Data Indexing

Purposes of Data Indexing

- What is Data Indexing?
- Why is it important?

Concept of File Systems

- Stores and organizes data into computer files.
- Makes it easier to find and access data at any given time.

Database Management Systems

- The file system that manages a database.
- Database - is an organized collection of logically related data.

How DBMS Accesses Data?

- The operations read, modify, update, and delete are used to access data from database.
- DBMS must first transfer the data temporarily to a buffer in main memory.
- Data is then transferred between disk and main memory into units called blocks.

Time Factors

- The transferring of data into blocks is a very slow operation.
- Accessing data is determined by the physical storage device being used.

Physical Storage Devices

- Random Access Memory – Fastest to access memory, but most expensive.
- Direct Access Memory – In between for accessing memory and cost
- Sequential Access Memory – Slowest to access memory, and least expensive.

More Time Factors

- Querying data out of a database requires more time.
- DBMS must search among the blocks of the database file to look for matching tuples.

Purpose of Data Indexing

- It is a data structure that is added to a file to provide faster access to the data.
- It reduces the number of blocks that the DBMS has to check.

Properties of Data Index

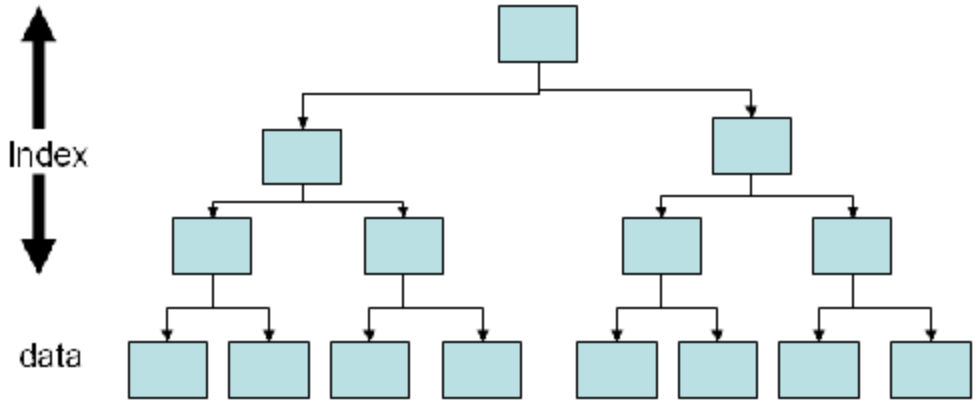
- It contains a search key and a pointer.
- Search key - an attribute or set of attributes that is used to look up the records in a file.
- Pointer - contains the address of where the data is stored in memory.
- It can be compared to the card catalog system used in public libraries of the past.

Two Types of Indices

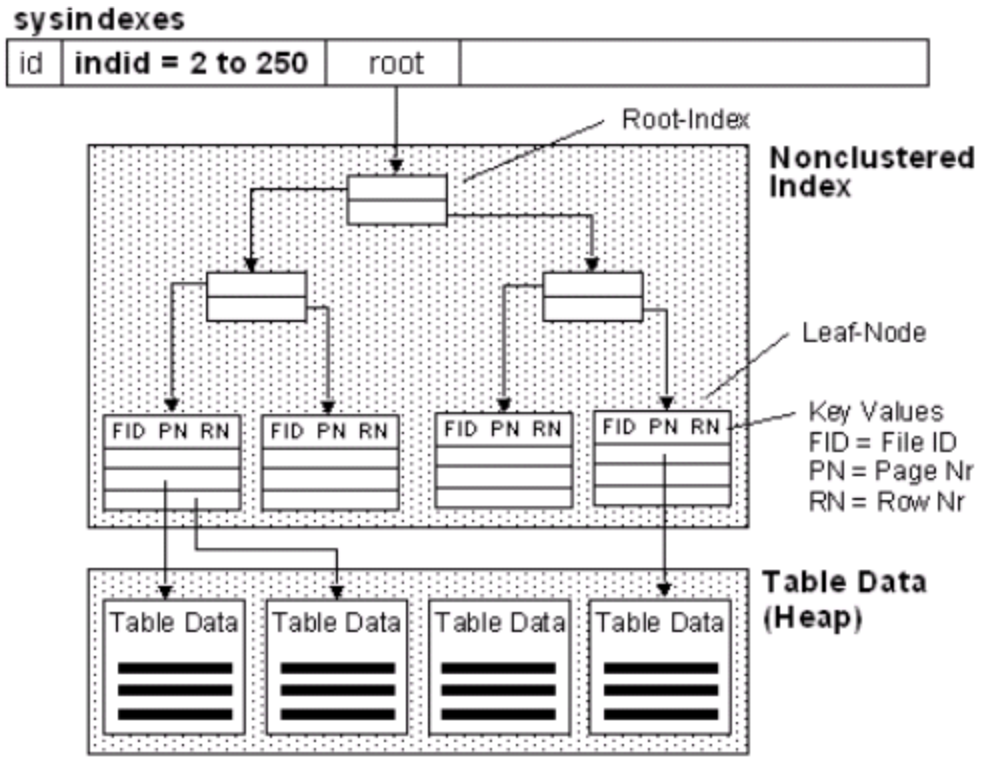
- Ordered index (Primary index or clustering index) – which is used to access data sorted by order of values.

