


 **Esercizio di riepilogo sull'uso di indici per l'accesso alle relazioni**

Dario Maio
<http://bias.csr.unibo.it/maio/>

 Esercizio – uso di indici 
1



 **Esercizio riassuntivo (1)**

- Consideriamo una relazione **Studenti**, che consiste di **NT = 200000** record di **160 byte** l'uno, organizzati in un **file ordinato su matricola**. Le pagine hanno dimensione **P = 4 KB** e il disco ha le seguenti caratteristiche:
 - Rotational latency: **Tr = 8.5 ms**
 - Average Seek time: **Ts = 12 ms**
 - Page transfer time: **Tt = 1 ms**
- Sono presenti anche due indici:

```
CREATE UNIQUE INDEX MatrIDX  
ON Studenti(matricola) CLUSTER
```



```
CREATE INDEX DataNascIDX  
ON Studenti(datanascita)
```

 Esercizio – uso di indici 
2

Esercizio riassuntivo (2)

Dimensione del file dati e degli indici:

per il file e per gli indici in ogni pagina il page header e la directory lasciano a disposizione uno spazio disponibile per i record pari a 4000 byte che viene riempito all'85%. I blocchi sono allocati in modo contiguo su una medesima traccia.

Si suppone che indici siano a un solo livello e i puntatori siano RID di 4 byte.

L'indice **MatrIDX** è **primario (e denso)**, quindi $NK(\text{MatrIDX}) = 200000$ con ogni coppia che occupa $(10 + 4)$ byte.

L'indice **DataNascIDX** è **secondario**; supponendo $NK(\text{DataNascIDX}) = 4000$, a ogni valore di chiave corrispondono in media 50 record; quindi valore di chiave e puntatori occupano $(10 + 50 \cdot 4) = 210$ byte.

Struttura	Numero pagine dati (NP) o foglia (NL)	Dimensione file
File dati	$\lceil 200000 / \lfloor 0.85 \cdot 4000 / 160 \rfloor \rceil = 9524$	37.20 MB
MatrIDX	$\lceil 200000 / \lfloor 0.85 \cdot 4000 / 14 \rfloor \rceil = 827$	3.23 MB
DataNascIDX	$\lceil 4000 / \lfloor 0.85 \cdot 4000 / 210 \rfloor \rceil = 250$	0.98 MB

Esercizio - uso di indici

3

Esercizio riassuntivo (3)

Ricerca per matricola:

Usando MatrIDX le letture sono random in quanto si effettua una ricerca binaria.

Si riportano per un raffronto anche i tempi relativi a organizzazioni heap e hash.

Cammino d'accesso	Costo di ricerca
Heap	$T_s + T_r + (NP+1)/2 \cdot T_t = 4020.5 \text{ ms}$
Hash	$T_s + T_r + T_t = 21.5 \text{ ms}$
Sequenziale	$\lceil \log_2(NP) \rceil \cdot (T_s + T_r + T_t) = 301 \text{ ms}$
MatrIDX	$\lceil \log_2(NL) \rceil \cdot (T_s + T_r + T_t) + (T_s + T_r + T_t) = 215 \text{ ms}$

- N.B. Il termine "cammino d'accesso sequenziale" non deve trarre in inganno, nel senso che l'organizzazione sottostante è comunque ad accesso diretto, dunque essendo il file ordinato su matricola si può far ricorso a una ricerca dicotomica.
- Il vantaggio che si ottiene usando un indice clustered è significativo, ma le prestazioni sono ancora lontane da quelle che si possono ottenere con un file hash, poiché si leggono comunque in modo random molte pagine dell'indice.

Esercizio - uso di indici

4

Esercizio riassuntivo (4)

Ricerca per intervallo di data di nascita

Si cercano tutti gli studenti nati in un certo periodo in cui vi sono 80 date di nascita (e quindi 4000 studenti).

Ogni foglia dell'indice contiene 16 valori di chiave, quindi oltre alla foglia individuata dalla ricerca binaria bisogna leggerne altre 4 (in sequenza).

Cammino d'accesso	Costo di ricerca per chiave
Sequenziale	$T_s + T_r + NP * T_t = 9544.5 \text{ ms}$
DataNascitaIDX	$(\lceil \log_2(NL) \rceil * (T_s + T_r + T_t) + 4 * T_t + 4000 * (T_s + T_r + T_t) = 86176 \text{ ms}$

L'uso di un indice unclustered può risultare svantaggioso se il numero dei record da reperire è elevato, in quanto ogni accesso al file dati comporta una lettura random.

