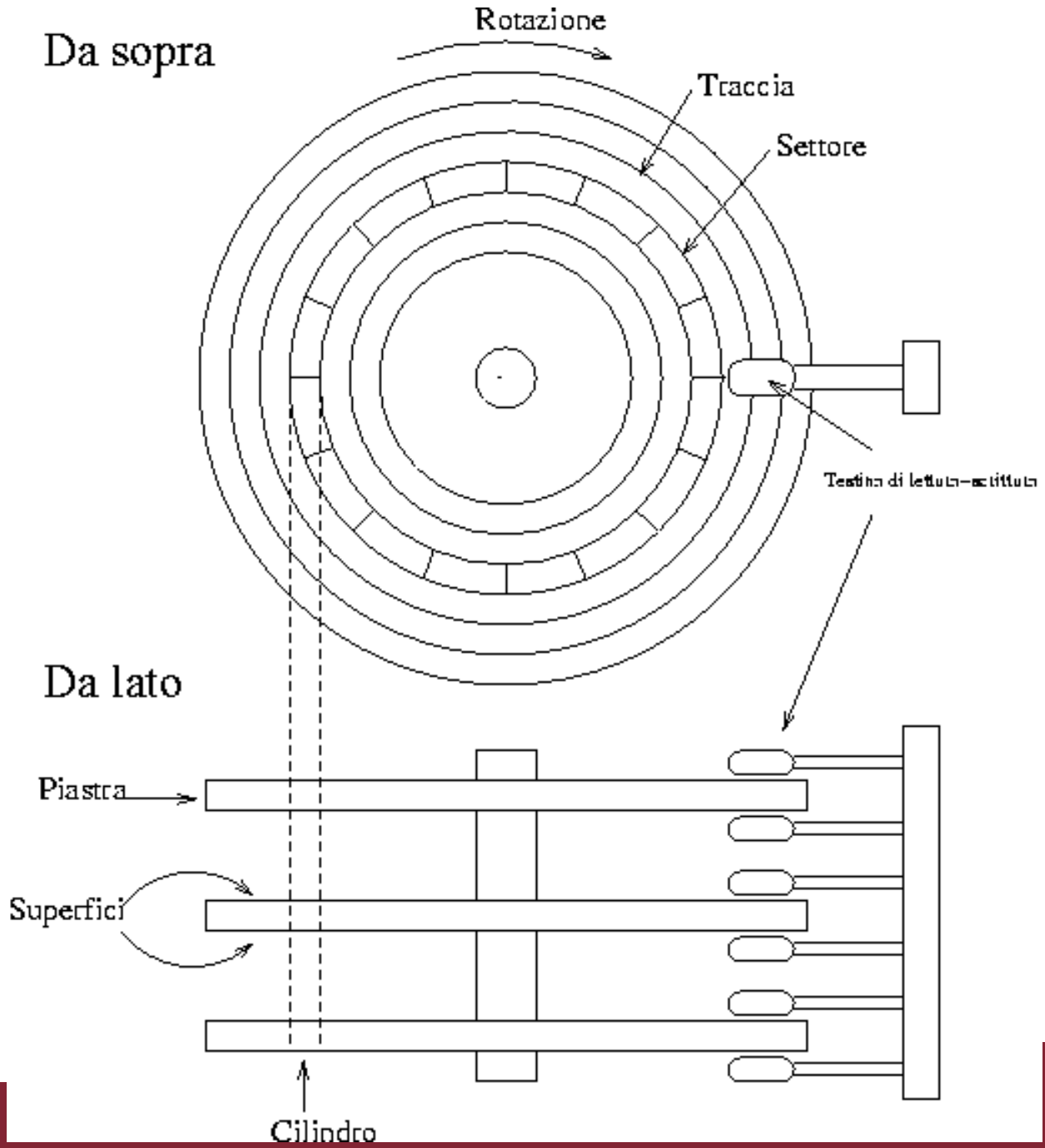
Lezione 21– Organizzazione fisica dei dati - concetti generali e primo esempio : file heap

Prof.ssa Maria De Marsico
demarsico@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Parte del materiale è tratto dal materiale preparato dalla prof.ssa Moscarini per il canale A-L


Da sopra
Rotazione
Traccia
Settore
Testina di lettura–scrittura
Da lato
Piastra
Superfici
Cilindro

Memorie a stato solido

- Nelle memorie a stato solido l'organizzazione dei dati è molto simile (blocchi, settori … , ma di taglia maggiore) ma non ci sono organi in movimento, e i tempi di accesso e archiviazione ridotti.