- Hash index (secondary index or non-clustering index) - used to access data that is distributed uniformly across a range of buckets.

Ordered Index



Hash Index


sysindexes
id
indid = 2 to 250
root
Root-Index
Nonclustered Index
Leaf-Node
FID PN RN
FID PN RN
FID PN RN
FID PN RN
Key Values
FID = File ID
PN = Page Nr
RN = Row Nr
Table Data
(Heap)
Table Data
Table Data
Table Data
Table Data

Definition of Bucket

- Bucket - another form of a storage unit that can store one or more records of information.

- Buckets are used if the search key value cannot form a candidate key, or if the file is not stored in search key order.

Choosing Indexing Technique

- Five Factors involved when choosing the indexing technique:
- access type
- access time
- insertion time
- deletion time
- space overhead

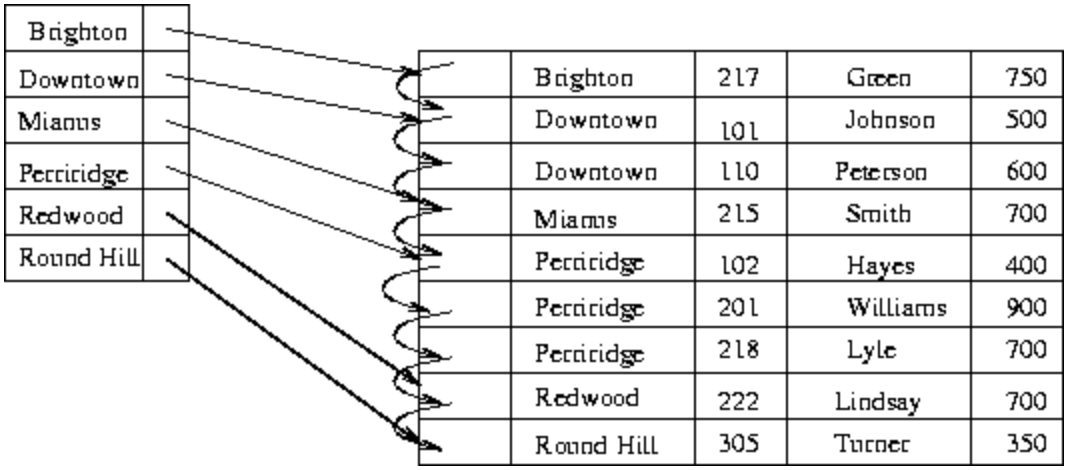
Indexing Definitions

- Access type is the type of access being used.
- Access time - time required to locate the data.
- Insertion time - time required to insert the new data.
- Deletion time - time required to delete the data.
- Space overhead - the additional space occupied by the added data structure.

Types of Ordered Indices

- Dense index - an index record appears for every search-key value in the file.
- Sparse index - an index record that appears for only some of the values in the file.

Dense Index


Brighton
Downtown
Miamus
Perridge
Redwood
Round Hill
Brighton 217 Green 750
Downtown 101 Johnson 500
Downtown 110 Peterson 600
Miamus 215 Smith 700
Perridge 102 Hayes 400
Perridge 201 Williams 900
Perridge 218 Lyle 700
Redwood 222 Lindsay 700
Round Hill 305 Turner 350

