Indexes as Access Paths

- A single-level index is an auxiliary file that makes it more efficient to search for a record in the data file.
- The index is usually specified on one field of the file (although it could be specified on several fields)
- One form of an index is a file of entries <**field value, pointer to record>**, which is ordered by field value
- The index is called an access path on the field.

Indexes as Access Paths (contd.)

- The index file usually occupies considerably less disk blocks than the data file because its entries are much smaller
- A binary search on the index yields a pointer to the file record
- Indexes can also be characterized as dense or sparse
  - 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**, on the other hand, has index entries for only some of the search values

Indexes as Access Paths (contd.)

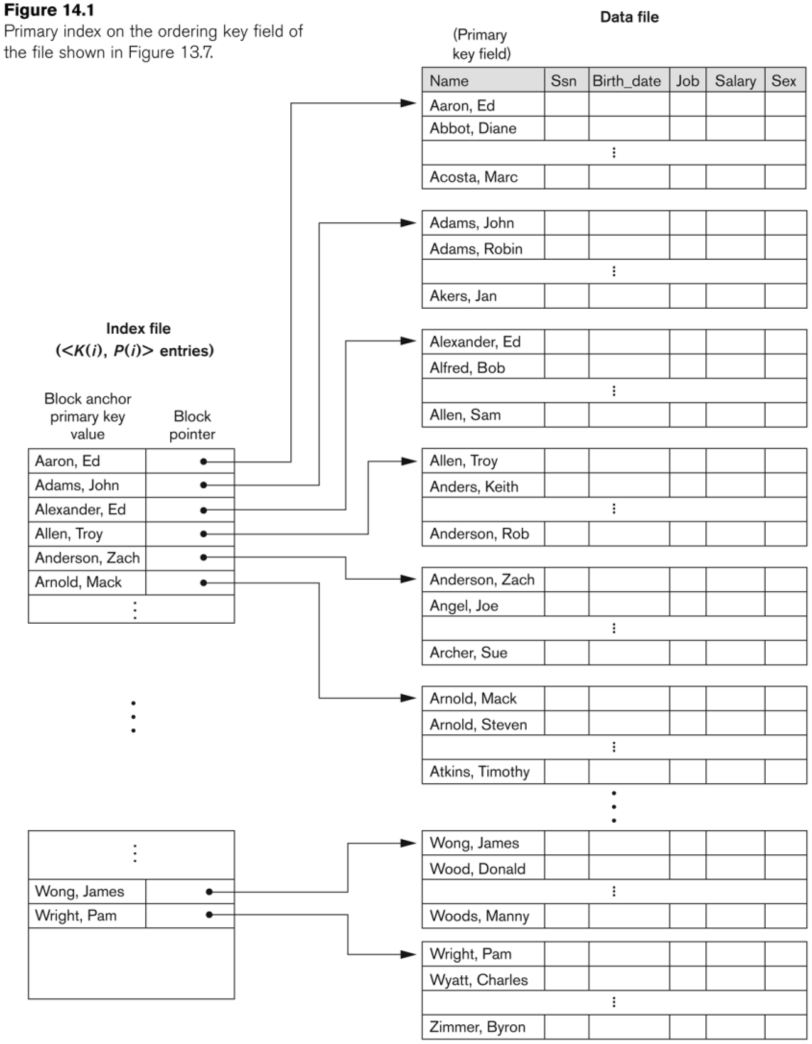
- Example: Given the following data file EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ...)
- Suppose that:
  - record size $R=150$ bytes block size $B=512$ bytes $r=30000$ records
- Then, we get:
  - blocking factor Bfr= B div R= 512 div 150= 3 records/block
  - number of file blocks b= (r/Bfr)= (30000/3)= 10000 blocks
- For an index on the SSN field, assume the field size $V_{SSN}=9$ bytes, assume the record pointer size $P_R=7$ bytes. Then:
  - index entry size $R_I=(V_{SSN}+ P_R)=(9+7)=16$ bytes
  - index blocking factor Bfr_I= B div R_I= 512 div 16= 32 entries/block
  - number of index blocks b= (r/ Bfr_I)= (30000/32)= 938 blocks
  - binary search needs $\log_2 bI= \log_2 938$= 10 block accesses
  - This is compared to an average linear search cost of:
    - (b/2)= 30000/2= 15000 block accesses
  - If the file records are ordered, the binary search cost would be:
    - $\log_2 b= \log_2 30000$= 15 block accesses

