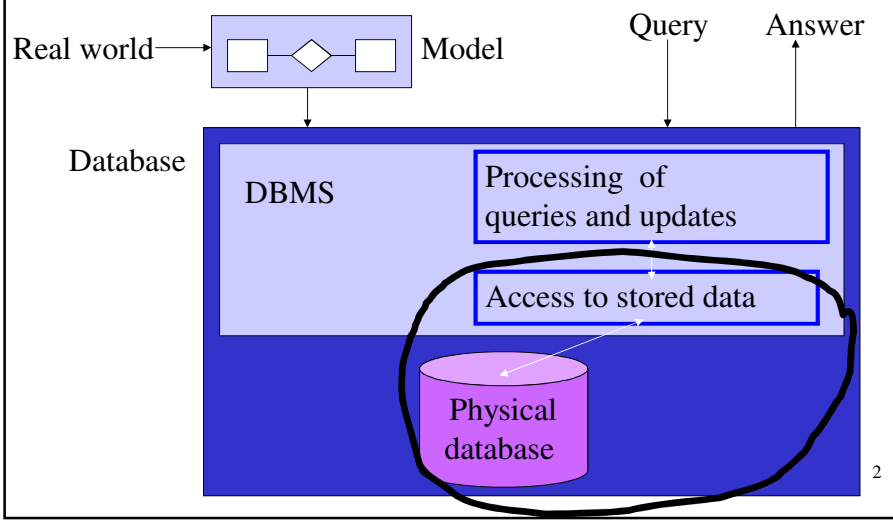
Lecture 7: Data structures for databases I

Jose M. Peña
jose.m.pena@liu.se

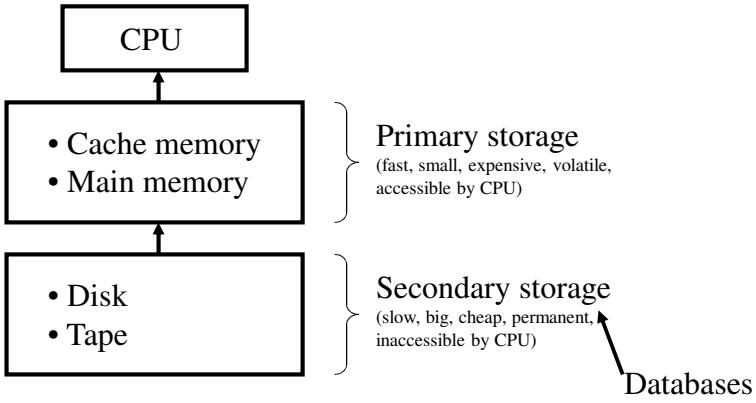
1

Database system



2

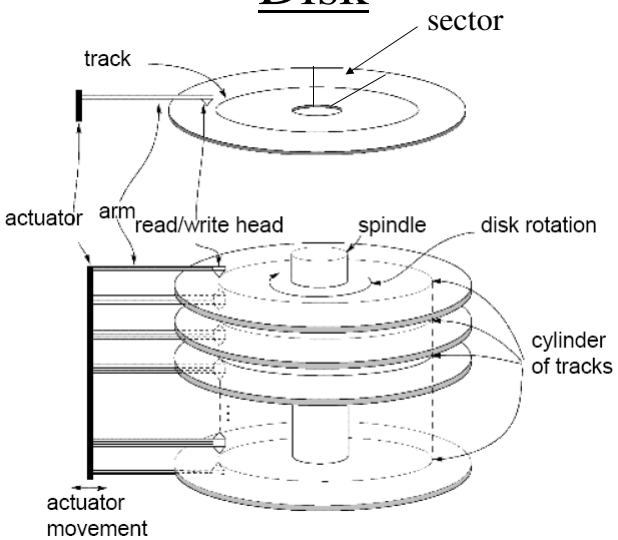
Storage hierarchy



- Important because it affects query efficiency.

3

Disk



4

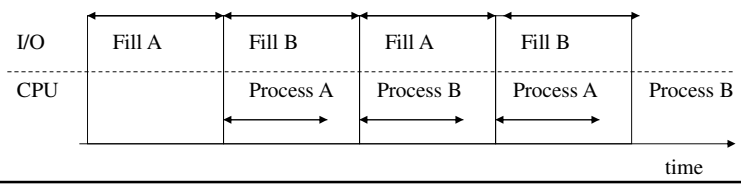
Disk

- Formatting divides the hard-coded sectors into equal-sized **blocks**.
- Block is the **unit of transfer of data** between disk and main memory, e.g.
  - Read = copy block from disk to buffer in main memory.
  - Write = the opposite way.
  - R/w time = seek time + rotational delay + block transfer time.
    search track: seek time; search block: rotational delay; 1-2 msec.: block transfer time; 12-60 msec.: total

5

Disk

- So, read/write to disk is a **bottleneck**, i.e.
  - Disk access $\approx 10^{-3}$ sec.
  - Main memory access $\approx 10^{-8}$ sec.
  - CPU instruction $\approx 10^{-9}$ sec.
- Double buffering helps to alleviate it (if several CPUs or at least a separate disk I/O processor is available).