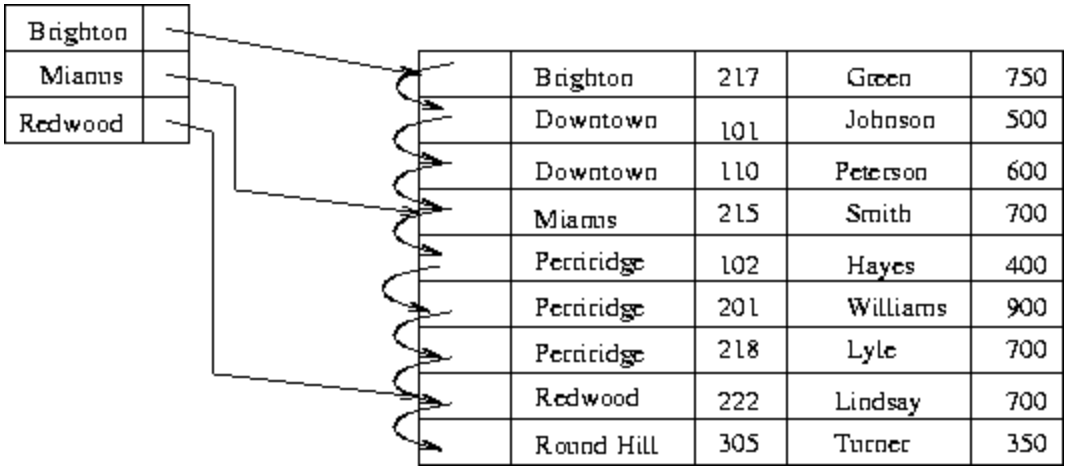
Dense Index Insertion

- if the search key value does not appear in the index, the new record is inserted to an appropriate position
- if the index record stores pointers to all records with the same search-key value, a pointer is added to the new record to the index record
- if the index record stores a pointer to only the first record with the same search-key value, the record being inserted is placed right after the other records with the same search-key values.

Dense Index Deletion

- if the deleted record was the only record with its unique search-key value, then it is simply deleted
- if the index record stores pointers to all records with the same search-key value, delete the point to the deleted record from the index record.
- If the index record stores a pointer to only the first record with the same search-key value, and if the deleted record was the first record, update the index record to point to the next record.

Sparse Index



Sparse Index Insertion

- first the index is assumed to be storing an entry of each block of the file.
- if no new block is created, no change is made to the index.
- if a new block is created, the first search-key value in the new block is added to the index.

Sparse Index Deletion

- if the deleted record was the only record with its search key, the corresponding index record is replaced with an index record for the next search-key value
- if the next search-key value already has an index entry, then the index record is deleted instead of being replaced;
- if the record being deleted is one of the many records with the same search-key value, and the index record is pointing particularly to it, the index record pointing to the next record with the same search-key value is updated as the reference instead.

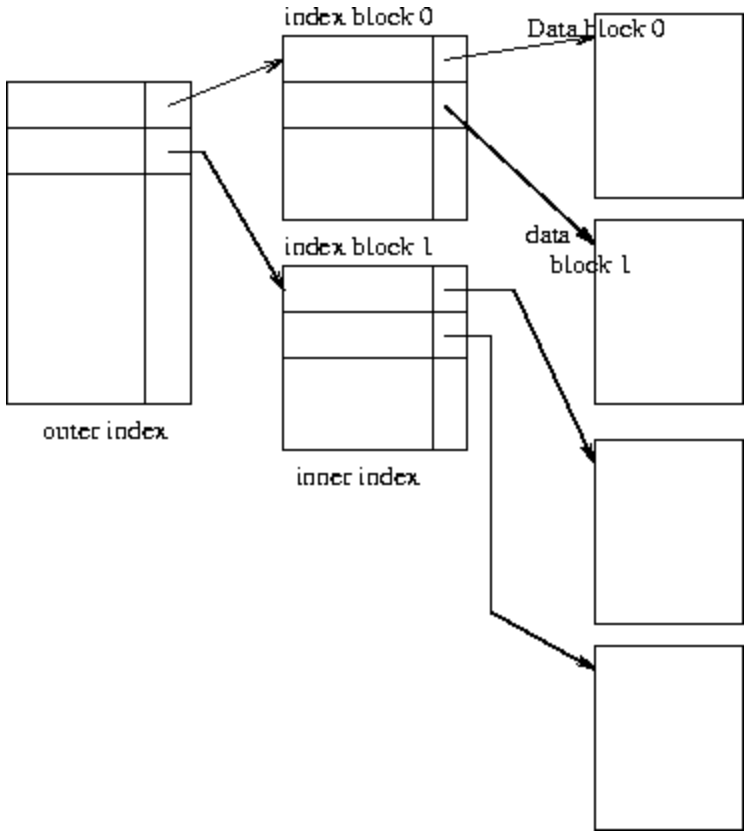
Index Choice

- Dense index requires more space overhead and more memory.
- Data can be accessed in a shorter time using Dense Index.
- It is preferable to use a dense index when the file is using a secondary index, or when the index file is small compared to the size of the memory.

