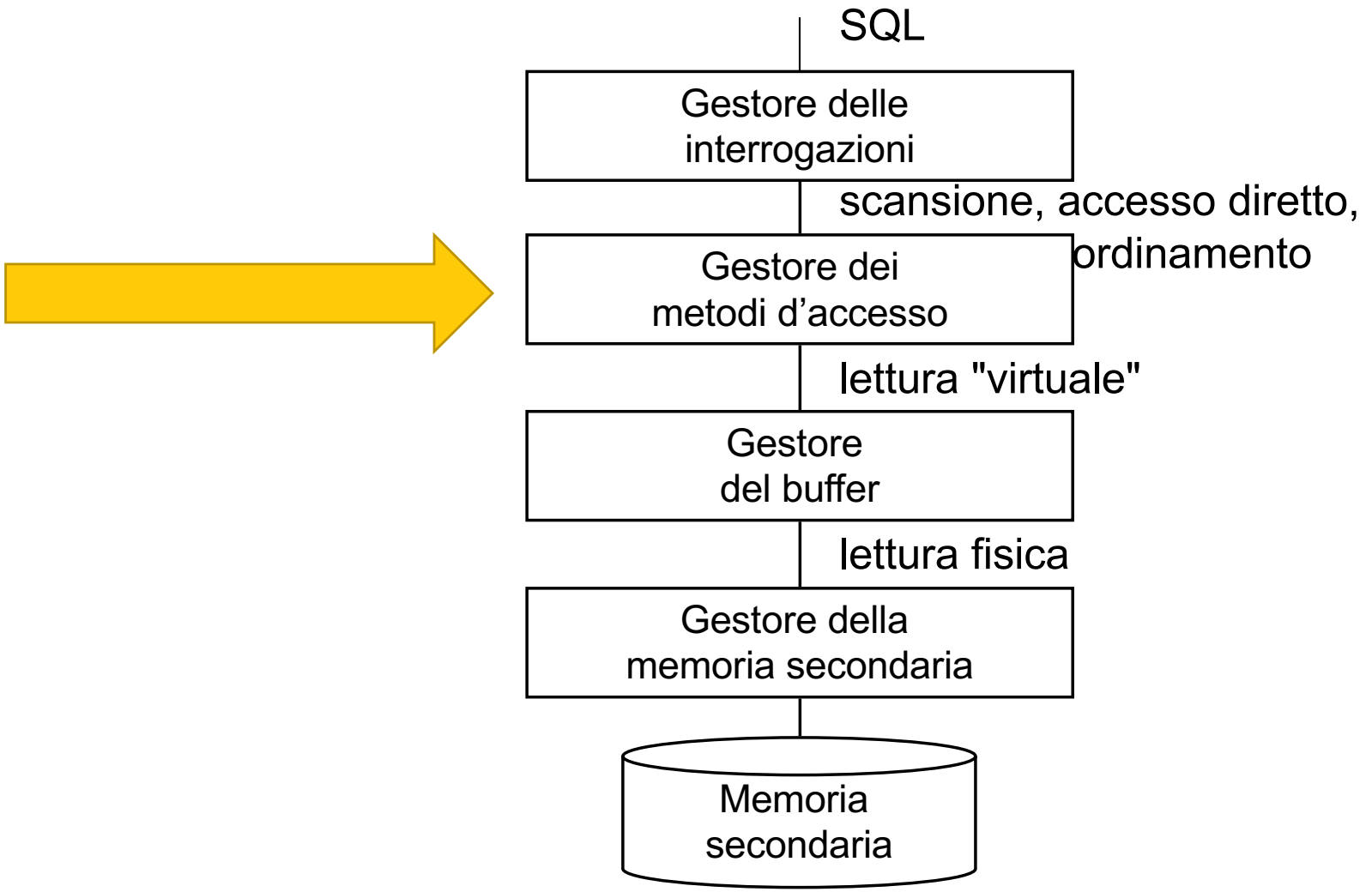
Gestore degli accessi e delle interrogazioni



Strutture ausiliari d'accesso - INDICI

- Struttura ausiliaria per l'accesso (efficiente) ai record di un file sulla base dei valori di un campo detto chiave
- INDICE: struttura d'accesso definita su campo chiave/campo di indicizzazione dove (per ogni) valore del campo chiave si memorizza il puntatore al blocco che memorizza i record con quel valore del campo
- IDEA: ordinare l'indice in base al valore del campo chiave (indici ordinati) per facilitare la ricerca
  - Analogo all'indice analitico di un libro: lista di coppie (termine, pagina), ordinata alfabeticamente sui termini
- Ordine sul campo chiave/indicizzazione permette ricerca binaria sul file indice

Indici ordinati

- Un indice $\mathcal{I}$ per un file f è un altro file con record/voce definiti su due campi:
  - Campo chiave/indicizzazione
  - Puntatore al record (numero blocco)

I record/voci dell'indice sono ordinati secondo i valori del campo chiave

In base all'organizzazione dei file dati, possiamo avere diversi tipi di indici ordinati:

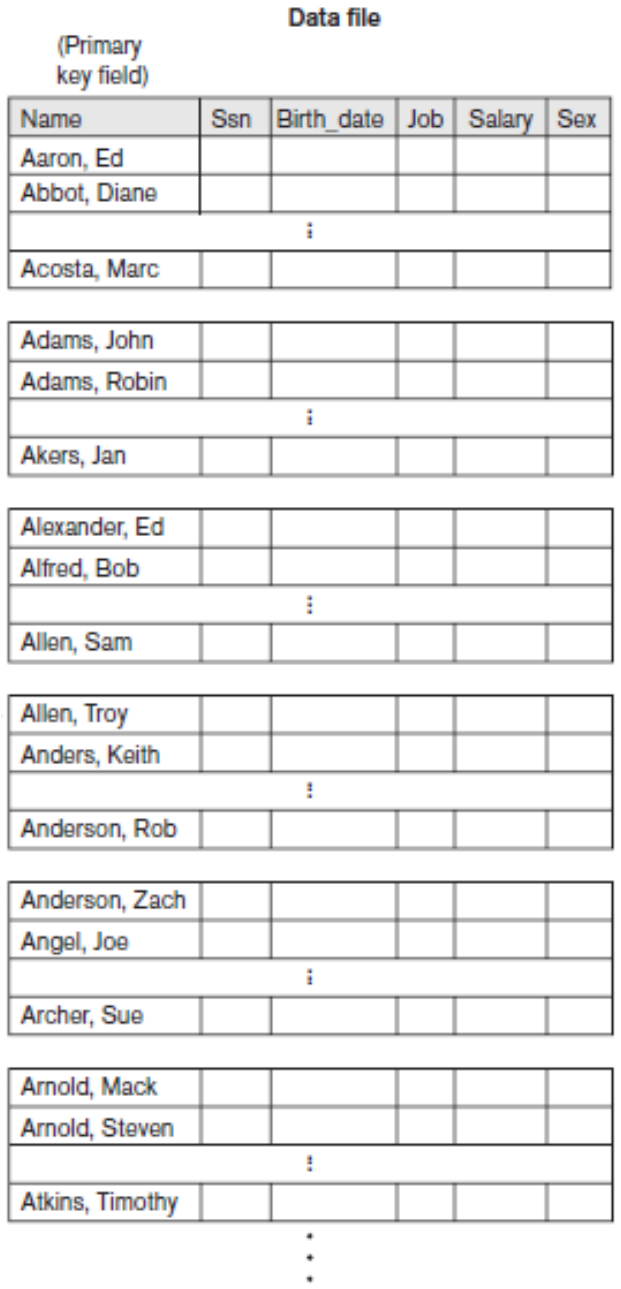
- Indici creati su campi usati per l'ordinamento fisico del file
  - Indice primario
  - Indice di clustering
- Indici creati su file non ordinati o creati su campi non usati per ordinamento del file
  - Indici secondari

