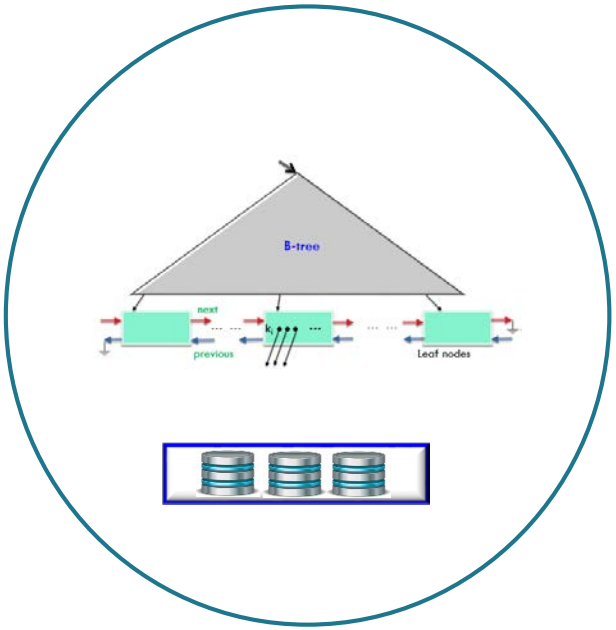



# Dispositivi e organizzazioni dei dati: esercizi

Corso di BASI DI DATI

Dario Maio

Università di Bologna

DISI (Dipartimento di Informatica — Scienza e Ingegneria)

# Esercizio n° 1

- È dato un file memorizzato in NP blocchi di disco; ogni blocco è accessibile in modo diretto specificandone l'indirizzo logico address (da 0 a NP-1); dato il seguente frammento di codice:

```
int nStep = 0;
if (IsOdd(NP))
     nStep = NP / 2;
else nStep = NP / 2 - 1;
for (int i = 0; i < nStep; i++)
   {
     int address = i + 1;
     for (int j = 0; j < nStep-i; j++)
       {
          SwapBlock(address, address + 1);
          address = address+2;
       }
    }
```

- SwapBlock(j,k) è una procedura che scambia il contenuto del blocco di indirizzo logico j con il contenuto del blocco di indirizzo logico k.
- Si determini l'effetto prodotto dal codice sul file e si valuti, in funzione di NP, il numero di operazioni di lettura e scrittura su disco.
- Si assuma per ipotesi di disporre di due buffer in memoria centrale, ciascuno atto a ospitare un singolo blocco, il primo I/O_BUFF da adibire alle operazioni di lettura e scrittura, il secondo AUX_BUFF da utilizzare come appoggio per lo scambio.

# Esercizio n° 1 – Soluzione (1)

- L'insieme dei blocchi può essere pensato come partizionato in due sottoinsiemi: il sottoinsieme dei blocchi di indirizzo dispari e il sottoinsieme dei blocchi di indirizzo pari.
- L'esecuzione del codice produce come effetto di compattare all'inizio tutti i blocchi che avevano indirizzo dispari e in fondo quelli che avevano indirizzo pari, conservando l'ordine relativo degli indirizzi all'interno di ogni sottoinsieme.
- Esempio:

```
int nStep = 0;
if (IsOdd(NP))
     nStep = NP / 2;
else nStep = NP / 2 - 1;
for (int i = 0; i < nStep; i++)
   {
     int address = i + 1;
     for (int j = 0; j < nStep-i; j++)
       {
            SwapBlock(address, address + 1);
            address = address+2;
       }
    }
```

prima dell'esecuzione:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | block address |
|---|---|---|---|---|---|---|---|
| a | b | c | d | e | f | g | content |

dopo l'esecuzione:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | block address |
|---|---|---|---|---|---|---|---|
| a | c | e | g | b | d | f | content |

# Esercizio n° 1 – Soluzione (2)

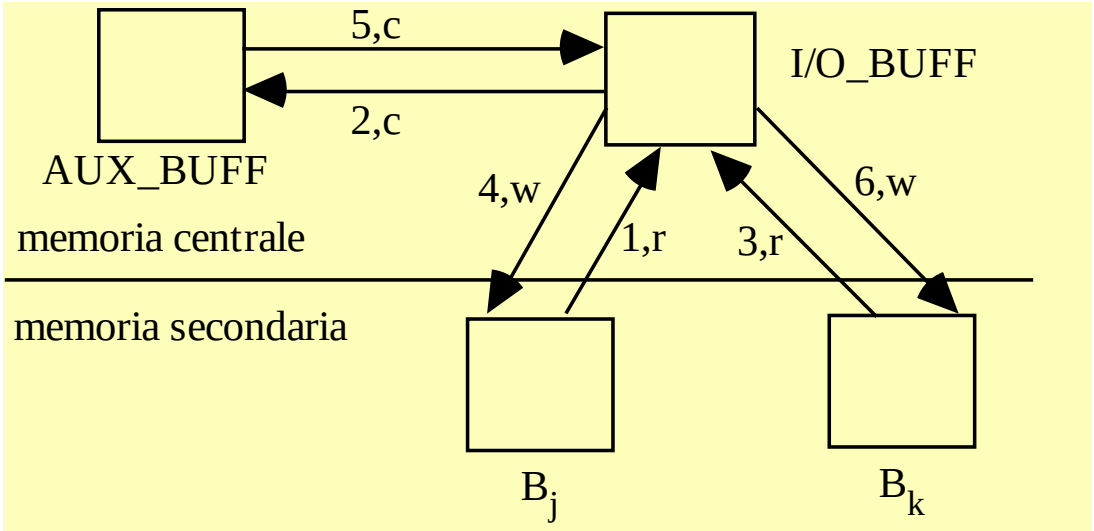
- Si deriva facilmente che il numero di chiamate NC alla procedura SwapBlock è esprimibile, in funzione di NP, come:

