File con indice - caratteristiche ed esercizi

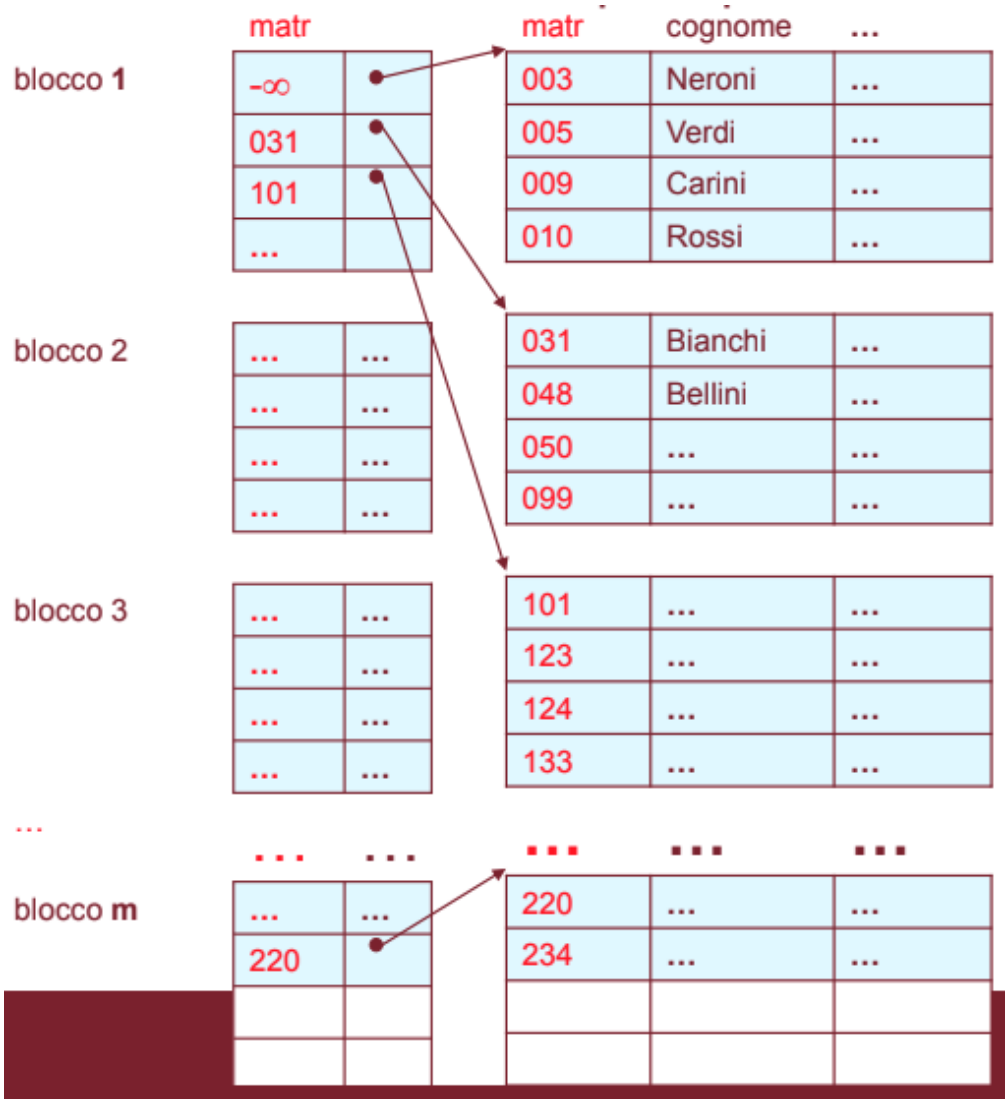
Introduction

Quando le chiavi ammettono un ordinamento significativo per l'applicazione, è più conveniente utilizzare un'organizzazione fisica dei fati che ne tanga conto (nel caso di campi multipli, si ordina sul primo campo, poi sul secondo e così via)

File ISAM

Il file **ISAM** (Indexed Sequential Access Method) è il primo esempio significativo della richiesta fatta.

Il file principale viene ordinato in base al valore della chiave di ricerca e generalmente viene lasciata una carta percentuale di spazio libero in ogni blocco (necessario per l'inserimento)



Nel file ISAM si ha un **file indice** (table of contents) e un **file principale**.
All'interno del file indice è presente una entrata per ogni blocco del file principale composta da chiave e puntatore all'inizio del blocco. La chiave in questo caso corrisponde alla chiave del primo record di ogni blocco del file principale (fatta eccezione per la prima entrata che è $-\infty$).
Ogni record del file indice inoltre è $\leq$ delle chiavi del blocco puntato ma strettamente maggiore delle chiavi del blocco puntato dal record precedente

---

Ricerca

Per ricercare un record con valore della chiave k occorre ricercare sul file indice un valore k' della chiave che **ricopre** k, cioè tale che:

- $k' \leq k$
- se il record con chiave k' non è l'ultimo record del file indice e k'' è il valore della chiave nel record successivo $k < k''$

La ricerca di un record con chiave k richiede una **ricerca sul file indice** + **1 accesso** in lettura sul file principale (nel conto del costo dobbiamo considerare solamente il numero di accessi in memoria, non importa la ricerca su un file che è stato già portato in memoria principale)

Poiché il file indice è ordinato in base al valore della chiave, la ricerca di un valore che ricopre la chiave può essere fatta in modo efficiente mediante la **ricerca binaria**

> se $k > k_1$ allora verifico tutte le chiavi successive all'interno del blocco e controllo se trovo k_n tale che $k_n < k < k_{n+1}$ in tal caso la ricerca dovrà proseguire nel blocco puntato da k_n (questa ricerca non aggiunge costo, sto leggendo un blocco già caricato); in caso contrario (k maggiore di tutte le chiavi del blocco) prosegue la ricerca binaria sui blocchi da

Ricerca binaria

Si fa un accesso in lettura al blocco del file indice $\frac{m}{2} + 1$ e si confronta k con k_1 (prima chiave del blocco):

