Soluzioni proposte degli esercizi sulla organizzazione fisica

1.

Abbiamo i seguenti dati:

il file contiene 250.000 record: NR = 250.000

ogni record occupa 300 byte: R = 300

il campo chiave occupa 75 byte: K = 75

ogni blocco contiene 1024 byte: CB = 1024

un puntatore a blocco occupa 4 byte: P = 4

a) Indichiamo con B il numero di bucket e con BD il numero di blocchi per la bucket directory. La bucket directory è essenzialmente un array di puntatori indicizzato da 0 a B-1. Allora ci occorrono B × P byte, ossia 1200 × 4 = 4800 bytes. Poiché ogni blocco contiene 1024 byte, ci occorrono BD = é4800/1024ù = 5 blocchi per la bucket directory (prendiamo la parte intera superiore perché i record devono essere interi, non essendo stato specificato diversamente dall'esercizio, e i blocchi allocati interamente, e quindi la frazione di blocco va arrotondata ad un blocco intero); siamo comunque sicuri che ogni blocco conterrà un numero intero di puntatori perché CB (=1024) è multiplo di P (=4);

b) Abbiamo record a lunghezza fissa, quindi supponiamo di non avere un direttorio di record all'inizio del blocco. Serve però un puntatore per linkare i blocchi dello stesso bucket. In un blocco dobbiamo quindi memorizzare il maggior numero possibile di record e in più un puntatore per un eventuale prossimo blocco nel bucket. Se indichiamo con M il massimo numero di record memorizzabili in un blocco, avremo $M \times R + P \leq 1024$, cioè $300M + 4 \leq 1024$, quindi $M \leq 1020/300 = 3{,}4$. M deve essere intero, perché non essendo stato detto altrimenti nella traccia, assumiamo che i record non possano trovarsi a cavallo di due o più blocchi, quindi assumiamo M = 3. Se la distribuzione dei record nei bucket è uniforme, indicando con RB il numero di

record in un bucket, avremo RB = é NR/B ù = é 250.000/1200 ù = 209 record per ogni bucket (prendiamo la parte intera superiore perché i record devono essere inseriti tutti, quindi la frazione di record va considerata per non tralasciare alla fine una parte dei record stessi). Indicando con NB il numero di blocchi per ogni bucket, occorrono quindi NB = é RB/M ù = é 209/3 ù = 70 blocchi per ogni bucket; indicando con BB il numero complessivo di blocchi per il file hash, avremo BB = NB × B = 70 × 1200 = 84.000 blocchi;

c) Se la distribuzione dei record nei bucket è uniforme, in un bucket avremo come detto NB = é RB/M ù = 70 blocchi per bucket. Poiché la ricerca avviene solo sul bucket individuato in base al risultato dell'applicazione della funzione hash alla chiave del record, avremo un numero di accessi a blocco pari a quello che si avrebbe su un heap della stessa dimensione del bucket (cioè 1/B rispetto alla dimensione originale). In media accederemo alla metà di questi blocchi, quindi indicando con MA il numero medio di accessi, avremo MA, pari a é NB/2 ù. Nel nostro caso quindi MA = é 70/2 ù = 35; a questi occorrerà aggiungerne 1 se il blocco della bucket directory relativo al bucket in cui si trova il record non si trova già in memoria principale;

d) per avere un numero di accessi a blocco inferiore o al massimo uguale a 10, riscriviamo l'espressione di MA in modo che vi compaia esplicitamente il numero di bucket B, e tralasciando per semplicità gli arrotondamenti che abbiamo effettuato via via nei calcoli tranne l'ultimo. Avremo MA = é NB/2 ù = é (RB/M)/2 ù = é (NR/B)/M/2 ù = é NR/2(B × M) ù. Vogliamo calcolare quindi B in modo che é NR/2(B × M) ù ≤ 10, cioè deve essere B ≥ NR/20M (l'inversione della disuguaglianza deriva da fatto che il numero medio di accessi decresce con l'aumentare del numero di bucket). Nel nostro caso dovremo avere allora B ≥ 250.000/(20× 3) = 4167. Verifichiamo infatti che in questo caso avremo RB = é NR/B ù = é 250.000/4167 ù = 60 record per ogni bucket, quindi NB = é RB/M ù = 20 record per bucket, e infine MA = é NB/2 ù = 10 accessi a