Choosing Multi-Level Index

- In some cases an index may be too large for efficient processing.
- In that case use multi-level indexing.
- In multi-level indexing, the primary index is treated as a sequence file and sparse index is created on it.
- The outer index is a sparse index of the primary index whereas the inner index is the primary index.

Multi-Level Index



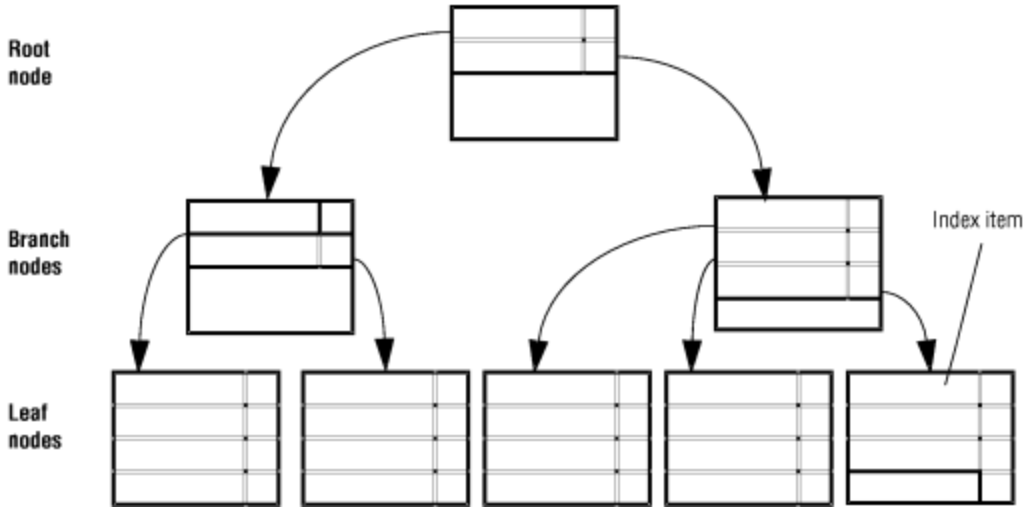
B-Tree Index

- B-tree is the most commonly used data structures for indexing.
- It is fully dynamic, that is it can grow and shrink.

Three Types B-Tree Nodes

- Root node - contains node pointers to branch nodes.
- Branch node - contains pointers to leaf nodes or other branch nodes.
- Leaf node - contains index items and horizontal pointers to other leaf nodes.