- se $k = k_1$ abbiamo finito
- se $k < k_1$ allora si ripete il procedimento sui blocchi da 1 a $\frac{m}{2}$
- se $k > k_1$ allora la ricerca prosegue sui blocchi da $\frac{m}{2} + 1$ ad m (il blocco $\frac{m}{2} + 1$ va riconsiderato perché abbiamo controllato solo la prima chiave)

Ci si ferma quando lo spazio di ricerca è ridotto ad un solo blocco, quindi dopo $\lceil \mathbf{\log_2 m} \rceil$ accessi

Ricerca per interpolazione

La **ricerca per interpolazione** è basata sulla conoscenza della **distribuzione** dei valori della chiave, ovvero deve essere disponibile una funzione f che dati tre valori k_1, k_2, k_3 della chiave fornisce un valore che è la **frazione dell'intervallo di valori** dalla chiave compresa tra k_2 e k_3 in cui deve trovarsi k_1 cioè la chiave che stiamo cercando (nella ricerca binaria questa frazione è sempre $\frac{1}{2}$). Ad esempio quando cerchiamo in un dizionario non partiamo sempre da metà

k_1 deve essere confrontato con il valore k della chiave del primo record del blocco i (del file indice), dove $i = f(k_1, k_2, k_3) \cdot m$; analogamente a quanto accade nella ricerca binaria. Se k_1 è minore di tale valore allora il procedimento deve essere ripetuto sui blocchi $1, 2, \ldots, i-1$, mentre se è maggiore il procedimento deve essere ripetuto sui blocchi $i, i+1, \ldots, m$, finché la ricerca si restringe ad un unico blocco

La ricerca per interpolazione richiede circa $\mathbf{1 + \log_2 \log_2 m}$ accessi che è molto più veloce ma è altrettanto difficile conoscere f e inoltre la distribuzione dei dati potrebbe cambiare nel tempo

---

Inserimento

Per inserire un record dobbiamo prima trovare il blocco corretto in cui metterlo e poi scriverlo; quindi il costo è **costo per la ricerca** + **1 accesso** per scrivere il blocco. Ciò però è possibile solo se nel blocco c'è spazio per inserire il nuovo record. Se così non è si controlla se è disponibile nel blocco successivo o nel precedente (ciò comporta riscalare tutte gli altri record in modo tale da mantenere l'ordine)

Nel caso in cui non fosse disponibile spazio né nel blocco precedente né nel blocco successivo occorre **richiedere un nuovo blocco al file system**, ripartire i record tra vecchio e nuovo blocco e riscrivere tutti i blocchi modificati. Per questo motivo viene lasciato dello spazio libero all'interno dei blocchi, per fare in modo che questo problema arrivi il più tardi possibile

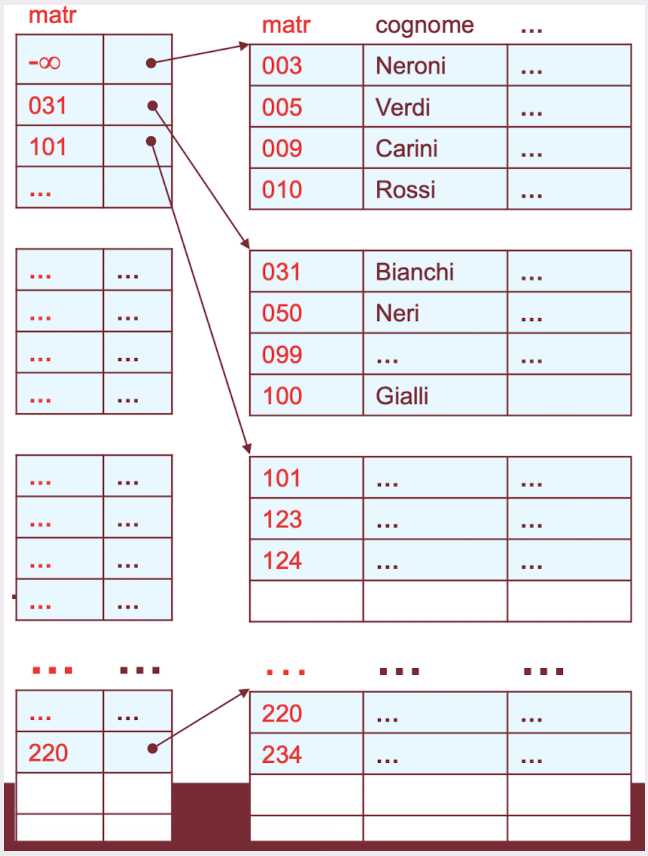
Esempi

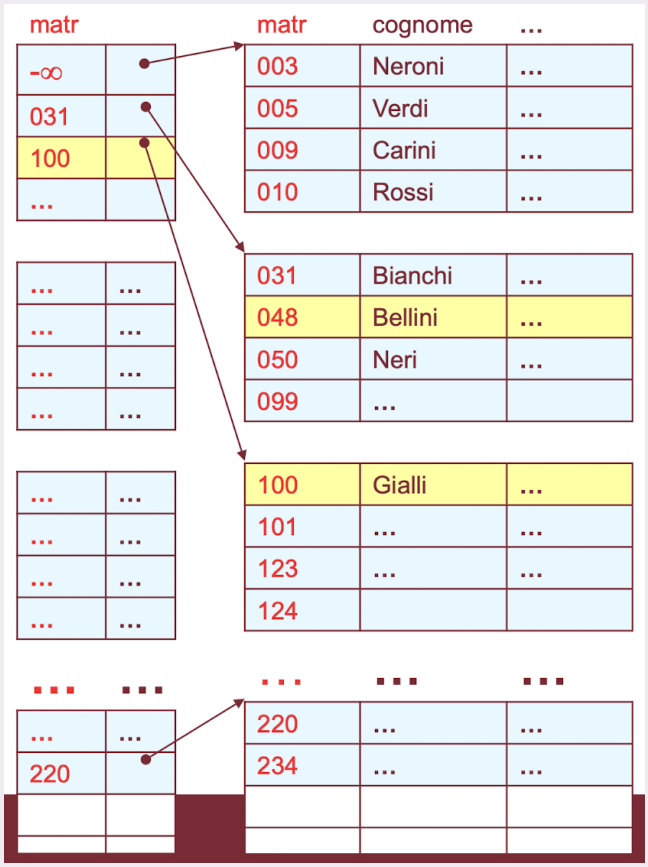
Spazio disponibile nel blocco precedente >

