PARTE II

3) Supponiamo di avere un file di 700.000 record. Ogni record occupa 180 byte, di cui 60 per il campo chiave. Ogni blocco contiene 2048 byte. Un puntatore a blocco occupa 5 byte. Non richiesto l'allineamento dei campi ad indirizzi multipli di 4 e i record non sono puntati. Usiamo una organizzazione B-tree. Qual il numero minimo di blocchi che dobbiamo utilizzare complessivamente per il file indice (totale dei blocchi di tutti i livelli) e per il file principale (livello foglia)?

---

4) Sia dato il seguente schedule S per l'insieme di transazioni T1, T2, T3, T4, T5:

T1	T2	T3	T4	T5
LOCK A				
	LOCK B			
UNLOCK A				
	UNLOCK B			
	LOCK A			
		LOCK B		
	UNLOCK A			
				LOCK A
		UNLOCK B		
				UNLOCK A
		LOCK A		
			LOCK B	
		UNLOCK A		
			UNLOCK B	
LOCK B				
			LOCK A	
			UNLOCK A	
UNLOCK B				
				LOCK B
				UNLOCK B

Verificare se lo schedule e' serializzabile, e in questo caso fornire lo schedule seriale equivalente (o gli schedule seriali equivalenti se ne esiste piu' di uno).

SOLUZIONI PARTE II

Esercizio 3

- Il numero minimo di blocchi viene utilizzato quando i blocchi sono interamente pieni. Ovviamente i record del file principale hanno occupazioni diverse da quelli del file indice, che contengono solo chiave e puntatore al blocco del livello inferiore. Per quanto riguarda i record indice fa eccezione il primo record di ogni blocco, che contiene solo il puntatore. Tale puntatore

viene seguito in fase di ricerca in corrispondenza di valori minori della chiave del record successivo.

- Ricordiamo che per **comodità di calcolo** (non per definizione del B-tree) è consentito utilizzare come massimo numero di record contenuti in un blocco un numero dispari: tale approssimazione ha senso quando dobbiamo calcolare la metà della capienza del blocco (i record devono essere pieni almeno per metà) e approssimarla per eccesso (infatti nella espressione **capienza = 2d-1** il numero **d** rappresenta la parte intera superiore della metà di un numero dispari), oppure quando vogliamo calcolare in anticipo l'altezza massima che potrebbe raggiungere l'albero. In generale infatti un blocco andrebbe riempito interamente (eventualmente con un numero pari di record) e, nei casi di occupazione minima, andrebbe verificato che la metà di questo numero di record riempia effettivamente **almeno** la metà del blocco.

Calcoliamo prima di tutto il numero di record che possono essere memorizzati in un blocco del file principale:

NRF = ëcapacità blocco / taglia recordû = ë2048 / 180û = ë11,37û = 11
Si prende la parte intera **inferiore** della divisione perché i record non devono essere spezzati, e la parte intera inferiore della divisione rappresenta appunto il massimo numero di record che entrano per intero in un blocco.

Possiamo ora calcolare quanti blocchi occorrono per memorizzare il file principale

NBF = énumero record / numero record per blocco ù = é700.000 / 11ù = é63.636,36ù = 63.637
Si prende la parte intera **superiore** della divisione perché i blocchi vanno allocati interamente, cioè non allochiamo frazioni di blocco.

Calcoliamo ora quanti record indice possono essere memorizzati in un blocco, tenendo presente che il primo record di ogni blocco contiene solo un puntatore. Posiamo effettuare il calcolo in due modi:

- se **N** e' il numero di record memorizzabili, dobbiamo calcolare **N** in modo che

(N-1)*(chiave) + N*(puntatore)≤ capacità blocco
(N-1) *60 + N*5 ≤2048 N*65 - 60 ≤2048 N*65≤2048+60 N≤2108/65
N≤32,43 e poiché N deve rappresentare un numero intero di record otteniamo N=32

- possiamo considerare la capacità del blocco dopo aver già memorizzato il primo puntatore, e calcolare quanti record interi composti da chiave e puntatore possono essere ancora memorizzati; questo numero va poi incrementato di uno per tener conto del puntatore memorizzato prima

N = ë(capacità blocco - puntatore) / taglia record indice completoû + 1= ë(2048-5) / 65û + 1
N = ë2043 / 65û + 1 = ë31,43û + 1 = 31 + 1 = 32

Come al solito si prende la parte intera **inferiore** della divisione perché i record non devono essere spezzati.

