Storing Data: Disks and Files

Linda Wu

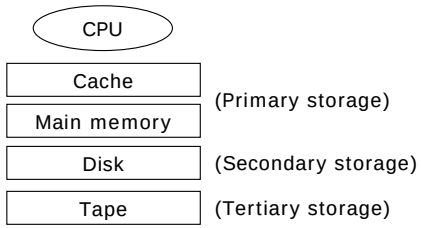
(CMPT 354 • 2004-2)

Topics

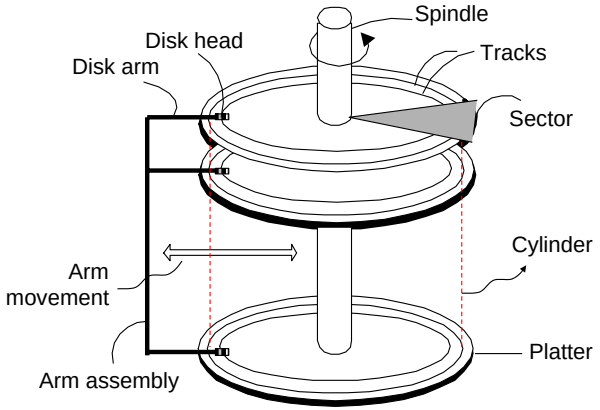
- Typical storage hierarchy
- Disks
- Disk space management
- Buffer management
- Record formats
- Page formats
- File of records

Typical Storage Hierarchy

- Main memory (RAM): currently used data
- Disk: the main database
  - READ: transfer data from disk to main memory
  - WRITE: transfer data from main memory to disk
- Tapes: to archive older versions of the data



Disks



Disks (Cont.)

- The platters spin (say, 90rps)
- Track: concentric rings on platters
- Cylinder: the set of all tracks with the same diameter
- Sector: each track is divided into fixed-sized arcs, called sectors
- Block: the unit in which data is written and read from disk; block size is a multiple of sector size
- Disk heads: move as a unit; only one head read/write at any one time
- Arm assembly: move in or out to position a disk head on a desired track

Disks (Cont.)

- Reading / Writing a disk block is called an I/O operation
  1) seek time (moving the disk head to the track with the desired block): 1~20 ms
  2) rotational delay (waiting for the desired block to rotate under the disk head): 0~10 ms; usually less than seek time
  3) transfer time (actually moving the data in the block): 1 ms per 4KB block

  * Seek time and rotational delay dominate
  * Relative placement of blocks on disk affect I/O operation cost

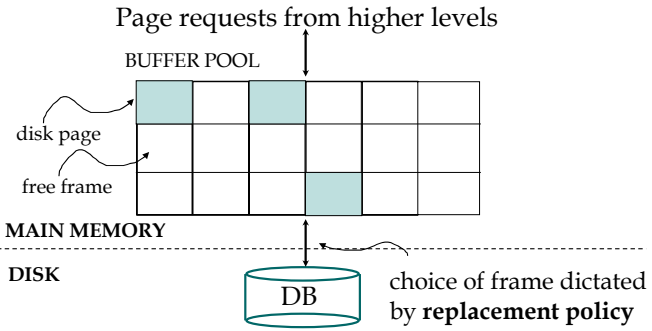
Disks (Cont.)

- Key to lower I/O cost: reduce seek/rotation delays!
- "Next" block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder
- To minimize seek and rotational delay, blocks in a file should be arranged sequentially on disk (by "next"): sequential access is faster than random access!

Disk Space Management

- Disk space manager
  - The lowest layer of DBMS software that manages space on disk
  - Support the concept of a page as a unit of data; size of a page = size of a disk block
  - Keep track of free blocks: linked list of free blocks, or, block bitmap
- Higher layers call upon this layer to:
  - allocate/deallocate a page
  - read/write a page
- Accessing data in sequential order must be satisfied by allocating the pages sequentially on disk

Buffer Management



- Data must be in RAM for DBMS to operate on it
- Table of <frame#, pageid> pairs is maintained

Buffer Management (Cont.)

