COS 597A:
Principles of
Database and Information Systems

File organization and access costs

Move down a level of abstraction

- Until now at level of user view of data
  - models
  - query languages
- Now: how actually store data and access
  - disk storage (low-level abstraction)
  - file organization (level between disk and user)
  - access costs
- Next: how compute query results efficiently
  - what are algorithms
  - what are costs

Disks

- platters containing tracks
- track read sequentially
- can seek from track to track
- tracks broken into sectors
  - smallest physical unit can read / address
  - typical size 512 Bytes

Disk access costs

- seek time
  - milliseconds
- rotational latency
  - milliseconds
- transfer rate
  - 100 MB/sec
- compare RAM
  - nanoseconds
  - factor of 10^6

•disk closeness
  - adjacent sectors
  - same track
  - same cylinder
  - adjacent cylinder

Why use disk?

- too much data for main memory
- need permanent storage

So far as technology advances, disk still gives significantly more space and less speed, regardless of how big/cheap RAM gets
  - voracious appetite for space!

File

- collection of records
- records grouped into blocks
  - block smallest unit read
  - "block" also known as "page"
  - typical 4-8 KB
- block is multiple of disk sectors
  - stored sequentially on disk
- want blocks of file physically close on disk
- read block into memory buffer
  - size of buffer in blocks
  - buffer as big as can afford

File storage management

- Who manages storage of files on disk
  1. custom OS for DBMS
  2. let OS do it
     - typically one file per relation
  3. define one OS file for whole DBMS
     - DBMS manages within file
- DBMS buffer manager
  - replacement strategy
  - pinning
  - forced-out blocks

Conceptual organization of file

- Heap file
  - no order records in blocks
  - linked list blocks or directory of blocks
- Hashing file
  - hash function applied to record gives address of first block of bucket
  - can be overflow
    - pointers to overflow blocks
    - where overflow blocks on disk?
  - try to keep blocks 80% full

Conceptual organization of file: cont.

- Sequential file
  - conceptually ordered set of records
  - order often sort on attributes of relation
  - stored in order
    - sequentially close => physically close
    - compact after delete
  - binary search?
    - need i^{th} block in sorted order in one disk I/O
- can have sorted file that is not sequential file

Access cost model

- B number of data blocks in file
- R number of records per block in full block
- D average time to R/W disk block
  - assume individual blocks not sequential on disk
    - no multi-block reads paying one seek cost
- Ignore CPU time

Simple average case time analysis

- Assumptions
  - Insert at end of heap
  - No overflow buckets for hash
    - Keep 80% occupancy
    - Inserts/deletes in balance
  - Sorted sequential file with binary search
  - Delete assumes have address of record
    - add search time if deleting by record value
- Use analysis for relative costs
  - TOO CRUDE for "on the fly" cost estimates

Avg. time	Heap	Sorted	Hashed
Scan	BD	BD	1.25 BD
Search = (unique)	.5BD	$D\log_2 B$	D
Search = (multiple)	BD	**$D(\log_2 B$ + # extra matching blocks)**	**D (1 + # extra matching blocks)**
Search range	BD	"	1.25 BD
Insert	2D	**Search + D + BD**	2D
Delete	2D	2D+BD	2D

Remarks

- to insert or delete from a block, must read and write block
  - gives rise to 2D factors for heap and hash
  - gives rise to search + D factor for sorted
- sequential storage of sorted file is maintained
  - gives rise to BD factor for insert/delete:

    on average move 1/2 records, touching1/2B blocks, with a read & write for each

    = 1/2B*2D