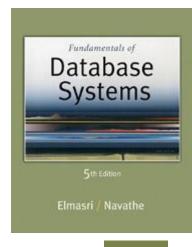


5th Edition

Elmasri / Navathe

## Chapter 14

Indexing Structures for Files





### **Chapter Outline**

- Types of Single-level Ordered Indexes
  - Primary Indexes
  - Clustering Indexes
  - Secondary Indexes
- Multilevel Indexes
- Dynamic Multilevel Indexes Using B-Trees and B+-Trees
- Indexes on Multiple Keys

#### 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.)

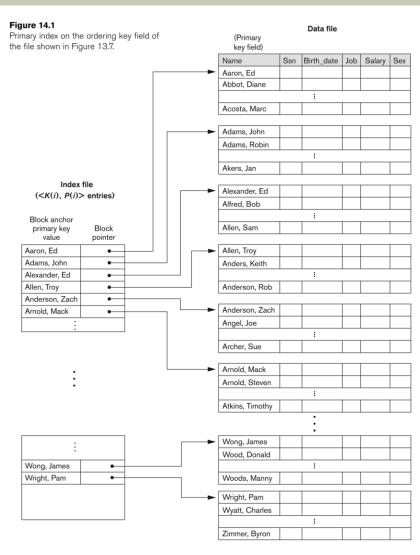
- Example: Given the following data file EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ...)
- Suppose that:
  - record size R=150 bytes
     block size B=512 bytes
     r=30000 records
- Then, we get:
  - blocking factor Bfr= B div R= 512 div 150= 3 records/block
  - number of file blocks b= (r/Bfr)= (30000/3)= 10000 blocks
- For an index on the SSN field, assume the field size V<sub>SSN</sub>=9 bytes, assume the record pointer size P<sub>R</sub>=7 bytes. Then:
  - index entry size  $R_I = (V_{SSN} + P_R) = (9+7) = 16$  bytes
  - index blocking factor Bfr<sub>i</sub>= B div R<sub>i</sub>= 512 div 16= 32 entries/block
  - number of index blocks b= (r/ Bfr<sub>I</sub>)= (30000/32)= 938 blocks
  - binary search needs log<sub>2</sub>bl= log<sub>2</sub>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:
    - $log_2b = log_230000 = 15$  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



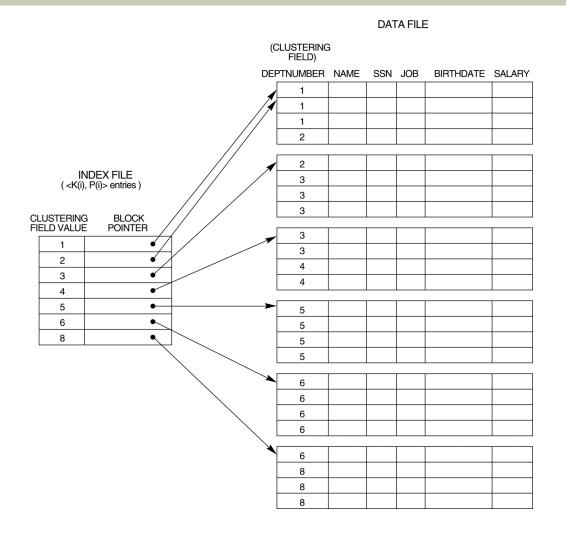
## Types of Single-Level Indexes

#### Clustering Index

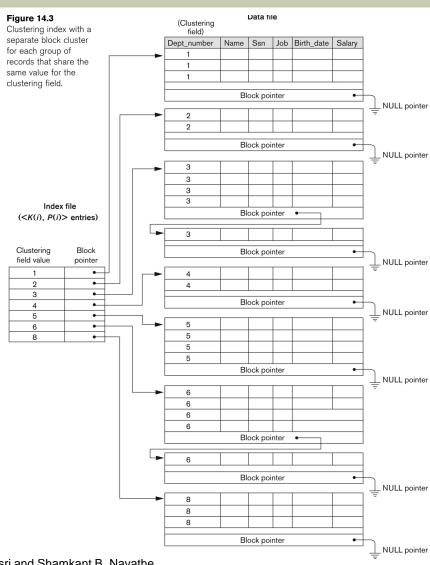
- Defined on an ordered data file
- The data file is ordered on a non-key field unlike primary index, which requires that the ordering field of the data file have a distinct value for each record.
- Includes one index entry for each distinct value of the field; the index entry points to the first data 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.