Esercizio – uso di indici



5

Indici: altri usi importanti

- Oltre che per risolvere predicati di ricerca, un indice può essere usato per accedere ai record secondo un certo ordine, e quindi può anche essere utile se nella query ci sono clausole ORDER BY e GROUP BY.

```
SELECT *  
FROM Studenti  
ORDER BY datanascita
```

- Per alcune query la presenza di un indice può addirittura evitare di dover accedere al file dati!

```
SELECT COUNT(DISTINCT datanascita)  
FROM Studenti
```

Se esiste un indice su (DeptNo, Salary), anche la seguente query non ha bisogno di accedere al file dati:

```
SELECT DeptNo, MAX(Salary)  
FROM Employee  
GROUP BY DeptNo
```

Esercizio – uso di indici



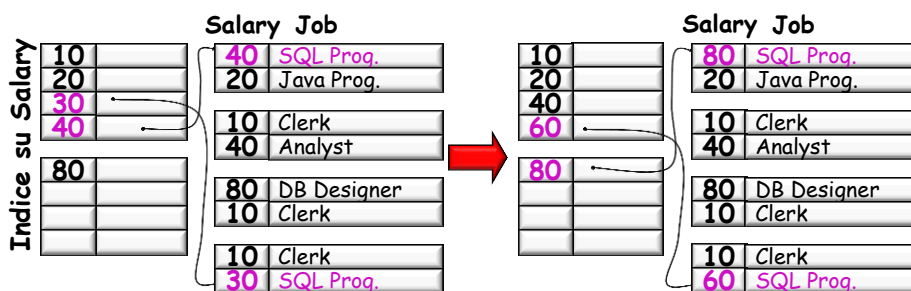
6

Indici e aggiornamenti

- Poiché ogni indice deve riflettere la situazione corrente nel file dati, è evidente che **ogni operazione di modifica dei dati si deve propagare anche agli indici interessati**. Ad esempio:

```
UPDATE Employee
SET     Salary = 2 * Salary
WHERE   Job = 'SQL Programmer'
```

richiede che si modifichino anche tutti gli indici costruiti su Salary.



Esercizio - uso di indici

7

Esercizio riassuntivo (5)

- Si consideri ancora la relazione Studenti, che consiste di **NT = 200000** record di **160 byte** l'uno, organizzati in un file ordinato su matricola. Le pagine hanno dimensione **P = 4 KB** e il disco ha le seguenti caratteristiche:

- Rotational latency: **Tr = 8.5 ms**
- Seek time: **Ts = 12 ms**
- Page transfer time: **Tt = 1 ms**

- Sono presenti anche due indici:

```
CREATE UNIQUE INDEX MatrIDX
ON Studenti(matricola) CLUSTER
```

```
CREATE INDEX DataNascIDX
ON Studenti(datanascita)
```

Esercizio - uso di indici

8

Continua ... (1)

- Confrontiamo le prestazioni ottenibili **usando B⁺-tree** per le query sulla relazione Studenti viste in precedenza:

Dimensione dei B⁺-tree:

Consideriamo il caso peggiore, in cui **ogni nodo diverso dalla radice è riempito al 50%**.

Per l'indice **MatrIDX**, si ha **g = 142**, e si deriva che è **h = 3**.

Per l'indice **DataNascIDX**, supponendo sempre **NK = 4000** e usando liste di puntatori nelle foglie (50 RID per valore di chiave), si ha **g = 9** con **h = 4**.

Struttura	Numero pagine dati (NP) o foglia (NL)	Dimensione file
File dati	$\lceil 200000 / \lfloor 0.85 * 4000 / 160 \rfloor \rceil = 9524$	37.20 MB
MatrIDX B ⁺ -tree	$\lceil 200000 / 142 \rceil = 1409$	5.54 MB
DataNascIDX B ⁺ -tree	$\lceil 4000 / 9 \rceil = 445$	1.93 MB

Esercizio - uso di indici



9

Continua... (2)

Ricerca per matricola:

Usando MatrIDX si leggono **h = 3** pagine indice e 1 pagina dati.

Si riportano per un raffronto i tempi relativi alle organizzazioni heap e hash.

Cammino d'accesso	Costo di ricerca
Heap	$T_s + T_r + NP/2 * T_t = 4020.5 \text{ ms}$
Hash	$T_s + T_r + T_t = 21.5 \text{ ms}$
Sequenziale	$\lceil \log_2(NP) \rceil * (T_s + T_r + T_t) = 301 \text{ ms}$
MatrIDX B ⁺ -tree	$(h + 1) * (T_s + T_r + T_t) = 86 \text{ ms}$

Esercizio - uso di indici



10

Continua ... (3)

Ricerca per intervallo di data di nascita

Si cercano tutti gli studenti nati in un certo periodo in cui vi sono 80 date di nascita (e quindi 4000 studenti).

Ogni foglia contiene 9 valori di chiave, quindi oltre alla foglia individuata bisogna leggerne altre 8 (le letture sono random perché non c'è nessuna garanzia che i nodi foglia siano allocati in modo contiguo sul disco)

Cammino d'accesso	Costo di ricerca per chiave
Sequenziale	$T_s + T_r + NP * T_t = 9544.5 \text{ ms}$
DataNascitaIDX B+-tree	$(h + 8) * (T_s + T_r + T_t) + 4000 * (T_s + T_r + T_t) = 86258 \text{ ms}$

- Anche in questo caso l'uso dell'indice non è conveniente, in quanto è **prevalente il costo di accesso al file dati**.