$$NC_{pari} = \sum_{i=1}^{\frac{NP}{2}-1} i = \frac{NP^2 - 2 \times NP}{8}, \text{ se NP è pari}$$

$$NC_{dispari} = \sum_{i=1}^{\frac{NP-1}{2}} i = \frac{NP^2 - 1}{8}, \text{ se NP è dispari}$$

# Esercizio n° 1 – Soluzione (3)

- Si consideri ora la figura seguente, che illustra l'ordinamento temporale delle operazioni compiute per lo scambio di due generici blocchi $B_j$ e $B_k$; a fianco del numero d'ordine dell'operazione è indicato il rispettivo tipo:
- r = lettura di un blocco da disco
- w = scrittura di un blocco su disco,
- c = copia di un blocco in memoria principale.




ogni scambio comporta:
2 letture da disco e
2 scritture su disco

per NP pari: $2 \times NC_{pari}$ letture ; $2 \times NC_{pari}$ scritture;
per NP dispari: $2 \times NC_{dispari}$ letture ; $2 \times NC_{dispari}$ scritture.

# Esercizio n° 2

- È dato un B-tree di ordine g=9 con NK=999999 chiavi. Si effettuano M=10 operazioni di cancellazione di chiave e ogni cancellazione avviene in una foglia lasciando un numero di chiavi residuo maggiore o uguale a g.
- Si stimi, nel caso peggiore, il costo complessivo delle M cancellazioni, in termini di numero di nodi letti e scritti, assumendo che la radice del B-tree resti disponibile in memoria principale dopo la prima cancellazione.

Algoritmo per la cancellazione di una chiave in un B-tree

```
{ ricerca la chiave y;
      if trovata
      then { if y in una foglia L
                 then cancella  y da L
                 else { ricerca nel sottoalbero di destra fino a una foglia
                        L seguendo la catena dei  puntatori q0;
                        sostituisci a y la prima chiave k di L;
                        cancella la prima chiave k di L
                      };
               if  L contiene meno di g chiavi
                 then attiva catenation e underflow
           }
}
```

# Esercizio n° 2 – Soluzione

- Per la prima cancellazione si leggono h nodi e si riscrive un nodo, per ogni altra cancellazione si leggono h–1 nodi e si riscrive un nodo.
- Dunque si operano M×(h–1)+1 letture e M scritture.
- Si ricorda che se il B-tree contiene NK chiavi vale la seguente relazione:

$$\left\lceil \log_{2g+1}(NK+1) \right\rceil \leq h \leq \left\lfloor 1+\log_{g+1}\left(\frac{NK+1}{2}\right) \right\rfloor$$

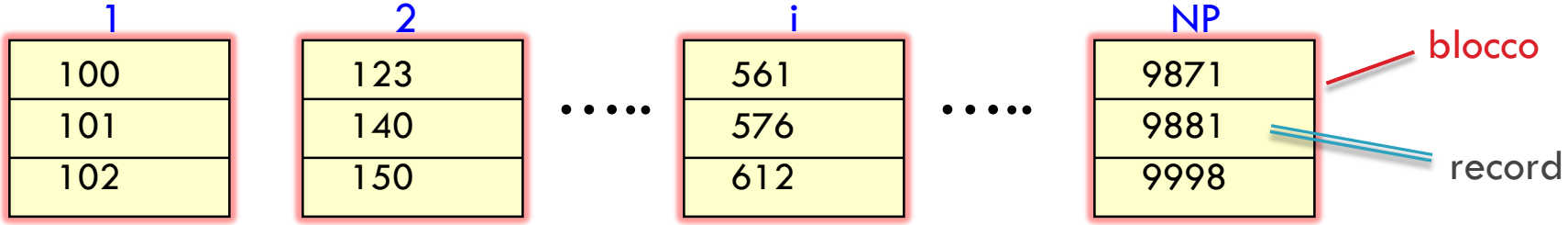
- Con i dati del problema si deriva che $5 \leq h \leq 6$, infatti:

$$\left\lceil \log_{19}(1000000) \right\rceil \leq h \leq \left\lfloor 1+\log_{10}\left(\frac{1000000}{2}\right) \right\rfloor$$

- Per il caso peggiore si fa riferimento all'altezza massima h=6 e quindi in totale si leggono 51 nodi e si riscrivono 10 nodi.