Types of Single-Level Indexes

- Primary Index
  - Defined on an ordered data file
  - The data file is ordered on a **key field**
  - Includes one index entry *for each block* in the data file; the index entry has the key field value for the *first record* in the block, which is called the *block anchor*
  - A similar scheme can use the *last record* in a block.
  - A primary index is a nondense (sparse) index, since it includes an entry for each disk block of the data file and the keys of its anchor record rather than for every search value.

Primary index on the ordering key field

Figure 14.1
Primary index on the ordering key field of the file shown in Figure 13.7.



Block anchor primary key value	Block pointer
Aaron, Ed	●
Adams, John	●
Alexander, Ed	●
Allen, Troy	●
Anderson, Zach	●
Arnold, Mack	●
⋮	
⋮	
⋮	
Wong, James	●
Wright, Pam	●

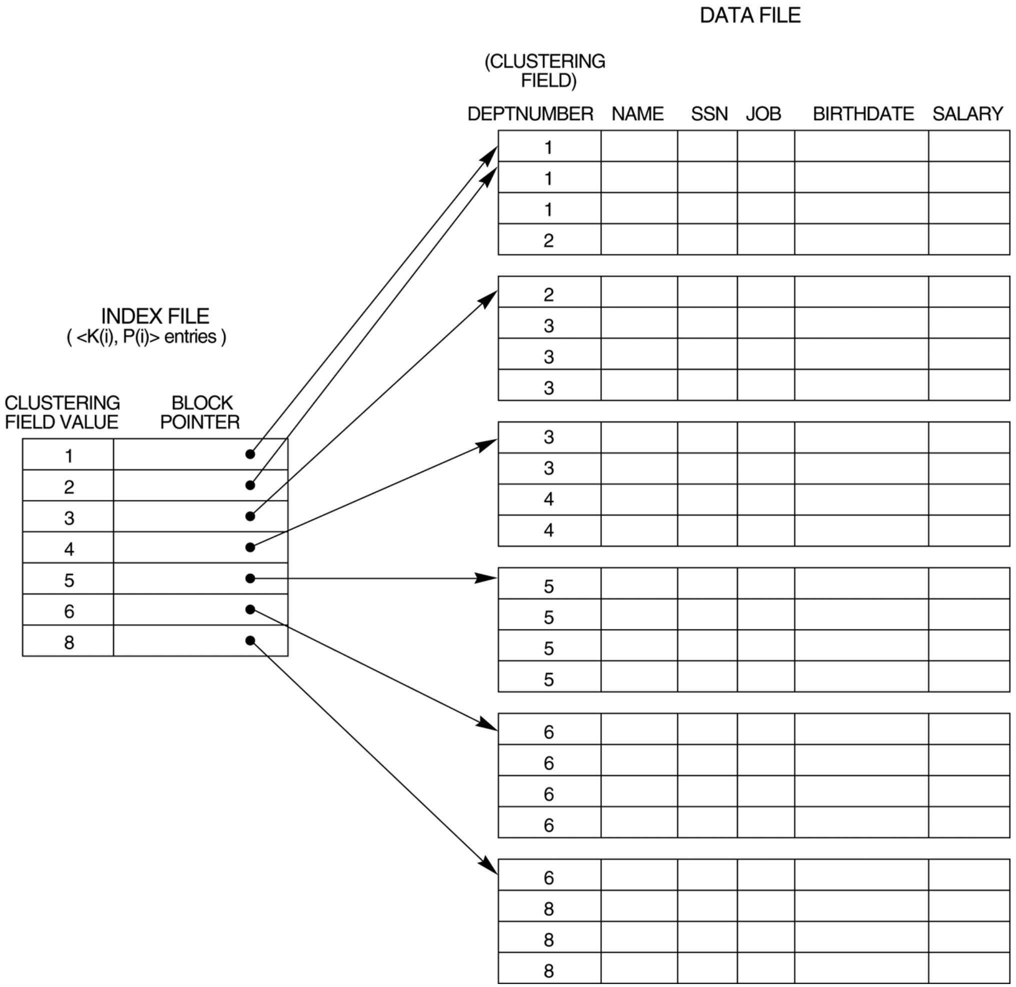
Name	Ssn	Birth_date	Job	Salary	Sex
Aaron, Ed					
Abbot, Diane					
⋮					
Acosta, Marc					
Adams, John					
Adams, Robin					
⋮					
Akers, Jan					
Alexander, Ed					
Alfred, Bob					
⋮					
Allen, Sam					
Allen, Troy					
Anders, Keith					
⋮					
Anderson, Rob					
Anderson, Zach					
Angel, Joe					
⋮					
Archer, Sue					
Arnold, Mack					
Arnold, Steven					
⋮					
Atkins, Timothy					
⋮					
Wong, James					
Wood, Donald					
⋮					
Woods, Manny					
Wright, Pam					
Wyatt, Charles					
⋮					
Zimmer, Byron					