blocco. Si poteva anche ragionare in un altro modo. Siccome MA = é NB/2 ù, per avere MA ≤ 10 dobbiamo fare in modo che sia NB ≤ 20 (ricordiamo che NB è il numero di blocchi in un bucket). Dobbiamo allora avere RB = M × NB ≤ M × 20, cioè nel nostro caso RB ≤ 60 (ricordiamo che RB è il numero di record per bucket). Per avere un numero di record per bucket inferiore a 60, deve essere NR/B ≤ 60, e quindi B ≥ (250.000/60), ottenendo lo stesso risultato.

2.

Abbiamo i seguenti dati:
il file contiene 150.000 record: NR = 150.000
ogni record occupa 250 byte: R = 250
il campo chiave occupa 50 byte: K = 50
ogni blocco contiene 1024 byte: CB = 1024
un puntatore a blocco occupa 4 byte: P = 4

a) I record sono di taglia fissa, quindi non occorrono puntatori all'inizio del blocco; inoltre, non essendo stato altrimenti specificato nella traccia, assumiamo che un record non possa superare i limiti di un blocco (quindi ogni blocco contiene un numero intero di record). Poiché i record non sono puntati, possono essere spostati, e quando un blocco si riempie, ne allochiamo uno nuovo che comporterà un aggiornamento dell'indice, con l'aggiunta di un nuovo record indice che punti al nuovo blocco. Di conseguenza, nei blocchi dati non occorre un puntatore al prossimo blocco. L'esercizio fornisce il fattore di occupazione attuale dei blocchi del file di dati, ma non indica un fattore di occupazione per i blocchi indice, quindi possiamo assumere che siano pieni. Per prima cosa dobbiamo stabilire quanti record indice occorrono, quindi sapendo che l'indice contiene un record per ogni blocco del file di dati, dobbiamo calcolare quanti sono questi blocchi. Sappiamo che i blocchi dati sono

occupati al 70% (indichiamo questa quantità con POD), quindi prima di tutto vediamo quanti record interi possono essere contenuti al massimo nella porzione indicata di blocco. Indicando con M questo numero, avremo che M = (CB × POD)/R, quindi nel nostro caso M = (1024 × 70/100)/250 = 2,87. Poiché M deve essere intero, avremo M = 2. Il numero di blocchi da indicizzare, indicato con BF, sarà quindi BF = éNR/Mù = é150.000/2ù = 75.000. Abbiamo bisogno quindi di 75.000 record indice. Ogni record indice, composto da chiave e puntatore al blocco, occupa RI = K + P byte, cioè RI = 54 byte. Dobbiamo ora calcolare quanti record indice interi possono essere contenuti in un blocco pieno (se anche in questo caso fosse precisata una percentuale di utilizzo, il calcolo procederebbe in maniera analoga a quella usata per i blocchi dati). Indicando con MI questo numero, avremo MI = ëCB/RI û= ë1024/54û= 18 record indice per blocco. In definitiva, indicando con BI il numero di blocchi indice, avremo BI = éBF/MIù = é75.000/18ù = 4167 blocchi indice; è da **notare** che il calcolo basato sul numero complessivo di byte necessari per l'indice RI × NR diviso per la capacità CB di ogni blocco darebbe un numero di blocchi pari a é(54 × 75.000)/1024ù = 3956, ma presenterebbe un errore di fondo, in quanto non assicurerebbe che ogni record sia contenuto interamente in un blocco;

