
10	
40	

10	
40	

30	
40	

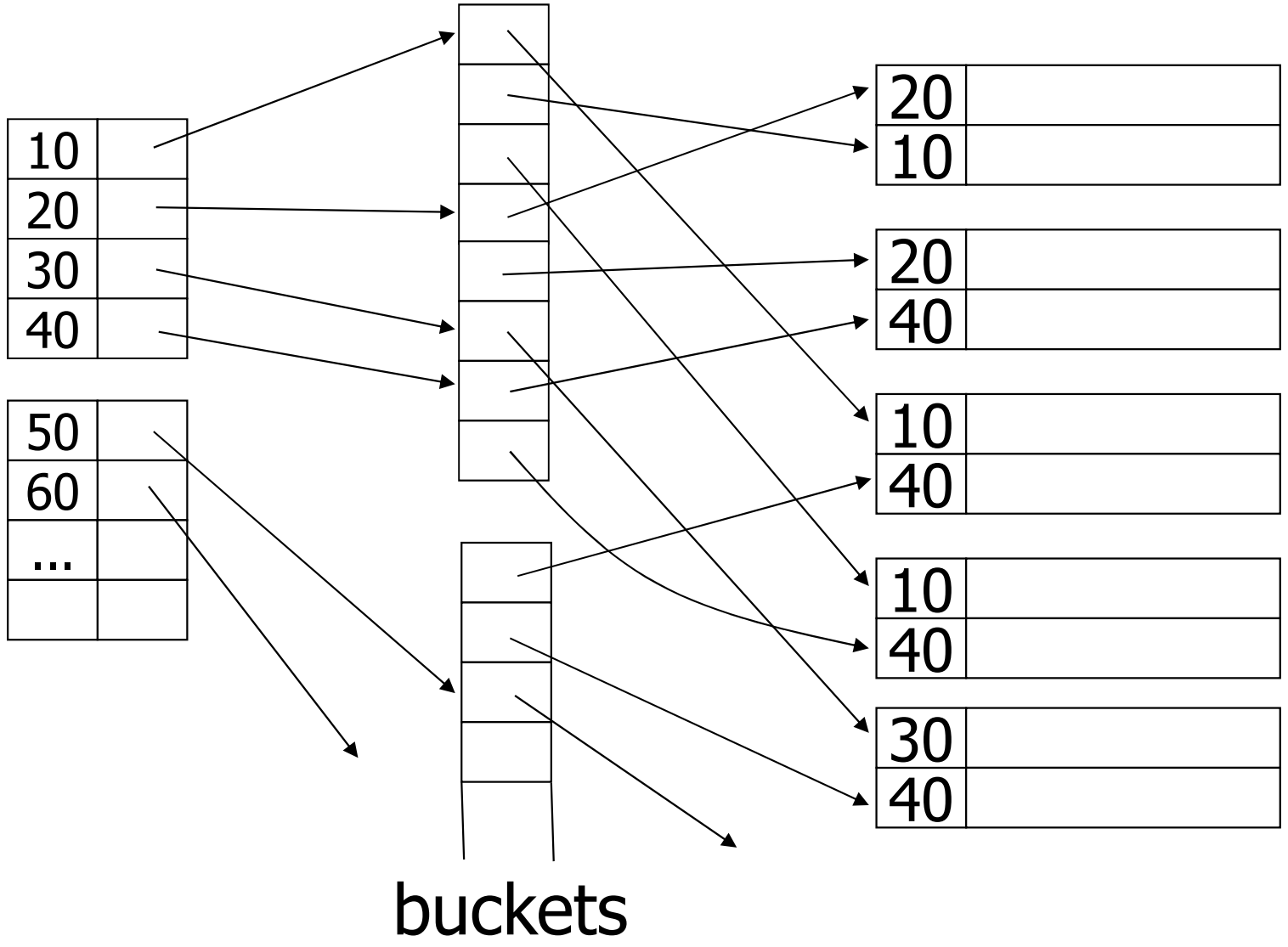
secondary indexes on non-key field

One option: duplicate index entries



Secondary indexes on non-key field

A common option (indirection): block pointer -> record pointer



Nondense Secondary Index (on non-key field)