# Esercizio n°3

- Si consideri una relazione con NR record memorizzata in un sorted sequential file con NP blocchi contigui su disco (NP>>1). L'ordinamento è mantenuto rispetto a un attributo numerico A che non presenta valori ripetuti.




- Assumendo che i valori di A siano uniformemente distribuiti nell'intervallo $[k_{min},k_{max}]$ si determini il numero di blocchi a cui si deve accedere in media per reperire tutti i record con $A \in [k_L,k_H]$, nel caso di successo della ricerca.
- Sono noti inoltre i seguenti parametri:
  - $T_s$ : average seek time;
  - $T_r$ : average rotational latency;
  - $T_t$ : average time to transfer a block.
- Assumendo che le letture per reperire il valore più piccolo siano random e che le letture degli eventuali successivi blocchi siano eseguite in modalità sequenziale B blocchi alla volta (al massimo), si derivi una formula per stimare in media la durata complessiva dell'operazione di ricerca suddetta.

# Esercizio n°3 – Soluzione

- Si assume che i record del file abbiano tutti la medesima lunghezza e che ogni blocco del file contenga un numero intero e costante di record (a parte eventualmente l'ultimo blocco che può contenere un numero inferiore di record).
- Per reperire il più piccolo valore di chiave $\geq k_L$ e $\leq k_H$ si utilizza una ricerca dicotomica, reperendo in media approssimativamente un numero di blocchi pari a:

$$\log_2 NP - 1$$

- Per gli altri valori dell'intervallo richiesto si procede sequenzialmente a leggere i successivi blocchi arrestando la ricerca col primo valore $> k_H$.
- Complessivamente il costo di I/O espresso in numero di operazioni di lettura di blocchi è stimabile come:

$$C_{I/O} = \left(\log_2 NP - 1\right) + \frac{\left(k_H - k_L\right)}{\left(k_{max} - k_{min}\right)} \times NP - 1 = \left(\log_2 NP - 1\right) + NP_x$$

- Pertanto il tempo speso in media per la ricerca è pari a:

$$T_{I/O} = \left(\log_2 NP - 1\right) \times (T_s + T_r + T_t) + \left\lceil \frac{NP_x}{B} \right\rceil \times \left(T_s + T_r\right) + NP_x \times T_t$$

Letture random — Letture sequenziali B blocchi alla volta

# Esercizio n°3 – Note SDD

- Nel caso in cui si usi un SDD in luogo di un HDD il concetto di random e sequential read deve essere rivisitato alla luce dell'allocazione nel dispositivo con blocchi (tipiche dimensioni: 128 KiB, 256KiB, 512 KiB) organizzati in pagine (dimensioni tipiche: 2KiB, 4KiB, 8KiB, 16 KiB).
- In primo luogo si deve operare una corrispondenza di termini e conoscere l'allocazione dei blocchi logici di un file nei blocchi fisici di un SDD.
- Si dovrebbe inoltre considerare il numero di canali disponibili e il grado di parallelismo consentito.
- Nel caso di operazioni di lettura i costruttori forniscono alcuni parametri:
  - $T_{rl}$ : average read latency;
  - $T_t$ : average time to transfer a page;
  - random I/O read IOPS;
  - sequential read in MiB/s.
- I miglioramenti in termini di prestazioni in lettura sono rilevanti; si tenga presente che per un SDD per server il tempo per una random read è di circa 0,03 - 0,05 ms.

# Esercizio n°4

- Si consideri una relazione POSTS che consiste di $NR_{POSTS}$=1000000 record di dimensione d=200 byte l'uno, con chiave primaria costituita dall'attributo IdPost.
- È costruito sull'attributo IdPost un primary dense clustered index di tipo $B^+$-tree. Le pagine del file dati e del file indice hanno entrambe dimensione D=8KB di cui 192 byte sono riservati. Sia 0.80 l'utilizzazione media di ogni pagina del file dati. Si consideri per l'indice il caso peggiore con u = 0.5, ovvero nodi riempiti al 50% a eccezione della radice.
- Si assumano lunghezze uguali per i separatori e per le chiavi dei record.
- Siano: len(IdPost) = 12 byte, len(p) = len(q) = 4 byte.
- Si calcolino:
  1. il numero totale delle pagine $NP_{POSTS}$ del file dati;
  2. l'ordine $g_{leaf}$(IX(POSTS.IdPost)) e il numero $NL_{IX(POSTS.IdPost)}$ di foglie dell'indice;
  3. l'altezza $h_{IX(POSTS.IdPost)}$ massima dell'indice.