Devi inserire il blocco

048	Bellini	…

All’interno di questo file ISAM



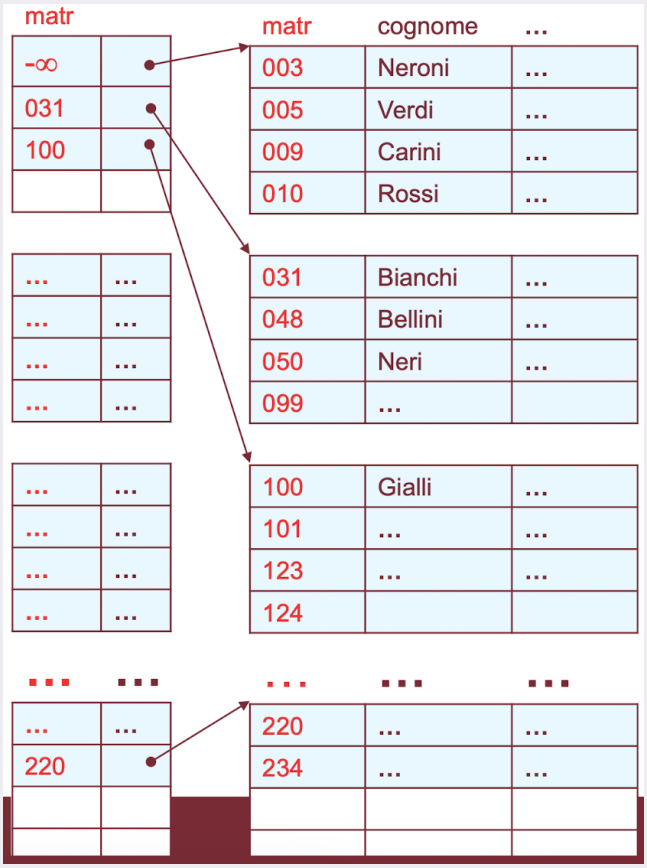


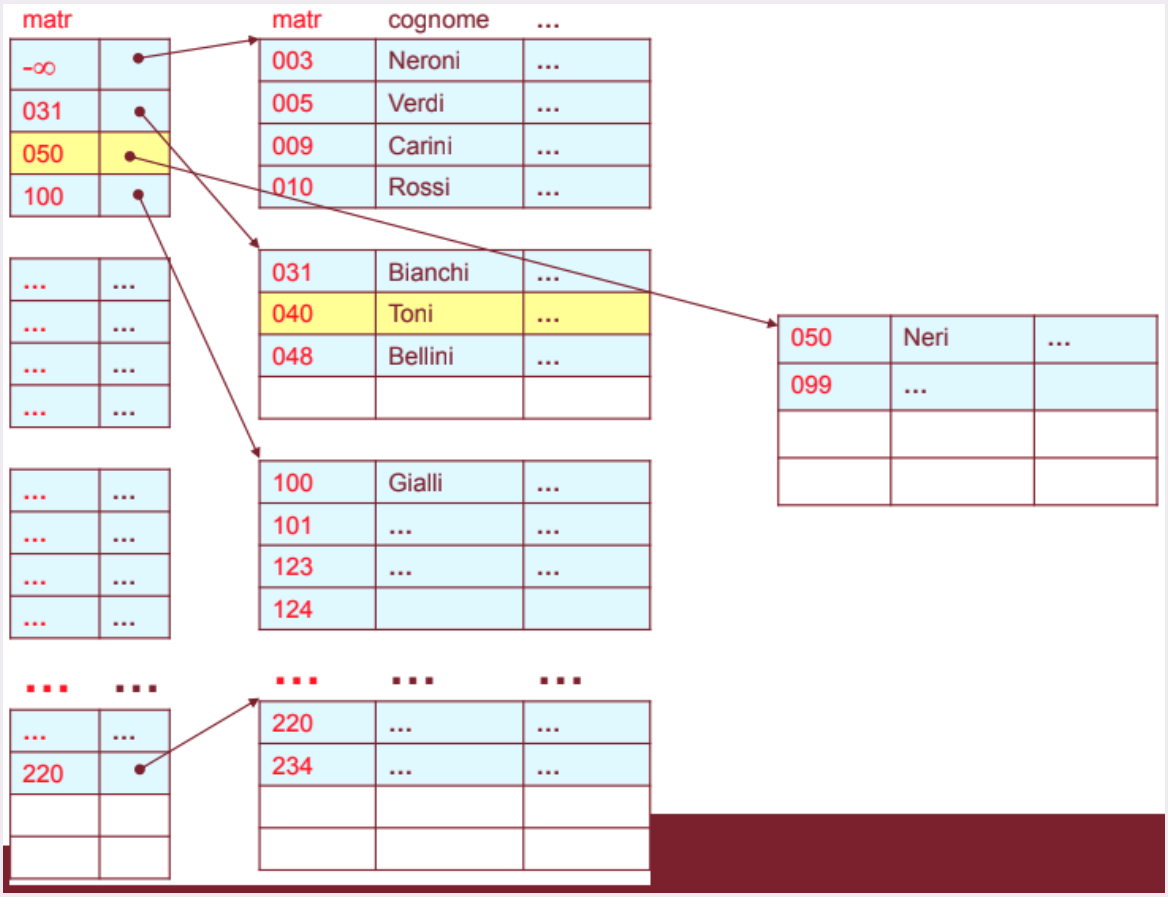
Spazio non disponibile nei blocchi adiacenti ›