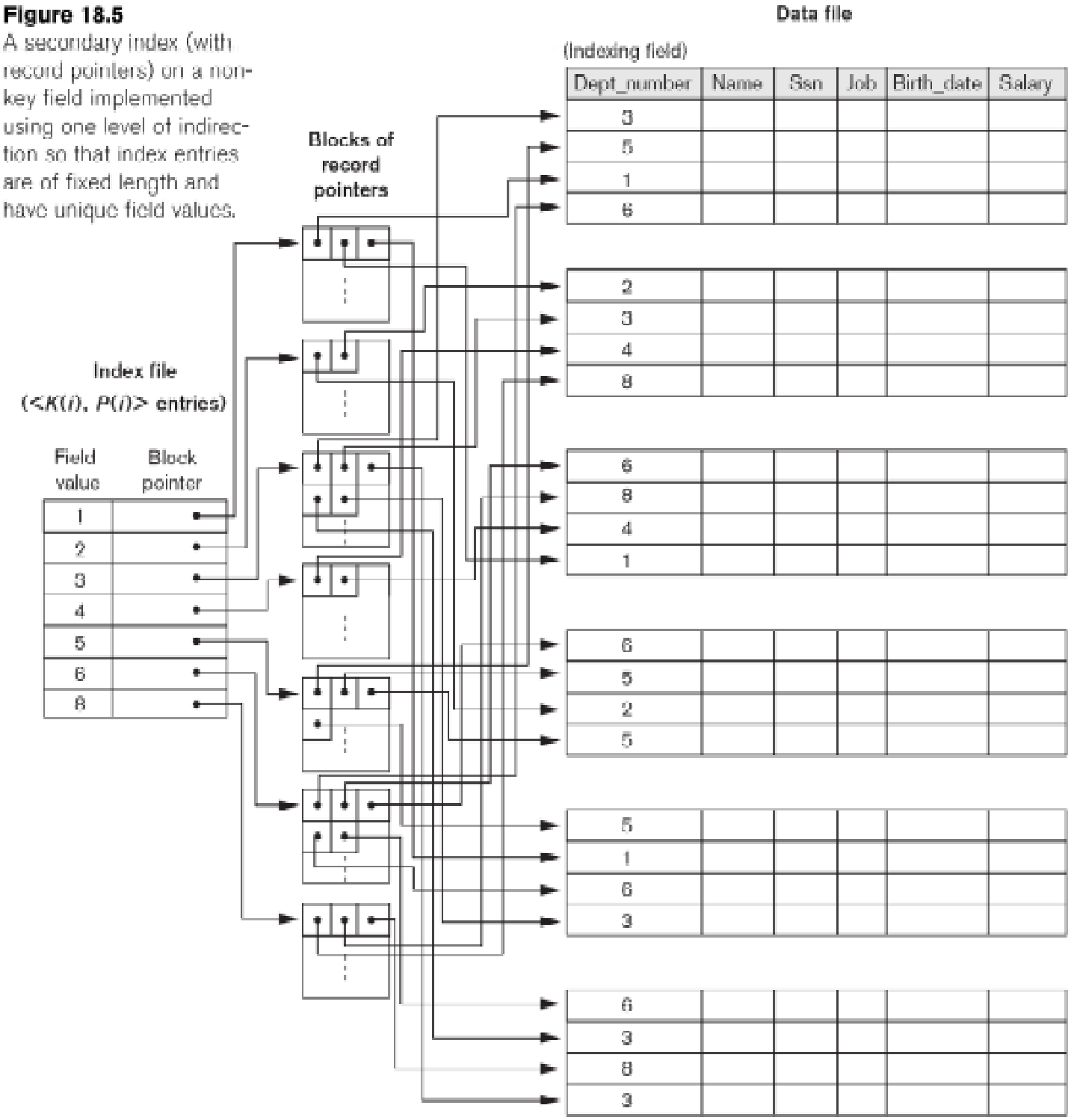
