Storing Data: Disks and File Organizations

"Yea, from the table of my memory
I'll wipe away all trivial fond records."
-- Shakespeare, *Hamlet*

Disks and Files

- DBMS stores information on ("hard") disks.
- This has major implications for DBMS design!
  - READ: transfer data from disk to main memory (RAM).
  - WRITE: transfer data from RAM to disk.
  - Both are high-cost operations, relative to in-memory operations, so must be planned carefully!

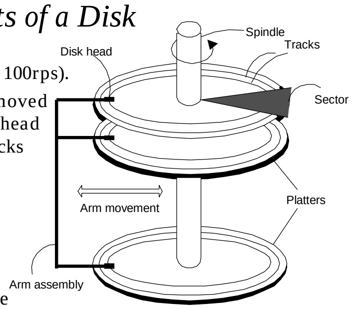
Why Not Store Everything in Main Memory?

- *Costs too much.* $1000 will buy you over 128MB of RAM or 7.5GB of disk today.
- *Main memory is volatile.* We want data to be saved between runs. (Obviously!)
- Typical storage hierarchy:
  - Main memory (RAM) for currently used data.
  - Disk for the main database (secondary storage).
  - Tapes for archiving older versions of the data (tertiary storage).

Disks

- Secondary storage device of choice.
- Main advantage over tapes: <u>*random access*</u> vs. *sequential.*
- Data is stored and retrieved in units called *disk blocks* or *pages.*
- Unlike RAM, time to retrieve a disk page varies depending upon location on disk.
  - Therefore, relative placement of pages on disk has major impact on DBMS performance!

Components of a Disk



- The platters spin (say, 100rps).
- The arm assembly is moved in or out to position a head on a desired track. Tracks under heads make a *cylinder* (imaginary!).
- Only one head reads/writes at any one time.
- *Block size* is a multiple of *sector size* (which is fixed).

Accessing a Disk Page

- Time to access (read / write) a disk block:
  - *seek time* (moving arms to position disk head on track)
  - *rotational delay* (waiting for block to rotate under head)
    - often called "rotational latency"
  - *transfer time* (actually moving data to / from disk surface)
- Seek time and rotational delay dominate.
  - Seek time varies from about 1 to 20msec
  - Rotational delay varies from 0 to 10msec
  - Transfer rate is about 1msec per 4KB page
- Key to lower I / O cost: reduce seek / rotation delays! Hardware vs. software solutions?

Arranging Pages on Disk

- ‘*Next*’ block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder
- Blocks in a file should be arranged sequentially on disk (by `next’), to minimize seek and rotational delay.
- For a sequential scan, *pre-fetching* several pages at a time is a big win!

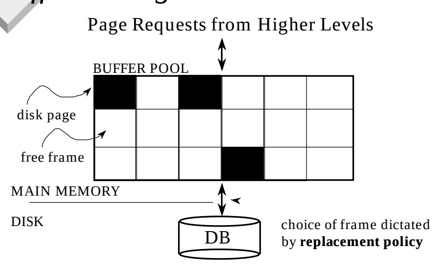
7

Disk Space Management

- Lowest layer of DBMS software manages space on disk.
- Higher levels call upon this layer to:
  - allocate / de-allocate a page
  - read / write a page
- One such “higher level” is the buffer manager, which receives a request to bring a page into memory and then, if needed, requests the disk space layer to read the page into the buffer pool.

8

Buffer Management in a DBMS



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

9

When a Page is Requested ...

- If requested page is not in pool:
  - Choose a frame for *replacement*; pin_count := 1
  - If frame is dirty, write it to disk
  - Read requested page into chosen frame
- Else:
  - increment pin_count
- Return its address.

☛ *If requests can be predicted (e.g., sequential scans) pages can be pre-fetched several pages at a time!*

10

More on Buffer Management

- Requestor of page must unpin it, and indicate whether page has been modified:
  - *dirty* bit is used for this.
- Page in pool may be requested many times:
  - a *pin count* is used. A page is a candidate for replacement iff *pin count* = 0.
- CC & recovery may entail additional I/O when a frame is chosen for replacement. (*Write-Ahead Log* protocol; more later.)

