COURSE OVERVIEW

Fundamental file processing operations

- open, close, read, write, seek (file as a stream of bytes)

Secondary Storage Devices and System Software

- how different secondary storage devices work (tapes, magnetic disks, CD-ROM)
- the role of different basic software and hardware in I/O (operating system (file manager), I/O processor, disk controller)
- buffering at the level of the system I/O buffers.

Logical view of files and file organization

- file as a collection of records (concepts of records, fields, keys)
- record and field structures
- sequential access; direct access (RRN, byte offset).

Organizing files for performance

- data compression
- reclaiming space in files: handling deletions, AVAIL LIST, etc.
- sorting and searching: internal sorting, binary searching, keysorting.

Cosequential processing

- main characteristics: **sequential access** to input and output files, and **co-ordinated access** of input files.
- main types of processing: matching (intersection) and merging (union).
- main types of application: the merging step in an external sorting method; posting of transactions to a master file.

Indexing

- Primary and secondary key indexes.
- Maintenance of indexed files; different index organizations.
- Inverted lists for secondary indexes.
- Alternative ways of organizing an index:
  - Simple index
    keeping index sorted by key (additions and deletions are expensive: about $O(n)$ disk accesses)

  - B trees and B+ trees
    improvement in time: searches, insertions and deletions in about $O(\log_k n)$ disk accesses, where k is the order of the tree and n is the number of records for B trees or the number of blocks of records for B+ trees.

  - Hashed index: searches, insertions and deletions in constant expected time, i.e. expected time $O(1)$, provided that the hash function disperses the keys well (approximately random).

B trees and B+ trees

- We started from the problem of maintenance of simple indexes.
- B trees: multi-level index that work from bottom up and guarantees searches and updates in about $O(\log_k n)$ disk accesses. More precisely, the worst case search time, insertion time and deletion time (in number of disk accesses) is in the worst-case proportional to the height of the B-tree. The maximum B tree height is $1 + \log_{\lceil k/2 \rceil} n/2$, where n is the number of keys and k is the order of the B tree.
- Then, we discussed the problem of having both an indexed and a sequential view of the file: B+ trees.
- B+ tree = sequence set + index set
  The index set is a B tree; the sequence set is like a doubly linked list of **blocks** of records.
- simple prefix B+ trees: B+ trees in which we store separators (short prefixes) rather than keys, in the index set.
- We learned B trees and B+ trees maintenance and operations.

Hashing and Extedible Hashing

- We have discussed **static hashing** techniques: expected time for access is $O(1)$ when when hash function is good and file doesn't change too much (not many deletions/additions).
- We have discussed: hash functions, record distribution and search length, the use of buckets, collision resolution techniques (progressive overflow, chained progressive overflow, chaining with a separate overflow area, scatter tables, patterns of record access).
- We have discussed **extendible hashing**. In extendible hashing, hashing is modified to become self-adjusting, allowing for the desired expected $O(1)$ access even when the file grows a lot; the address space changes after a lot of insertions or deletions. A good hash function is still required.
- If you cannot predict which one could be a good hash function for a dataset, it is preferable to use a B tree or B+ tree (e.g. general purpose data base management systems). If the hash function is not good, hashing may lead to an $O(n)$ performance (e.g. assignment 3), which is prohibitive, while B trees and B+ trees have a guaranteed worst case performance which is quite good.

What's next: Database Management Systems CSI3317

Different layers:

Database management systems
File systems
Physical storage devices

Each layer hides details of the lower level.
This course focused in **file processing**. In order to do file processing efficiently we studied some key issues concerning **physical storage devices**.
CSI3317 will focus on **database management systems**.

As the number of users and applications in an organization grows, file processing evolves into database processing.
A **database management system** is a large software that maintains the data of an organization and mediates between the data and the application programs.
Each application program asks the DBMS for data, the DBMS figures out the best way to locate the data.
The applications are coded without knowing the physical organization of the data.
File processing is done by the DBMS rather than the application program.
The application program describes the database at a high-level, conceptual view; this high-level description is called a data model.
One of the most popular data models is the relational data model; SQL is a commercially used, standard relational-database language.

Difficulties and issues handled by a DBMS:

- data independence: changes in file organization should not require changes in the application programs; example of common changes: add new fields to records of a file, add an index to a file, add and remove secondary indexes.
- data sharing: care must be taken when several users may be reading and modifying the data.
- data integrity: ensuring consistency of data (when data is updated, related data must be updated)

You you learn a lot about this in CSI 3317 …