# Esercizio n°4 – Soluzione (1)

- Poiché l'indice è primario si ha $NK_{POSTS.IdPost} = NR$.
- Lo spazio utile in una pagina dati o indice è D* = 8192-192 = 8000 byte
- Quesito 1): il numero delle pagine dati del file si ricava come segue:

$$NP_{POSTS} = \left\lceil \frac{NR_{POSTS}}{\left\lfloor \frac{u_P \times D^*}{d} \right\rfloor} \right\rceil = \left\lceil \frac{1000000}{\left\lfloor \frac{0.80 \times 8000}{200} \right\rfloor} \right\rceil = 31250$$

- Quesito 2): si ricorda che in ogni foglia si possono memorizzare al più $2g_{leaf}$ valori di chiave e $2g_{leaf}$ puntatori a record, oltre ai 2 puntatori rispettivamente alla foglia precedente e alla foglia successiva:

  $2g_{leaf} \times len(k) + (2g_{leaf}) \times len(p) + 2 \times len(q) \leq D^*$ da cui si ricava l'ordine

$$g_{leaf}\left(IX(POSTS.IdPost)\right) = \left\lfloor \frac{D^* - 2 \times len(q)}{2 \times (len(k) + len(p))} \right\rfloor = \left\lfloor \frac{8000 - 2 \times 4}{2 \times (12 + 4)} \right\rfloor = 249$$

# Esercizio n°4 – Soluzione (2)

- Pertanto il numero delle foglie, considerando l'utilizzazione u = 0.5 nel caso peggiore, è stimabile come:

$$NL_{IX(POSTS.IdPOSTS)} = \left\lceil \frac{NR_{POSTS}}{2 \times g_{leaf} \times u} \right\rceil = \left\lceil \frac{1000000}{2 \times 249 \times 0.5} \right\rceil = 4017$$

- Domanda 3): per ipotesi i separatori hanno la stessa lunghezza delle chiavi dunque $g = g_{leaf}$ e l'altezza massima si può dunque derivare come:

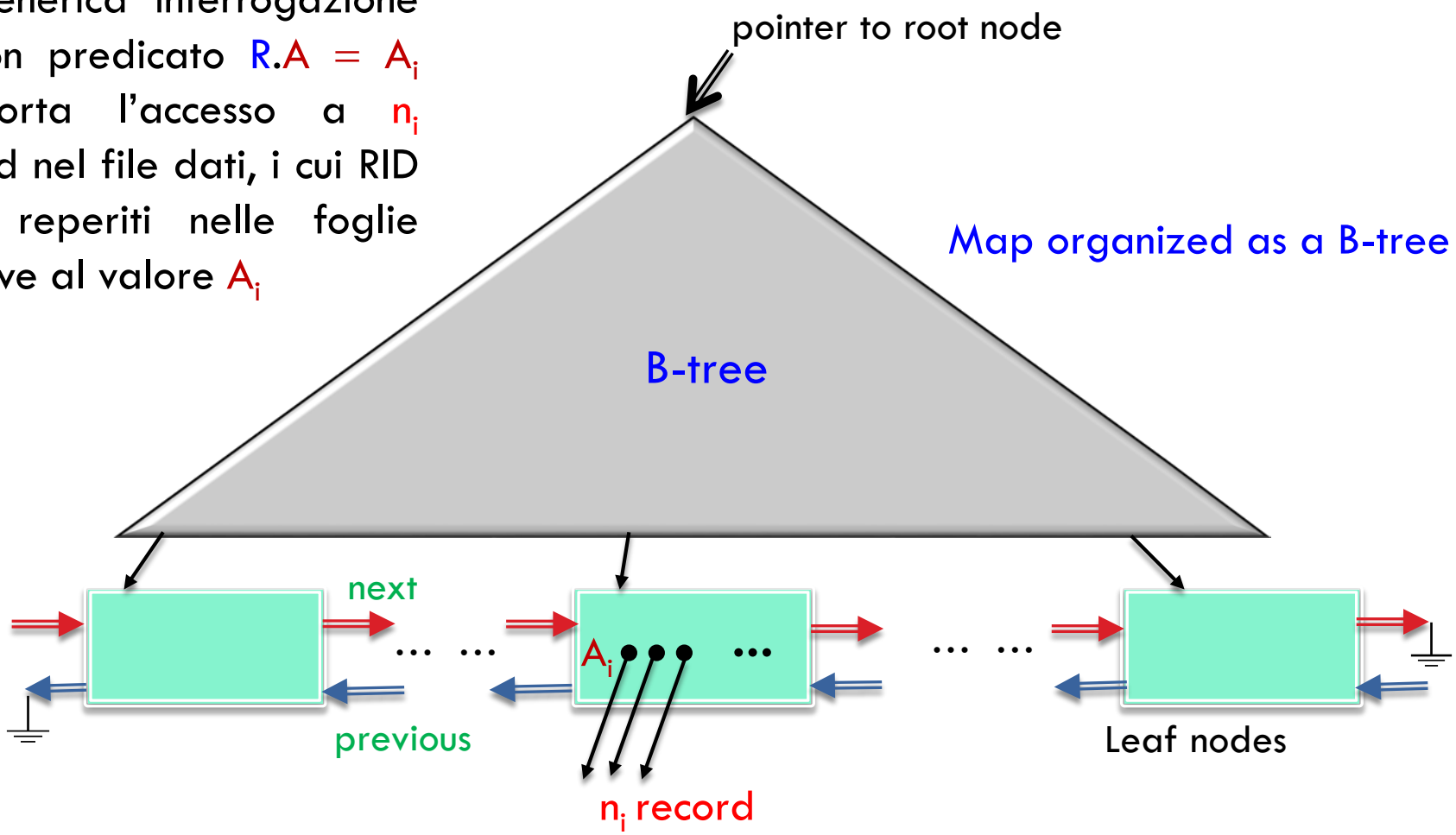
$$h_{IX(POSTS.IdPOSTS)} = 2 + \left\lfloor \log_{g+1} \frac{NL}{2} \right\rfloor = 2 + \left\lfloor \log_{250} \frac{4017}{2} \right\rfloor = 3$$