Types of Single-Level Indexes

- Clustering Index
  - Defined on an ordered data file
  - The data file is ordered on a *non-key field* unlike primary index, which requires that the ordering field of the data file have a distinct value for each record.
  - Includes one index entry *for each distinct value* of the field; the index entry points to the first data block that contains records with that field value.
  - It is another example of *nondense* index where Insertion and Deletion is relatively straightforward with a clustering index.

A Clustering Index Example

- FIGURE 14.2
  A clustering index on the DEPTNUMBER ordering non-key field of an EMPLOYEE file.



CLUSTERING FIELD VALUE	BLOCK POINTER
1	●
2	●
3	●
4	●
5	●
6	●
8	●

DEPTNUMBER	NAME	SSN	JOB	BIRTHDATE	SALARY
1					
1					
1					
2					
2					
3					
3					
3					
3					
3					
4					
4					
5					
5					
5					
5					
6					
6					
6					
6					
6					
8					
8					
8					

Another Clustering Index Example

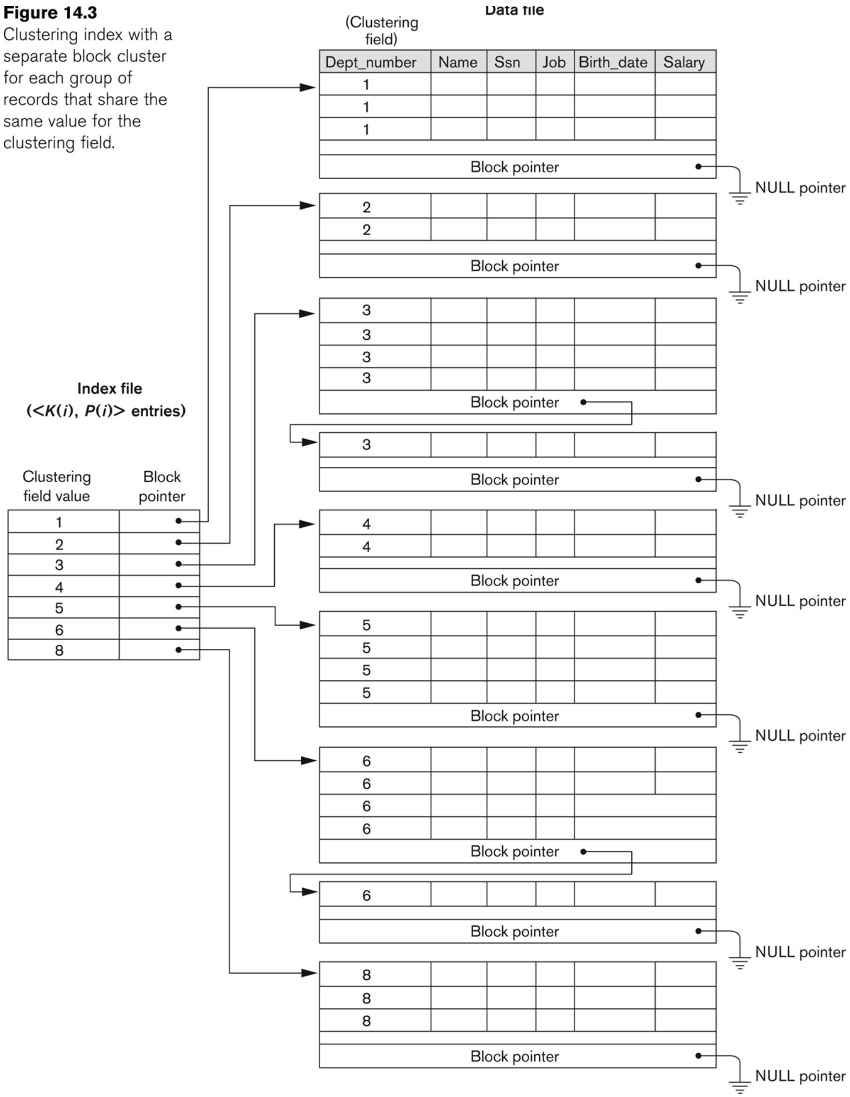


Figure 14.3
Clustering index with a separate block cluster for each group of records that share the same value for the clustering field.

Index file
(<*K*(*i*), *P*(*i*)> entries)

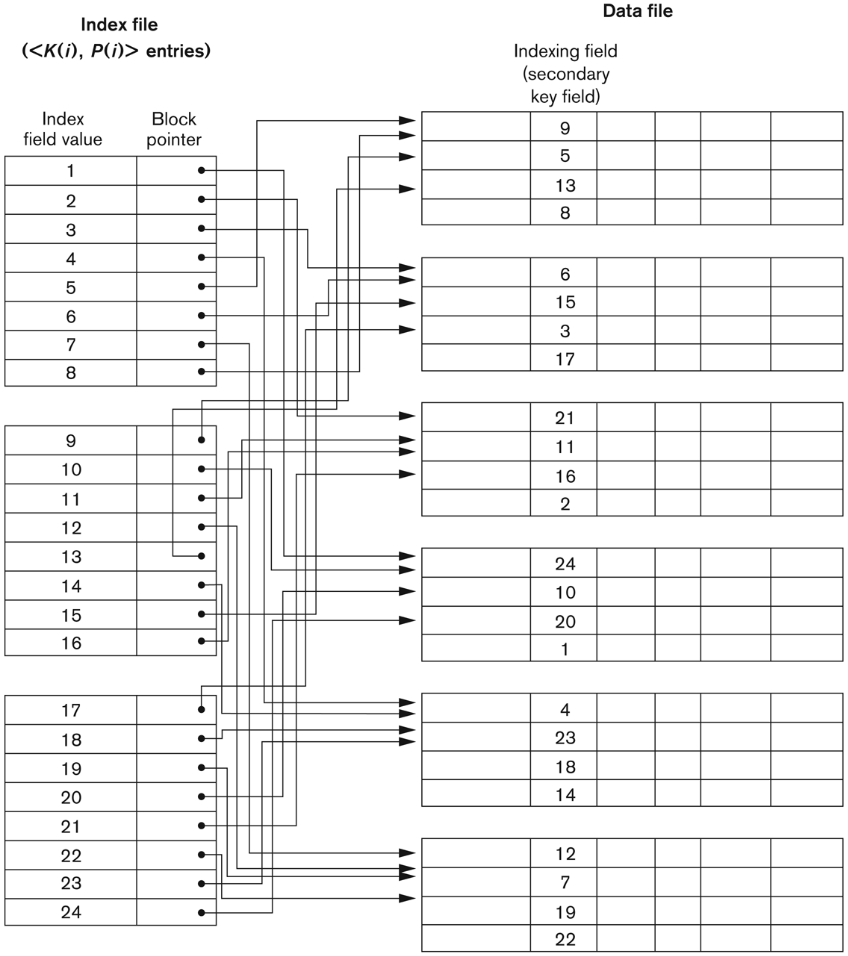
Clustering field value	Block pointer
1	●
2	●
3	●
4	●
5	●
6	●
8	●