Devi inserire il blocco

040	Toni	…

All’interno di questo file ISAM





Cancellazione

Per cancellare un record dobbiamo prima trovarlo all'interno del file principale e poi cancellarlo; quindi il costo è **costo per la ricerca** + **1 accesso** per scrivere il blocco. Se il record cancellato è il primo di un blocco sono necessari ulteriori accessi. Se il record cancellato è l'unico record del blocco, il blocco viene restituito al sistema e viene modificato il file indice

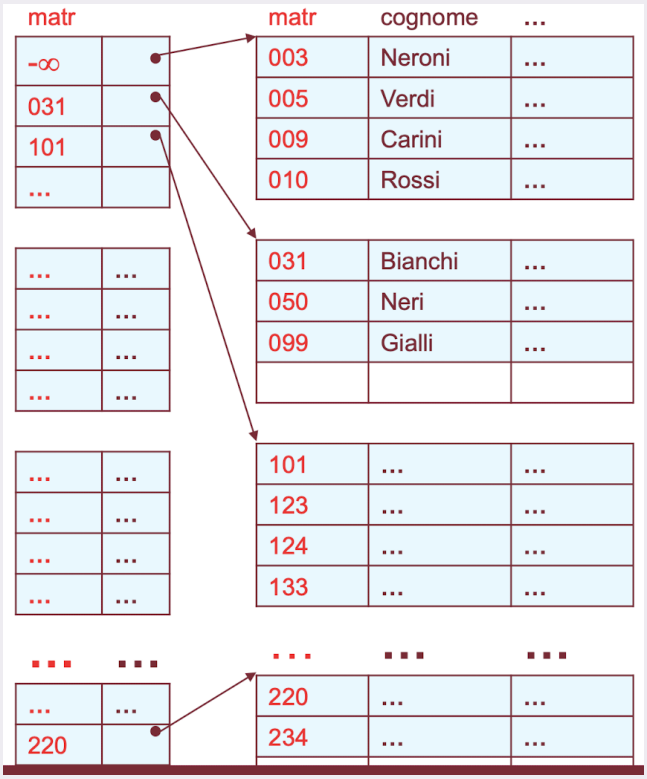
Esempi

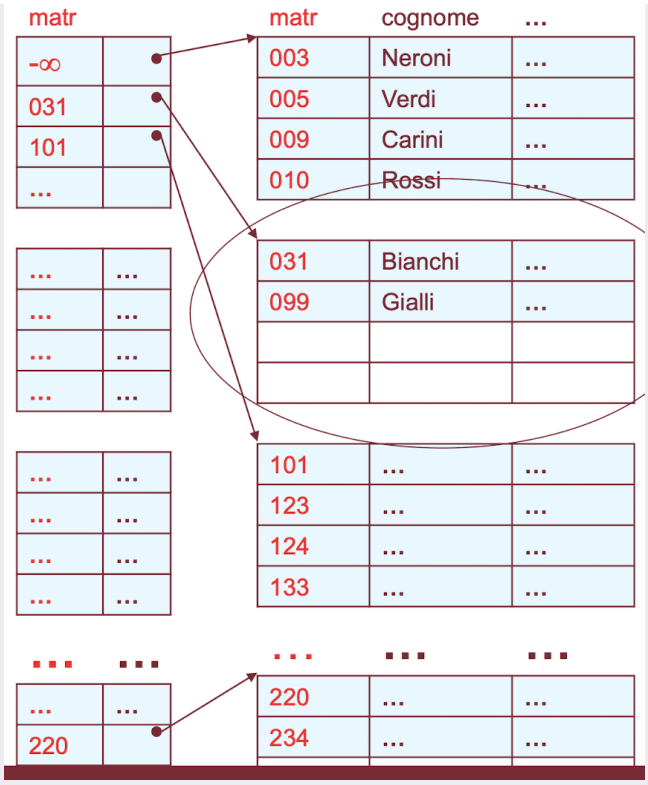
Record si trova al centro del blocco >

Immaginiamo di dover elminare

050	Neri	...

Dal seguente file ISAM