#### Another Clustering Index Example

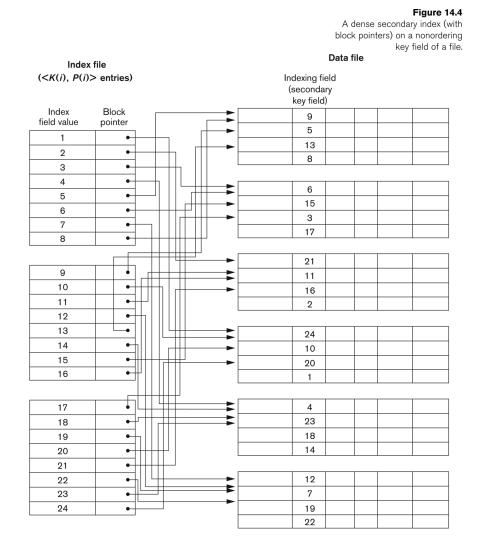


## Types of Single-Level Indexes

#### 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



## An Example of a Secondary Index

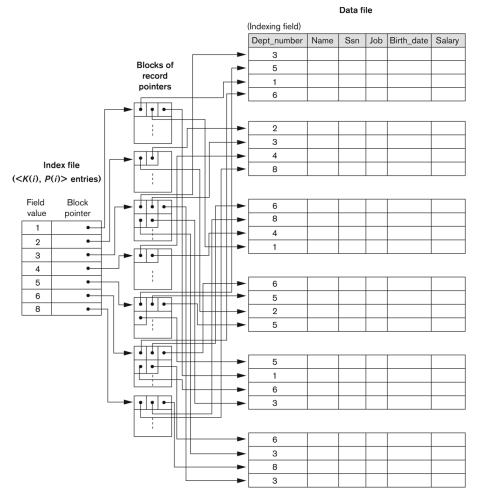


Figure 14.5

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 <sup>a</sup>
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	Number of records <sup>b</sup> or Number of distinct index field values <sup>c</sup>	Dense or Nondense	No

<sup>&</sup>lt;sup>a</sup>Yes if every distinct value of the ordering field starts a new block; no otherwise.

<sup>&</sup>lt;sup>b</sup>For option 1.

<sup>&</sup>lt;sup>c</sup>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

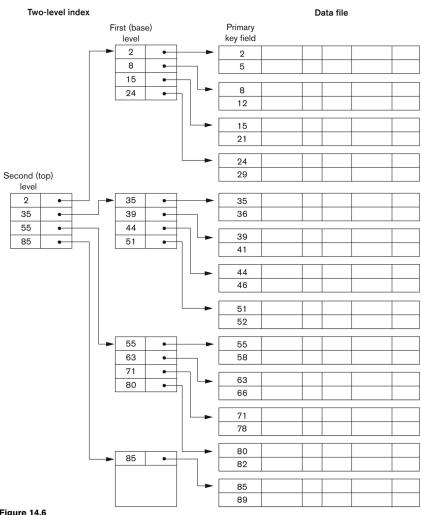


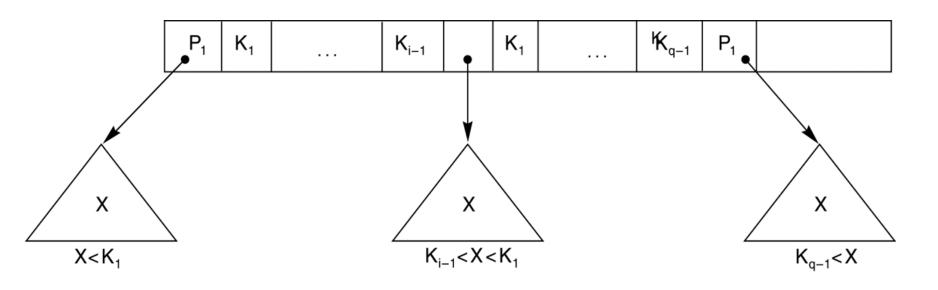
Figure 14.6
A two-level primary index resembling ISAM (Index Sequential Access Method) organization.

