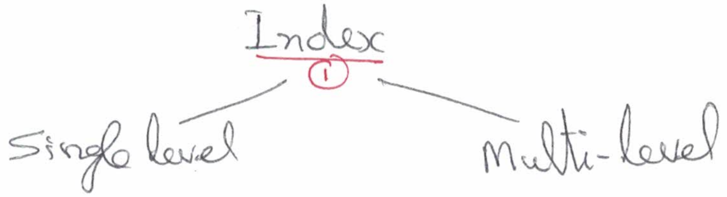
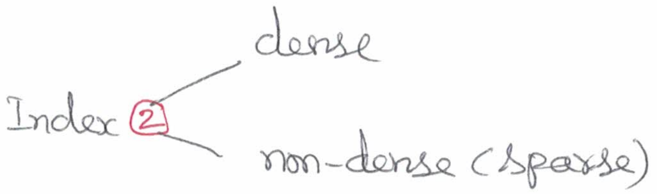
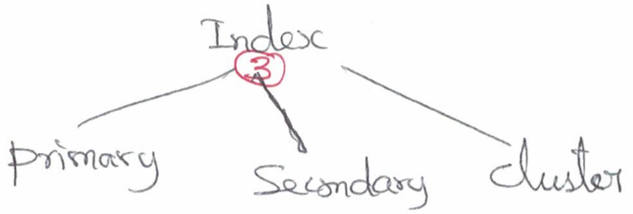






Single-level index

Each index entry has access path to its record

Each entry: $(\underbrace{\overset{\text{7 bytes}}{\text{Field value}}, \overset{\text{5 bytes}}{\text{Record pointer}}}_{\text{12 bytes}})$ PID

No. of entries: 46,000 each 12 bytes.

Block size = 1,024 bytes (1KB)

No. of index entries/block $= \dfrac{1024}{12} = 85$ entries/block } block factor.

No. of disk blocks needed $= \dfrac{46000}{85} = 542$ Disk blocks.

Average # of disk block access for any index search } $= \overset{271+1}{272}$ < best case: 1 / worst case: 543

$$\text{Index density} = \frac{\text{No. of distinct index key values}}{\text{Total no. of records.}}$$

Dense index: e.g. Index on Student (PID)

Non-Dense (Sparse) index: e.g. Index on Student (Zipcode)

Comparison of number of disk block accesses for sequential file and index file

Sequential file without index

Best case: 1 block access (record is found in the first data block)

No. of student records/block = 1024 / 300 = 3 records/block

Total no. of data blocks = 46000 / 3 = 15,334

Worst case: 15,334 blocks access (record is found in the last data block)

Average for an arbitrary record search: 7,667 blocks accessed.

Index file with Single-Level index

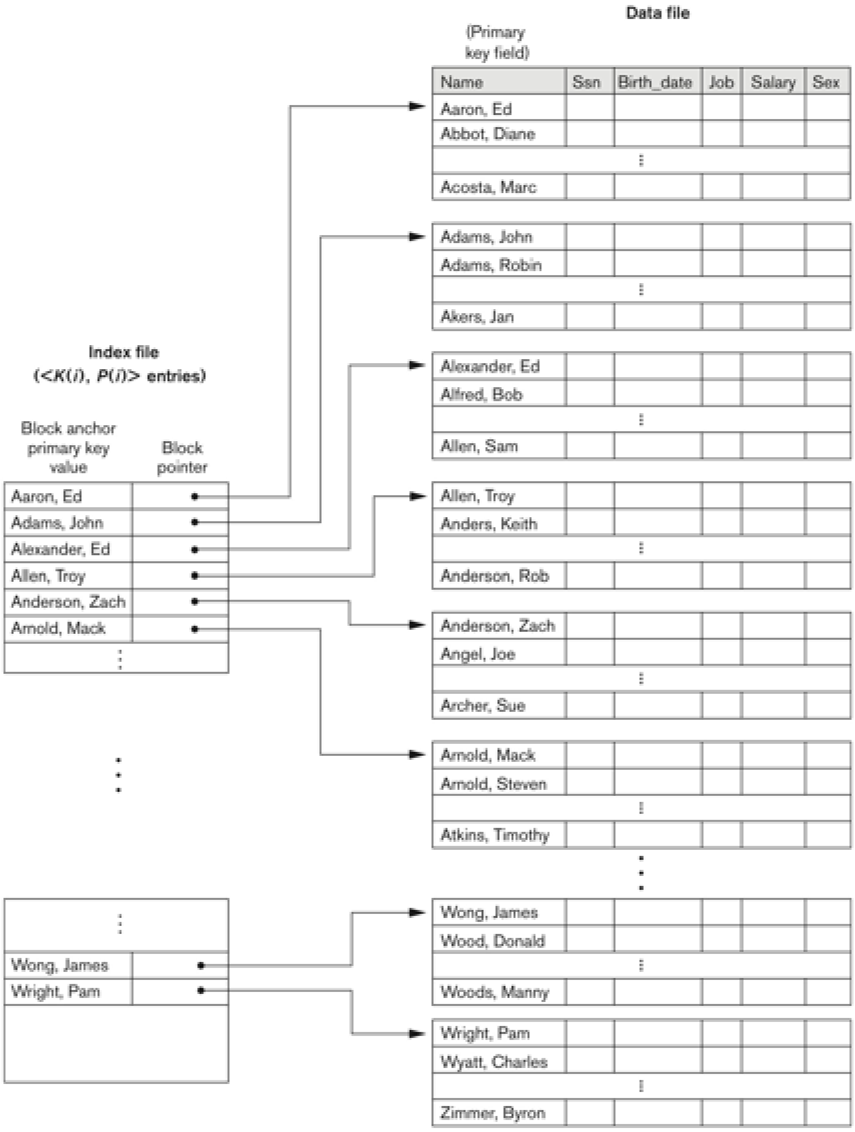
Best case: 1 + 1
= 2 block access (record is found in the first index block and the data block access)