- Si lavora nell'ordine dei decimi di millisecondo, mentre il tempo di accesso dei dischi magnetici è oltre 50 volte maggiore, attestandosi invece tra i 5 e i 10 millisecondi.

- In ogni caso, il tempo di trasferimento dati, soprattutto in scrittura, è più lento del tempo di elaborazione della CPU,

Blocchi (o pagine) e accessi a memoria secondaria

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Al momento della formattazione del disco, ogni traccia e suddivisa in **blocchi** (o **pagine**) di dimensione fissa (compresa tra 2^9 e 2^{12} byte).

- Per **accesso** si intende il trasferimento di un **blocco** da memoria secondaria a memoria principale (**lettura** di un blocco) o da memoria principale a memoria secondaria (**scrittura** di un blocco).

Accessi a memoria secondaria

- Il tempo necessario per un accesso è dato dalla somma di:

➢ tempo di posizionamento (della testina sulla traccia in cui si trova il blocco)

➢ ritardo di rotazione (rotazione necessaria per posizionare la testina all'inizio del blocco)

➢ tempo di trasferimento (dei dati contenuti nel blocco)

Accessi a memoria secondaria

- Il tempo richiesto per un accesso a memoria secondaria è dell'ordine dei millisecondi, quindi notevolmente superiore a quello di **elaborazione** dei dati in memoria principale.

- Per questo motivo il costo di un'operazione sulla base di dati è definito in termini di numero di accessi.

Memorizzazione di relazioni

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Ad ogni relazione corrisponde un file di record che hanno tutti lo stesso tipo (numero e tipo dei campi).
- Ad ogni attributo corrisponde un campo.
- Ad ogni tupla corrisponde un record.
- Nel modello relazionale in ogni file ci sono record appartenenti ad **un'unica** relazione (... ad una tabella!)

Record

In un record oltre ai campi che corrispondono agli attributi ci possono essere altri campi che contengono **informazioni sul record** stesso o **puntatori** ad altri record/blocchi.

Informazioni sul record

- All'inizio di un record alcuni byte possono essere utilizzati per:

- specificare il tipo del record (è necessario quando in uno stesso **blocco** sono memorizzati record di tipo diverso) (cioè record appartenenti a più file)
- specificare la lunghezza del record (se il record ha campi a lunghezza variabile)
- contenere un bit di "cancellazione" o un bit di "usato/non usato"

Informazioni sul record (per accedere ad un campo)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

offset: numero di byte del record che precedono il campo

Se tutti i campi del record hanno lunghezza fissa, l'inizio di ciascun campo sarà sempre ad un numero fisso di byte dall'inizio del record

Informazioni sul record (per accedere ad un campo)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Se il record contiene campi a **lunghezza variabile**:

- all'inizio di **ogni campo** c'è un **contatore** che specifica la lunghezza **del campo** in numero di byte

•oppure

- all'inizio **del record** ci sono **gli offset dei campi a lunghezza variabile** (tutti i campi a lunghezza fissa **precedono** quelli a lunghezza variabile).
  - ***offset***: numero di byte del record che precedono il campo
  - Se tutti i campi del record hanno lunghezza fissa, l'inizio di ciascun campo sarà sempre **ad un numero fisso di byte** dall'inizio del record

- **Nota**: Nel primo caso per individuare la posizione di un campo **bisogna esaminare i campi precedenti** per vedere quanto sono lunghi; quindi la prima strategia **è meno efficiente** della seconda

Puntatori

- Un puntatore ad un record/blocco è un dato che permette di accedere rapidamente al record/blocco:
- l'indirizzo dell'inizio (primo byte) del record/blocco sul disco

•oppure

- nel caso di un record, una coppia (b,k) dove
  - b è l'indirizzo del **blocco** che contiene il record
  - k è il valore della chiave

- **Nota**: Nel secondo caso e' possibile spostare il record **all'interno del blocco**, nel primo no altrimenti potremmo avere dei **dangling pointer**

Blocchi

In un blocco ci possono essere:

informazioni sul blocco stesso e/o **puntatori** ad altri blocchi.