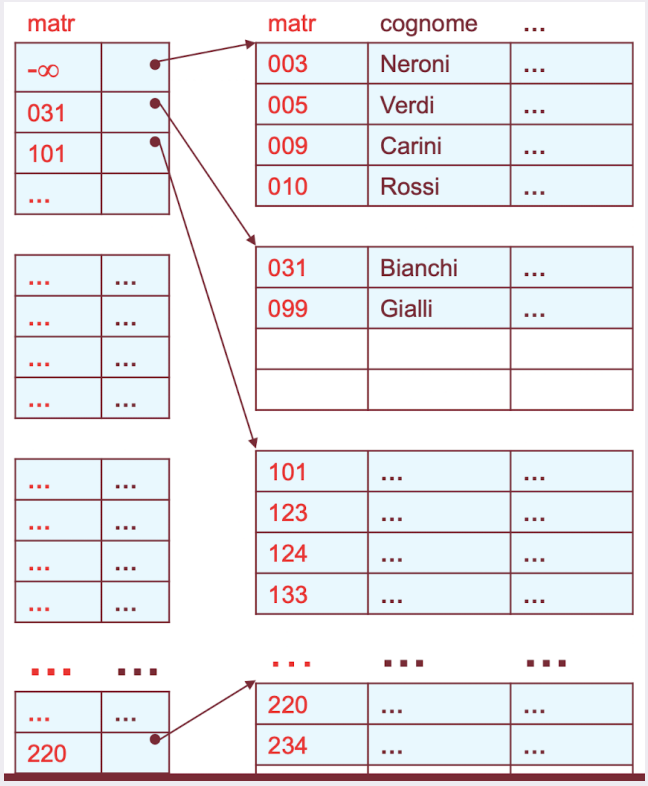


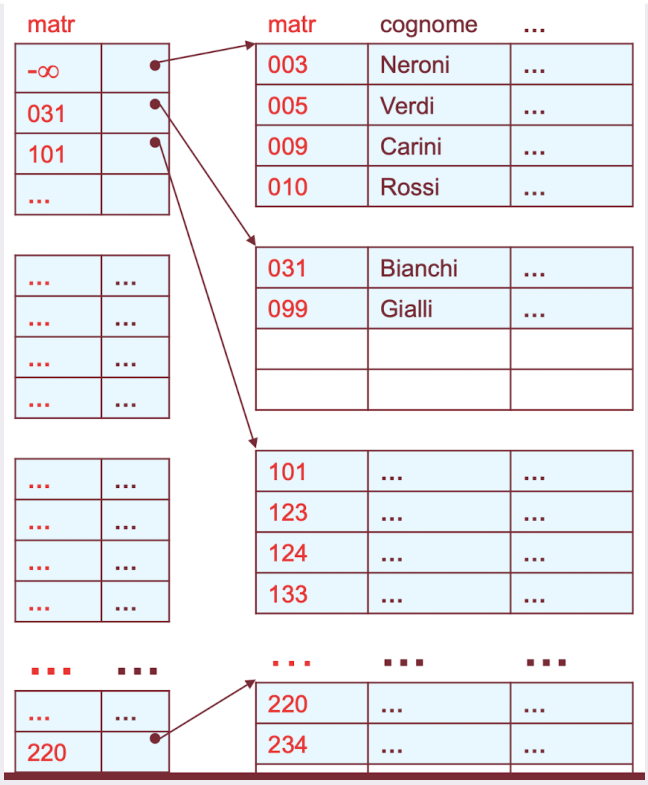
Il record è il primo del blocco

Il record è il primo del blocco
Immaginiamo di dover eliminare

031	Bianchi	...

Dal seguente file ISAM:



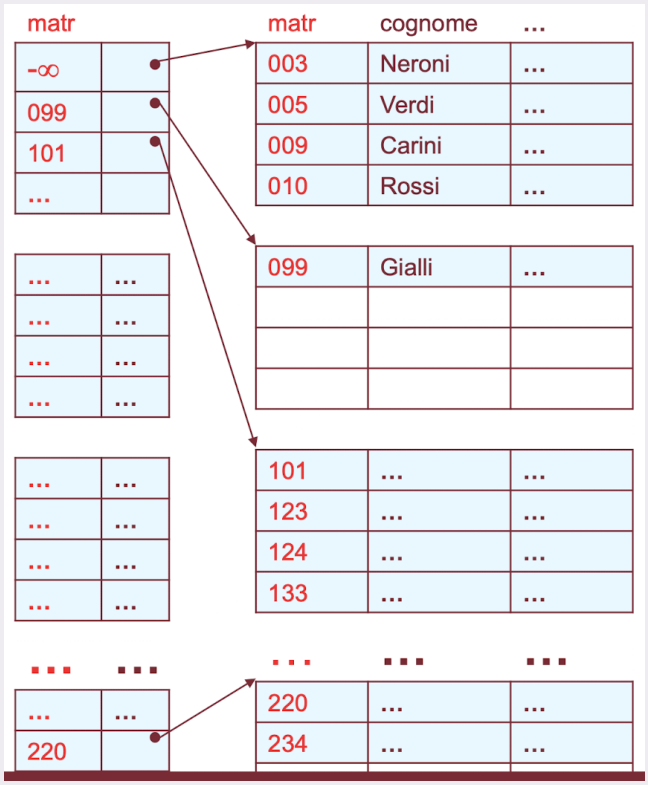


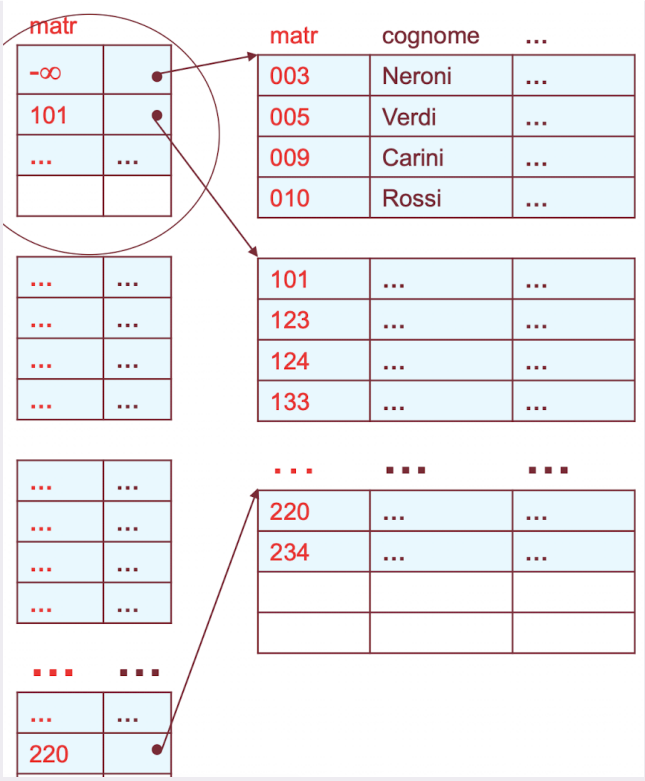
Il record è l'ultimo rimanente del blocco

Immaginiamo di dover eliminare

099	Gialli	...

Dal seguente file ISAM:





Modifica

Per modificare un record dobbiamo prima trovarlo tramite la chiave all'interno del file principale e poi modificarlo con ciò che vogliamo; quindi il costo è **costo per la ricerca** + **1 accesso** per scrivere il blocco.