# Esercizio n°5

- È data una relazione R con $NR_R$ record memorizzati in $NP_R$ pagine. Sull'attributo R.A è disponibile un secondary unclustered index IX(R.A) strutturato come un $B^+$-tree di altezza $h_{IX(R.A)}$ con $NL_{IX(R.A)}$ foglie.
- Nelle foglie dell'indice, per ogni singolo valore dell'attributo R.A, i RID sono mantenuti ordinati.
- Il numero di valori distinti di R.A è pari a $NK_{R.A}$ e, per ogni possibile valore $A_i$, $i = 1,2, ...., NK_{R.A}$, si conosce il numero $n_i$ di record che presentano quel valore $A_i$ nella relazione.
- È dato inoltre un insieme $Q = \{q_1, q_2, ...., q_m\}$ di interrogazioni con predicato di selezione (R.A = valore) che sono risolte utilizzando l'indice suddetto come via di accesso.
- Si stimi il valor medio del costo globale di accesso a pagine dati e indice, necessario per risolvere tutte le m interrogazioni, sapendo che la probabilità $p_i$ che una generica interrogazione $q_k$ interessi il valore $A_i$ è pari a $n_i/NR_R$.

# Esercizio n°5 – Soluzione (1)

La generica interrogazione $q_k$ con predicato $R.A = A_i$ comporta l'accesso a $n_i$ record nel file dati, i cui RID sono reperiti nelle foglie relative al valore $A_i$



# Esercizio n°5 – Soluzione (2)

- La generica interrogazione $q_k$ viene risolta via indice.
- Per ipotesi si conosce il fattore di selettività del predicato: $f_{(R.A=A_i)} = n_i/NR_R$
- Per l'accesso all'indice è sensato assumere che il singolo valore $A_i$ comporti un numero di foglie pari a:

$$EL_{R.A=A_i} = \left\lceil \frac{n_i}{NR_R} \times NL_{IX(R.A)} \right\rceil$$

- Pertanto quando $q_k$ interessa il valore $A_i$ si ha un costo di accesso all'indice pari a:

$$C_I = h_{IX(R.A)} - 1 + \left\lceil \frac{n_i}{NR_R} \times NL_{IX(R.A)} \right\rceil$$

# Esercizio n°5 – Soluzione (3)

- Per quanto riguarda l'accesso ai dati si può ricorrere alla formula di Cardenas tenendo conto che il numero di record con R.A = $A_i$ è pari a $n_i$.
- Pertanto, per la generica interrogazione che richiede $n_i$ record con valore $A_i$ dell'attributo R.A, il costo di accesso ai dati è pari al numero di pagine attese che si può stimare come:

$$C_D = \Phi\left(n_i, NP_R\right)$$

- Ne consegue che il costo medio globale per l'insieme Q delle interrogazioni è:

$$m \times \sum_{1}^{NK_A} p_i \times \left(C_I + C_D\right)_i = m \times \sum_{1}^{NK_A} p_i \times \left( h_{IX(R.A)} - 1 + \left\lceil \frac{n_i}{NR_R} NL_{IX(R.A)} \right\rceil + \Phi(n_i, NP_R) \right)$$

# Esercizio n°6

- È data una relazione R con $NR_R$ record memorizzati in $NP_R$ pagine. Sull'attributo R.A è disponibile un secondary unclustered index IX(R.A) strutturato come un $B^+$-tree di altezza $h_{IX(R.A)}$ con $NL_{IX(R.A)}$ foglie. Nelle foglie, per ogni singolo valore dell'attributo R.A, i RID sono mantenuti ordinati.
- È data un'interrogazione di range che seleziona $EK_{R.A}$ valori dell'attributo R.A.
- Per quale valore di $EK_{R.A}$ è conveniente risolvere l'interrogazione con accesso via indice invece di una scansione completa della relazione? Si assuma distribuzione uniforme sia dei valori di R.A nei record sia dei record nel file che memorizza la relazione.
- <u>Soluzione</u>

