Esercizi: EX 1. STRUTTURE PRIMARIE

Si consideri una tabella R(A,B), con A di 18 byte e B di lunghezza variabile, massimo 20 byte. R è memorizzato in un file disordinato, con blocchi dati di B=400 byte e con record, di lunghezza variabile, che richiedono 2 byte per la gestione (e.g., cancellazione, puntatori etc). Gli spazi dei record eliminati sono marcati come liberi e non riutilizzati. Nel caso di modifica del record, dato la lunghezza variabile, se la lunghezza del record cambia, l'operazione viene gestita come cancellazione ed inserimento del record modificato. La tabelle è inizialmente vuota

Si supponga di eseguire le seguenti operazioni:

1. Inserimento di 1000 record con campo B di lunghezza nulla
2. Aggiornamento di 100 record, con modifica del valore di B, che occupa ora 20 byte
3. Riorganizzazione del file con ricompattazione dei blocchi

Indicare il numero di blocchi occupati dalla tabella dopo ciascuna operazione.

Esercizi: EX 1. STRUTTURE PRIMARIE

(1) 1000 record di lunghezza di A + 2 byte di gestione.
1000 record di lunghezza 20 byte.
Fattore di blocco 400/20= 20 record per blocco.
blocchi dopo inserimento 1000/20= 50

(2) i 100 record non possono essere sovrascritti. Si cancellano e si aggiungono 100 record nuovi.
Dimensione dei nuovi record: dimensione A + dimensione B + 2 byte gestione= 18+20+2= 40
Fattore di blocco= 400/40= 10 record per blocco
Blocchi aggiunti dopo la modifica: 100/10= 10
blocchi dopo modifica= 50+10=60

(3) Nella riorganizzazione del file sono cancellati dai blocchi i 100 cancellati.
900 record di lunghezza: dimensione A + 2 byte= 20 byte
fattore di blocco: 400/20= 20
blocchi: 900/20= 45
100 record di lunghezza: dimensione A+ dimensione B + 2 byte gestione= 40
fattore di blocco: 400/40=10
blocchi 100/10= 10
totale blocchi: 45 + 10 = 55.

Esercizi: Ex.2 INDICI PRIMARI/SECONDARI/MULTILIVELLO

Si consideri la seguente configurazione:

Dimensione blocco, B = 512 byte.

Puntatore a blocco, P = 6 byte

#record r = 100.000 IMPIEGATO

Campi record IMPIEGATO :

NOME (30 byte),
SSN (9 byte),
NUMERO_DIPARTIMENTO (9 byte),
DATA_NASCITA (8 byte),
SESSO (1 byte),
Un ulteriore byte viene utilizzato come indicatore di cancellazione.

D1) Si assuma che il file dati di IMPIEGATO sia ordinato con il campo chiave SSN e che si è creato un indice primario su SSN. Calcolare:
- il fattore di blocco dell'indice bfri
- il numero delle voci e il numero di blocchi dell'indice del primo livello;
- il numero di livelli necessari se si vuole passare a un indice multilivello;
- il numero totale di blocchi richiesti dall'indice multilivello;
- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice primario
- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice multilivello

D2. Si supponga che il file non sia ordinato, e si voglia creare un indice secondario su SSN.
- il fattore di blocco dell'indice bfri
- il numero delle voci e il numero di blocchi dell'indice secondario;
- il numero di livelli necessari se si vuole passare a un indice multilivello;
- il numero totale di blocchi richiesti dall'indice multilivello;
- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice secondario
- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice multilivello

Esercizi: Ex.2 INDICI PRIMARI/SECONDARI/MULTILIVELLO

D1) Si assuma che il file dati di IMPIEGATO sia ordinato con il campo chiave SSN e che si è creato un indice primario su SSN. Calcolare:

- il fattore di blocco dell'indice bfri

  Dimensione voce= SSN+ Puntatore Blocco= 9 + 6= 15 byte

  Fattore blocco dell'indice: bfri= |_ 512/15 _| = 34

- il numero delle voci e il numero di blocchi dell'indice del primo livello;

  File impiegato ordinato su SSN, indice primario è sparso

  Numero voci pari al numero di blocchi dati che memorizzano i record di impiegati

  dimensione record: 58 byte

  fattore di blocco record bfr= |_ 512/58 _|= 8

  # blocchi per memorizzate file dati, b= ⌈100.000/8⌉= 12.500

  # voci indice primario su SSN 12.500

  # blocchi per memorizzare indice primo livello: ⌈12.500/34⌉= 368

- il numero di livelli necessari se si vuole passare a un indice multilivello;

  - primo livello, indice primario su SSN
  - secondo livello:

    indice per un file ordinato (indice primario)

    numero voci secondo livello:

    368 (una per ogni blocco dell'indice primario – indice sparso)

    numero blocchi per secondo livello: ⌈368/34⌉= 11

    serve un altro livello?