INDICE PRIMARIO

- Si applica nel caso di file dati ordinato
- L'indice primario è un indice specificato sul campo chiave utilizzato nel file dati per ordinare i record
  - Il file dati è ordinato su campo K
  - Indice primario è creato su campo K
- Record/voce dell'indice primario:
  - Campo chiave (chiave primaria): stesso tipo del campo di ordinamento del file dati
  - Puntatore al blocco che memorizza il record con quel valore di campo

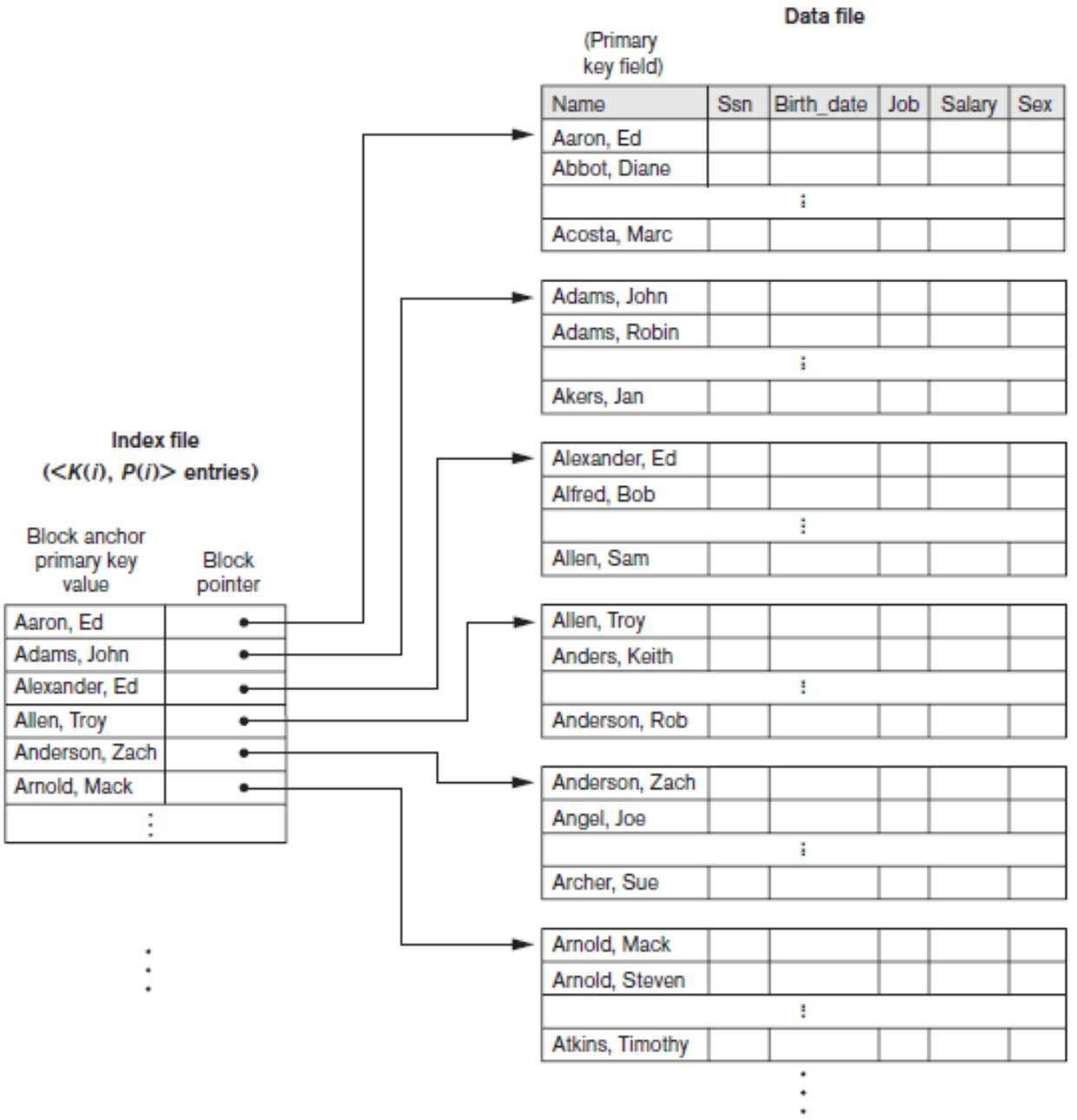
E' necessario un record/voce per ogni valore del campo chiave?

INDICE PRIMARIO



INDICE PRIMARIO

- Sfrutto l'ordinamento del file dati per avere una voce che punta al primo record di ogni blocco (o ultimo)
- Indice primario ha un numero di record/voci pari al numero di blocchi del file dati
- Indice primario è SPARSO
  - Contiene voci solo per alcuni valori del campo chiave



INDICE PRIMARIO

- Costo d'accesso?
- Un file indice per un indice primario occupa meno blocchi rispetto al file dei dati:
  - ha meno voci/record
  - ogni voce/record è più piccolo
- La ricerca binaria su un indice primario è più efficiente che una ricerca binaria su file ordinato

Esercizio: Costo ricerca in termini di numero accessi

- Si supponga un file ordinato su campo ID con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.
- Si assuma un indice primario su campo ID dove:
  - Dimensione ID è 9 byte
  - Dimensione puntatore è 6 byte

Mediamente, quanti accessi sono richiesti per una ricerca sul campo ID?