  Si ricorda che l'accesso via indice comporta:

  - un costo $C_I$ per reperimento delle foglie che contengono gli $EK_{R.A}$ valori distinti di R.A al fine di trovare, per ognuno dei valori $A_i$ ($i = 1, 2, \ldots, EK_{R.A}$), i RID dei record che presentano il valore $A_i$.
  - un costo $C_D$ per reperire le pagine dati che contengono i record interessati.
- Rispondere al quesito significa risolvere la disequazione: $C_I + C_D < NP_R$

# Esercizio n°6 - Soluzione

- Si ha per la query di range:

$$C_I = h_{IX(R.A)} - 1 + \left\lceil \frac{EK_{R.A}}{NK_{R.A}} \times NL_{IX(R.A)} \right\rceil$$

$$C_D = EK_{R.A} \times \Phi\left( \frac{NR_R}{NK_{R.A}}, NP_R \right)$$

- Dunque perché l'indice sia conveniente si deve risolvere in $EK_{R.A}$ la seguente disequazione:

$$h_{IX(R.A)} - 1 + \left\lceil \frac{EK_{R.A}}{NK_{R.A}} \times NL_{IX(R.A)} \right\rceil + EK_{R.A} \times \Phi\left( \frac{NR_R}{NK_{R.A}}, NP_R \right) < NP_R$$

# Esercizio n°7

- Per l'accesso a un insieme di P = 100 pagine di una relazione memorizzata su disco sia HR = 0.8 l'Hit Ratio nella cache del controller.
- Sono noti i seguenti parametri:
  - $t_{I/O}$ = 10 ms : tempo medio per un'operazione di lettura o scrittura di pagina su disco;
  - $t_C = 0.01 \times t_{I/O}$ : tempo medio di accesso alla cache.
- Si consideri inoltre nullo il tempo necessario a riconoscere se una pagina è già presente nella cache.

- Si calcoli l'effettivo tempo medio per l'accesso alle P pagine nei seguenti due casi:
  1. per ogni pagina richiesta non già presente nella cache non si deve mai operare rimpiazzamento;
  2. per ogni pagina richiesta che non è già presente nella cache si deve operare un rimpiazzamento con riscrittura della pagina rimpiazzata.

# Esercizio n°7 - Soluzione

Caso 1

- Il tempo effettivo per la lettura di una pagina è:

$$t_E = HR \times t_C + (1 - HR) \times t_{I/O} = 0.80 \times 0.01 \times 10 + 0.20 \times 10 = 2.08 \text{ms}$$

- Pertanto per leggere P pagine il tempo medio speso è pari a:

$$P \times t_E = 100 \times 2.08 = 208 \text{ ms} .$$

Caso 2

$$t_E = HR \times t_C + (1 - HR) \times 2 \times t_{I/O} = 0.80 \times 0.01 \times 10 + 0.20 \times 2 \times 10 = 4.08 \text{ms}$$

- Pertanto si ha un tempo medio complessivo pari a: 408 ms.

# Esercizio n°8

- Calcolare l'occupazione di memoria di un secondary $B^+$-tree con i seguenti dati:

| | |
|---|---|
| dimensione di un nodo: | D = 4096 con 96 byte di header |
| lunghezze puntatori: | len(p) = len(q) = 4 byte |
| lunghezza chiave: | len(k) = 10 byte |
| numero record: | NR = 1000000 |
| numero valori distinti di chiave: | NK = 1000 |
| utilizzazione: | u = ln 2 |

- Si assuma che la lunghezza dei separatori nella mappa B-tree sia pari alla lunghezza delle chiavi. Si trascuri per semplicità l'occupazione dei puntatori alla foglia precedente e a quella successiva. Si assuma inoltre che la dimensione di un nodo coincida con quella di un blocco di disco.

# Esercizio n°8 – Soluzione (1)

- Il numero delle foglie NL dipende anche dal numero di valori di chiave distinti, NK. Lo spazio utile per un nodo è pari a D* = D–96 = 4000 byte. Trascurando i puntatori alla foglia precedente e successiva NL si può stimare come:

$$NL = \left\lceil \frac{NK \times len(k) + NR \times len(p)}{D^* \times u} \right\rceil = \left\lceil \frac{1000 \times 10 + 1000000 \times 4}{4000 \times \ln(2)} \right\rceil = 1447$$

- L'ordine g si può ricavare ricordando che in ogni nodo non foglia deve valere:

$$2g \times len(k) + (2g+1) \times len(q) \leq D^*$$

- Pertanto si ottiene:

$$g = \left\lfloor \frac{D^* - len(q)}{2(len(k) + len(q))} \right\rfloor = 142$$

# Esercizio n°8 – Soluzione (2)

- Per calcolare l'altezza dell'albero occorre considerare che NK è minore di NL pertanto la formula per calcolare l'altezza minima e l'altezza massima dell'albero è:

$$1+\left\lceil \log_{2g+1} NK \right\rceil \le h \le 2+\left\lfloor \log_{g+1} \frac{NK}{2} \right\rfloor$$