Worst case: 542 + 1 = 543 blocks access (record is found in the last index block)

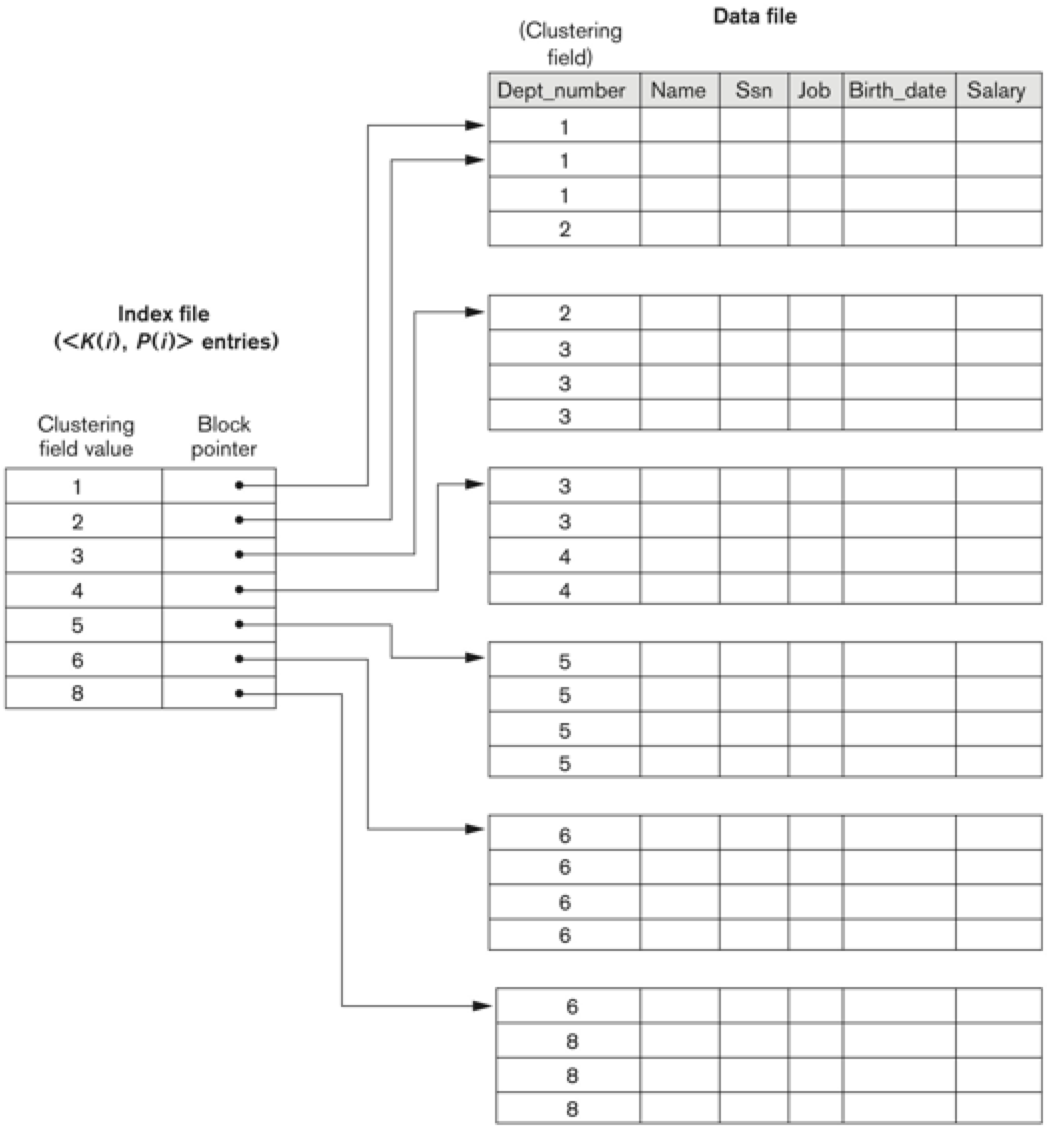
Avg no. blocks accessed: 271 (index blocks) + 1 (data block)
= 272 block accesses