> ⚠ **Warning**
>
> Nel caso in cui dobbiamo modificare la chiave di un record, ciò consiste in una cancellazione e un inserimento. Infatti non possiamo modificare la chiave (cambia l'ordine) ma solo gli altri campi del record

---

File con record puntati

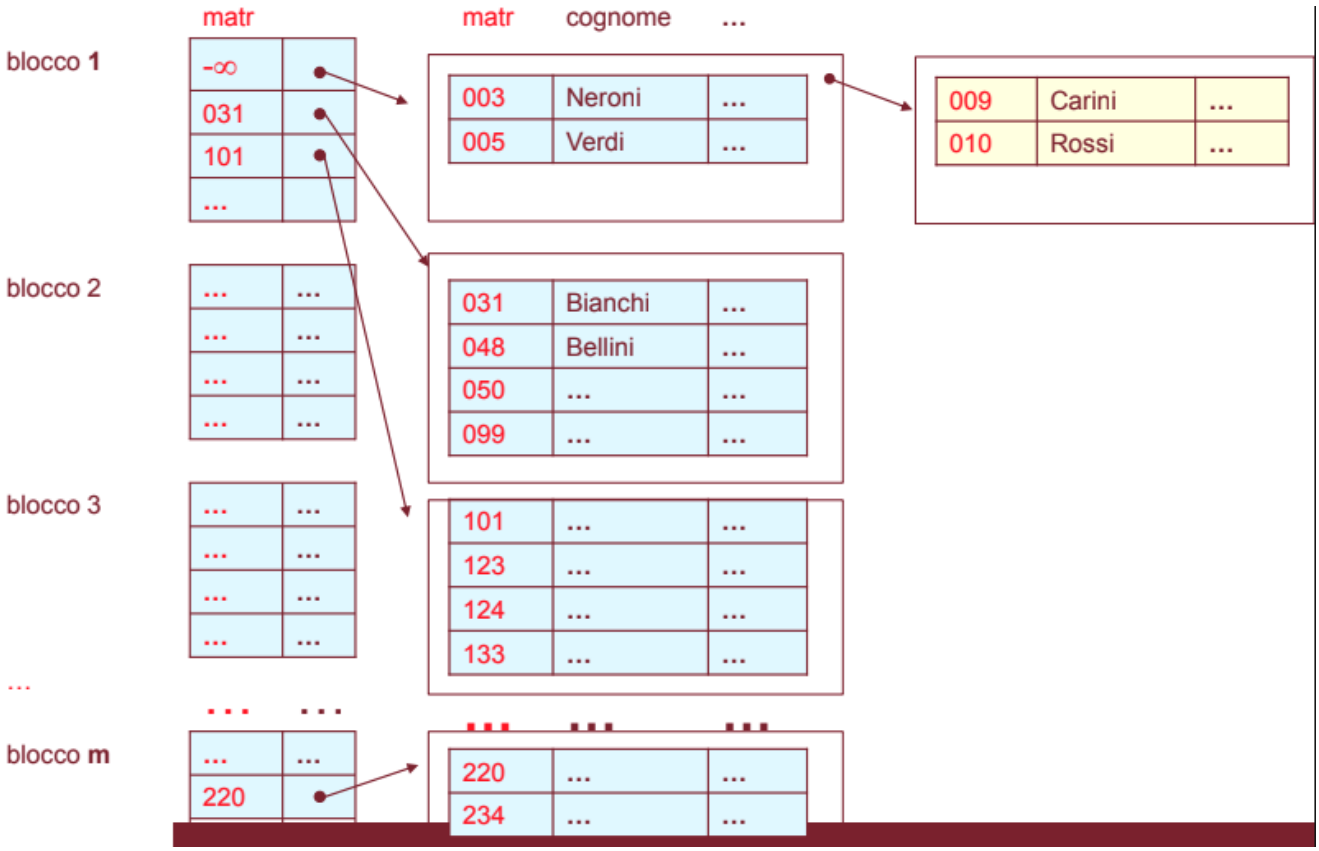
Consideriamo ora il caso in cui il file principale contiene **record puntati**.

Nella fase di inizializzazione è preferibile lasciare **più spazio libero** nei blocchi per successivi inserimenti. Poiché i record sono puntati, non possono essere spostati per mantenere l'ordinamento

Se **non c'è sufficiente spazio** in un blocco B per l'inserimento di un nuovo record, occorre richiedere al sistema un **nuovo blocco che viene collegato a B tramite un puntatore**; in tal modo ogni record del file indice punta al primo blocco di un bucket e il file indice non viene mai modificato (a meno che le dimensioni dei bucket non siano diventate tali da richiedere una riorganizzazione dell'intero file)

La **ricerca** di un record con chiave v richiede la ricerca sul file indice di un valore della chiave che ricopre v e quindi la scansione del bucket corrispondente
La **cancellazione** di un record richiede la ricerca del record e quindi la modifica dei bit di cancellazione nell'intestazione del blocco
La **modifica** di un record richiedere una ricerca del record; quindi, se la modifica non coinvolge campi della chiave, il record viene modificato e il blocco riscritto. Altrimenti la modifica equivale ad una cancellazione seguita da un inserimento; in questo caso non è sufficiente modificare il bit di cancellazione del record cancellato, ma è necessario inserire in esso un puntatore al nuovo record inserito in modo che questo sia raggiungibile da qualsiasi record che contenga un puntatore al record cancellato

Poiché non è possibile mantenere il file principale ordinato, se si vuole avere la possibilità di esaminare il file seguendo l'ordinamento della chiave occorre **inserire in ogni record un puntatore al record successivo nell'ordinamento**