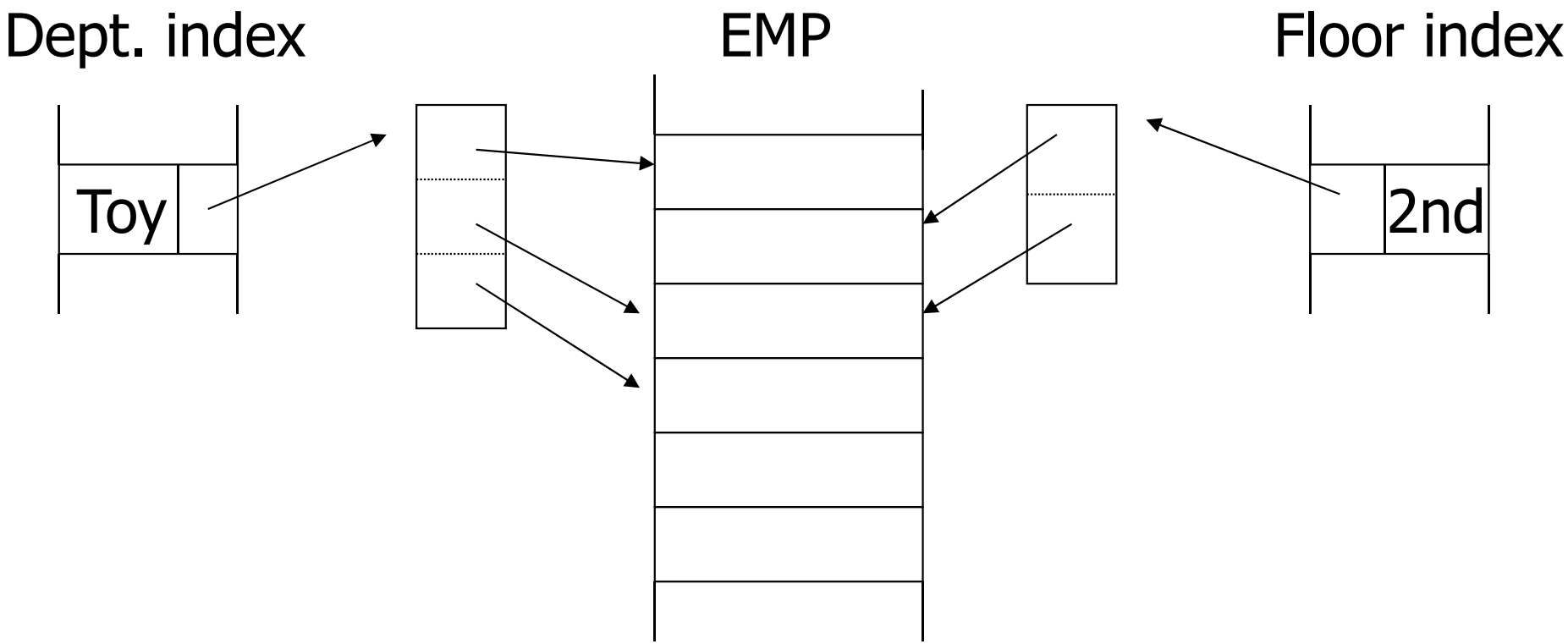