Speed up with index file = 7667 / 272 = 28.2 times

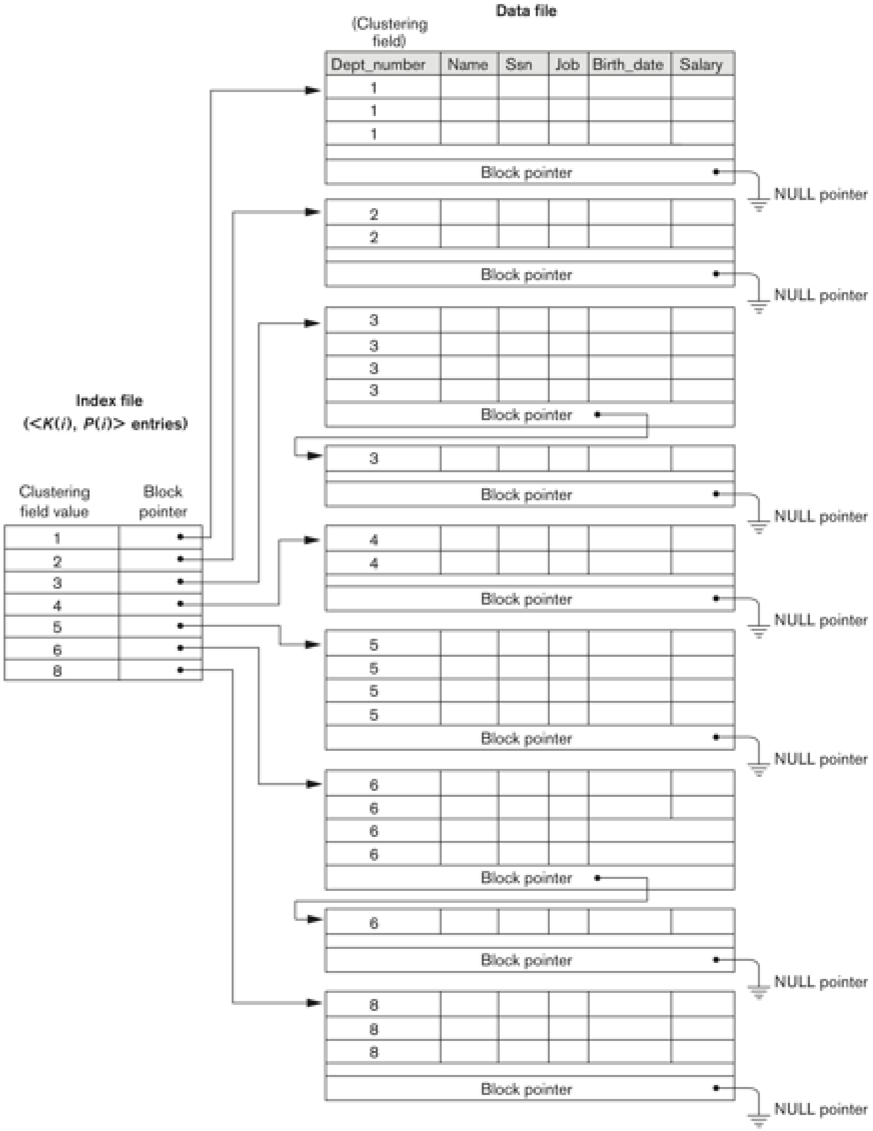
In this case, disk access on indexed file will be approximately 28 times faster than sequential file.



