B+Trees

Glossary for B+Trees

There are a few terms we use in B+Trees

- **Data Record**: an element of data information to be managed by the B+Tree, e.g. a student record with ID number, name, email address etc.
- **Key value**: the value by which we identify the record, e.g. the ID number or the student name.
- **Discriminator**: a value used to decide which path to take down the tree in searching for a record. Almost always the key value of some record, but, in principle, it doesn't have to be.
- **Disk Address**: the offset from the start of the disk file to a particular block in the file. A file read/write can be executed by rquesting the operating system to "*seek*" to this address and read/write a block from/to the file. Think of a disk address as a pointer to disk memory.

Order of a B+Tree

B+Trees nodes are designed to fit in disk blocks so that reading or writing a node corresponds to reading or writing a single disk block (or a sequence of consecutive disk blocks)

Most descriptions of B+Trees start with the *order* of a B+Tree, which is variously defined as the minimum or maximum number of children or the minimum or maximum number of keys in a non-root internal node of the tree (these can be 4 different numbers for the same tree!)

However, in real-world B+Tree implementations, first of all the key values are often variable length strings, and second the the limiting factor on the number of children in each node is not some arbitrary order specification, but instead is decided by how many keys and disk addresses can fit in a single disk block.

- Note that, since keys can be of variable length, the maximum number of children of an internal node of the tree is not a fixed number.

B+Trees

There are 2 variants of B+Trees in common use which we can call *Record-Embedded B+Trees*[1] and *Index B+Trees*[1]

- **Record-Embedded B+Trees** store the data records in the leaf nodes of the B+Tree. This is a suitable data structure when
    - you only need to find records by one search key, e.g. you look up student records by student ID numbers, but not by student name.
    - you want the B+Tree to do all management of the data records, e.g. if you delete the B+Tree, you delete the student records.
- **Index B+Trees** store only key-values and disk addresses in the leaf nodes, which identifies the disk block in the separate file of data records where the record with the corresponding key values reside:
    - The data records are kept in blocks separate from the B+Tree blocks, so dropping the B+Tree does not delete the data records
    - Multiple B+Tree indexes can be created on a collection of data records, e.g. one indexing on student IDs, another on student names.

[1]Not standard terminology, but there is none for these variants