6

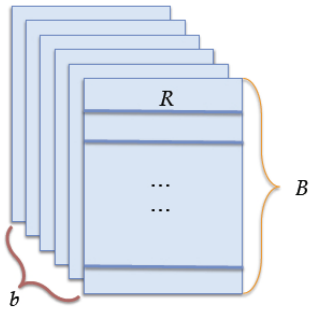
Files and records

- Data stored in **files**.
- File is a **sequence** of **records**.
- Record is a set of field values.
- For instance, file = relation, record = entity, and field = attribute.
- Records are allocated to file **blocks**.

7

Files and records

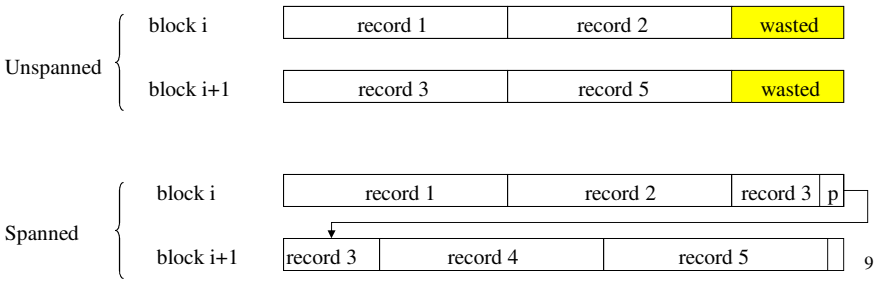
- Let us assume
  - B is the size in bytes of the block.
  - R is the size in bytes of the record.
  - r is the number of records in the file.
- **Blocking factor**, i.e. number of recors per block: $bfr = \left\lfloor \frac{B}{R} \right\rfloor$
- Blocks needed to store the file: $b = \left\lceil \frac{r}{bfr} \right\rceil$
- What is the space wasted per block ?



8

Files and records

- Wasted space per block = B – bfr * R.
- Solution: **Spanned** records.



9

File allocation

- How to allocate **file** blocks to **disk** blocks.
- **Contiguous** allocation: The file blocks are allocated one after another in disk. Then, cheap sequential access but expensive record addition.
- **Linked** allocation: The file blocks are allocated in a linked list of disk blocks. Then, expensive sequential access but cheap record addition.
- **Linked clusters** allocation. Hybrid of the two above.
- **Indexed** allocation.

10

File organization

- How the records are arranged in the file.
- **Heap** files.
- **Sorted** files.
- **Hash** files.

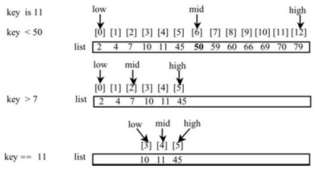
- File organization != access method, although it determines the **primary** access method.

11

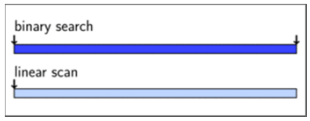
Heap files

- Records are added to the **end** of the file. Hence,
  - **Cheap** record addition.
  - **Expensive** record retrieval, removal and update, since they imply **linear search**:
    - **Average** case: $\left\lceil \frac{b}{2} \right\rceil$ block accesses.
    - **Worst** case: b block accesses.
  - Moreover, record removal implies **waste** of space. So, periodic reorganization.
- Heap file, contiguous allocation, and unspanned blocks. What is the disk block and record of the i-th file record?

12



Sorted files



- Records **ordered** according to some field. So,
  - **Cheap** ordered record retrieval (on the ordering field, otherwise expensive):
    - All the records: Access the blocks sequentially.
    - Next record: Probably in the same block.
    - Random record: **Binary search**, then worst case implies $\lceil \log_2 b \rceil$ block accesses.
  - **Expensive** record addition, but less expensive record deletion (deletion markers + periodic reorganization).
- Is record updating cheap or expensive ?

13

Internal hash files

- The hash function is applied to the hash field and returns the **position** of the record in the **file**. E.g.

  position = field **mod** r
- **Collision**: different field values hash to the same position. Solutions:
  - Check subsequent positions until one is empty.
  - Use a second hash function.
  - Put the record in the **overflow** area and link it.

14

External hash files

- The hash function returns a **bucket** number, where a bucket is one or several contiguous disk blocks. A table converts the bucket number into a disk block address.
- Collisions are typically resolved via overflow area.
- **Cheapest** random record retrieval (when searching for **equality**).
- **Expensive** ordered record retrieval.
- Is record updating cheap or expensive ?

15