- Con i dati del problema si ricava: $h_{min} = h_{max} = 3$.
- Per calcolare l'occupazione di memoria si può procedere in via approssimata supponendo che al livello intermedio si abbiano g+1 = 143 puntatori. In questo caso si hanno:
- 1447 blocchi al livello 3 (foglie);
- $\left\lfloor \frac{NK}{g+1} \right\rfloor = \left\lfloor \frac{1000}{143} \right\rfloor = 6$ al livello 2;
- 1 al livello 1 – radice dell'albero.

L'occupazione totale è pertanto:

$$(1447+6+1)\times D =$$
$$= 1454 \times 4096 = 5816\ KB$$

# Esercizio n°9

- È data la relazione USERS (alias U) con $NR_U$ record memorizzati in $NP_U$ pagine. Sull'attributo U.JOB è costruito un secondary unclustered $B^+$-tree a PID (Page Identifier), cioè nelle foglie per ogni valore di chiave, la lista di puntatori consiste di tanti PID quante sono le pagine del file dati che contengono almeno un record con quel valore di chiave.
- Siano rispettivamente $NK_{JOB}$ il numero di valori distinti dell'attributo U.JOB e $DK_{JOB}$ la cardinalità del dominio dell'attributo U.JOB.
- Si calcoli la probabilità di aggiornare l'indice (ovvero aggiungere un nuovo PID in una foglia) inserendo un nuovo record nella relazione USERS, sapendo che il record viene allocato in una delle $NP_U$ pagine già esistenti.

# Esercizio n°9 – Soluzione (1)

- Due sono le situazioni in cui, a seguito dell'inserimento di un nuovo record nel file dati, è necessario aggiungere un nuovo PID nell'indice.
  - Caso 1: il valore dell'attributo U.JOB del nuovo record non è già presente in nessun altro record della relazione e quindi non compare nell'indice stesso;
  - Caso 2: il valore dell'attributo U.JOB del nuovo record, pur già presente in uno o più record della relazione, non è presente nella pagina dati dove il record deve essere inserito.
- Con ipotesi di uniformità e supponendo che ogni pagina contenga mediamente un ugual numero di record pari a $NR_U/NP_U$, la probabilità di aggiornare l'indice a causa di un inserimento di un nuovo record può essere calcolata come:

$$\left(1-\frac{NK_{JOB}}{DK_{JOB}}\right)+\frac{NK_{JOB}}{DK_{JOB}}\times\left(1-\frac{1}{NK_{JOB}}\right)^{\frac{NR_U}{NP_U}}$$

Caso 1 Caso 2

# Esercizio n°9 – Soluzione (2)

- Infatti con le ipotesi di uniformità si ha quanto appresso descritto.
- La probabilità che il valore U.JOB del nuovo record non appartenga a quelli che già esistono nella relazione è pari a: $\left(1-\frac{NK_{JOB}}{DK_{JOB}}\right)$
- La probabilità che il valore U.JOB del nuovo record appartenga a quelli che già esistono nella relazione è pari a: $\frac{NK_{JOB}}{DK_{JOB}}$
- Se già esiste nella relazione il valore U.JOB del nuovo record allora la probabilità che non sia già presente nella pagina dove deve essere inserito è pari a: $\left(1-\frac{1}{NK_{JOB}}\right)^{\frac{NR_U}{NP_U}}$
- La probabilità di aggiornare l'indice è: $\left(1-\frac{NK_{JOB}}{DK_{JOB}}\right)+\frac{NK_{JOB}}{DK_{JOB}}\times\left(1-\frac{1}{NK_{JOB}}\right)^{\frac{NR_U}{NP_U}}$