Types of Single-Level Indexes

- Secondary Index
  - A secondary index provides a secondary means of accessing a file for which some primary access already exists.
  - The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
  - The index is an ordered file with two fields.
    - The first field is of the same data type as some **non-ordering field** of the data file that is an indexing field.
    - The second field is either a **block** pointer or a record pointer.
    - There can be *many* secondary indexes (and hence, indexing fields) for the same file.
  - Includes one entry *for each record* in the data file; hence, it is a *dense index*

Example of a Dense Secondary Index

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



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				

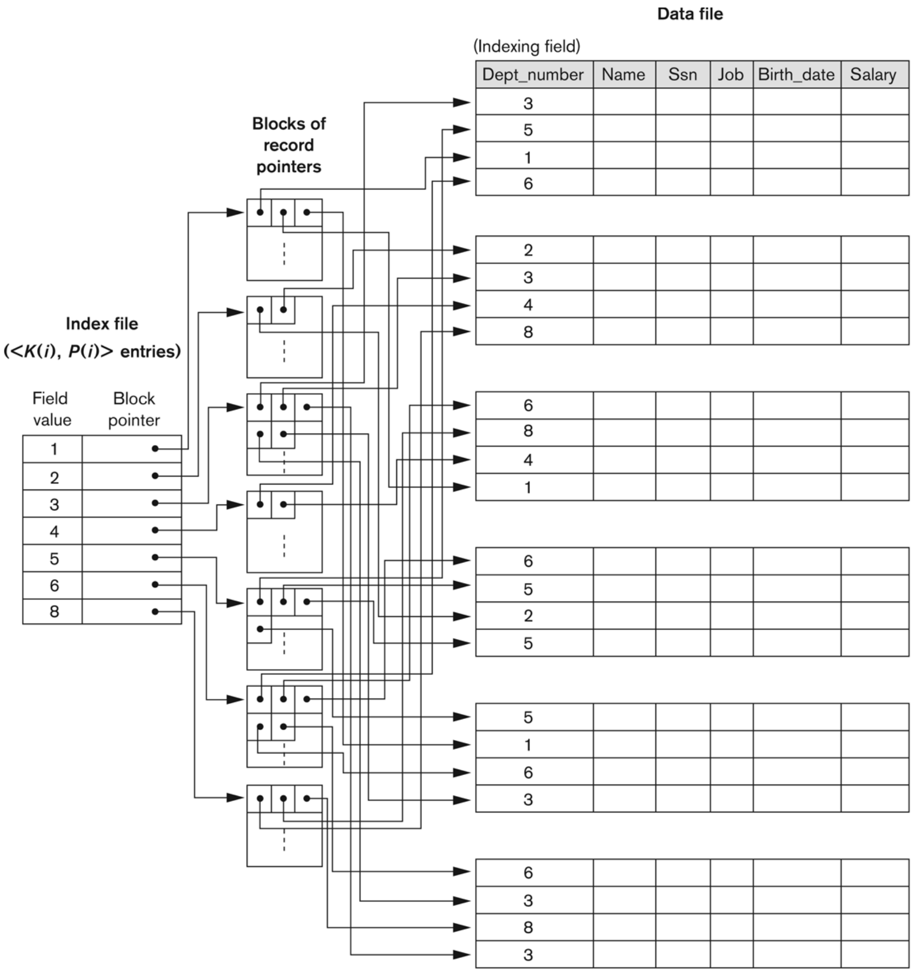
	21				
	11				
	16				
	2				

	24				
	10				
	20				
	1				

	4				
	23				
	18				
	14				

	12				
	7				
	19				
	22				

An Example of a Secondary Index



Field value	Block pointer
1	•
2	•
3	•
4	•
5	•
6	•
8	•

Dept_number	Name	Ssn	Job	Birth_date	Salary
3					
5					
1					
6					
2					
3					
4					
8					
6					
8					
4					
1					
6					
5					
2					
5					
5					
1					
6					
3					
6					
3					
8					
3					

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.

Properties of Index Types

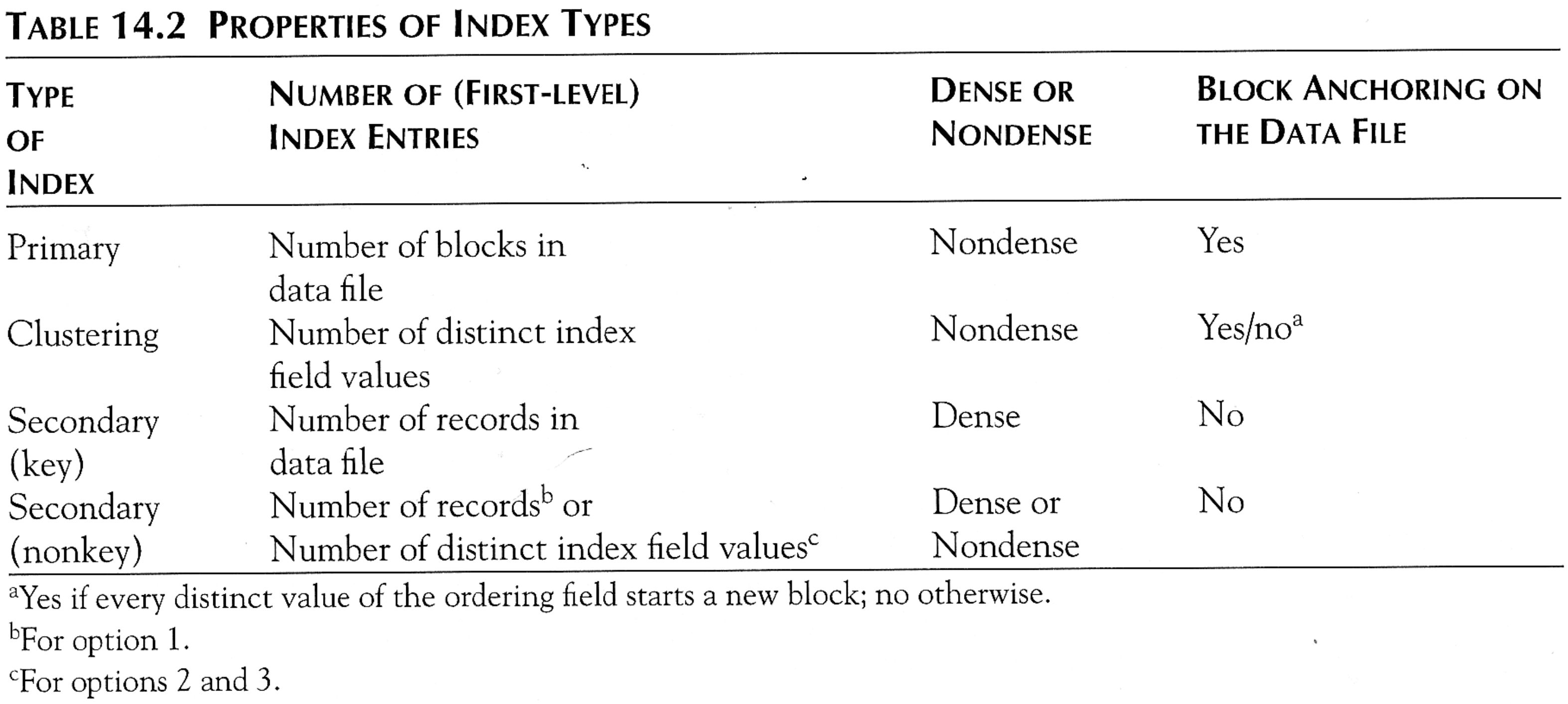


TABLE 14.2 PROPERTIES OF INDEX TYPES

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.