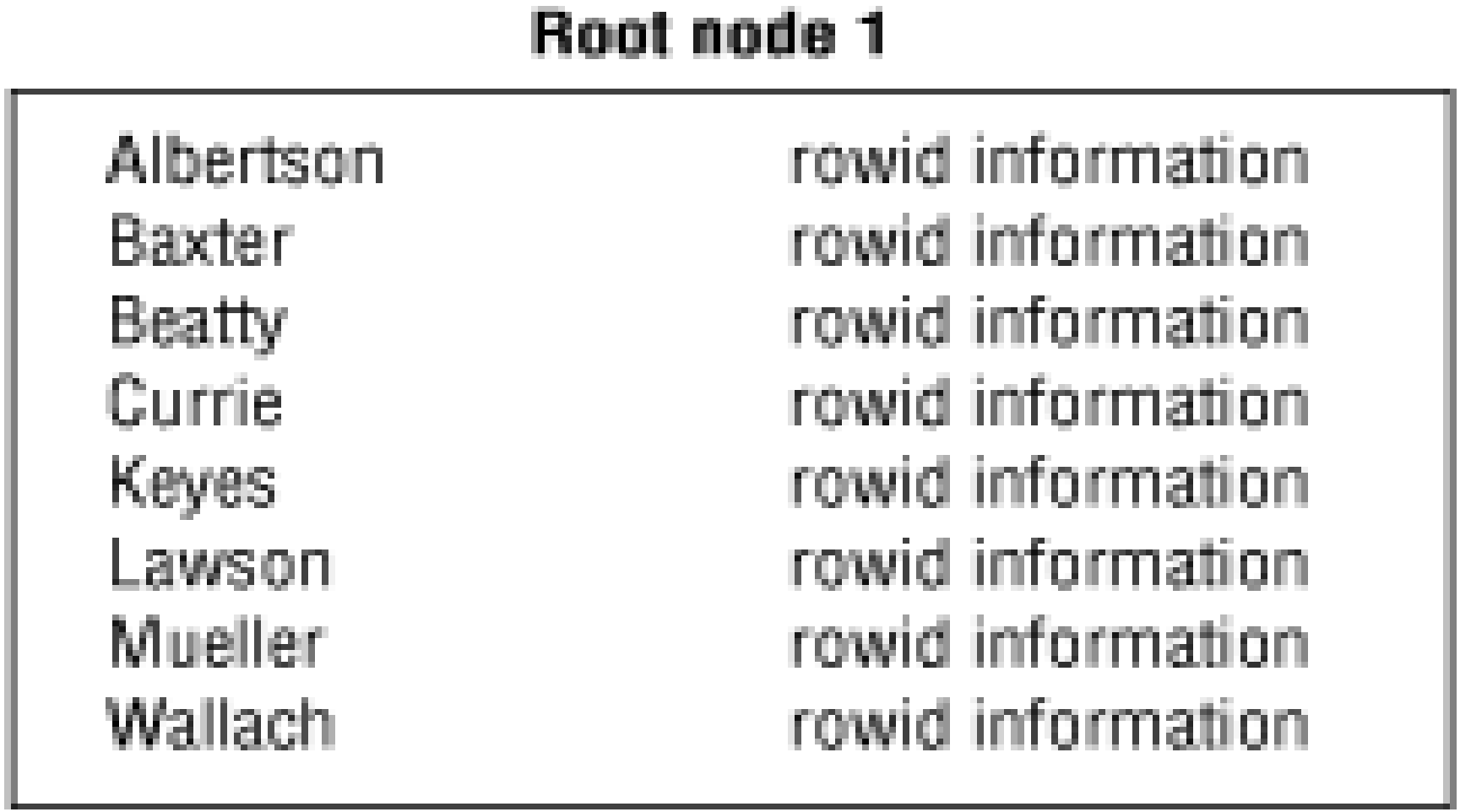
Full B-Tree Structure



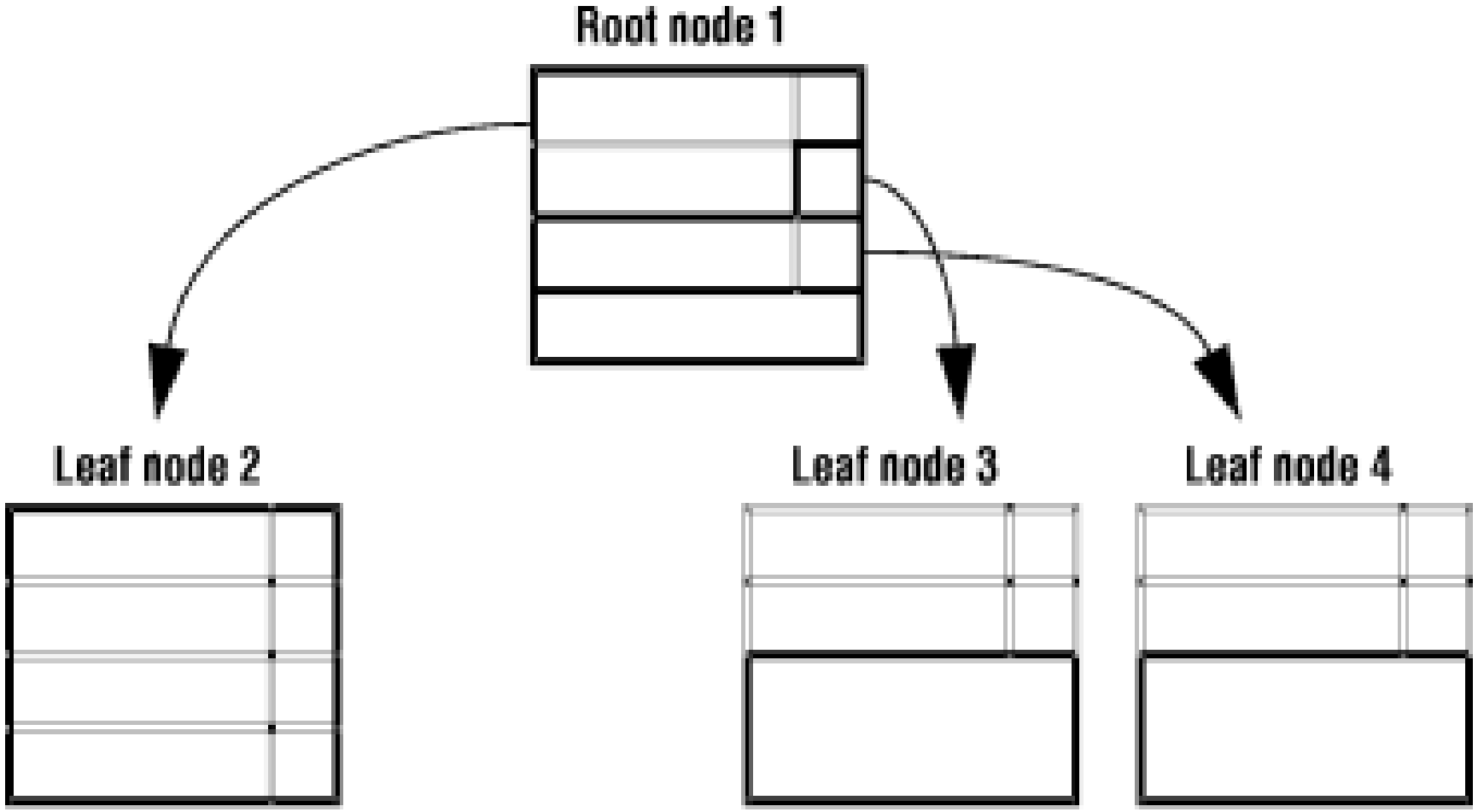
B-Tree Insertion

- First the DBMS looks up the search key value
- if the search key value exists in a leaf node, then a file is added to the record and a bucket pointer if necessary
- if a search-key value does not exist, then a new record is inserted into the file and a new bucket (if necessary) are added
- if there is no search key value and there is no room in the node, then the node is split. In this case, the two resulting leaves are adjusted to a new greatest and least search-key value. After a split, a new node is inserted to the parent. The process of splitting repeats when it gets full.