Figure 18.5
A secondary index (with record pointers) on a non-key field implemented using one level of indirection so that index entries are of fixed length and have unique field values.

Example

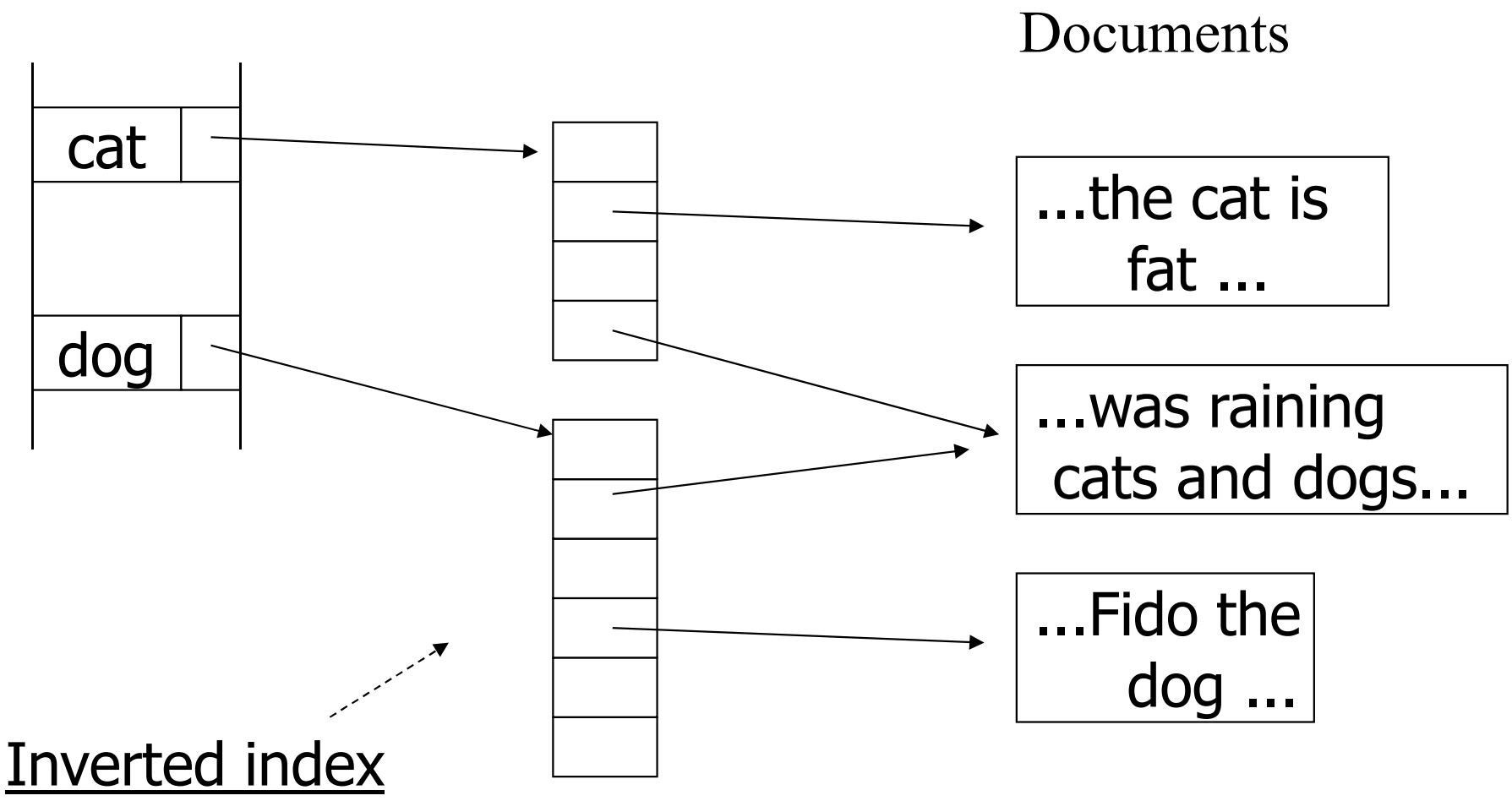


Primary index: name

Secondary indexes: Dept, Floor

Query: Select employees in (Toy Dept) $\wedge$ (2nd floor)

Text information retrieval



Properties of Index Types

Table 18.2 Properties of Index Types

Type of Index	Number of (First-level) Index Entries	Dense or Nondense (Sparse)	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

SQL

- PRIMARY KEY declaration automatically creates a primary index
- UNIQUE key automatically creates a secondary index
- Secondary index can be created on non-key attribute(s)
  - CREATE INDEX StudentGPAIndex ON Student(GPA);