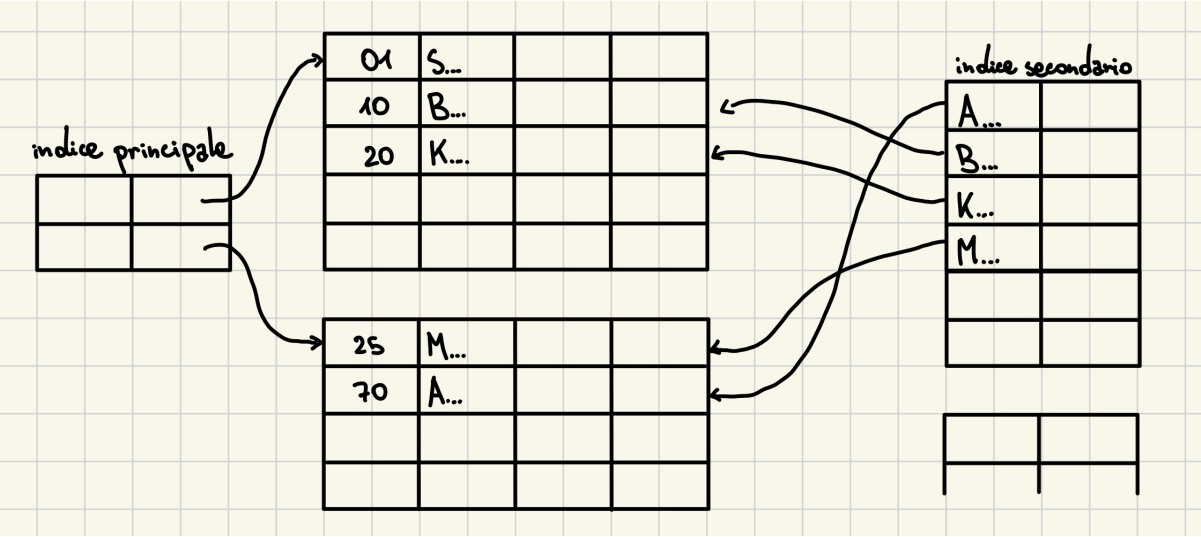


Quando i record del file principale sono puntati, una volta inseriti non possono essere spostati per mantenere l'ordinamento (quindi in pratica l'ordinamento stretto dei record vale solo all'inizializzazione). Viceversa, a differenza da quanto accade per l'ISAM

classico, i record dei blocchi indice non vengono mai modificati, quindi ciò che rimane valido è la **ripartizione degli intervalli delle chaivi**.
Se un record indice punta ad un'area di dati con valori di chiave comprese tra k_1 e k_2, questa condizione deve rimanere valida. Se il blocco originario si riempie, e arrivano nuovi record con valori di chiave compresi tra k_1 e k_2, dobbiamo allocare un nuovo blocco che però, anziché essere puntato dall'indice, sarà linkato al blocco originario, e così per ogni nuovo blocco che si riempie (abbiamo cioè una lista di **blocchi di overflow** che partono da quello originario, dove ognuno punta al successivo)

Indici sparsi e densi

Le precedenti strutture dati presentano tutte un indice sparso, ovvero nell'indice si ha una entrata per ogni blocco del file principale.
Nell'indice denso (indice secondario) si ha come chiave all'interno di un record del file indice un attributo unique che non è chiave primaria. In questo caso dunque, al posto di avere una entrata per ogni blocco, si ha un'entrata per ogni record (dato che non si possono avere contemporaneamente due ordinamenti)



Esercizi

Esercizio 1 ›

Supponiamo di avere un file di 150.000 record. Ogni record occupa 250 byte, di cui 50 per il campo chiave. Ogni blocco contiene 1024 byte. Un puntatore a blocco occupa 4 byte

1. Se usiamo un indice ISAM sparso, e assumiamo che i record non siano puntati e che il fattore di utilizzo dei blocchi del file principale sia 0.7 (cioè i blocchi non sono completamente pieni, ma pieni al più al 70%), quanti blocchi dobbiamo usare per l'indice?
2. Se usiamo un indice ISAM sparso, e assumiamo che i record siano puntati e chie i blocchi del file principale siano pieni, quanti blocchi dobbiamo utilizzare per l'indice (senza liste di overflow)?
3. Se utilizziamo la ricerca binaria, quale è il numero massimo di accessi a blocco per ricercare un record presente nel file dei casi 1. e 2., supponendo nel caso 2. di non avere liste di overflow?

Abbiamo i seguenti dati:

- il file contiene 150.000 record $\rightarrow NR = 150.000$
- ogni record occupa 250 byte $\rightarrow R = 250$
- il campo chiave occupa 50 byte $\rightarrow K = 50$
- ogni blocco contiene 1024 byte $\rightarrow CB = 1024$
- un puntatore a blocco occupa 4 byte $\rightarrow P = 4$

1

I record sono di taglia fissa, quindi non occorrono puntatori all'inizio del blocco; inoltre, non essendo stato altrimenti specificato nella traccia, assumiamo che un record non possa superare i limiti di un blocco (quindi ogni blocco contiene un numero intero di record). Poiché i record non sono puntati, possono essere spostati, e quando un blocco si riempie, ne allochiamo uno nuovo che comporterà un aggiornamento dell'indice. Di conseguenza, nei blocchi del file principale non occorre un puntatore al prossimo blocco.
L'esercizio fornisce il fattore di occupazione attuale dei blocchi del file principale, ma non indica un fattore di occupazione per i blocchi indice, quindi possiamo assumere che siano pieni

Dobbiamo stabilire quanti record indice occorrono quindi sapendo che l'indice contiene un record per ogni blocco del file principale, dobbiamo calcolare quanti sono questi blocchi.
Sappiamo che i blocchi dati sono occupati al più al 70% (indichiamo questa quantità con PO), quindi prima di tutto vediamo quanti record interi possono essere contenuti al massimo nella porzione indicata dal blocco, indicando con M questo numero

$$M = \frac{CB \cdot PO}{R} = \left\lfloor \frac{1024 \cdot 70}{100} \right\rfloor = \left\lfloor \frac{716}{250} \right\rfloor = \lfloor 2.86 \rfloor = 2$$

Il numero di blocchi da indicizzare (numero di record indice), indicato con BF, sarà

$$BF = \left\lceil \frac{NR}{M} \right\rceil = \left\lceil \frac{150.000}{2} \right\rceil = 75.000$$

