CSE 132C : Database System Implementation

PROJECT 2: BADGERDB B+ TREE IMPLEMENTATION

Discussion : Friday, Apr 28, 2023

Presenter : Kyle Luoma

Project Assignment

- Build internals of a simple RDBMS
- Project 1 : Buffer Manager - > (done)
- Project 2 : B+ Tree Index

Relations And Index

- Relations are stored in files
- However, searching for records can be expensive if we have to scan the whole relation each time
- Solution: Use index to speed up access to record
- One such index is B+ Tree

Tasks

- Implement following functions in Btree.cpp
  - BTreeIndex()
  - ~BTreeIndex()
  - insertEntry()
  - startScan()
  - scanNext()
  - endScan()

BTreeIndex - Constructor

- The constructor has the following parameters:
  - relationName: Name of relation on which to build the index.
  - outIndexName: Return the name of index file.
  - bufMgrIn: Buffer Manager Instance (**The buffer manager**)
  - attrByteOffset The byte offset of the attribute in the tuple on which to build the index.
  - attrType: Datatype of attribute over which index is built

BTreeIndex - Constructor

Tasks:

- Check to see if the corresponding index file exists
  - If it does, open it
  - If not, create it and insert entries for every tuple in the base relation

BTreeIndex - Constructor

Dependencies - Storing Index Nodes:

- Both PageFile and BlobFile implement file interface
  - PageFile is used to store relations
  - BlobFile is an 8KB blob, used to store B+ Tree nodes

- Blob files can be modified arbitrarily

BTreeIndex - Constructor

Dependencies - Iterating over tuples of a given relation:

- Use FileScan Class to get records from relation file

Filescan:

- Scans records in a file
- Used for base relation - not index
- Methods:
  - FileScan()
  - ~FileScan()
  - scanNext()
  - getRecord()

BTreeIndex - Constructor

Tasks:

- Open BlobFile() - to store index nodes
- Get Records from relation file: use FileScan Class
- Create BTree: leaf and non leaf nodes for each level
  - Leaf nodes have <key, recordID> pair
  - Non leaf nodes have <pageNo, key> pair

~BtreeIndex - Destructor

Tasks:

- Clear up state variables
- Call endScan() to stop the fileScan
- Flush file (bufMgr->flushFile())
- Delete BlobFile (used for index)

insertEntry - Insert a new pair <key,rid>

- Calls another recursive function
  - insertRecursive() - recursively traverses the tree to find appropriate place to put entry
  - Call above function with rootNode as starting pageNo.

insertRecursive - Insert a new pair <key,rid>

- Check if node is a leaf node or non leaf
- If non-leaf:
  - find next node to follow: nextNode = findPageNoInNonLeaf(currentNode, key)
  - call insertRecursive() with above node
- If leaf node:
  - split if full: splitLeafNode()
  - insert <Key, RecordID> in new (or old) node
  - set sibling pointers in case of split

insertRecursive - Insert a new pair <key,rid>

- Return from recursive call
  - check if below node was split
  - if so, need to insert key in present node
  - split current node if full: splitNonLeafNode()
  - add key and return
- Reach root node: if root was split:
  - update IndexMetaInfo (struct holding metadata for index file)

startScan - Begin a filtered scan of the index

- E.g., if called with (1, GT, 10, LTE) it means seek entries greater than 1 and less than or equal to 10
- Traverses down the tree: stops at first leaf node containing record matching the key constraint
- Calls ScanNext() to get recordID of next record matching the criteria
- Throws if not found any leaf satisfying the criteria

scanNext - Get next record the matches the scan

- Fetches the record id of the next tuple matching the scan criteria set in startScan
- Throws IndexScanCompletedException if no record satisfying the criteria found

endScan - Terminates current scan

- Unpins pages pinned during scan process
- Resets scan specific variables

Thank you!