B-Tree Root Node



Insertion into B-Tree

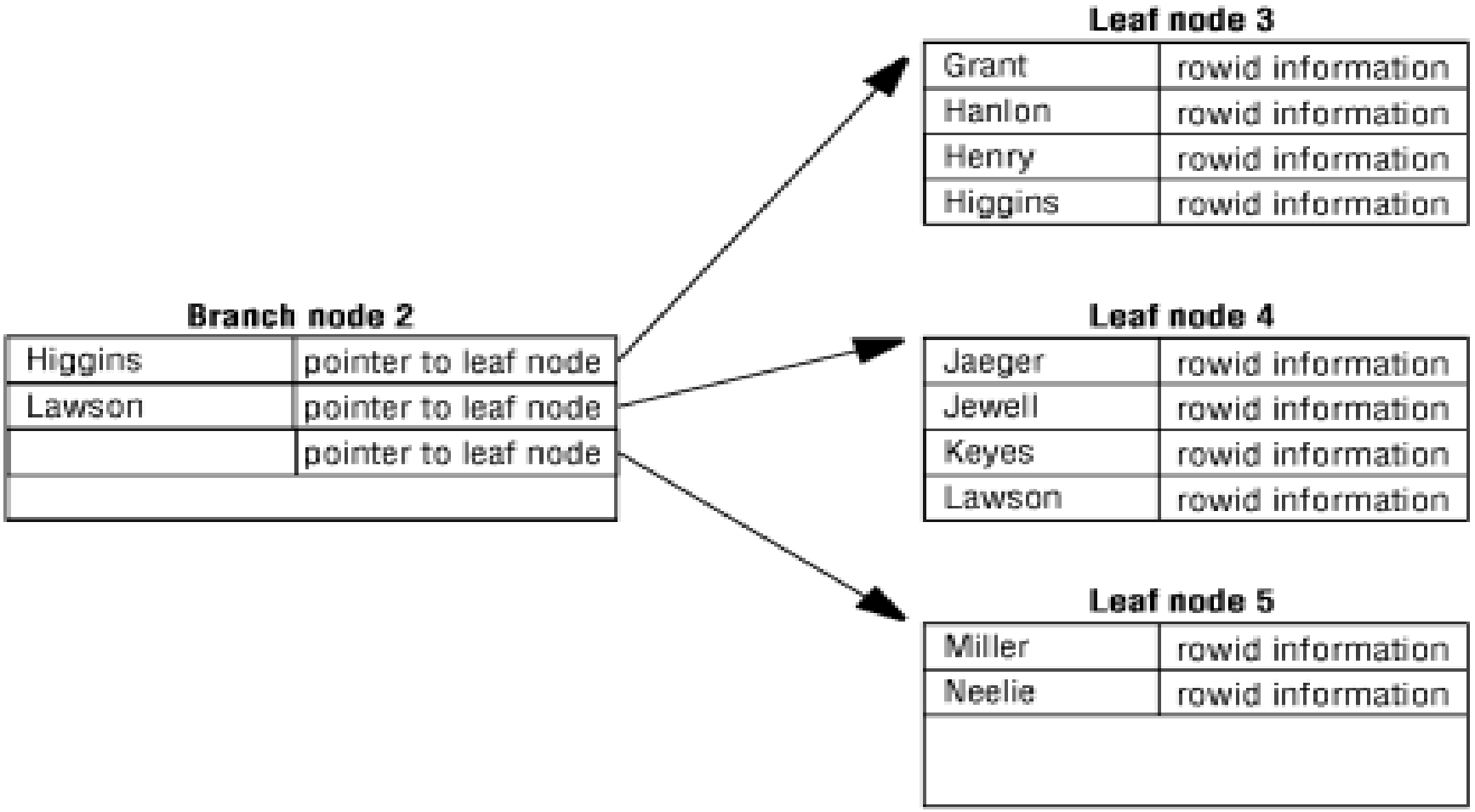


B-Tree Structure

- This process results in a four-level tree, with one root node, two branch levels, and one leaf level.

- The B-tree structure can continue to grow in this way to a maximum of 20 levels.

Branch Node Example



Branch Nodes Pointing to Leaf Nodes

- The first item in the left branch node contains the same key value as the largest item in the leftmost leaf node and a node pointer to it.

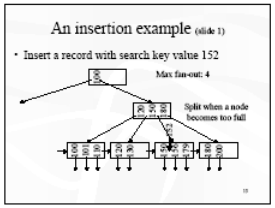
- The second item contains the largest item in the next leaf node and a node pointer to it. The third item in the branch node contains only a pointer to the next higher leaf node.

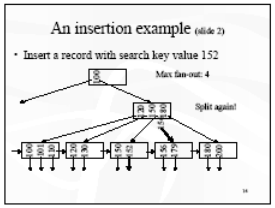
- Depending on the index growth, this third item can contain the actual key value in addition to the pointer at a later point during the lifespan of the index.

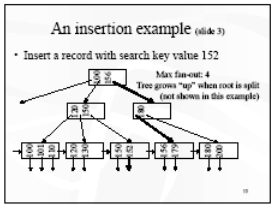
B-Tree Deletion

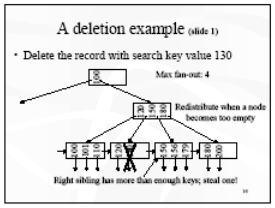
- the DBMS first look up the record and removes it from file
- if no bucket is associated with its search-key value or is empty, the search-key value is removed
- if there are too few pointers in a node, the pointers is then transferred to a sibling node, and it is delete thereafter
- if transferring pointers gives a node to many pointers, the pointers are redistributed.

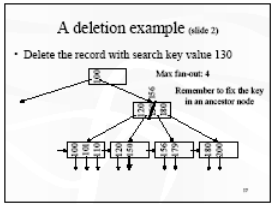
Insertion/Deletion Examples

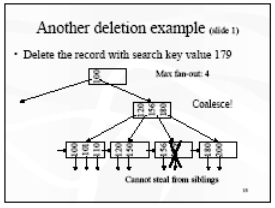












References

- Dr. Lee's Data Indexing Lecture.
- A. Silberschatz, H.F. Korth, S. Sudarshan: **Database System Concepts**, 5th Ed., McGraw-Hill, 2006.
- Umanath, Narayan S. Scamell, Richard W, Data Modeling and Database Design: Thomson, 2007.
- http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.adref.doc/adref235.htm
- http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.adref.doc/adref235.htm