- Buffer manager
  - Buffer pool
    - The main memory page in the buffer pool are called frames
  - Two variables for each frame
    - *pin_count*: # of current users of the page
    - *dirty* bit: has the page been modified since it was brought into the buffer pool?

Buffer Management (Cont.)

- When a page is requested:
  - The buffer manager checks the buffer pool to see if some frame contains the requested page
  - If so, (*pin_count* ++) for that frame;
  - If the requested page is not in the buffer pool,
    - Choose a frame for replacement
    - If the replacement frame is dirty, write it back to disk
    - Read the requested page into the replacement frame
  - Pin the requested page and return its address in the main memory

Buffer Management (Cont.)

- More on replacement
  - If a requested page is not in the buffer pool and a free frame is not available
    - A frame with *pin_count* = 0 is chosen for replacement according to the buffer replacement policy
    - If no page in the pool has *pin_count* 0, the buffer manager has to wait till some page is released; the requesting transaction may be simply aborted

Buffer Management (Cont.)

- The requestor of a page must unpin it and inform the buffer manager whether the page has been modified
  - *dirty* bit is used for this
- If a page is requested by several different transactions
  - Each transaction should obtain a lock on the page before it read or modify the page (locking protocol)

Buffer Management (Cont.)

- Buffer Replacement Policy
  - Least recently used (LRU)
    - A queue of pointers to frames with *pin_count* 0
    - A frame is added to the end of the queue when it becomes a candidate for replacement
    - The frame at the head of queue is chosen for replacement
  - Clock
  - First in first out (FIFO)
  - Most recently used (MRU)

Buffer Management (Cont.)

- The replacement policy can have big impact on # of I/O's, depending on the access pattern
- Sequential flooding: nasty situation caused by LRU and repeated sequential scans
  - If # buffer frames < # pages in file, each page request causes an I/O with LRU. MRU is much better in this situation (but not in all situations, of course).

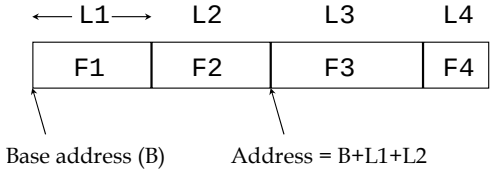
DBMS vs. OS File System

Why not let OS manage these disk space and buffer?

- Self-contained DBMS code: portability issues
- Some limitations, e.g., files can't span disks
- Buffer management in DBMS requires ability to:
  - Adjust replacement policy, and pre-fetch pages based on page reference patterns in typical DB operations
  - Pin a page in buffer pool, force a page to disk (important for implementing CC & recovery)

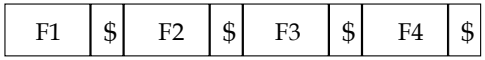
Record Formats

- Fixed-length records
  - Information about # of files and field types is same for all records in a file; it is stored in system catalogs
  - The fields can be stored consecutively
  - Finding i'th field does not require scan of record



Record Formats (Cont.)

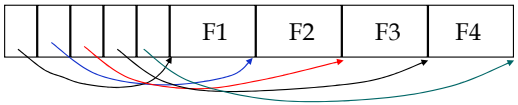
- Variable-length records
  - # of fields is fixed; the field length is not fixed
  - Format 1
    - Fields are stored consecutively; separated by delimiters
    - Finding i'th field requires scan of record



Fields delimited by special symbols

Record Formats (Cont.)

- Format 2
  - Reserve some space at the beginning of a record for use as an array of field offsets
  - Offer direct access to any field
  - Efficient storage of NULL values
  - May store fixed length records as well



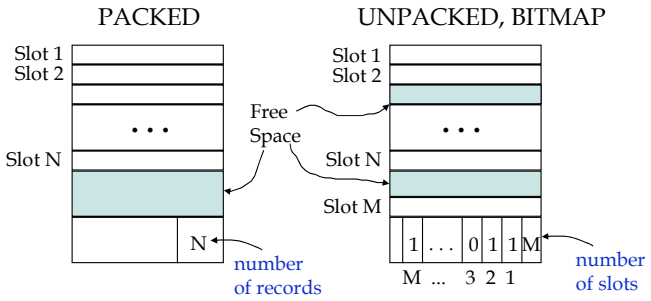
Array of Field Offsets

Page Formats

- Page is a collection of slots
- Each slot contains a record
- A record is uniquely identified by record id = <page id, slot #>

