CS 1555

Lecture 25

**Data Storage**

Hash files

- Also called direct files

- External hashing maps keys to disk blocks

Collision

- Insertion of a new record may lead to collision

- Probing or conflict resolution

- Open addressing – use different hash functions until free space is found

- Not very efficient for databases

- Chaining – use overflow buckets

Hashing functions

- Good hash functions must be computed efficiently and minimize collisions

Pros and cons of hashing

- Excellent performance for searching on equality on the key used for hashing

- Records are not ordered

- Any operation that doesn’t involve equality is slow

- Prediction of total number of buckets is difficult

Dynamic hashing methods

- Allow file size to change as records are added or deleted

**Access paths**

Access paths or methods

- Translate attribute values into Rid or other type of internal addresses

- Selection predicate: condition based on which the associate access is done

Index structures

- Auxiliary file that makes it more efficient to search for a record in a data file

- Index usually specified on one field value of the data file

- Ordered file of entries <field-value, pointer>

Index-Sequential Access Method (ISAM)

- Primary index, defined on ordered data file based on a key value

- First record in a data block is called block anchor

- Includes one index entry for each block in data file

Advantages of a primary index

- Might be stored in multiple blocks, but occupies much smaller space than a data file

Summary

|  |  |  |
| --- | --- | --- |
|  | **Number of Entries** | **Dense/Sparse** |
| **Primary** | Number of blocks in data file | Sparse |
| **Clustering** | Number of distinct values | Sparse |
| **Secondary (key)** | Number of records in data file | Dense |
| **Secondary (non-key, 1st level)** | Number of distinct values | Sparse |
| **Secondary (non-key, 2nd level)** | Number of records/pointers | Dense |

Multi-level indexes

- Because a single-level index is an ordered file, create a primary index to the index itself

Drawbacks of ISAM

- Static structure

- Needs monitoring for dynamic databases

- Insertion/deletion of new index entry is a problem because every level of the index is an ordered file

- Insertion is handled by some form of overflow blocks

- Active files need frequent reorganization

- No guaranteed performance for searching based on the key for active files

Multi-level index as a tree

- Multi-level index is a form of search tree

- B+ trees: all pointers to data file are at leaf nodes

B-trees and B+-trees

- Variations of search trees that allow efficient insertion and deletion of new values

- Each node in tree is a disk block

- Each node is kept between half-full and completely full

B/B+ tree performance

- Insertion

- Efficient if there is space

- Otherwise a full node is split

- Splitting may propagate to other levels

- Deletion

- Efficient if it does not cause the node to be less than half-full

- Otherwise it must be merged with its sibling node or, if not possible, accept half of the keys of its sibling nodes

B-tree vs B+-tree

- In a B+-tree, all pointers to data records exist at the leaf level nodes

- In a B-tree, pointers to data records exist at all levels

- A B+-tree can have fewer levels than a corresponding B-tree

- In a B+-tree, the search cost is the same for any value, O(log n); the tree is always balanced

B+-tree index

- Tree order is the number of pointers

- Every node, except for the root, has between n/2 and n children or pointers

- Leaf nodes are chained to form a linked-list (fast sequential access)