Semih Kaldırım: File Organization, ISAM, B+ Tree and Bulk Loading

Date: Week 13 - 28 April 2015

File Organization

- File is a set of related records.
  Ex: Student(stNo, stName, stDept, stAddress)

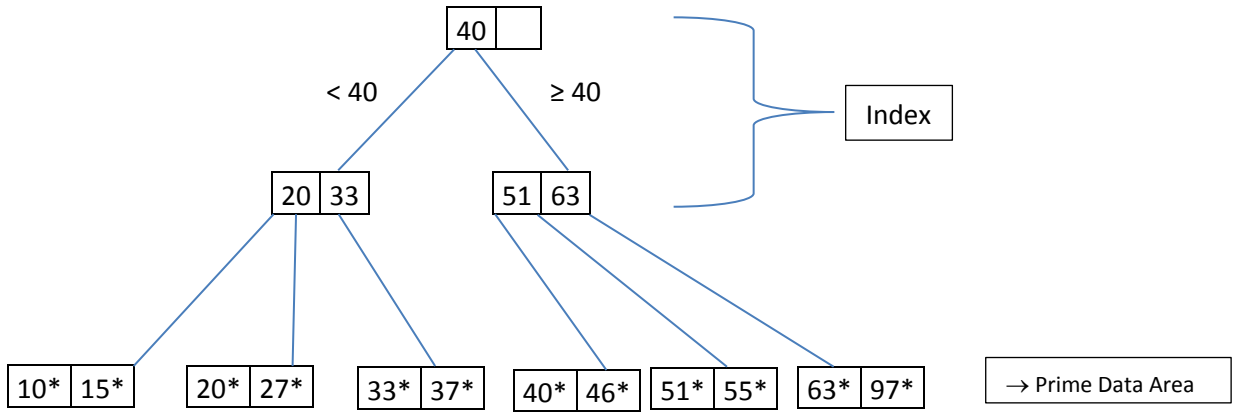
  10 Oya IE Bilkent → Record
  1020 Zeynep IE Çankaya → Record

- How does it work?: i) Index structure: Something we use for searching purposes (speeding up the data retrieval) ii) Page/Block: Unit we access in harddisk/ hard drive (not one record)
- ISAM, B+ Tree and Extendible Hashing are possible index structures

ISAM Structure

- IBM program for mainframes.

ISAM Example:



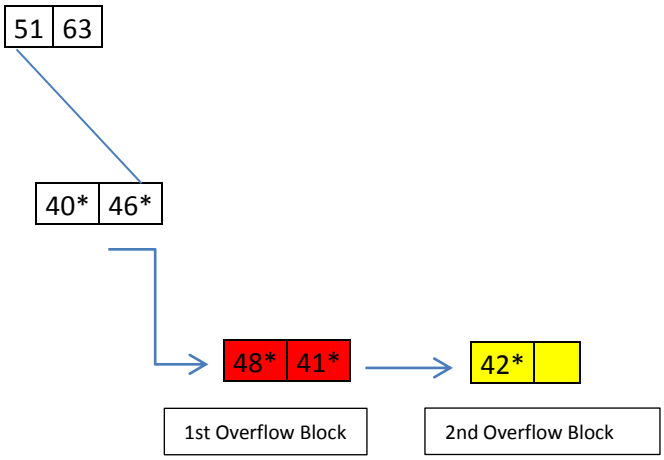
- The expression 10 is a key while 10* is a record

Example: 1) 10 2) 40 3) 15 4) 46 : Process in the given order causes twice processing for 10 and 15. In order to avoid this we accumulate and sort the transactions in the following way:

→ (10 15) & (40 46): Both are applicable with a single (one) disk access.

Note: Prime data area size is fixed.