b) quando i record del main file sono puntati, una volta inseriti non possono essere spostati per mantenere l'ordinamento (quindi in pratica l'ordinamento stretto dei record vale solo all'inizializzazione). Viceversa, a differenza da quanto accade per l'ISAM classico, i record dei blocchi indice non vengono mai modificati, quindi ciò che rimane valido è la ripartizione degli intervalli delle chiavi. In altre parole, se un record indice punta ad un'area di dati con valori di chiave comprese tra k1 e k2, questa condizione deve rimanere valida. Se quindi il blocco originario si riempie, e arrivano nuovi record con valori di chiave compresi sempre tra k1 e k2, dobbiamo allocare un nuovo blocco che però, anzichè essere puntato dall'indice, sarà linkato al blocco originario, e così per ogni nuovo blocco che si riempie (abbiamo cioè una lista di blocchi di overflow che partono da quello originario, dove ognuno punta al

successivo). Di conseguenza ogni blocco originario del main file deve essere visto come potenzialmente il primo di un bucket, e quindi dobbiamo anche prevedere spazio per un puntatore. L'esercizio ci dice in questo caso che i blocchi dati sono pieni, e lo stesso possiamo assumere per i blocchi indice, visto che non è specificato altrimenti. Altra assunzione che possiamo fare, visto che non è stato specificato altrimenti, è che non ci siano liste di overflow, cioè che tutti i record di dati siano contenuti in blocchi puntati direttamente dall'indice. Vediamo prima di tutto in queste nuove condizioni quanti record interi di dati entrano al massimo in un blocco. Indicando questo numero con MP avremo la condizione $(MP \times R) + P \leq CB$, quindi nel nostro caso $(MP \times 250) + 4 \leq 1024$, cioè $MP \leq 1020/250 = 4{,}08$. Poiché MP deve essere intero avremo MP = 4. Vediamo allora quanti blocchi vanno indicizzati (nel caso di non avere liste di overflow). Indicando questo numero con BF avremo $BF = \lceil NR/MP \rceil = \lceil 150.000/4 \rceil = 37.500$. A questo punto calcoliamo il numero di blocchi indice. La struttura dl blocco indice è identica a quella del caso precedente, quindi ricordiamo che la taglia del record indice è RI = 54 byte , che il numero massimo di record indice per blocco è MI = 18 e quindi occorrono $BI = \lceil BF/MI \rceil = \lceil 37.500/18 \rceil = 2084$ blocchi indice;

c) in entrambi i casi la ricerca si effettua prima di tutto sui blocchi indice; nel caso in cui effettuiamo una ricerca binaria, dimezziamo ad ogni passo l'insieme di **blocchi** in cui cercare la chiave che ricopra quella voluta, fino a ricondurci a cercare in un unico blocco indice; a questo punto i due casi si differenziano, perché nel caso di record non puntati dobbiamo ancora leggere in memoria un solo blocco di record di dati, mentre nel caso di record puntati l'indice punta in realtà, come abbiamo visto, ad un bucket che potrebbe contenere più blocchi di overflow, che vanno tutti esaminati. Nel nostro caso però l'esercizio dice esplicitamente che non ci sono liste di overflow, quindi in entrambe le configurazioni dobbiamo aggiungere un solo accesso a blocco, quindi nella configurazione a) avremo $A = \lceil \log_2 BI \rceil + 1 = \lceil \log_2 4167 \rceil + 1 = 14$, mentre nella configurazione b) avremo $A = \lceil \log_2 BI \rceil + 1 = \lceil \log_2 2084 \rceil + 1 = 13$.

3.

