Physical Layer

Davood Rafiei (Copyright 2001-2022)

Why physical layer?
- better understanding of the overhead and the cost
- organizing data outside DBMS
  - not every data management problem is best solved using a DBMS

Physical Layer involves
1. organizing files on disk
2. structuring data in files (file structures)?
3. creating and maintaining indexes

Computer memory hierarchy (from the fastest to the slowest)
- processor registers: very fast, very expensive, very small, volatile

Our focus is on Disk and RAM:
Typical setting: Data is stored on disk (or tape) and fetched to RAM when needed

Organizing data on a disk:
==========================
- disk components: surfaces, tracks, sectors, cylinders
- speeds: 5400rpm, 7200 rpm, 10k rpm, 15k rpm (not mainstream)

Q1. How much data can we store in a disk pack?

Example. A Barracuda disk from Seagate consists of 930,408 cyliners with 2 tracks per cylinder, 63 sectors per track and 512 bytes per sector. What is the disk capacity in bytes?

Data units (for reading/writing disks):

Sector:

Page:

Q2. How long does it take to read a block?

Access time = seek time + rotational delay + transfer time

Seek Time: the time needed to move the head to the right track

specified as : min seek time (eg. 1 msec)
max seek time (eg. 22 msec)
avg seek time (eg. 9 msec)

Rotational Delay (or latency) : the time needed for the beginning of the desired sector to rotate into position under the disk head.

min = 0
max = time for one disk rotation
avg =

Transfer Time: the time needed to read the data
= (# bytes transferred / # bytes on a track) * time for one disk rotation

Example . Consider a Barcuda disk from Seagate with 63 sectors per track, 512 bytes per sector and average seek time of 9 msec. The disk platters rotate at 7200 rpm (revolution per minute). How long does it take to read a block of 5 sectors?

-- Slides on reducing I/O costs --

Organizing records in a file:
==============================
Experiment: Create a file (say t.txt) with a few characters inside and check out its size on disk. If using lab machines, try it on a local disk say at /tmp since the behaviour is different on nfs files.
  du -hs t.txt

1. Heap files;
  - with or w/o gaps
  - operations: insert, delete, search
  - good performance for insert and full scan
  - bad performance for searches and deletes (need a search first)

2. Sorted files
- with or w/o gaps
- operations: insert, delete, search
- good for searches (binary search), deletes (tagging!)
- bad for inserts (difficult to keep the file sorted)
- full scan is the same as in a heap file

3. Indexed files

To provide a fast access to data
- sorting (dictionary, phone book)
- indexing (book index):

(1) hash indexes,
(2) tree-structured indexes

Sorting:

- given a list of 10 numbers, how can you sort them?
  (internal sort)
- given a list of 1 billion records, how can you sort them?
  (external sort)

Indexing
- efficiently locate rows without searching the entire table
- search key: id, name, title (can have duplicate values)
  keys with the same values are stored together.

- index entry:
  (1) <search key, full record>; e.g. entries in a phone book or a dictionary
  This choice results in an integrated index.
  (2) <search key, id or address>
    address: rid, page id, a key (e.g. emp id), location address
  This typically results in separate index and data files.

- Clustered vs. unclustered index
  (1) Clustered index
  Index entries and data records are ordered on the same columns.
  At most one clustered index. Good for range queries.
  (2) Unclustered index: Index entries and data records are not ordered on the same columns. Any number of unclustered indexes are possible.

- Dense vs Sparse
  (1) Dense index: index entry for each data record
  (1) Sparse index: index entry for each data page

- Index overhead
  (1) accessing the data
  (2) updating the data

Example. Consider a data file with 10,000 pages and a range query that returns 100 rows. Suppose there are 20 rows/page and 200 index entries/page. Assuming that an index access to retrieve an entry takes at most 2 page transfers, which one of the following file organizations does involve fewer page transfers?