# Esercizio n°10

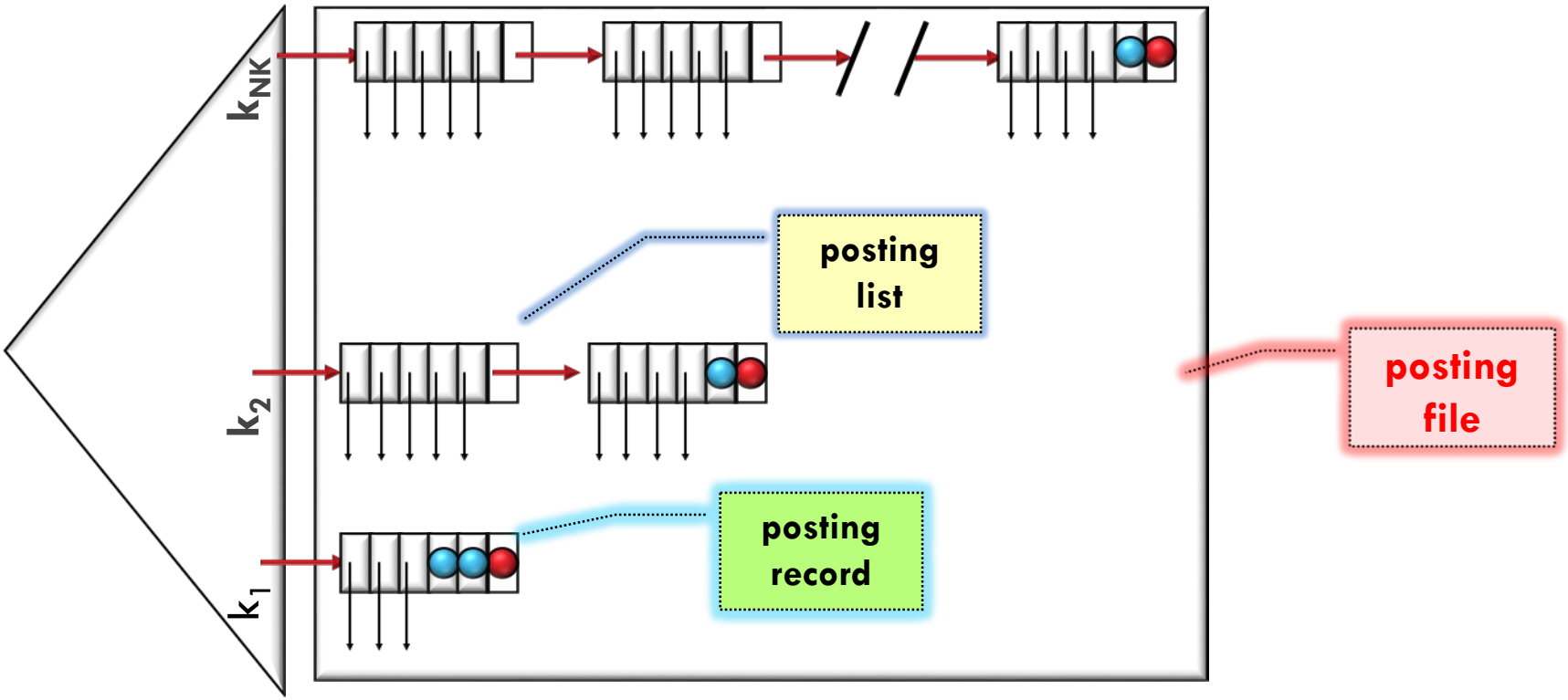
- È dato un secondary unclustered index organizzato come un B+-tree che usa un posting file separato per la memorizzazione delle liste di RID:

| | |
|---|---|
| dimensione di un nodo: | D = 4096 con 96 byte di header |
| lunghezze puntatori: | len(p) puntatore alla testa della lista |
| | = len(q) = 4 byte |
| lunghezza chiave: | len(k) = 20 byte |
| numero record: | NR = 1000000 |
| numero pagine dati | NP = 40000 |
| numero valori distinti di chiave: | NK = 1000 |
| ordine: | g = 83 |
| utilizzazione: | u = ln 2 |

- Si ha una range query che porta a reperire EK = 10 valori di chiave nel file dati; utilizzando per la ricerca l'indice suddetto si stimi il numero di accessi a pagine, assumendo pari a $C_{post}$ il costo pagato per accedere a una lista di RID associata a un valore di chiave.

# Esercizio n°10 – Soluzione (1)

- Il posting file memorizza le liste di RID associate a singoli valori di chiave e le foglie del B+-tree contengono solo valori di chiave ciascuno seguito da puntatore alla relativa lista.



# Esercizio n°10 – Soluzione (2)

- Lo spazio utile per un nodo è pari a $D^* = D - 96 = 4000$ byte.
- Trascurando i puntatori alla foglia precedente e successiva NL si può stimare come:

$$NL = \left\lceil \frac{NK \times len(k) + NK \times len(p)}{D^* \times u} \right\rceil = \left\lceil \frac{1000 \times 20 + 1000 \times 4}{4000 \times \ln(2)} \right\rceil = 9$$

- L'altezza massima è pari a: $h = 2 + \left\lfloor \log_{g+1} \frac{NL}{2} \right\rfloor = 2$ .
- Per l'accesso alle foglie il costo è pari a:

$$C_I = h - 1 + \left\lceil \frac{EK}{NK} \times NL \right\rceil = 2 - 1 + \left\lceil \frac{10}{1000} \times 9 \right\rceil = 2$$

- Per l'accesso alle EK liste di RID si ha:

$$C_P = EK \times C_{post} = 10 \times C_{post} .$$

- Per l'accesso alle pagine dati, con ipotesi di uniformità, si ha:

$$C_D = EK \times \Phi\left(\frac{NR}{NK}, NP\right) \simeq 10 \times 988 = 9880$$