#### Multi-Level Indexes

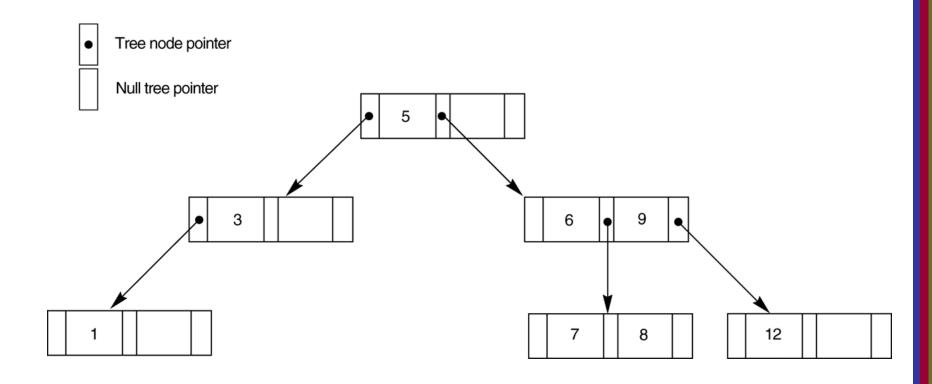
- Such a multi-level index is a form of search tree
  - However, insertion and deletion of new index entries is a severe problem because every level of the index is an *ordered file*.

## A Node in a Search Tree with Pointers to Subtrees below It

■ FIGURE 14.8



# FIGURE 14.9 A search tree of order p = 3.



## Dynamic Multilevel Indexes Using B-Trees and B+-Trees

- Most multi-level indexes use B-tree or B+-tree data structures because of the insertion and deletion problem
  - This leaves space in each tree node (disk block) to allow for new index entries
- These data structures are variations of search trees that allow efficient insertion and deletion of new search values.
- In B-Tree and B+-Tree data structures, each node corresponds to a disk block
- Each node is kept between half-full and completely full

# Dynamic Multilevel Indexes Using B-Trees and B+-Trees (contd.)

- An insertion into a node that is not full is quite efficient
  - If a node is full the insertion causes a split into two nodes
- Splitting may propagate to other tree levels
- A deletion is quite efficient if a node does not become less than half full
- If a deletion causes a node to become less than half full, it must be merged with neighboring nodes

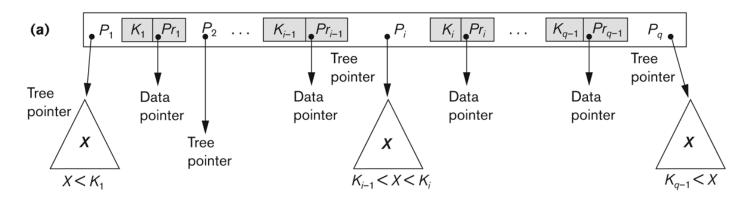
#### Difference between B-tree and B+-tree

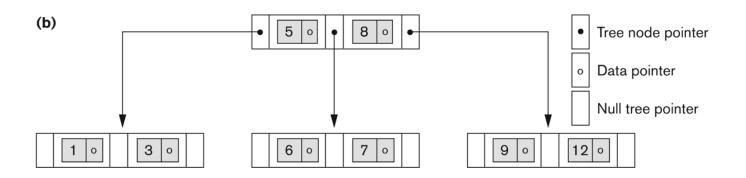
- In a B-tree, pointers to data records exist at all levels of the tree
- In a B+-tree, all pointers to data records exists at the leaf-level nodes
- A B+-tree can have less levels (or higher capacity of search values) than the corresponding B-tree

