ORGANIZZAZIONE FISICA (ripasso generale)

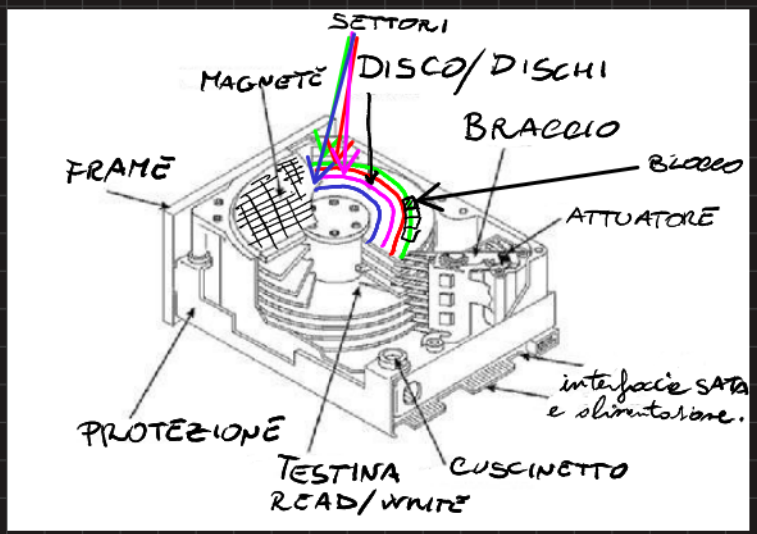
Dove vengono immagazzinati i dati?

- Può sembrare anacronistico ma ancora oggi anche le più importanti e le più grandi basi di dati utilizzano i dischi meccanici per immagazzinare i dati. Queste "vecchie" tecnologie si preferisce perché è molto più economica delle controparte a stato solido (S.S.D) e perché grazie a delle particolari strutture dati risulta essere "lenta" come tecnologia.

HDD



(Comunissimi)



- Le parti principali di un H.D.D. sono il braccio che ha le testine alle sue estremità queste testine sono in grado di leggere e scrivere sul disco.
- Il disco/i dischi sono divisi in settori, su ogni settore ci sono diversi blocchi, le testine legge e scrive un blocco per volta, le

selezione del blocco è attuata grazie ad un motore BRUSH-LESS che mette in rotazione i dischi.
La struttura degli HDD comporta 3 Delay quando si ricerca un dato.

1) ROT Delay : tempo che serve al disco per posizionare sotto alla testina il blocco cercato
2) SEEK Time : tempo che serve all'attuatore per selezionare il settore corretto
3) TRASFER Time : tempo di trasferimento dei dati (SATA 3 TRASFR TIME = MAX 6Gbit/sec)

Questi 3 Delay sommati sono detti service time.

Strutture dati per ottimizzare la situazione

- Abbiamo capito che l'Hardware a nostra disposizione non è il migliore, per ottimizzare i tempi di risposta dobbiamo ottimizzare via software con delle strutture dati.

HEAP E FILE SEQUENZIALI

HEAP

In queste strutture dati i file sono scritti sul disco in "ordine di arrivo". La ricerca di un Record risulta essere molto lenta ($O(n)$) ma per inserire un nuovo record non necessita la modifica dell'ordine di quelli già presenti in precedenza.

- PER TROVARE UN RECORD ESEGUO $O(n)$ Accessi sequenziali dove n è il numero di blocchi che occupano i record.

FILE SEQUENZIALI

In una struttura "File sequenziali" i Record sono ordinati in base alla chiave, posso quindi eseguire una ricerca binaria sui record che mi costerà $O\log_2(N)$ Accessi Random.

- L'ultimo punto a sfavore di queste strutture è che se devo aggiungere un nuovo record, dovrò cambiare l'ordine di tutti i record già presenti nelle base di dati.

● HASH

Si serve di un algoritmo di Hashing sulle chiavi per mappare i record in agglomerati di blocchi detti Bucket. Fornite le chiave l'algoritmo di Hashing mi restituirà ad un bucket che è un insieme di blocchi, effettuando degli accessi sequenziali ai blocchi troverà il record cercato. I Bucket hanno dimensione fissata, quindi se un record non entra nel bucket sarà mappato al primo blocco disponibile.

RIEPILOGANDO.

#accessi medi se record in bucket = 1 RBA + $\left(\frac{nblk_pBk}{2}\right)$ SBA

#accessi se record non in bucket = 1 RBA + $(nblk_pBk)$ SBA + 1 SBA.

- Ci deve essere un certo trade-off sulla grandezza dei bucket.

● ISAM

Queste strutture hanno un file detto INDEX e un file principale. Il file principale viene

diviso in sezioni grandi quanto un blocco del disco, nel file indice ci sono liste con chiave e puntatore al primo record di ogni sezione e queste liste sono ordinate per chiave. Per cercare un record si può effettuare una ricerca binaria sul file indice che costa $O\log_2(n)$ dove n è il numero di blocchi occupati dal file principale.

$Am(\text{RICERCA BINARIA SU INDEX}) = \log_2(NBF_{index}) + 1\ RBA$

$Am(\text{RICERCA BINARI SU FILE P}) = (\log_2(NBF_{principale}))\ RBA.$

B - tree

Partendo dal modello degli alberi m-nari un B-tree è un albero m-ario bilanciato.
Ogni nodo dell'albero è grande un blocco, contiene m puntatori e m-1 Kiavi al massimo

• BILANCIAMENTO

1) ogni nodo (NO FOGLIE/NO RADICI) hanno almeno $\lceil \frac{m}{2} \rceil$ figli e massimo m figli.
2) nodo radice ha almeno due figli
3) nodi pieni almeno al 50%
4) foglie tutte allo stesso livello.

Un nodo dell'albero è del tipo:

| P_1 | K_1 | P_2 | K_2 | ... |

Dove $K_i < K_{i+1}$ e il nodo a cui punta P_i avrà tutte chiavi minori di K_i.
La ricerca di un record avviene in maniera ricorsiva e quando si trova la chiave c'è avrà un puntatore al record creato nel file principale.

- Il numero di accessi medi per ricercare un record usando questa struttura dati è $O(h)$ dove h è l'altezza dell'albero
  h_{MAX} se blocchi pieni al $\lceil 50\% \rceil$
  h_{MIN} se blocchi pieni al 100%

B+ - tree

- Variante dei B-TREE.

PROPRIETÀ

1) ogni nodo contiene chiavi contenute nelle foglie
2) le foglie hanno un puntatore alla foglia adiacente
3) ogni nodo contiene sottoinsiemi di chiavi contenuti nelle foglie.

- LA RICERCA È PIÙ EFFICIENTE PERCHÉ SONO GENERALMENTE PIÙ BASSI DEI B-TREE