Effettuato questo calcolo, si poteva specificare di assumere un numero di record dispari (ma **non è** richiesto dalla **definizione** del B-tree) e memorizzare quindi 31 record per blocco.

A questo punto cominciamo a calcolare l'occupazione in blocchi del file indice, procedendo per livelli e ricordando che ad ogni livello sono è presente un record indice per ogni blocco del livello

inferiore, quindi al primo livello di indice dobbiamo inserire un record per ogni blocco del file principale, al secondo livello un record per ogni blocco del primo, e cos via fino ad arrivare alla radice composta da un solo blocco; inoltre i blocchi devono essere allocati interamente (viene riportato il calcolo per entrambe le capienze):

capienza = 32 record:

BI1 = ⌈63.637 / 32⌉ = ⌈1988,6⌉ = 1989 blocchi
BI2 = ⌈1.989 / 32⌉ = ⌈62,15⌉ = 63 blocchi
BI3 = ⌈63/ 32⌉ = ⌈1,96⌉ = 2 blocchi
BI4 = ⌈2 / 32⌉ = 1 blocco

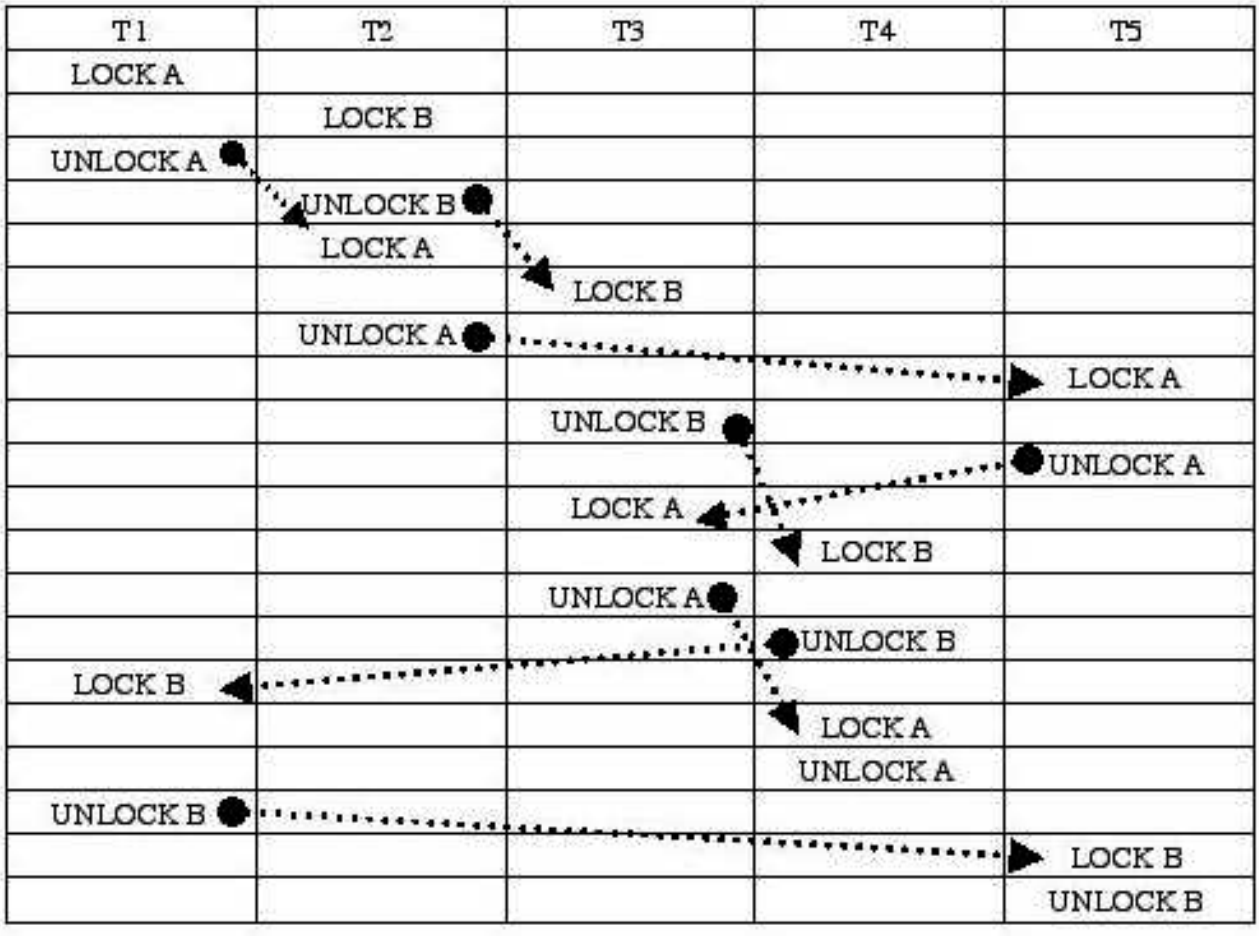
Occupazione totale indice 1989+63+2+1 = 2055