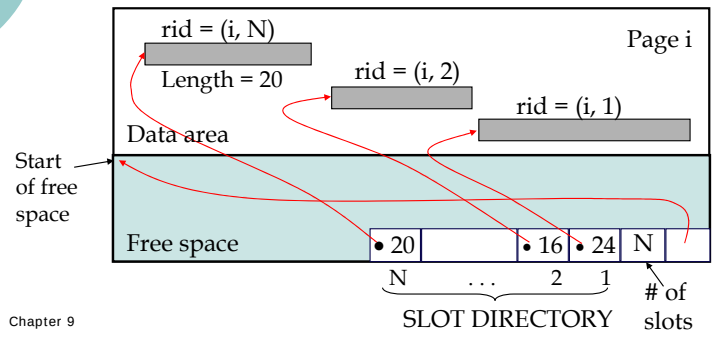
Page Formats (Cont.)

- Fixed-length record



Page Formats (Cont.)

- Variable-length records
  - records can be moved on page without changing rid



Files of Records

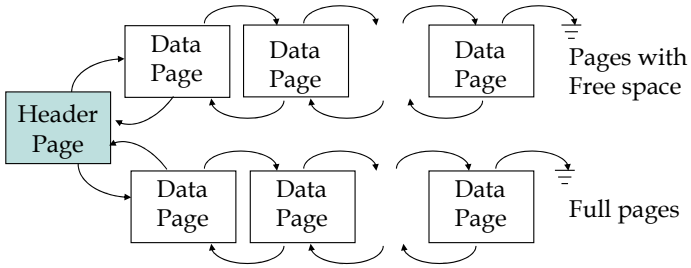
- Disk/buffer manager works with pages and blocks, but higher levels of DBMS operate on records, and files of records
- File: a collection of pages, each containing a collection of records
- Supported operations on a file:
  - insert/delete/modify a record with a given rid
  - read a particular record with a given rid
  - scan all records (possibly with some conditions on the records to be retrieved)

Files of Records (Cont.)

- Heap files
  - The simplest file structure that contains records in no particular order
  - As file grows and shrinks, disk pages are allocated and de-allocated
  - To support record level operations, we must
    - keep track of the *pages* in a file
    - keep track of *free space* on pages
    - keep track of the *records* on a page
  - There are many alternatives for keeping track of this

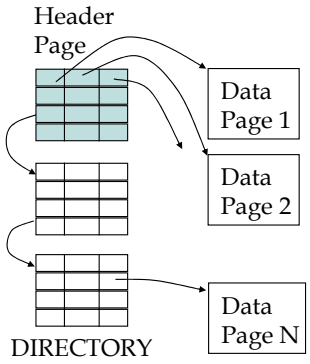
Files of Records (Cont.)

- Heap file implemented as linked list
  - The pairs <file_name, header_page_addr> must be stored in a known location on disk
  - Each page contains 2 pointers (page id) plus data



Files of Records (Cont.)

- Heap file using a page directory
  - The directory entry for a data page can include the number of free bytes on the page
  - The directory is a collection of pages
    - Much smaller than the linked list of all heap file pages



Summary

- Disks provide cheap, non-volatile storage
- It is important to arrange data sequentially to minimize seek and rotation delays
- Buffer manager brings pages into RAM
  - Page stays in RAM until released by requestor
  - Written to disk when frame is chosen for replacement
  - Choice of frame to replace based on replacement policy
  - Tries to pre-fetch several pages at a time

Summary (Cont.)

- Variable length record format with field offset directory offers support for direct access to i'th field and null values
- Slotted page format supports variable length records and allows records to move on page
- At file layer, pages with free space are identified using linked list or directory structure