Esercizio: Costo ricerca in termini di numero accessi

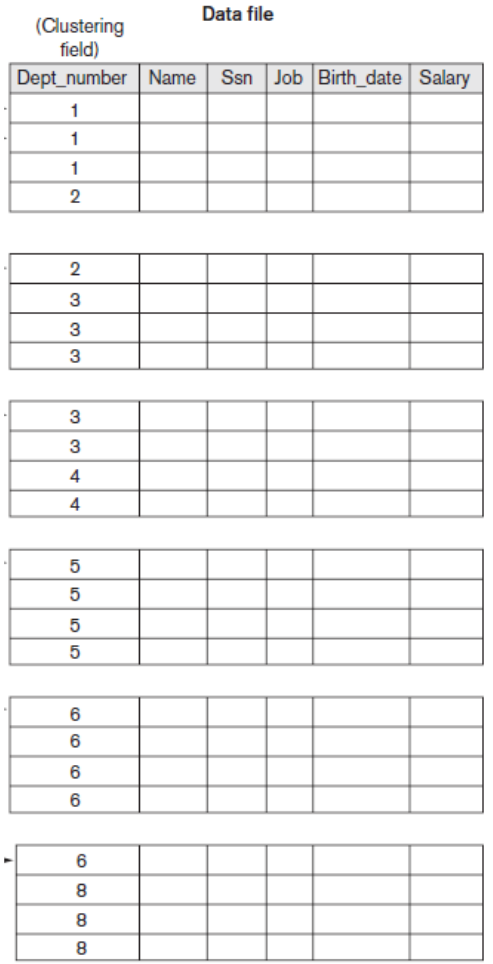
- Si supponga un file ordinato su campo ID con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.
- Si assuma un indice primario su campo ID dove:
  - Dimensione ID è 9 byte
  - Dimensione puntatore è 6 byte

Mediamente, quanti accessi sono richiesti per una ricerca sul campo ID?

- Fattore di blocco per il file dati $\lfloor 4096/100 \rfloor = 40$
- Numero blocchi per memorizzare i record: $\lceil 300.000/40 \rceil = 7500$
- Fattore di blocco per il file indice: $\lfloor 4096/15 \rfloor = 273$
- Nel file indice, una voce per ogni blocco di dati, quindi
  - Numero blocchi per memorizzare indice: $\lfloor 7500/273 \rfloor = 28$
- Numero accesso ai blocchi file indice per ricerca binaria $\lceil \log_2 28 \rceil = 5$
- N. Accessi per ricercare record 5+1

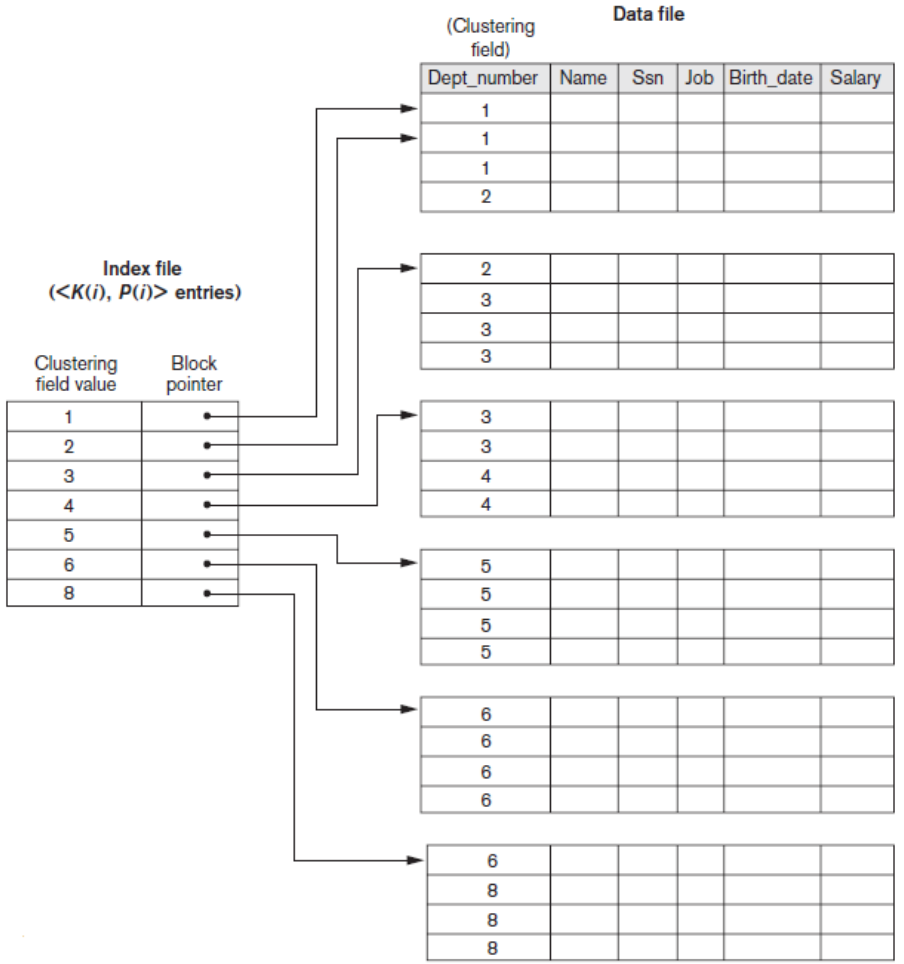
INDICE di CLUSTERING

- Si applica nel caso di file dati ordinato su un campo che non è UNIQUE/chiave primaria
  - Campo di clustering o campo di raggruppamento
  - File clustered
- Utile per velocizzare ricerca in caso di tanti record con lo stesso valore di raggruppamento