Informazioni sul blocco

Se un blocco contiene **solo** record di lunghezza **fissa**:

- il blocco è suddiviso in **aree** (sottoblocchi) di lunghezza **fissa** ciascuna delle quali **può contenere** un record
- i bit "**usato/non usato**" sono raccolti in uno o più byte all'inizio del blocco.

- **Nota**: Se bisogna inserire un record nel blocco occorre cercare un'area non usata; **se** il bit "usato/non usato" è **in ciascun record** ciò può richiedere la scansione **di tutto il blocco**; per evitare ciò si possono raccogliere tutti i bit "usato/non usato" in uno o più byte all'inizio del blocco.

Informazioni sul blocco

Se un blocco contiene record **di lunghezza variabile:**

- si pone **in ogni record** un campo che ne specifica la lunghezza in termini di numero di byte

oppure

- si pone **all'inizio del blocco una directory contenente i puntatori (offset) ai record nel blocco**

Informazioni sul blocco (directory)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

La directory può essere realizzata in uno dei modi seguenti:

- è preceduta da un campo che specifica **quanti sono i puntatori** nella directory
- è una **lista** di puntatori (la fine della lista è specificata da uno **0**)
- ha dimensione **fissa** e contiene il valore **0** negli spazi che **non contengono puntatori**

Operazioni sulla base di dati

Un'**operazione** sulla base di dati consiste di:

- **ricerca**
- **Inserimento (implica ricerca se vogliamo evitare duplicati)**
- **cancellazione (implica ricerca)**
- **modifica (implica ricerca)**

di un **record**

- **Notare che la ricerca è alla base di tutte le altre operazioni**

Operazioni sulla base di dati

- Un requisito **fondamentale** di un DBMS è **l'efficienza**, cioè la capacità di rispondere alle richieste **dell'utente il più rapidamente possibile**

- Una particolare **organizzazione fisica** dei dati (cioè una particolare **organizzazione dei record nei file**) può rendere efficiente la elaborazione di particolari richieste …
- … quindi **l'amministratore** della base di dati durante il **progetto fisico** della base di dati deve tener presente **quali operazioni saranno effettuate più frequentemente**.
- Generalmente **ad ogni oggetto base di un modello logico** (**schemi di relazione nel modello relazionale**, segmenti nel modello gerarchico, tipi di record nel modello a rete) corrisponde **un file di record che hanno tutti lo stesso formato**, cioè **gli stessi campi** (che corrispondono agli attributi nel caso del modello relazionale e ai campi di segmenti e di tipi di record negli altri due modelli).
- **Attenzione**: stesso formato **non significa** stessa lunghezza, ma stesso **numero e tipo** di campi!

Diverse organizzazioni dei file

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- In questa parte del corso esamineremo diversi tipi di organizzazione fisica di file che consentono la ricerca di record in base al valore di uno o più **campi chiave**.

- **ATTENZIONE!** Il termine “chiave” non va inteso nel senso in cui viene usato nella teoria relazionale, in quanto un valore della chiave **non necessariamente** identifica univocamente un record nel file.

File heap

- Cominciamo da una **NON** organizzazione, cioè una collocazione dei record nei file in un ordine determinato solo dall'ordine di inserimento

- **Attenzione**: non parliamo dell'heap – albero di ricerca del corso di algoritmi!

- Il fatto di non adottare **nessun particolare accorgimento** nell'inserimento dei record che possa poi **facilitare** la ricerca, ci fornisce le **prestazioni peggiori** in termini di numero di accessi in memoria richiesti dalle operazioni di **ricerca** mentre l'inserimento è molto veloce se ammettiamo **duplicati**

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	...
blocco 2	...	...	...
	...	...	...
	...	...	...
	...	...	...
...	...	...	...
blocco n	...	...	...

In un file heap un record viene inserito sempre come **ultimo record del file.**
Pertanto tutti i blocchi, **tranne l'ultimo,** sono **pieni.**
L'accesso al file avviene attraverso la directory (**puntatori ai blocchi**)

Ricerca

Se si vuole ricercare un record occorre scandire **tutto** il file, iniziando dal **primo** record fino ad **incontrare** il record desiderato.

Ricerca

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	...
...	...	...	...
blocco i	...	...	...
	...	...	...
	...	...	...
	...	...	...
...	...	...	...
blocco n	...	...	...