    Si, per indicizzare i 368 voci nei 11 blocchi
  - terzo livello:

    indice per un file ordinato (anche il secondo livello è ordinato)

    numero voci terzo livello:

    11 (una per ogni blocco dell'indice di secondo livello – indice sparso)

    numero blocchi per terzo livello: ⌈11/34⌉= 1

    serve un altro livello?

    No. 11 voci si indicizzano con un blocco solo

- il numero totale di blocchi richiesti dall'indice multilivello;

  # blocchi primo livello + # blocchi secondo livello + # blocchi terzo livello= 368+11+ 1= 380

- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice primario

  ricerca binaria sul indice primario: Log2(368)= 9 + accesso al blocco dati: 1

  tot 9+1=10

- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice multilivello

  un accesso per ogni livello + accesso al blocco dati: 3+1=4

Esercizi: Ex.2 INDICI PRIMARI/SECONDARI/MULTILIVELLO

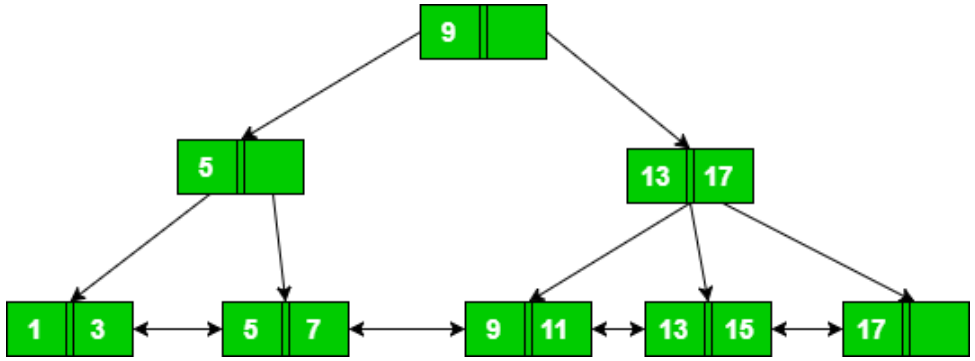
D2. Si supponga che il file non sia ordinato, e si voglia creare un indice secondario su SSN.
- il fattore di blocco dell'indice bfri: come prima
- il numero delle voci e il numero di blocchi dell'indice secondario;
  il file impiegato non è ordinato, l'indice è denso.
  # voci pari al numero record: 100.000
  # blocchi per memorizzare indice secondario: ⌈100.000/34⌉= 2.942
- il numero di livelli necessari se si vuole passare a un indice multilivello;
  - primo livello, indice secondario su SSN
  - secondo livello: Indice per un file ordinato (indice secondario è ordinato)
    numero voci secondo livello: 2942 (una per ogni blocco dell'indice secondario – indice sparso)
    numero blocchi per secondo livello: ⌈2942/34⌉= 87
    serve un altro livello? Si, per indicizzare i 2942 voci nei 87 blocchi
  - terzo livello: indice per un file ordinato (anche il secondo livello è ordinato)
    numero voci terzo livello: 87 (una per ogni blocco dell'indice di secondo livello – indice sparso)
    numero blocchi per terzo livello: ⌈87/34⌉= 3
    serve un altro livello?
    si. 87 voci si indicizzano con 3 blocchi
  - quarto livello: indice per un file ordinato
    numero voci quarto livello: 3 (una per ogni blocco dell'indice di terzo livello – indice sparso)
    numero blocchi per quarto livello: ⌈3/34⌉= 1
    serve un altro livello? No. 3 voci si indicizzano con 1 blocchi
- il numero totale di blocchi richiesti dall'indice multilivello;
  # blocchi primo livello + # blocchi secondo livello + # blocchi terzo livello + # blocchi quarto livello= 2.942+87+ 3 + 1= 3.033
- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice secondario
  ricerca binaria sul indice primario: Log2(2942)= 12
  accesso al blocco dati: 1
  tot 12+1= 13
- il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di SSN, usando l'indice multilivello
- un accesso per ogni livello + accesso al blocco dati: 4+1=4

B1. Si consideri un B+-tree con 5 valori di chiave al massimo nei nodi. Qual'è il minimo numero di valori di ricerca nei nodi?