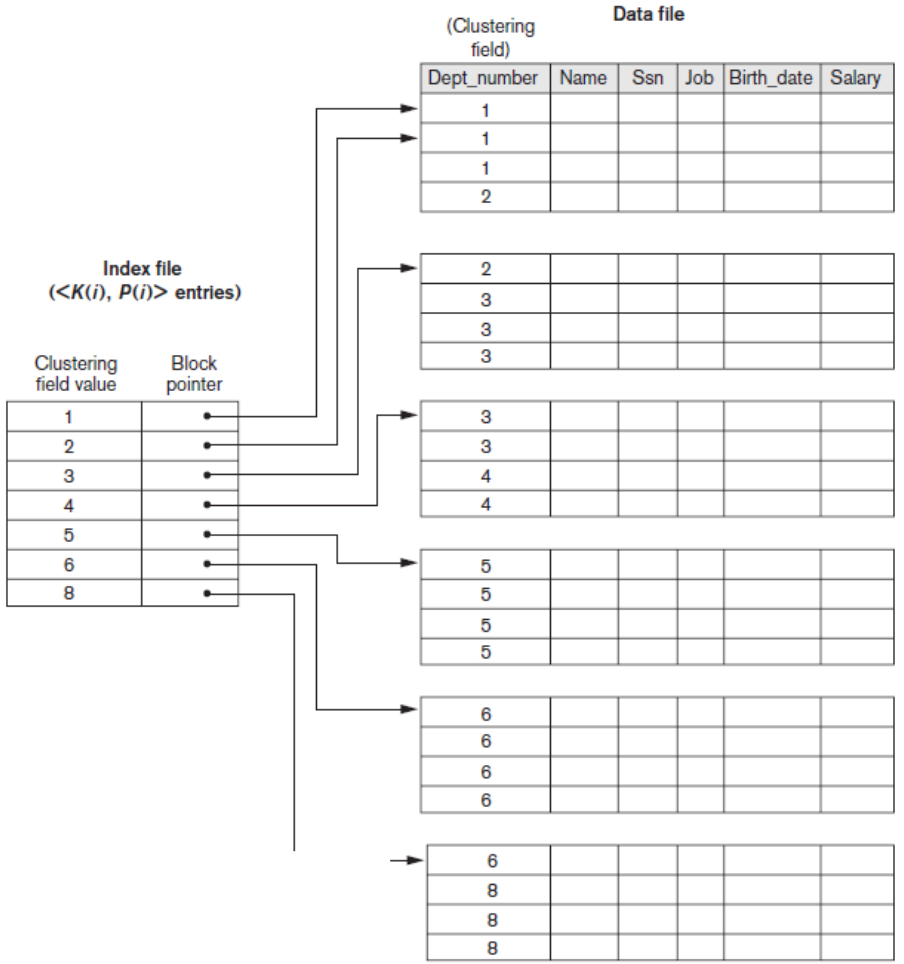
INDICE di CLUSTERING

- Si applica nel caso di file dati ordinato su un campo che non è UNIQUE/chiave primaria
  - Campo di clustering o campo di raggruppamento
  - File clustered
- Utile per velocizzare ricerca in caso di tanti record con lo stesso valore di raggruppamento
- Record nell'indice di clustering:
  - Valore del campo di raggruppamento
  - Puntatore al primo blocco che contiene il primo record con quel valore
- Numero voci nel file indice pari ai possibili valori di raggruppamento
- Indice SPARSO



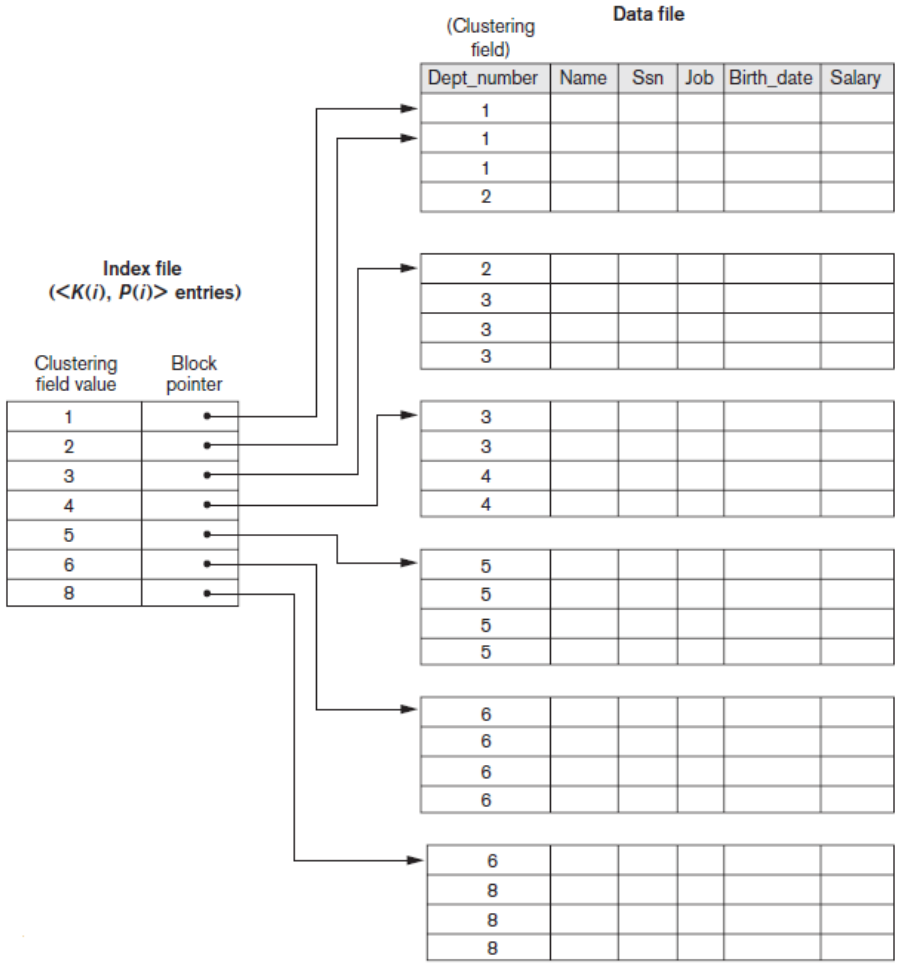
INDICE di CLUSTERING

- Si applica nel caso di file dati ordinato su un campo che non è UNIQUE/chiave primaria
  - Campo di clustering o campo di raggruppamento
  - File clustered
- Utile per velocizzare ricerca in caso di tanti record con lo stesso valore di raggruppamento
- Record nell'indice di clustering:
  - Valore del campo di raggruppamento
  - Puntatore al primo blocco che contiene il primo record con quel valore
- Numero voci nel file indice pari ai possibili valori di raggruppamento
- Indice SPARSO
- Problema: inserimento/cancellazione richiede riordino dei record



