

# COMP282

## Lecture 17 Introduction to B-Trees

## 2-3 Trees vs B-Trees

- 2-3 Trees have a disadvantage: They are memory based. The whole tree and all its data has to fit in core memory.
- Or does it?...
- B-Trees are used for storing large amounts of data. More than can fit in memory. And they are quite a lot like 2-3 Trees.

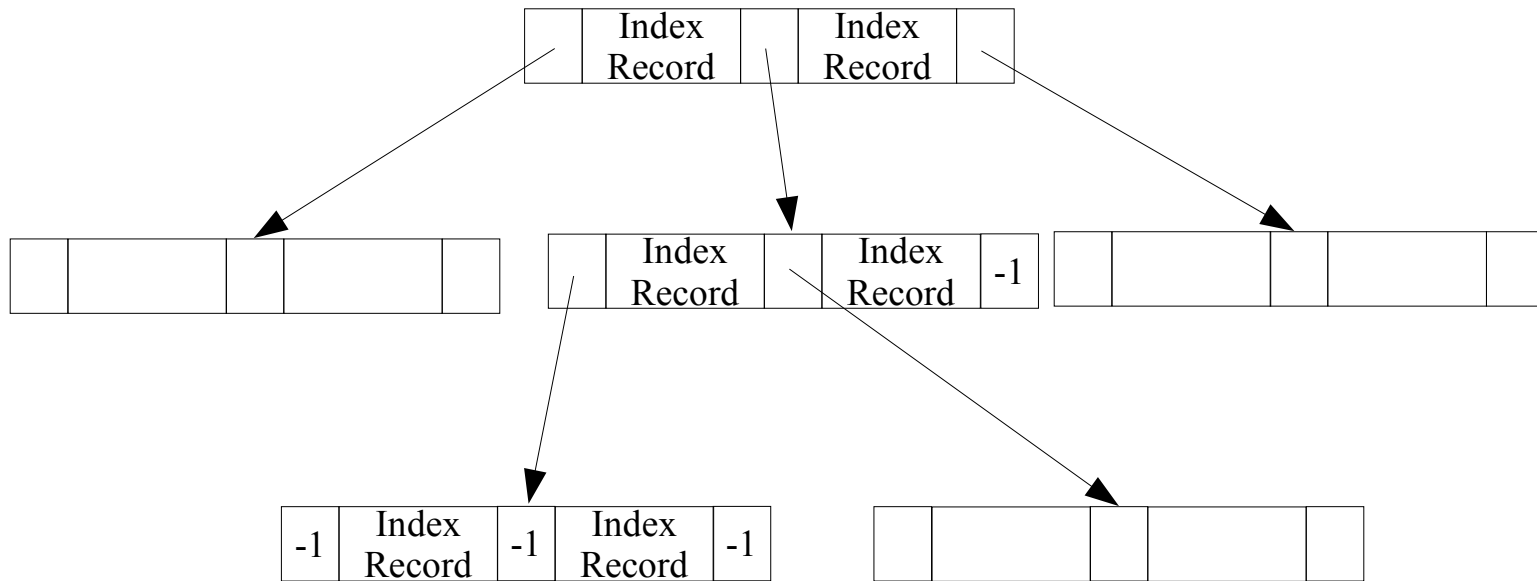
# Key Mapping

- Balanced binary search trees are generally implemented such that key values are mapped to, or associated with, an object that you are interested in storing.
  - `java.util.TreeMap.put(Comparable key, Object o)`
- Only the keys are used in creating and maintaining the topology of the 2-3 Tree.
- Keys are generally much, much smaller than the objects associated with them.

# B-Tree advantage

- B-Trees take advantage of this by maintaining a balanced binary tree structure through the use of two files:
  - index file: contains all the keys and the tree's topology is represented by the organization of data in this file.
  - data file: a file that contains all the objects and information stored by the “tree”. Objects contained here are referenced by block pointer references stored in the index file.

# Index file topology



Block number of left child	key	pointer to data	Block number of middle child	key	pointer to data	Block number of right child
----------------------------	-----	-----------------	------------------------------	-----	-----------------	-----------------------------

## 2-3-4-5-6-7-8-etc Trees

- In 2-3 Trees it was pointed out that 2-3 and 2-3-4 Trees provide advantages but larger nodes didn't appear useful because:
- There is a trade off in terms of the height of the tree and the number of comparisons that are required at each branch point.
- too many items in the nodes and the tree begins to act linear.

# B-Trees and Comparisons

- B-Trees are file based.
- “Descending” the tree requires you to read another block from a random location in the index file.
- reading blocks is far, far more time consuming than processing the information contained in the block.
- We could perform thousands of comparisons in the time required to fetch a single block.

# B-Tree nodes then...

- So we actually want to pack as many index records into each block as possible.
- How large is an index record?
  - number of bytes to maintain a key
  - number of bytes to maintain a data file block reference.
- You also need to account for the child block references.

# index record count

- $N$ =number of bytes in a block
- $b$ =number of bytes for a block reference
- $i$ =number of bytes for an index record
- $m$ =number of index records per block
- $N = mi + (m+1)b$
- $N = m(i+b) + b$
- For simplifying algorithms always choose  $m$  to be odd.