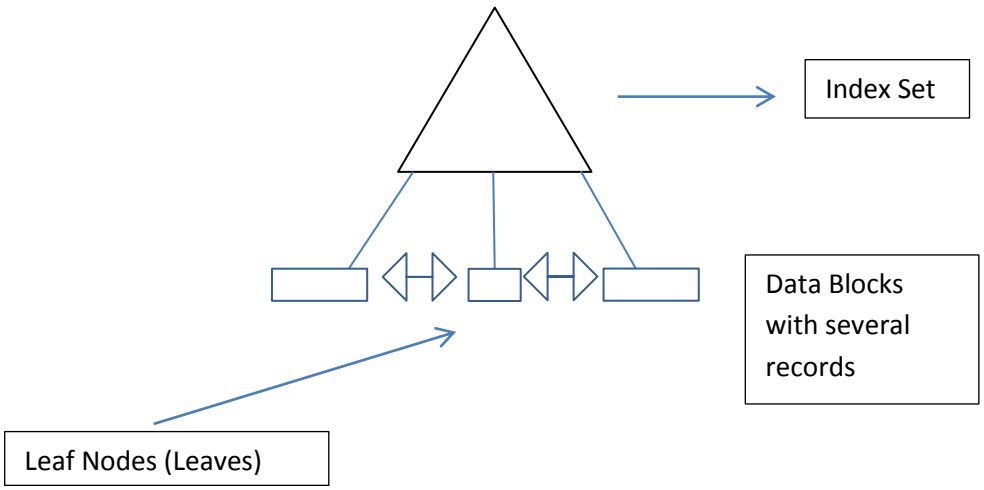
- INSERT 48, 41 and 42 to the previous ISAM index structure:



- In this form, accessing different records require different amounts of time: (40 -46): one disk access, 42: non-uniform response time.
- Due to the records in the overflow area, larger prime data area is needed. To make a larger prime data area file maintenance is required: Copy the file into new ISAM file with a larger prime data area. In addition, for the cases like deletion, file maintenance is also needed if we have several deleted records in the file. These problems with the ISAM Structure are the motivations for the B+ tree structure.

<u>B+ Tree Structure</u>

- Universal File Structure

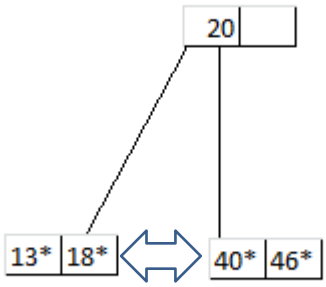


d: Degree (order), 2d: entries / page block: d=1 means two entries, d=2 means four entries

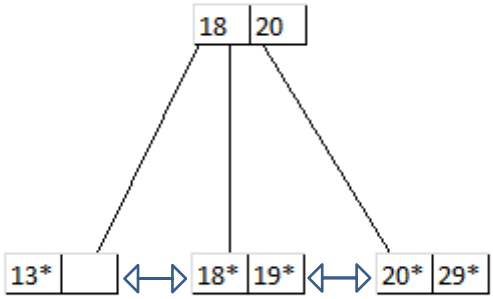
Main Characteristics of B+ Tree

1) Tree is balanced: All leaf nodes are at same level (one level down from the previous level, uniform access time, each data record requires the same time)
2) Each node is at least 50% full, typically 70% full (ROOT EXCEPTION: <50% when d>1)
3) No overflow blocks.
4) d=100 typically.

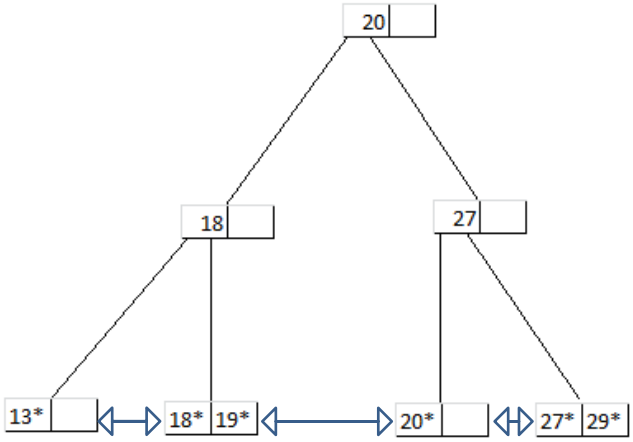
Example:



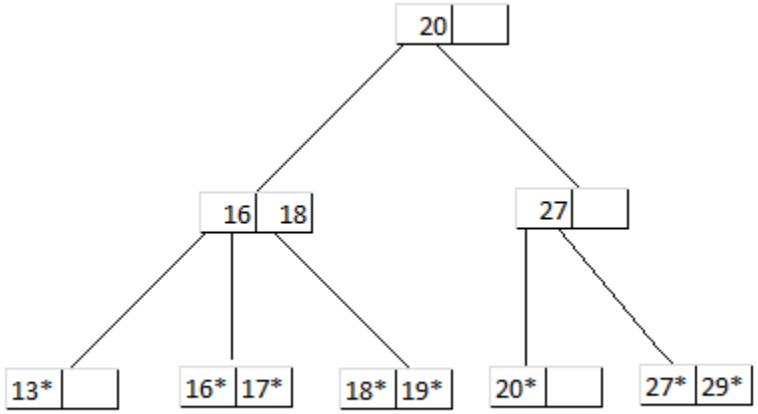
Insert 19: No room for 19. Split the node into two parts. Between 13-18-19, 18 is in the middle. Move it to the upper node:



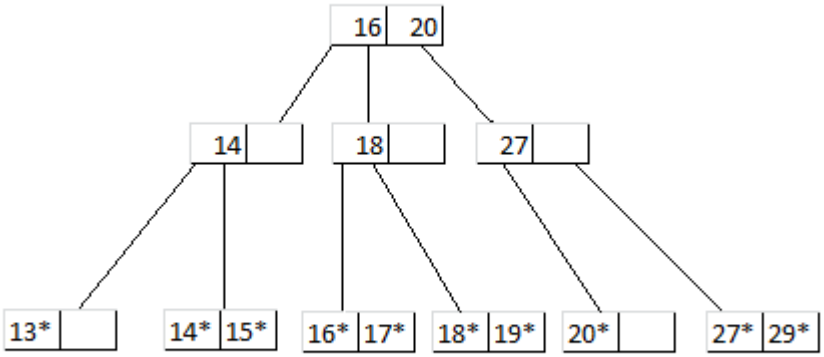
Insert 27: No room for 27. Between 20-27-29, 27 moves to upper node. Between 18-20-27, 20 moves to the upper node:



Insert 16 and 17:



Insert 14 and 15:



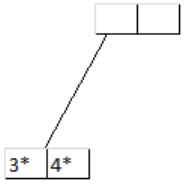
Bulk Loading

- Creationof B+ Tree one by one is expensive (Carry half into the new nodes).

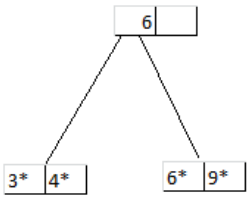
1) Sort the data entries according to the order
2) Allocate an empty page to serve as the root and insert a pointer to the first page of entries into it.
3) When the root is full, split the root and create a new root page.
4) Keep inserting entries to the right most index (node) just above the leaf level until all entries are indexed.

Bulk Loading Example: a) 3*- 4* b) 6*- 9* c) 10* - 11* d) 12* - 13* e) 20*- 22*f) 23* - 31* g) 35* - 36* h) 38* - 41* i) 44* -

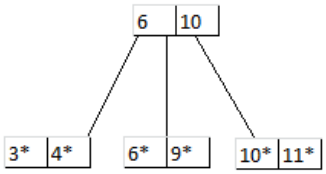
-



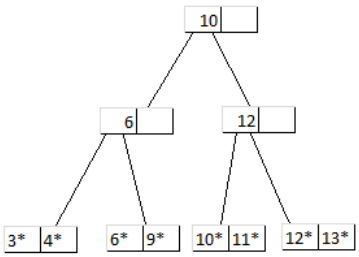
- Insert b:



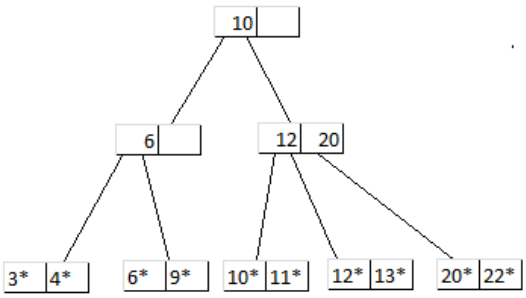
- Insert c:



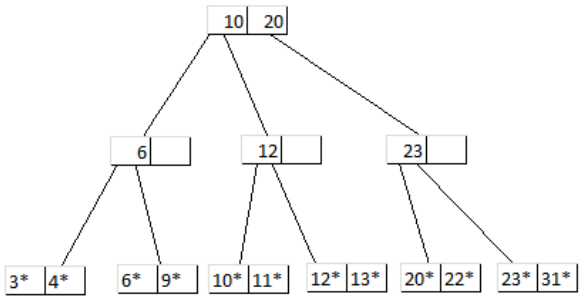
- Insert d:



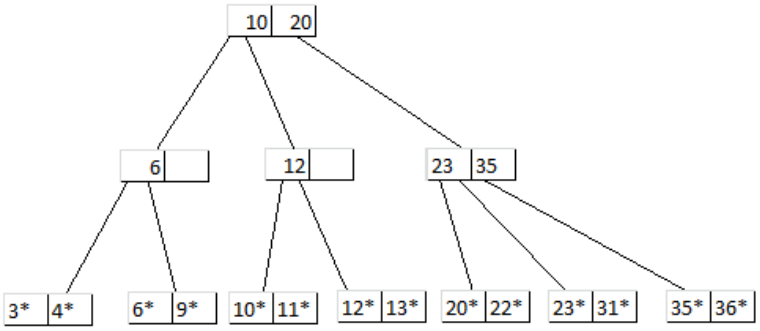
- Insert e:



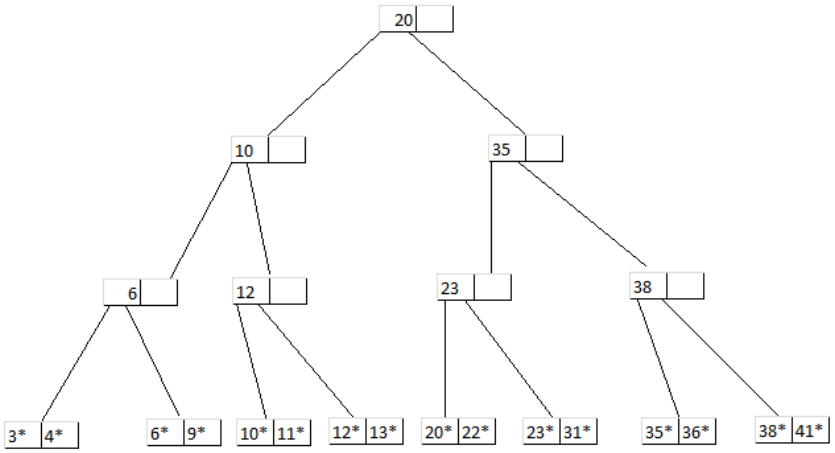
- Insert f:



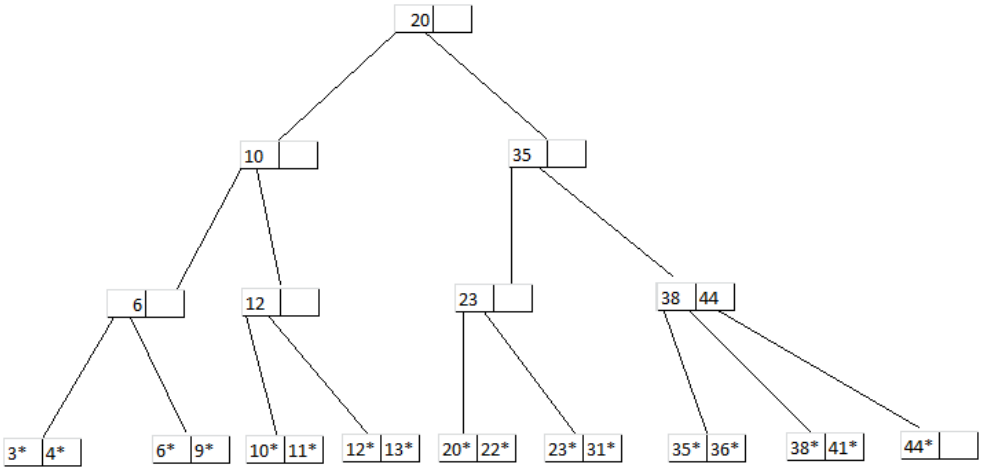
- Insert g:



- Insert h:



- Insert i:



- IMPORTANT NOTE: Despite not shown in some occasions, there is always transition (double-sided arrows) between the leaves of B+ Tree for security procedures.