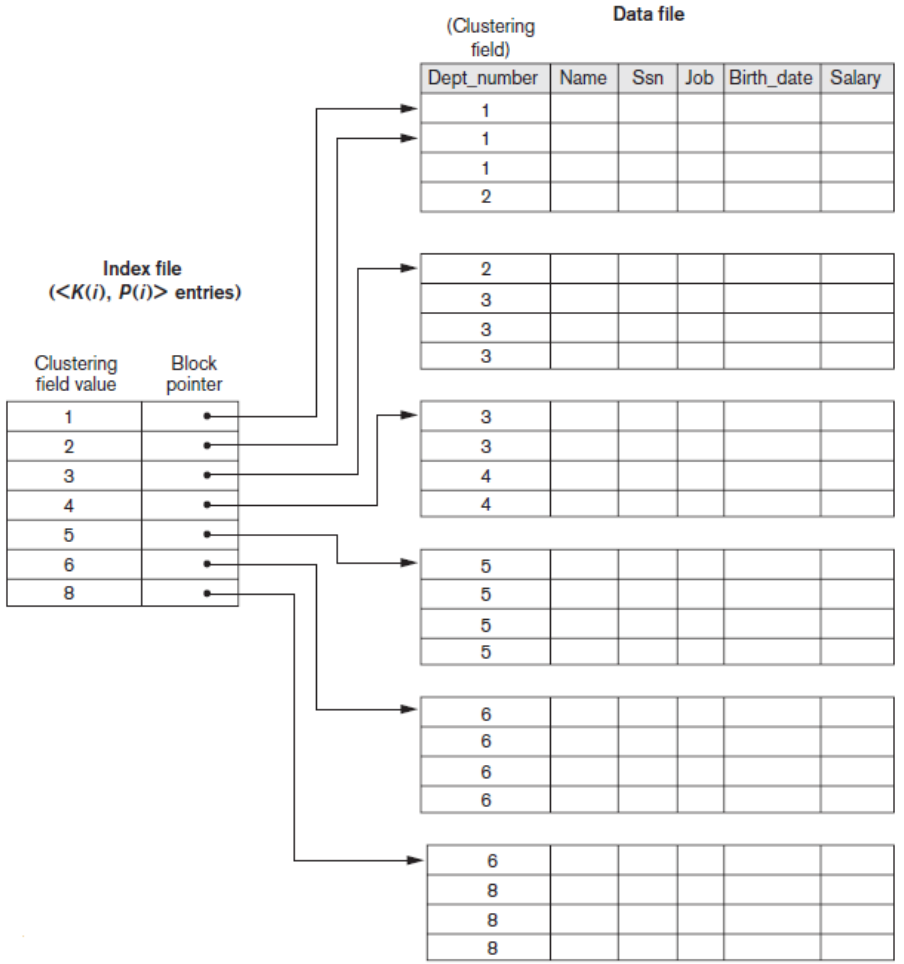
INDICE di CLUSTERING

- Si applica nel caso di file dati ordinato su un campo che non è UNIQUE/chiave primaria
  - Campo di clustering o campo di raggruppamento
  - File clustered
- Utile per velocizzare ricerca in caso di tanti record con lo stesso valore di raggruppamento
- Record nell'indice di clustering:
  - Valore del campo di raggruppamento
  - Puntatore al primo blocco che contiene il primo record con quel valore
- Numero voci nel file indice pari ai possibili valori di raggruppamento
- Indice SPARSO
- Problema: inserimento/cancellazione richiede riordino dei record
- Soluzione: un blocco differente o gruppi di blocchi per ogni valore di raggruppamento



