File heap

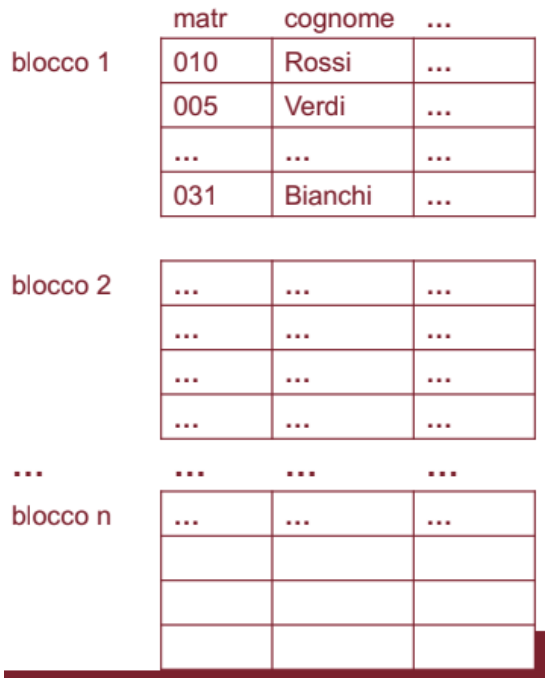
Index

Introduction

Nei file heap artiamo da una non organizzazione dei record, cioè una collocazione dei record nei file in un ordine determinato solo dall'ordine di inserimento

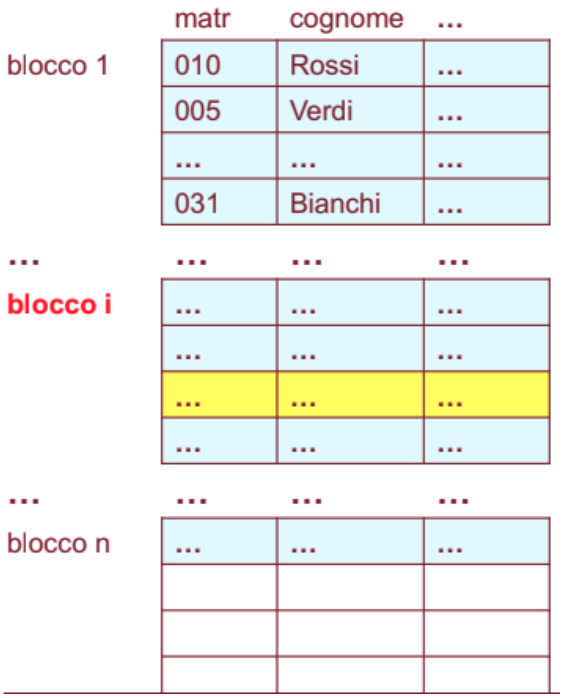
> ⚠ **Non si parla dell'heap inteso come albero di ricerca**

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



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



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

Esempio

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

Record interi per ogni blocco

$$\left\lfloor \frac{65-4}{30} \right\rfloor = \lfloor 2.03 \rfloor = 2$$

Non ho infatti spazio per gli ultimi 0.03 di record, e non posso memorizzarli in un nuovo blocco

Numero di blocchi che occorrono per memorizzare N record

$$\left\lceil \frac{N}{R} \right\rceil = \left\lceil \frac{151}{2} \right\rceil = \lceil 75.5 \rceil = 76$$

Devo allocale anche 0.5 perché 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

Cominciamo dalla ricerca quando la chiave ha un valore che non ammette duplicati
$N \rightarrow$ numero di record
$R \rightarrow$ numero di record che possono essere memorizzati in un blocco
$n = \frac{N}{R}$
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

$$\frac{1R + 2R + \cdots + nR}{N} = \frac{R(1 + 2 + \cdots + i + \cdots + n)}{N} = \frac{R}{N}\frac{n(n+1)}{2} = \\ = \frac{1}{n}\frac{n(n+1)}{2} \approx \frac{n}{2}$$

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

Per l'inserimento è necessario solamente un accesso in lettura (per portare l'ultimo blocco in memoria principale) e un accesso in scrittura (per riscrivere l'ultimo blocco in memoria secondaria dopo aver aggiunto il record)
Questo però non vale se decidiamo di non ammettere duplicati, in questo caso infatti l'inserimento deve essere preceduto dalla ricerca, quindi dobbiamo aggiungere una media di $\frac{n}{2}$ accessi per verificare che non esista già un record con la chiave data

---

Modifica

Per la modifica è necessario il costo della ricerca più un accesso in scrittura (per riscrivere in memoria secondaria il blocco, dopo aver modificato il record). Se ammettessimo duplicati ciò deve essere fatto per ogni occorrenza della chiave

---

Cancellazione

Per la modifica è necessario il costo della ricerca più un accesso in lettura (per leggere l'ultimo blocco) più due accessi in scrittura (per riscrivere in memoria secondaria il blocco modificato e l'ultimo blocco).
Se volessimo riutilizzare riutilizzare spazio ed evitare "buchi" ci basta riempire il posto mancante con il nuovo record, risulta invece essere più complicato nel caso in cui i record sono a lunghezza variabile, in quel caso dobbiamo spostare verso l'alto tutti i record successivi modificando eventuali puntatori