In un B-tree di ordine p, ogni nodo può avere al massimo p puntatori e p-1 valori di chiave. Con 5 valori di chiavi, l'ordine è p=5+1=6. Inoltre, in ogni nodo il numero minimo di puntatori è p/2= 3, quindi 2 valori di chiave

B2- Si consideri il B+-tree rappresentato in figura. Quanti nodi devono essere letti per eseguire la ricerca dei record con campo chiave >=7 e <=15?

5



Esercizi: Ex.4 progettazione fisica

Si considera una tabella R(A,C, ...) con 1.000.000 di record di lunghezza fissa di 100 byte. In R, A è campo chiave. La dimensione di A è 4 byte, la dimensione di C è 4 byte. La memoria secondaria è organizzata in blocchi di dimensione B = 1000 byte.

Considerare il seguente carico di lavoro:

Op1. inserimento di un record (con verifica del vincolo di chiave), con frequenza giornaliera f1 = 1;

Op2. ricerca di un record sulla base del valore della chiave A, con frequenza giornaliera f2 = 1000;

Op3. ricerca di un record sulla base del valore dell'attributo C, con frequenza giornaliera f3 = 10.000;

Assumere, inoltre, che di notte il sistema esegue la riorganizzazione delle strutture fisiche

Definire i costi giornalieri valutando le seguenti soluzioni

(A) Struttura primaria ordinata sull'attributo A, con eventuali inserimenti effettuati in coda al file

(B) Struttura primaria non ordinata con indice B^+-tree sull'attributo A

(C) Struttura primaria ordinata sull'attributo A con indice indice B^+-tree su campo C, con eventuali inserimenti effettuati in coda al file

(D) Struttura primaria non ordinata, con indice B^+-tree sull'attributo A e indice B^+-tree sull'attributo C

Si assuma, indici B^+-tree sono usati con fattore di riempimento a 70%, un puntatore P al blocco occupa 6 byte, un puntatore Pr al record occupa 7 byte. Come costo stimare il tempo di esecuzione in termini di accessi alla memoria secondaria.

Esercizi: Ex.4 progettazione fisica

(A) Struttura primaria ordinata sull'attributo A, con eventuali inserimenti effettuati in coda al file

Op1. inserimento di un record (con verifica del vincolo di chiave), con frequenza giornaliera f1 = 1;

- la verifica del vincolo di chiave richiede una ricerca per verificare se esiste un record con il campo chiave uguale a quello del nuovo record da inserire

- il file dati è ordinato, si esegue una ricerca binaria sui blocchi dati.

  - # blocchi dati: fattore di blocco= 1000/100=10

  # blocchi: 1.000.000/10= 100.000

  - ricerca binaria: log2(100.000) = 17

Op1. 17 accessi per la ricerca, f1=1 ==> 17 accessi

Op2. ricerca di un record sulla base del valore della chiave A, con frequenza giornaliera f2 = 1000

ricerca binaria sul file dati (come sopra): 17 accessi

Op2. 17 accessi, f2=1000 ==> 17*1.000= 17.000 accessi

Op3. ricerca di un record sulla base del valore dell'attributo C, con frequenza giornaliera f3 = 10.000

Il file non è ordinato sull'attributo C, quindi la ricerca è sequenziale. In media, è necessario accedere a metà dei blocchi:

ricerca sequenziale: 100.000/2= 50.000

Op3. 50.000 accessi, f3=10.000 ==> 50.000*10.000= 500.000.000

(B) Struttura primaria non ordinata con indice B$^+$-tree sull'attributo A

Definisco B$^+$-tree sull'attributo A:

Calcolo $p_{interno}$ massimo su blocco di 1.000 byte: Struttura $p_{interno}$ (p-1) chiavi + p puntatori al blocco