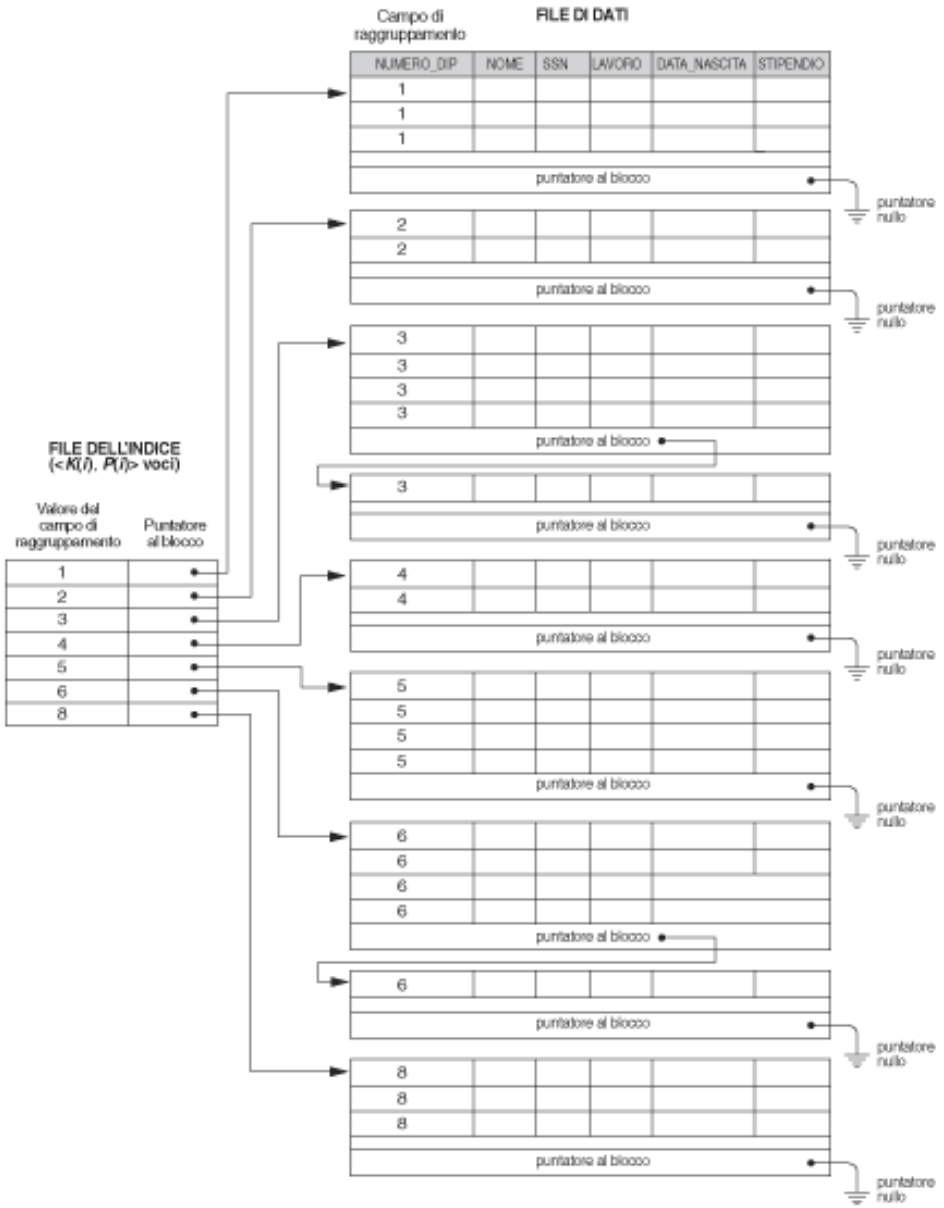
INDICE di CLUSTERING

- Si applica nel caso di file dati ordinato su un campo che non è UNIQUE/chiave primaria
  - Campo di clustering o campo di raggruppamento
  - File clustered
- Utile per velocizzare ricerca in caso di tanti record con lo stesso valore di raggruppamento
- Record nell'indice di clustering:
  - Valore del campo di raggruppamento
  - Puntatore al primo blocco che contiene il primo record con quel valore
- Numero voci nel file indice pari ai possibili valori di raggruppamento
- Indice SPARSO
- Problema: inserimento/cancellazione richiede riordino dei record
- Soluzione: un blocco differente o gruppi di blocchi per ogni valore di raggruppamento



INDICE di CLUSTERING

- Si applica nel caso di file dati ordinato su un campo che non è UNIQUE/chiave primaria
  - Campo di clustering o campo di raggruppamento
  - File clustered
- Utile per velocizzare ricerca in caso di tanti record con lo stesso valore di raggruppamento
- Record nell'indice di clustering:
  - Valore del campo di raggruppamento
  - Puntatore al primo blocco che contiene il primo record con quel valore
- Numero voci nel file indice pari ai possibili valori di raggruppamento
- Indice SPARSO
- Problema: inserimento/cancellazione richiede riordino dei record
- Soluzione: un blocco differente o gruppi di blocchi per ogni valore di raggruppamento

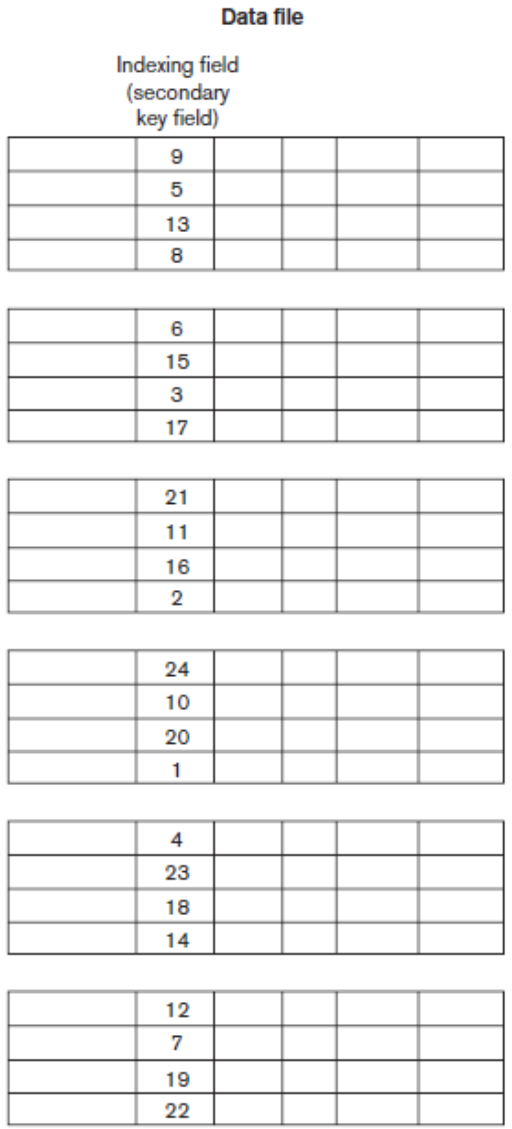


INDICE SECONDARIO

- L’indice secondario fornisce una struttura d’accesso indipendentemente dalla struttura fisica
  - Il file di dati può essere ordinato, non ordinato, hash
- E’ possibile realizzare per il medesimo file dati diversi indici secondari
- L’indice secondario si può creare su un campo del record che:
  - è chiave primaria/UNIQUE
  - non è chiave primaria/UNIQUE