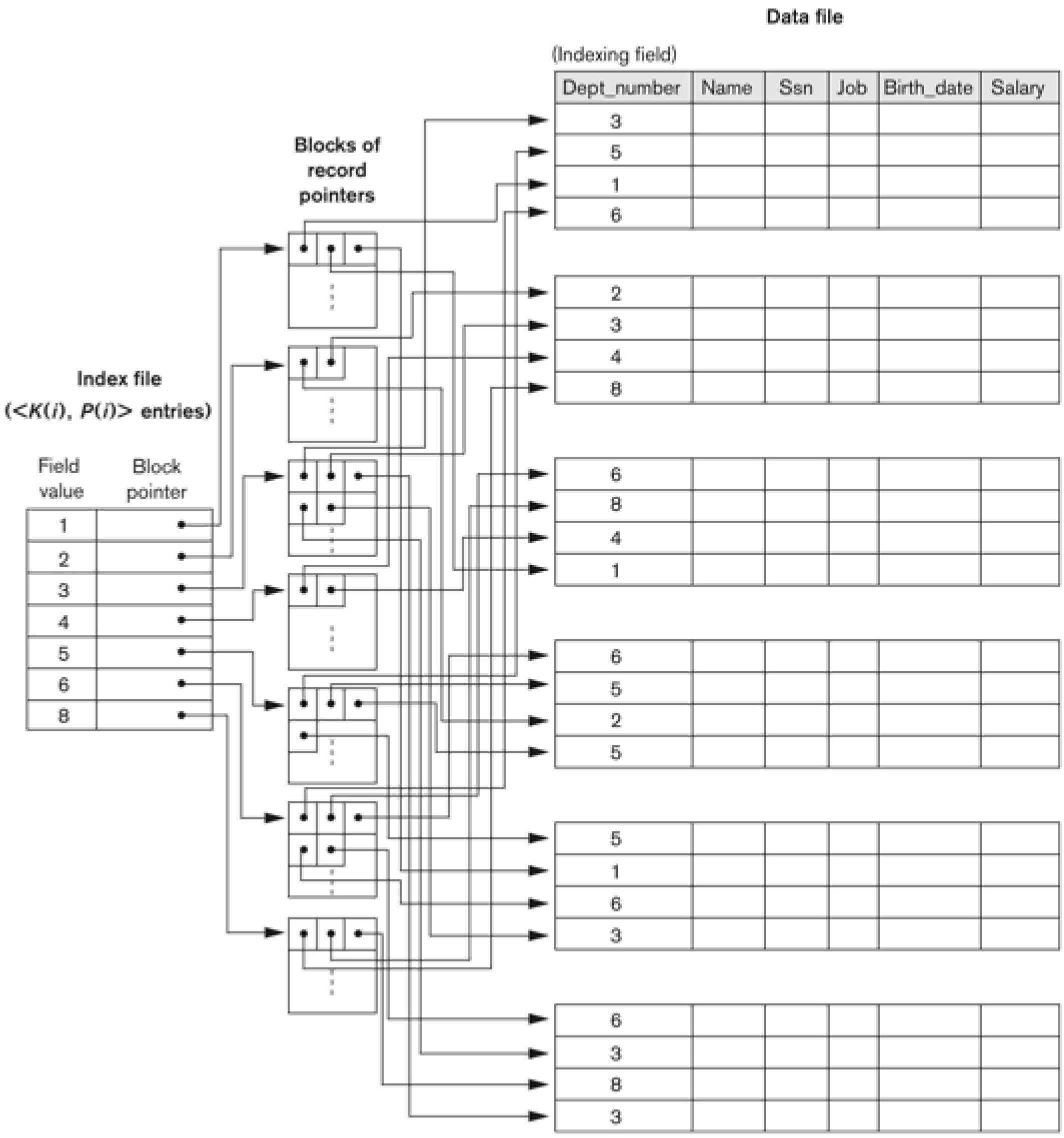
Primary index on the ordering key field