#### **B-tree Structures**

Figure 14.10

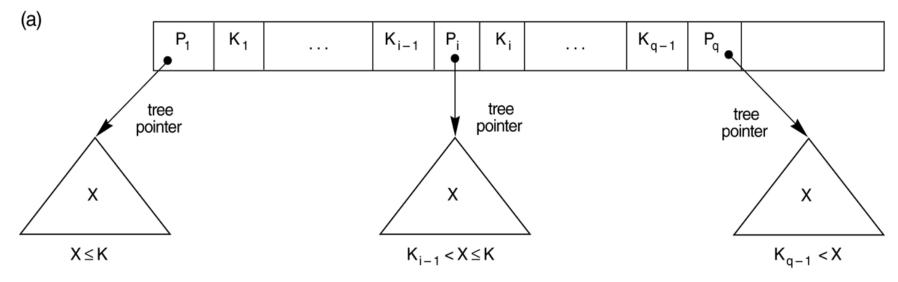
B-Tree structures. (a) A node in a B-tree with q-1 search values. (b) A B-tree of order p=3. The values were inserted in the order 8, 5, 1, 7, 3, 12, 9, 6.

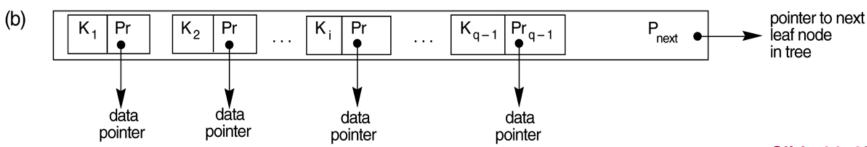




#### The Nodes of a B+-tree

- FIGURE 14.11 The nodes of a B+-tree
  - (a) Internal node of a B+-tree with q −1 search values.
  - (b) Leaf node of a B+-tree with q 1 search values and q 1 data pointers.

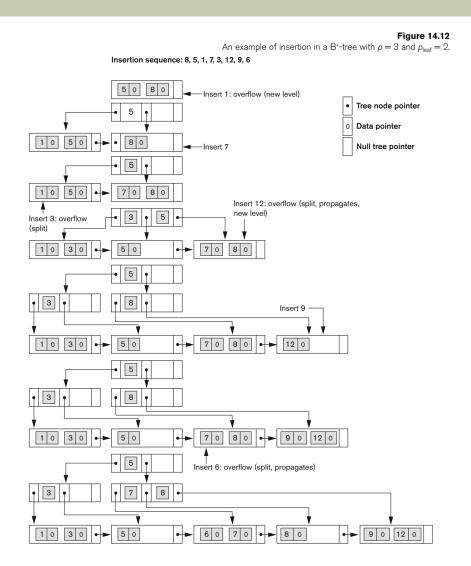




Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

Slide 14-25

#### An Example of an Insertion in a B+-tree



## An Example of a Deletion in a B+-tree

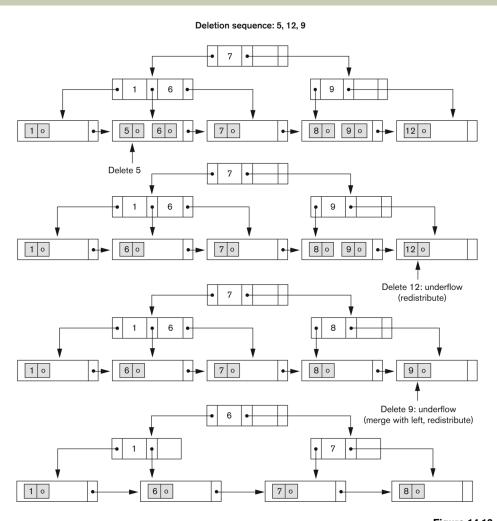


Figure 14.13 An example of deletion from a B+-tree.

### Summary

- Types of Single-level Ordered Indexes
  - Primary Indexes
  - Clustering Indexes
  - Secondary Indexes
- Multilevel Indexes
- Dynamic Multilevel Indexes Using B-Trees and B+-Trees
- Indexes on Multiple Keys