Abbiamo i seguenti dati:

il file contiene 170.000 record: NR = 170.000

ogni record occupa 200 byte: R = 200

il campo chiave occupa 20 byte: K = 20

ogni blocco contiene 1024 byte: CB = 1024

un puntatore a blocco occupa 4 byte: P = 4

a) Poiché abbiamo record di taglia fissa, non abbiamo bisogno di un direttorio dei record all'inizio del blocco. Inoltre nei blocchi dati non abbiamo bisogno di puntatori al prossimo blocco, quindi possiamo sfruttare tutto lo spazio per i dati. In un B-tree, per comodità, il numero massimo dei record contenuti nei blocchi viene assunto essere dispari, cioè del tipo 2K-1, e per definizione ogni blocco è sempre pieno almeno per metà. Per semplicità questo controllo si fa basandosi sulla parte intera superiore della metà dei record possibili, quindi K. (**NOTA:** K potrebbe rappresentare un numero di record che non riempie la metà dei byte del blocco, ma il calcolo effettuato ragionando sui byte sarebbe più complesso**).** Per quanto riguarda i blocchi di dati nel livello foglia indichiamo questo numero con 2E-1: per quanto detto prima avremo $2E\text{-}1 \leq \lfloor CB/R \rfloor = \lfloor 1024/200 \rfloor = 5$. Il $\leq$ è dovuto al fatto che se il risultato fosse un numero pari, prenderemmo il numero dispari immediatamente precedente, ma nel nostro caso possiamo assumere 2E-1 = 5. Questo significa che, per definizione, un blocco dati non conterrà mai meno di E = 3 record, e comunque non può contenerne più di 2E-1 = 5. L'esercizio indica che i blocchi sono pieni al minimo, quindi ogni record contiene E record. A questo punto possiamo calcolare il numero di blocchi nel livello foglia, che sarà $BF = \lceil NR/E \rceil = \lceil 170.000/3 \rceil = 56.667$. Il primo livello di indice contiene un record per ognuno di questi blocchi. A questo punto dobbiamo calcolare la capacità dei blocchi indice. Nei livelli indice del B-tree, ogni record, eccetto il primo, contiene una coppia (chiave, puntatore); il primo record invece contiene solo un puntatore (relativo al blocco dati che contiene i record con

chiave minore della chiave contenuta nel record indice successivo). Il numero complessivo di record deve essere comunque dispari, quindi indicando con 2D-1 tale numero dovremo avere (2D-2) × K + (2D-1) × P ≤ CB (ricordiamo che abbiamo una chiave in meno), quindi nel nostro caso (2D-2) × 20 + (2D-1) × 4 ≤ 1024, cioè esplicitando i passaggi di calcolo

(2D-1) × 20 – 20 + (2D-1) × 4 ≤ 1024, (2D-1) × 24 ≤ 1044, 2D-1 ≤ ë1044/24û = 43.

Analogamente al caso precedente il numero ottenuto è dispari, quindi possiamo assumere 2D-1=43 e D=22. L'esercizio indica che anche i blocchi indice sono pieni al minimo, quindi dobbiamo sempre considerare D record per blocco. Al primo livello di indice avremo quindi B1 = éBF/Dù (che possiamo anche scrivere éNR/EDù) = é56.667/22ù = 2576 blocchi indice. Ad ognuno di questi corrisponderà un record al secondo livello di indice, quindi B2 = éB1/Dù (che possiamo anche scrivere éNR/ED^2ù) = é2576/22ù = 118 blocchi indice. Iterando il ragionamento avremo B3 = éB2/Dù (che possiamo anche scrivere éNR/ED^3ù) = é118/22ù = 6 blocchi indice. A questo punto **attenzione**: anche questi ultimi sono blocchi a cui devono corrispondere dei record indice. Arriviamo così alla radice, che deve contenere sempre un unico blocco, e per la quale si rilascia il vincolo che sia piena almeno per metà (infatti nel nostro caso conterrà solo 6 record). Abbiamo allora B4 = 1 blocco indice.