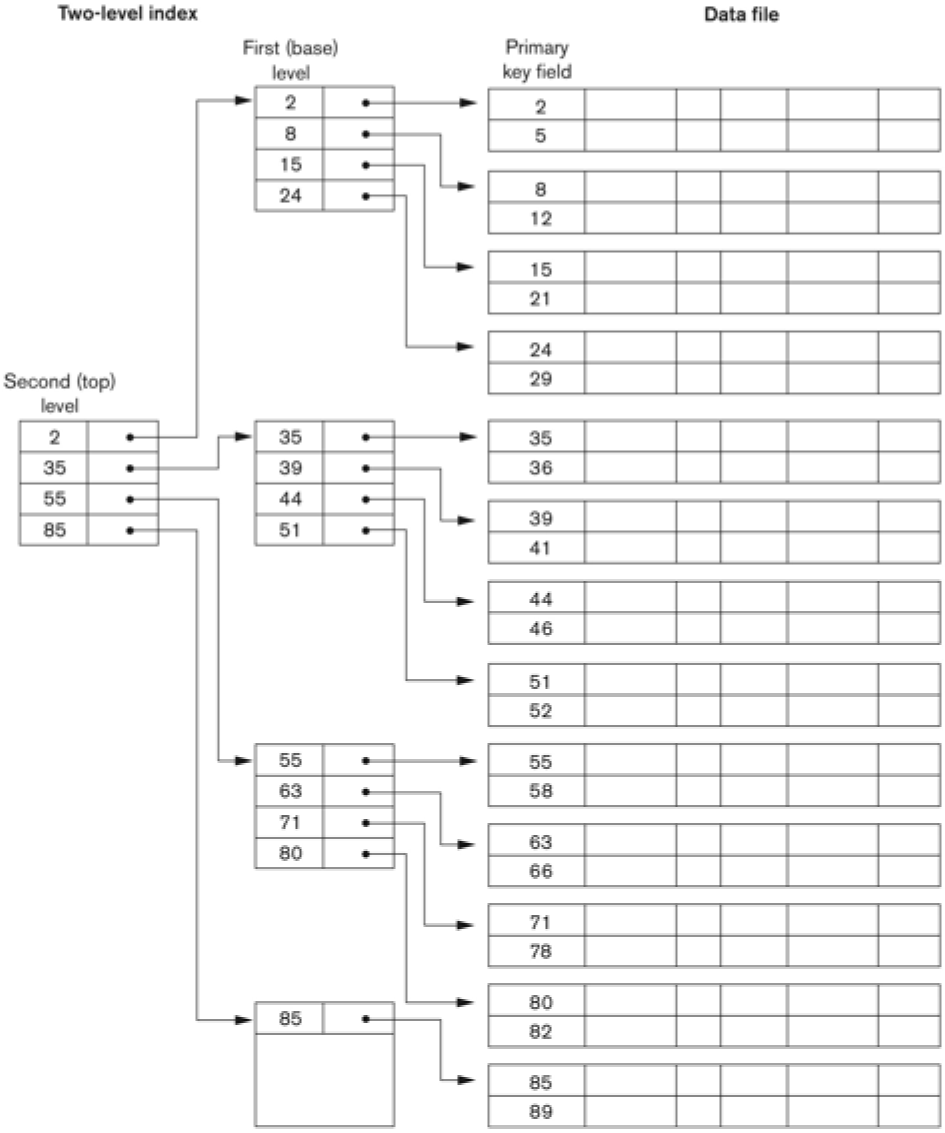
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file.