Multi-Level Indexes

- Because a single-level index is an ordered file, we can create a primary index *to the index itself*;
  - In this case, the original index file is called the *first-level index* and the index to the index is called the *second-level index*.
- We can repeat the process, creating a third, fourth, ..., top level until all entries of the *top level* fit in one disk block
- A multi-level index can be created for any type of first-level index (primary, secondary, clustering) as long as the first-level index consists of *more than one* disk block

A Two-level Primary Index

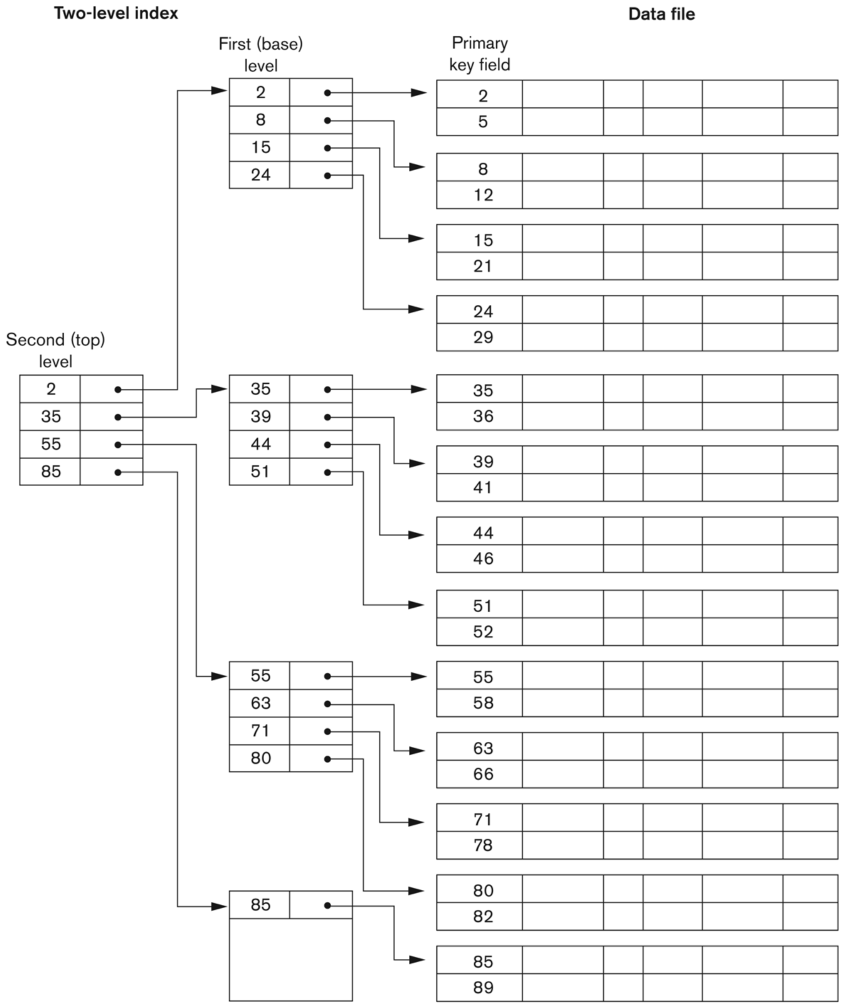


Figure 14.6
A two-level primary index resembling ISAM (Index Sequential Access Method) organization.