Il costo della ricerca varia in base a dove si trova il record: se il record che si cerca si trova nell'**i-esimo** blocco occorre effettuare **i accessi in lettura**

Pertanto ha senso valutare il **costo medio** di ricerca

N = 151 record
Ogni record 30 byte
Ogni blocco contiene 65 byte
Ogni blocco ha un puntatore al prossimo blocco (4 byte)

Record interi X blocco -> (65-4)/30 = parte intera inferiore di 2,03 = 2 (non ho abbastanza spazio per gli ultimi 0,03 cioè 3/100 di record, e non posso memorizzarli in un nuovo blocco)

Numero blocchi che occorrono per memorizzare N record n -> N/R = 151/2 =parte intera superiore di 75,5 = 76 (devo allocare anche 0,5 cioè mezzo blocco perchè ci va 1 record)

In una ricerca, devo scorrere la lista di 76 blocchi: se sono fortunato trovo il record nel primo blocco, ma potrebbe anche trovarsi nell'ultimo, o in uno qualsiasi intermedio tra i due.

Costo medio della ricerca

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Cominciamo dalla ricerca quando la chiave ha un valore che non ammette duplicati

N: numero di record
R: numero di record che possono essere memorizzati in un blocco
n=N/R

Per ottenere il costo medio occorre sommare i costi per accedere ai singoli record e quindi dividere tale somma per il numero dei record. Per ognuno degli R record nell'**i-esimo** blocco sono necessari **i accessi**.

$$\frac{R\times 1+R\times 2+\ldots+R\times i+\ldots+R\times n}{N}=\frac{R\times(1+2+\ldots+i+\ldots+n)}{N}=$$

$$=\frac{R}{N}\times\frac{n(n+1)}{2}=\frac{1}{n}\frac{n(n+1)}{2}\approx\frac{n}{2}$$

Costo medio della ricerca

Se vogliamo cercare **tutti** i record con una certa chiave (che non è chiave in senso relazionale, cioè ammettiamo duplicati) **dovremo comunque accedere a n blocchi**, perché non possiamo dire quando abbiamo trovato **l'ultima** occorrenza di record con la chiave cercata.

Inserimento

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	...
blocco 2	...	...	...
	...	...	...
	...	...	...
	...	...	...
...	...	...	...
blocco n	...	...	...

1 accesso in lettura
(per portare l'ultimo blocco in memoria principale)

Inserimento

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	...
blocco 2	...	...	...
	...	...	...
	...	...	...
	...	...	...
...	...	...	...
blocco n	...	...	...
	022	Belli	...

1 accesso in lettura
(per portare l'ultimo blocco in memoria principale)

+

1 accesso in scrittura
(per riscrivere l'ultimo blocco in memoria secondaria, dopo aver

Inserimento

… ma se non ammettiamo duplicati l'inserimento deve essere preceduto dalla ricerca, quindi dobbiamo aggiungere una **media di n/2 accessi per verificare che non esista già un record con la chiave data**

Modifica

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	..
blocco 2	...	...	...
	003	Neri	...
	...	...	...
	...	...	...
...	...	...	...
blocco n	...	...	...

Costo ricerca

Modifica

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	..
blocco 2	...	...	...
	003	Neroni	...
	...	...	...
	...	...	...
...	...	...	...
blocco n	...	...	...

Costo ricerca

+

1 accesso in scrittura
(per riscrivere in memoria secondaria il blocco , dopo aver modificato il record)

per ogni occorrenza della chiave, se ammettiamo duplicati

Cancellazione

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	...
blocco 2	048	Bellini	...
	003	Neroni	...
	009	Carini	...
	...	...	...
...	...	...	...
blocco n	...	...	...
	123	Stella	...

Costo ricerca

+

1 accesso in lettura
(per leggere l'ultimo blocco)

se vogliamo riutilizzare spazio ed evitare «buchi».. più complicato se i record sono a lunghezza variabile … i quel caso non possiamo direttamente «riempire» lo spazio vuoto, ma dobbiamo spostare verso l'alto tutti i record successivi, modificando eventuali puntatori)

Cancellazione

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	matr	cognome	...
blocco 1	010	Rossi	...
	005	Verdi	...
	...	...	...
	031	Bianchi	...
blocco 2	048	Bellini	...
	123	Stella	...
	009	Carini	...
	...	...	...
...	...	...	...
blocco n	...	...	...

Costo ricerca

+

1 accesso in lettura
(per leggere l'ultimo blocco)

+

2 accessi in scrittura
(per riscrivere in memoria secondaria il blocco modificato e l'ultimo blocco)