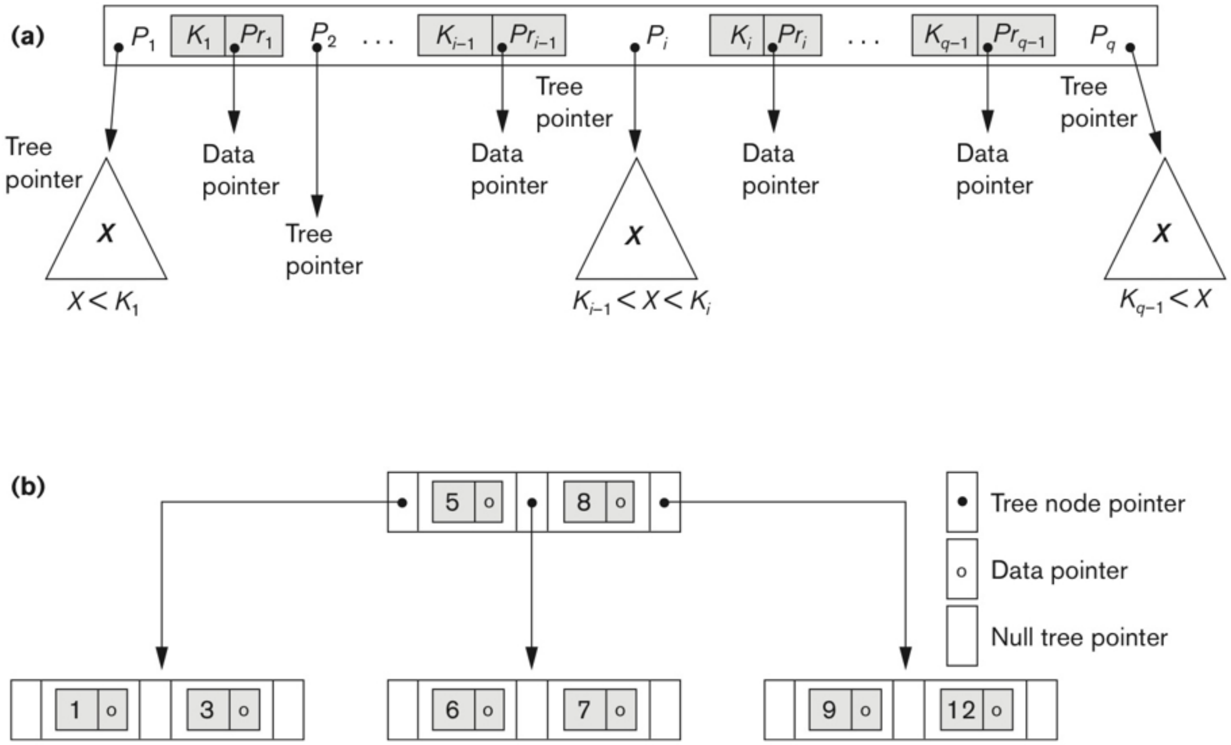
Clustering index with a separate block cluster for each group of records that share the same value for the clustering field.



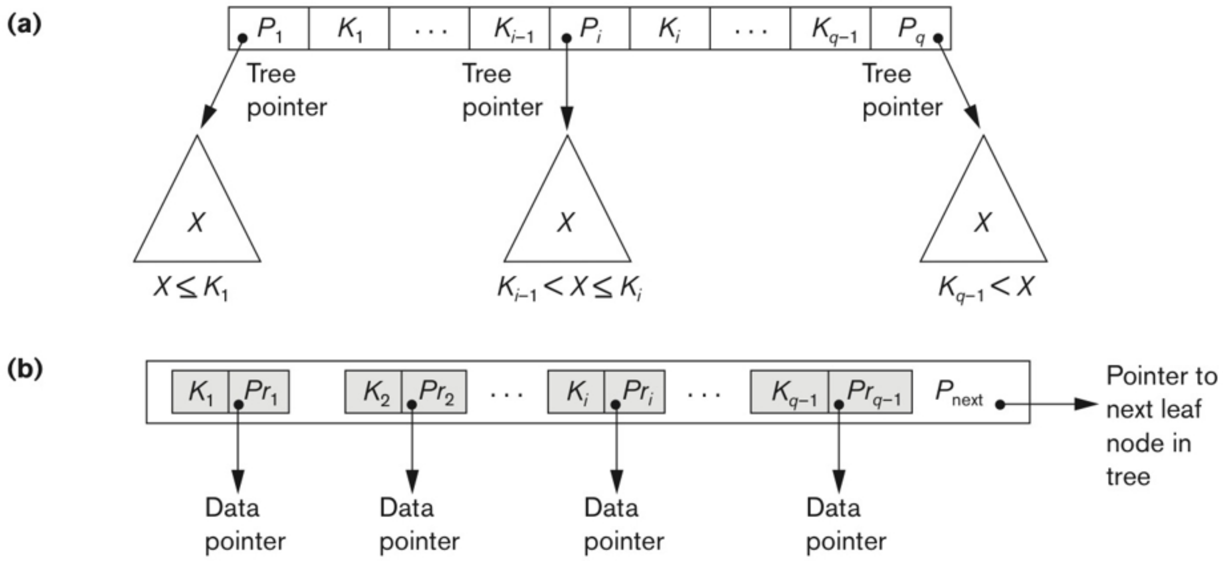
A secondary index (with record pointers) on a nonkey field implemented using one level of indirection so that index entries are of fixed length and have unique field values.



A two-level primary index resembling ISAM (indexed sequential access method) organization.



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. (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.