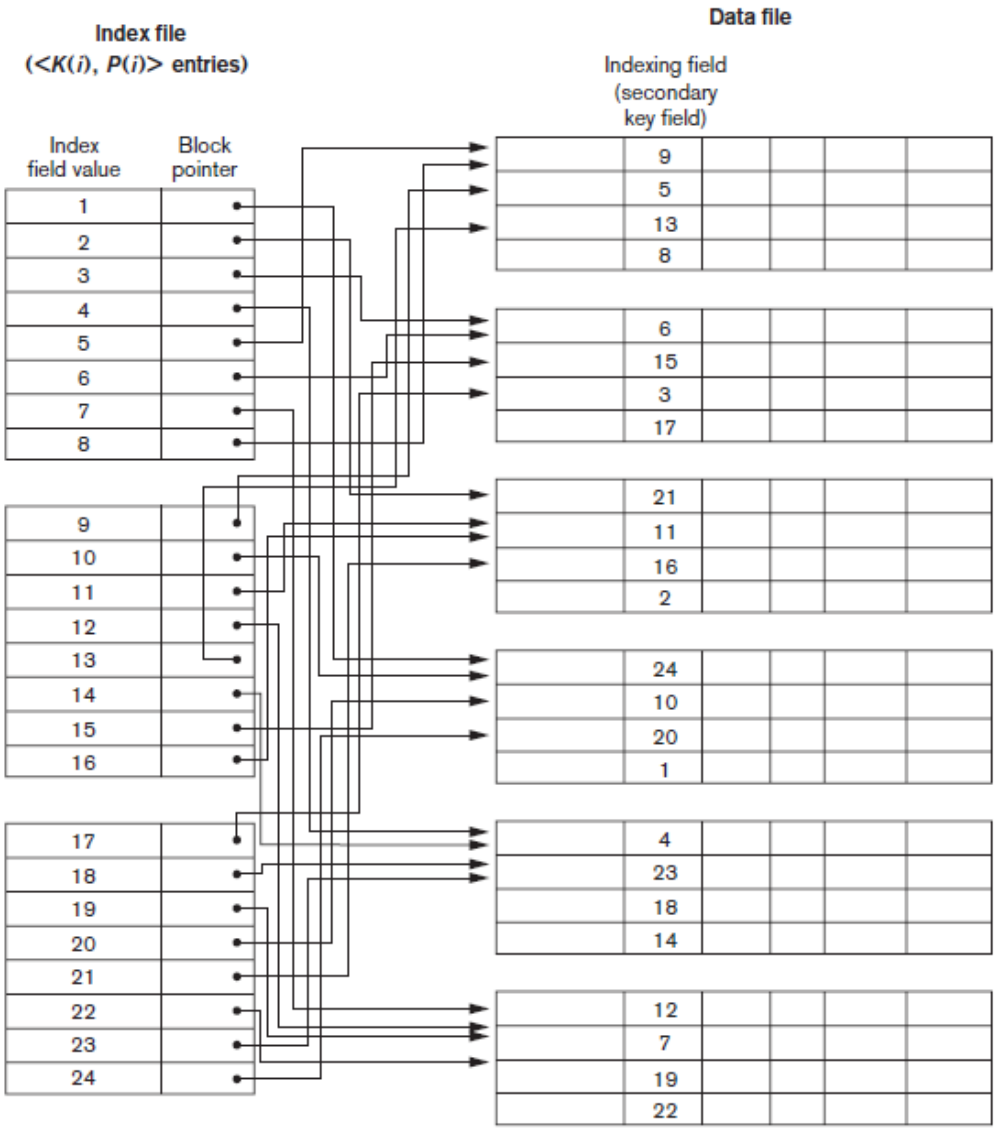
INDICE SECONDARIO
su chiave/UNIQUE

- I record nel file NON sono fisicamente ordinati rispetto ai valori del campo dell'indice secondario,
  - non è possibile riferire solo al blocco che contiene il primo record



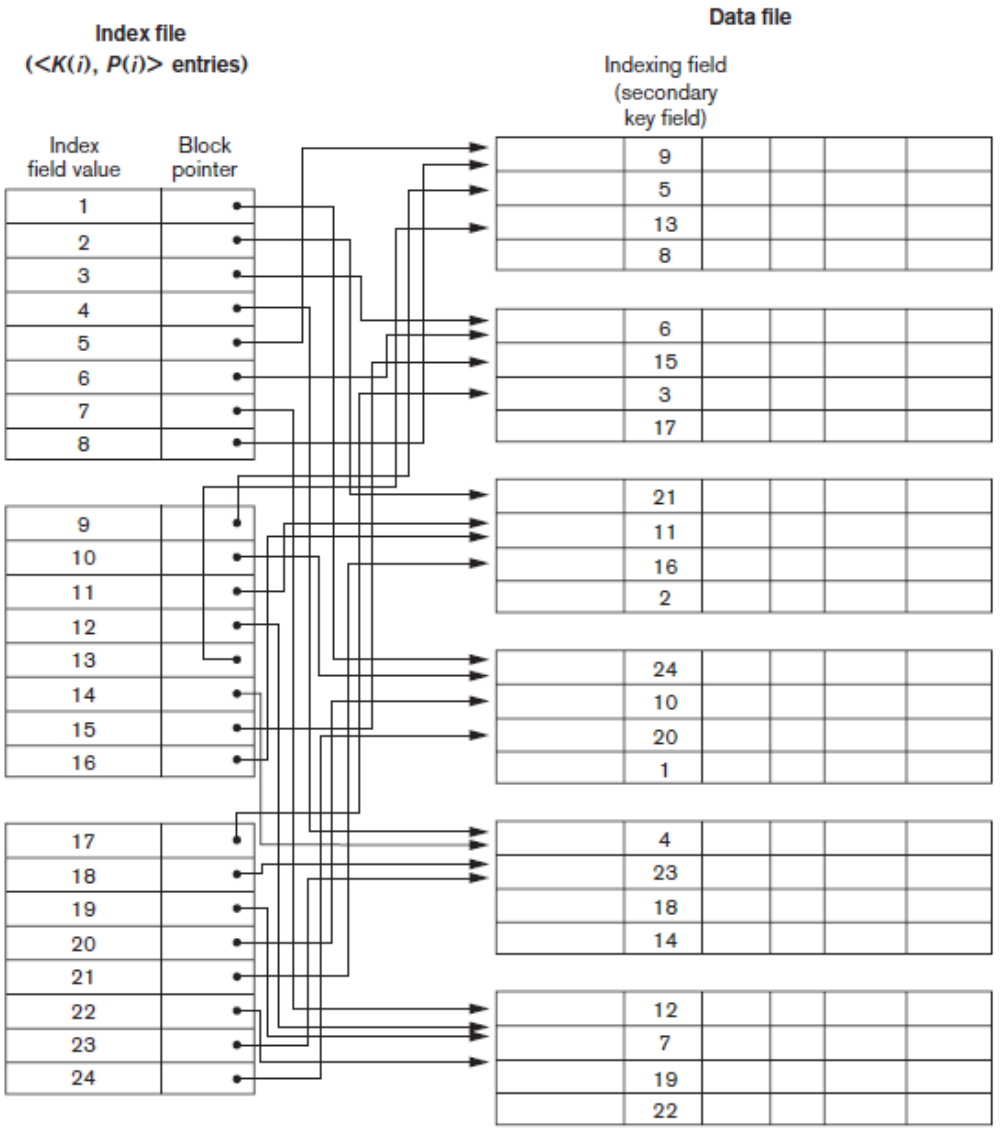
INDICE SECONDARIO
su chiave/UNIQUE

- I record nel file NON sono fisicamente ordinati rispetto ai valori del campo dell'indice secondario,
  - non è possibile riferire solo al blocco che contiene il primo record
- Nel file indice serve una voce per ogni possibile valore del campo dell'indice secondario
  - Indice DENSO



INDICE SECONDARIO
su chiave/UNIQUE

- I record nel file NON sono fisicamente ordinati rispetto ai valori del campo dell'indice secondario,
  - non è possibile riferire solo al blocco che contiene il primo record
- Nel file indice serve una voce per ogni possibile valore del campo dell'indice secondario
  - Indice DENSO
- Un indice secondario ha bisogno di più spazio di memorizzazione e richiede un maggior tempo di ricerca rispetto all'indice primario



Esercizio: Costo ricerca in termini di numero accessi

- Si supponga un file con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.
- Si assuma un indice secondario su campo V dove:
  - Dimensione V è 9 byte
  - Dimensione puntatore è 6 byte