11

Buffer Replacement Policy

- Frame is chosen for replacement by a *replacement policy:*
  - Least-recently-used (LRU), Clock, MRU, etc.
- Policy can have big impact on # of I/O’s; depends on the *access pattern*.
- *Sequential flooding*: Nasty situation caused by LRU + repeated sequential scans.
  - # buffer frames < # pages in file means each page request causes an I/O. MRU much better in this situation (but not in all situations, of course).

12

DBMS vs. OS File System

OS does disk space & buffer mgmt: why not let OS manage these tasks?

- Differences in OS support: portability issues
- Some limitations, e.g., files can't span disks.
- Buffer management in DBMS requires ability to:
  - pin a page in buffer pool, force a page to disk (important for implementing CC & recovery),
  - adjust *replacement policy*, and pre-fetch pages based on access patterns in typical DB operations.

13

Files of Records

- Page or block is OK when doing I/O, but higher levels of DBMS operate on *records*, and *files of records*.
- FILE: A collection of pages, each containing a collection of records. Must support:
  - insert/delete/modify record
  - read a particular record (specified using *record id*)
  - scan all records (possibly with some conditions on the records to be retrieved)

14

Unordered (Heap) Files

- Simplest file structure: 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.

15

Alternative File Organizations

Many alternatives exist, *each ideal for some situation , and not so good in others:*

- Heap files: Suitable when typical access is a file scan retrieving all records.
- Sorted Files: Best if records must be retrieved in some order, or only a `range' of records is needed.
- Hashed Files: Good for equality selections.
  - File is a collection of *buckets*. Bucket = *primary* page plus zero or more *overflow* pages.
  - *Hashing function* **h**: **h**(*r*) = bucket in which record *r* belongs. **h** looks at only the search fields of *r*.
- Clustered Files: Tuples from > 1 relation stored [illegible]

16

Cost Model for Our Analysis

We ignore CPU costs, for simplicity:

- **B:** The number of data pages
- **R:** Number of records per page
- **D:** (Average) time to read or write disk page
- Measuring number of page I/O's ignores gains of pre-fetching blocks of pages; thus, even I/O cost is only approximated.
- Average-case analysis; based on several simplistic assumptions.

☛ *Good enough to show the overall trends!*

17

Assumptions in Our Analysis

- Single record insert and delete.
- Heap Files:
  - Equality selection on key; exactly one match.
  - Insert always at end of file.
- Sorted Files:
  - Files compacted after deletions.
  - Selections on sort field(s).
- Hashed Files:
  - No overflow buckets, 80% page occupancy.

18

Cost of Operations (I/O only)

	Heap File	Sorted File	Hashed File
Scan all recs	**BD**	**BD**	**1.25 BD**
Equality Search	**0.5 BD**	**D $\log_2 B$**	**D**
Range Search	**BD**	**D ($\log_2 B$ + # of pages with matches)**	**1.25 BD**
Insert	**2D**	**Search + BD**	**2D**
Delete	**Search + D**	**Search + BD**	**2D**

☛ *Several assumptions (see previous slide)!*

19

Disk and File Summary

- Disks provide cheap, non-volatile storage.
  - Random access, but cost depends on location of page on disk; 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(s).
  - Written to disk when frame chosen for replacement (which is after all requestors release the page).
  - Choice of frame to replace based on *replacement policy*.
  - Tries to *pre-fetch* several pages at a time.

20

Disk and File Summary (Contd.)

- DBMS vs. OS File Support
  - DBMS needs features not found in many OS's, e.g., forcing a page to disk, controlling the order of page writes to disk, files spanning disks, ability to control pre-fetching and page replacement policy
- File layer keeps track of pages in a file, and supports abstraction of a collection of records.
  - Pages with free space identified using linked list or directory structure (similar to how pages in file are kept track of).
- Many alternative file organizations exist, each appropriate in some situation.

21