In conclusione abbiamo BF = 56.667 blocchi dati al livello foglia, e complessivamente BI = B1 + B2 + B3 + B4 = 2576 + 118 + 6 + 1 = 2701 blocchi nei 4 livelli di indice;

b) per un indice secondario il livello foglia non contiene i blocchi dati (che tipicamente saranno stati ordinati secondo la chiave primaria) ma un ulteriore livello di indice (indice denso) che contiene un record indice per ciascun record del file dati (main file), e in cui anche il primo record di ogni blocco contiene la chiave di riferimento. Inoltre ogni chiave che comparirà potrà far riferimento a più record, perché potrebbe non essere una chiave in senso stretto (cioè una chiave alternativa alla chiave primaria, ma che comunque identifica univocamente i record), ma

semplicemente un valore di un certo attributo in base al quale possiamo selezionare gruppi di record. Assumiamo di organizzare il nostro livello foglia con record che contengono la chiave e una serie di puntatori ai record del main file che presentano quel valore di chiave. Abbiamo quindi dei record di taglia variabile. Per semplicità di calcolo però non consideriamo il direttorio dei record all'inizio dei blocchi, e non preoccupiamoci del fatto che un blocco sia contenuto interamente in un record, altrimenti dovremmo fare ulteriori assunzioni sulla distribuzione dei record rispetto chiavi. Inoltre non applichiamo all'indice denso i vincoli di occupazione dei B-tree. Con queste semplificazioni, procediamo a calcolare l'occupazione dei record dell'indice denso in termini di blocchi, tenendo conto della percentuale di utilizzo data (60%). Abbiamo bisogno di 2500 campi chiave ognuno di 10 byte, che devono indicizzare complessivamente (a prescindere dalla distribuzione per ogni chiave) i 170.000 record del main file, quindi, considerando anche la percentuale PO di occupazione di blocchi, avremo BF = $\lceil (NK \times LK + NR \times PR)/(CB \times PO) \rceil$ = $\lceil (2500 \times 10 + 170.000 \times 3)/(1024 \times 60/100) \rceil$ = 872 blocchi. C'è una differenza a questo punto rispetto all'esercizio presentato in aula. In quel caso il numero di blocchi era superiore al numero di chiavi, ma potevamo puntare solo a quelli che contenevano effettivamente una chiave, e quindi avere al primo livello indice un numero di record pari al numero di chiavi. In questo caso invece il numero di blocchi è inferiore a quello delle chiavi (ogni blocco contiene effettivamente record relativi a un intervallo di chiavi), e quindi adottiamo la strategia di puntare a tutti i blocchi dell'indice denso. Ai livelli indice intermedi abbiamo la consueta struttura di record con chiave e puntatore al blocco foglia (quindi al blocco dell'indice denso), con il primo record con il solo puntatore. Impostiamo il calcolo della capacità come nella lettera a) con la lunghezza delle chiavi secondarie, quindi dobbiamo avere (2D-2) × K + (2D-1) × P ≤ CB (ricordiamo che abbiamo una chiave in meno), quindi nel nostro caso (2D-2) × 10 + (2D-1) × 4 ≤ 1024, cioè esplicitando i passaggi di calcolo

(2D-1) × 10 – 10 + (2D-1) × 4 ≤ 1024, (2D-1) × 14 ≤ 1034, 2D-1 ≤ $\lfloor 1034/14 \rfloor$ = 73

che ci dà ancora 2D-1=73 e D=37. In questo caso l'esercizio indica che i blocchi ai livelli indice sono pieni al massimo, quindi ognuno contiene 2D-1 = 73 record. Al primo livello indice ci occorrono quindi B1 = éBF/(2D-1)ù = é872/73ù = 12 blocchi. I record indice che puntano a questi 12 blocchi possono essere contenuti in un unico blocco al livello successivo, che quindi risulta essere la radice con B2 = 1 blocco. Considerando anche l'indice denso, avremo complessivamente BI = BF + B1 + B2 = 872 + 12 + 1 = 885 blocchi indice.