Calcoliamo quindi il numero di record contenuti in ogni blocco del file indice, indicato con MI

$$MK = \left\lfloor \frac{1024}{50 + 4} \right\rfloor = 18$$

Infine indichiamo con BI il numero di blocchi indice

$$BI = \left\lceil \frac{BF}{MI} \right\rceil = \left\lceil \frac{75.000}{18} \right\rceil = 4167$$

Info

Il calcolo basato sul numero complessivo di byte necessari per l'indice $RI \cdot NR$ diviso per la capacità CB di ogni blocco darebbe un numero pari a $\left\lceil \frac{54 \cdot 75.000}{1024} \right\rceil = 3956$, ma presenterebbe un errore di fondo, in quando non assicurerebbe che ogni record sia contenuto interamente in un blocco

2

Ogni blocco del file principale deve essere visto come parte di un bucket, e quindi dobbiamo anche prevedere spazio per un puntatore. L'esercizio ci dice in questo caso che i blocchi dati sono pieni (come i blocchi indice)

Altra assunzione che possiamo fare, visto che non è stato specificato altrimenti, è che non ci siano liste di overflow, cioè che tutti i record di dati siano contenuti in blocchi puntati direttamente dall'indice

Vediamo prima di tutto in queste nuove condizioni quanti record interi di dati entrano al massimo in un blocco

Ogni blocco del file principale deve essere visto come parte di un bucket, e quindi dobbiamo anche prevedere spazio per un puntatore. L'esercizio ci dice in questo caso che i blocchi dati sono pieni, e lo stesso possiamo assumere per i blocchi indice, visto che non è specificato altrimenti. Altra assunzione che possiamo fare, visto che non è stato specificato altrimenti, è che non ci siano liste di overflow, cioè che tutti i record di dati siano contenuti in blocchi puntati direttamente dall'indice

Vediamo prima di tutto in queste nuove condizioni quanti record interi di dati entrano al massimo in un blocco

$$M = \left\lfloor \frac{CB - P}{R} \right\rfloor = \left\lfloor \frac{1020}{250} \right\rfloor = \lfloor 4.08 \rfloor = 4$$

Vediamo ora quanti blocchi vanno indicizzati (numero di record per l'indice)

$$BF = \left\lceil \frac{NR}{MR} \right\rceil = \left\lceil \frac{150.000}{4} \right\rceil = 37.500$$

Adesso utilizzo il numero di record per blocco indice dal punto 1., quindi ho

$$BI = \left\lceil \frac{BF}{MI} \right\rceil = \left\lceil \frac{37.500}{18} \right\rceil = 2084$$

3

In entrambi i casi la ricerca si effettua prima di tutto sui blocchi indice; assumiamo una ricerca binaria; i due casi si differenziano, perché nel caso di record non puntati dobbiamo ancora leggere in memoria un solo blocco di record di dati, mentre nel caso di record puntati l'indice punta ad un bucket che potrebbe contenere più blocchi di overflow, che vanno tutti esaminati

Nel nostro caso però l'esercizio dice esplicitamente che non ci sono liste di overflow, quindi in entrambe le configurazioni dobbiamo aggiungere un solo accesso a blocco, quindi nella configurazione

$$\text{caso 1. } A = \lceil \log_2 BI \rceil + 1 = \lceil \log_2 4167 \rceil + 1 = 14$$
$$\text{caso 2. } A = \lceil \log_2 BI \rceil + 1 = \lceil \log_2 2084 \rceil + 1 = 13$$

Per calcolare il numero massimo di record che ci possono essere mantenendo 13 accessi (caso 1)

$$2^{13} \cdot 18 \cdot \left\lfloor \frac{1024}{250} \right\rfloor = 589.824$$

Esercizio 2 >

Supponiamo di avere un file di 200.000 record. Ogni record occupa 150 byte, di cui 40 per il campo chiave. Ogni blocco contiene 512 byte. Un puntatore a blocco occupa 4 byte. Usiamo un indice ISAM sparso e il fattore di utilizzo sia dei blocchi

del file dati sia dei blocchi dell'indice sia 0.8 (cioè i blocchi non sono completamente pieni, ma pieni al più all'80%)

1. Quanti blocchi dobbiamo utilizzare per il file principale?
2. Quanti blocchi dobbiamo utilizzare per l'indice?
3. Calcolare il numero massimo di accessi per la ricerca di un record nel file principale, utilizzando la ricerca binaria sul file indice

Abbiamo i seguenti dati:

- il file contiene 200.000 record $\rightarrow NR = 200.000$
- ogni record occupa 150 byte $\rightarrow R = 150$
- il campo chiave occupa 40 byte $\rightarrow K = 40$
- ogni blocco contiene 512 byte $\rightarrow CB = 512$
- un puntatore a blocco occupa 4 byte $\rightarrow P = 4$
- capacità blocco effettiva $(80\%) \rightarrow CB = 409$

1

Calcoliamo il numero di record (RB) entrano in un blocco

$$RB = \left\lfloor \frac{CB}{R} \right\rfloor = \left\lfloor \frac{409}{150} \right\rfloor = 2$$

Calcoliamo il numero di blocchi per il file principale che corrisponde al numero di record del file indice(BF)

$$BF = \left\lceil \frac{NR}{RB} \right\rceil = \left\lceil \frac{200.000}{2} \right\rceil = 100.000$$

2

Calcoliamo la dimensione in termini di record dei blocchi per il file indice (IB)

$$IB = \left\lfloor \frac{CB}{K + P} \right\rfloor = \left\lfloor \frac{409}{44} \right\rfloor = 9$$

Calcoliamo i blocchi necessari per l'indice (BI)

$$BI = \left\lceil \frac{BF}{IB} \right\rceil = \left\lceil \frac{100.000}{9} \right\rceil = 11.112$$

3

Il numero massimo di accessi (MA) utilizzando la ricerca binaria è

$$MA = \lceil \log_2 BI \rceil + 1 = \lceil \log_2 11.112 \rceil + 1 = 14 + 1 = 15$$