(p-1)*A + p*P <=1.000
(p-1)*4 + p*6 <=1.000
4p -4 + 6p <=1.000
10p <=1004 p= 100

Calcolo p_{foglia} massimo: Struttura p_{foglia} p chiavi + p puntatori al record + Puntatore blocco

p*A + pPr + P <=1.000
4p + 7p + 6 <=1.000
11p <= 994 p= 90

Con riempimento 70%: $p_{interno}$=100*0.70= 70 p_{foglia}=90*0.70= 63

Livello	#nodi	#valori di chiavi	#puntatori record
Root	1	(70-1)=69	-
1 liv.	70	70 x 69= 4830	-
2 liv.	70^2=4.900	4900 x 69 = 338.100	-
3 liv./foglie	70^3 =343.000	343.000 x 63= 21.609.000	21.609.000

Op1. inserimento di un record (con verifica del vincolo di chiave), con frequenza giornaliera f1 = 1;
la verifica del vincolo di chiave richiede una ricerca per verificare se esiste un record con il campo chiave uguale a quello del nuovo record da inserire:

Ricerca su B$^+$-tree: 4 accessi (root + 3 livelli)

Op1. 4 accessi per la ricerca, f1=1 ==> 4 accessi

Op2. ricerca di un record sulla base del valore della chiave A, con frequenza giornaliera f2 = 1000

Ricerca su B$^+$-tree: 4 accessi (root + 3 livelli)

Op2. 4 accessi per la ricerca, f2=1000 ==> 4.000 accessi

Op3. ricerca di un record sulla base del valore dell'attributo C, con frequenza giornaliera f3 = 10.000 Il file non è ordinato sull'attributo C, quindi la ricerca è sequenziale. I media deve accedere metà blocchi: 100.000/2= 50.000

Op3. 50.000 accessi, f3=10.000 ==> 50.000*10.000= 500.000.000

(C) Struttura primaria ordinata sull'attributo A con indice indice B$^+$-tree su campo C, con eventuali inserimenti effettuati in coda al file

Op1. inserimento di un record (con verifica del vincolo di chiave), con frequenza giornaliera f1 = 1;

Come (A): ricerca binaria sul file dati: 17 accessi

Op1. 17 accessi per la ricerca, f1=1 ==> 17 accessi

Op2. ricerca di un record sulla base del valore della chiave A, con frequenza giornaliera f2 = 1000

Come (A): ricerca binaria sul file dati: 17 accessi

Op2. 17 accessi, f2=1000 ==> 17*1.000= 17.000 accessi

Op3. ricerca di un record sulla base del valore dell'attributo C, con frequenza giornaliera f3 = 10.000

Definisco B+-tree sull'attributo C: l'attributo A e C hanno la stessa dimensione. B$^+$-tree sull'attributo C avrà gli stessi ordini e stessa profondità del B$^+$-tree costruito su A

Ricerca record su B$^+$-tree costruito sul campo C: 4 accessi (root + 3 livelli)

Op3. 4 accessi, f3=10.000 ==> 40.000

Esercizi: Ex.4 progettazione fisica

(D) Struttura primaria non ordinata, con indice B^+-tree sull'attributo A e indice B^+-tree sull'attributo C

Op1. inserimento di un record (con verifica del vincolo di chiave), con frequenza giornaliera f1 = 1;

come (B): la verifica del vincolo di chiave richiede una ricerca per verificare se esiste un record con il campo chiave uguale a quello del nuovo record da inserire

Ricerca su B^+-tree, 4 accessi (root + 3 livelli)

Op1. 4 accessi per la ricerca, f1=1 ==> 4 accessi

Op2. ricerca di un record sulla base del valore della chiave A, con frequenza giornaliera f2 = 1000

Come (B) - Ricerca su B^+-tree, 4 accessi (root + 3 livelli)

Op1. 4 accessi per la ricerca, f2=1.000 ==> 4.000 accessi

Op3. ricerca di un record sulla base del valore dell'attributo C, con frequenza giornaliera f3 = 10.000

Come (C): Definisco B+-tree sull'attributo C: l'attributo A e C hanno la stessa dimensione. B+-tree sull'attributo C avrà gli stessi ordini e stessa profondità

Ricerca record su B+-tree costruito sul campo C: 4 accessi (root + 3 livelli)

Op3. 4 accessi, f3=10.000 ==> 40.000

TABELLA RIASSUNTIVA

	A	B	C	D
Op1	17	4	17	4
Op2	17.000	4.000	17.000	4.000
Op3	500.000.000	500.000.000	40.000	40.000