capienza = 31 record:

BI1 = ⌈63.637 / 31⌉ = ⌈2052,8⌉ = 2053 blocchi
BI2 = ⌈2053 / 31⌉ = ⌈66,22⌉ = 67 blocchi
BI3 = ⌈67/ 31⌉ = ⌈2,16⌉ = 3 blocchi
BI4 = ⌈3 / 31⌉ = 1 blocco

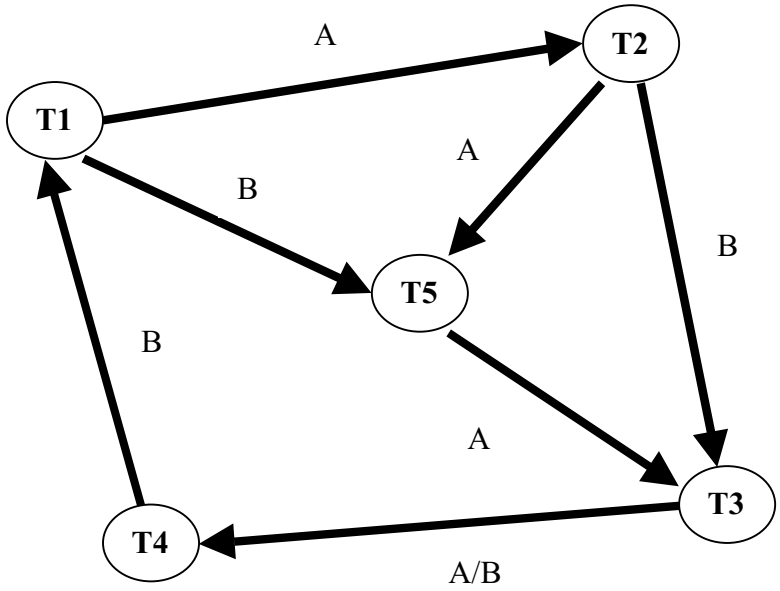
Occupazione totale indice 2053+67+3+1 = 2124

Esercizio 4



Applichiamo l'algoritmo per costruire il grafo di serializzazione: i nodi corrispondono alle transazioni e che esiste un arco tra il nodo T_i ed il nodo T_j se la transazione T_j la prima nello schedule ad effettuare un lock su un item su cui T_i ha effettuato un unlock. Per comodit possiamo visualizzare queste situazioni sulla tabella dello schedule, come fatto nella figura sopra.

Costruiamo a questo punto il grafo, e per chiarezza riportiamo su ogni arco l'item che porta all'inserimento dellàrco stesso.



Il grafo presenta vari cicli, tra cui quello che coinvolge i nodi T1, T2, T3, T4. Possiamo quindi concludere che lo schedule dato non serializzabile.