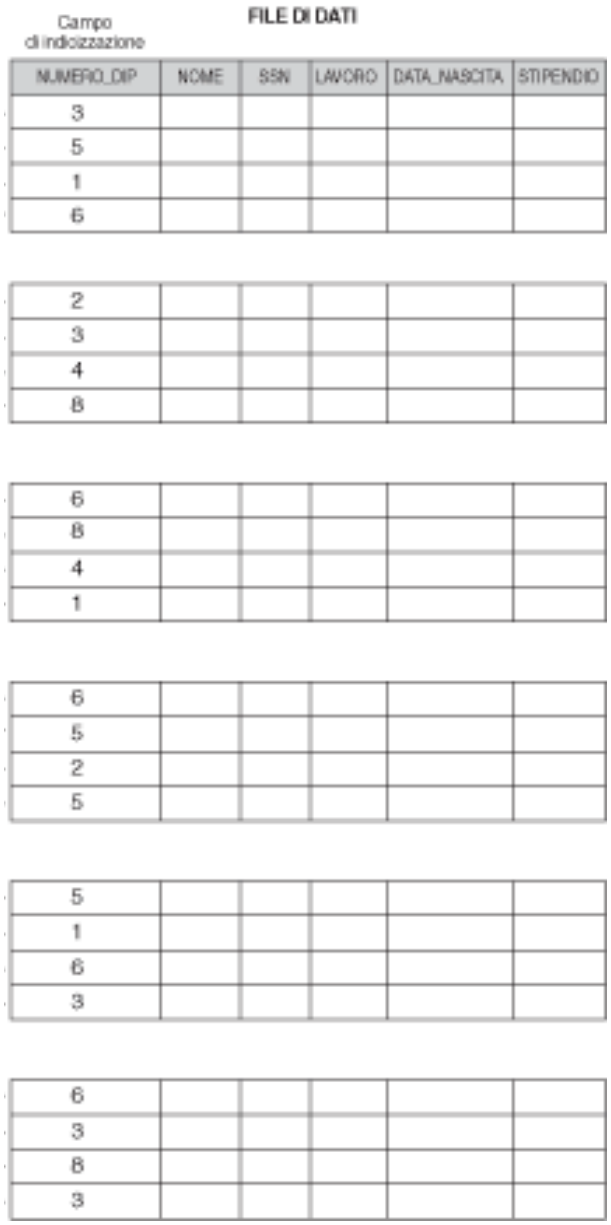
Mediamente, quanti accessi sono richiesti per una ricerca sul campo V?

- Fattore di blocco per il file dati $\lfloor 4096/100 \rfloor = 40$
- Numero blocchi per memorizzare i record: $\lceil 300.000/40 \rceil = 7500$
- Fattore di blocco per il file indice: $\lfloor 4096/15 \rfloor = 273$
- Nell'indice secondario, una voce per record, quindi
  - Numero di blocchi per memorizzare indice: $\lfloor 300.000/273 \rfloor = 1099$
- Numero accesso ai blocchi file indice per ricerca binaria $\lceil \log_2 1099 \rceil = 11$
- N. Accessi per ricercare record 11+1

INDICE SECONDARIO

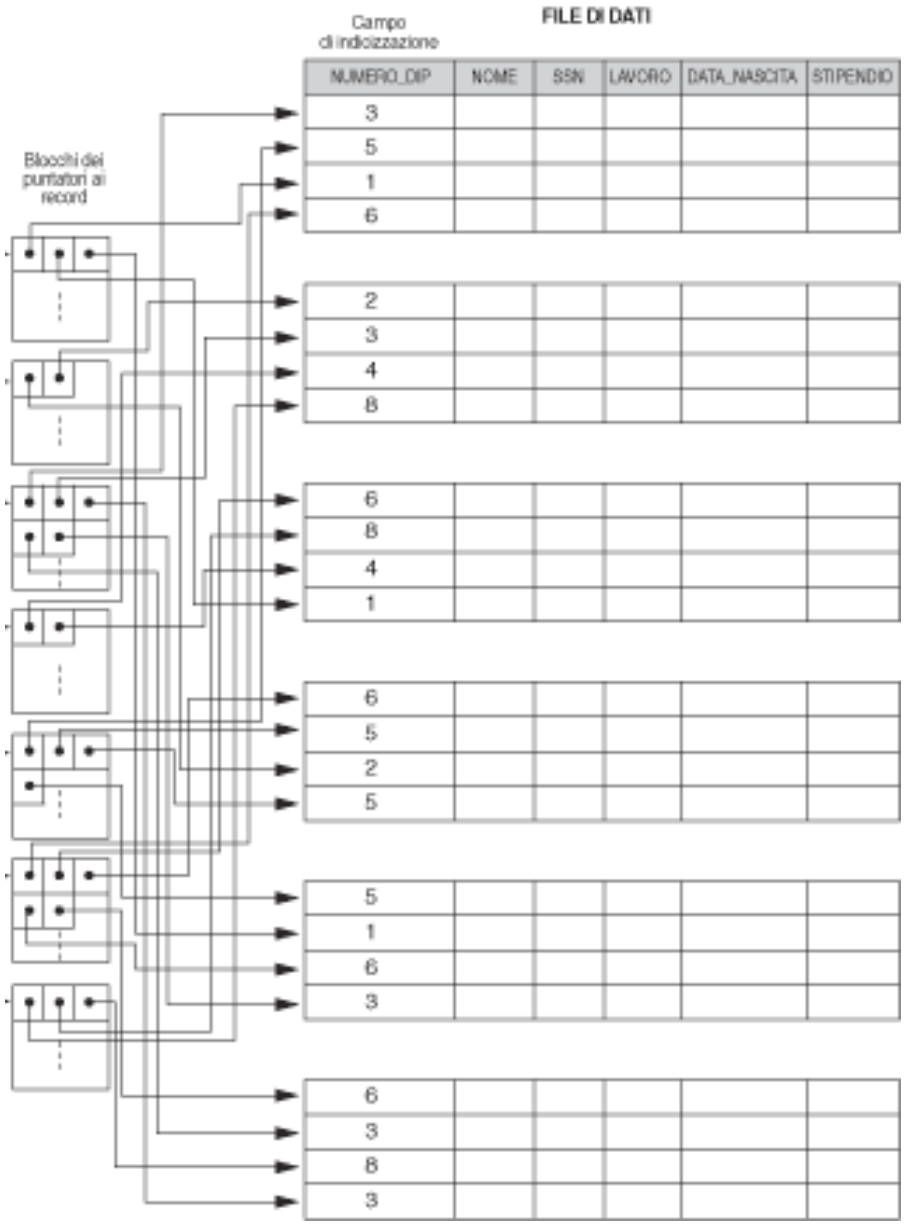
su campo NON chiave

- Numerosi record del file possono avere lo stesso valore del campo indicizzazione
- Varie soluzioni per file indice
  - Inserire più voci con stesso valore del campo di indicizzazione
  - Usare record di lunghezza variabile per ospitare tutti i puntatori ai record



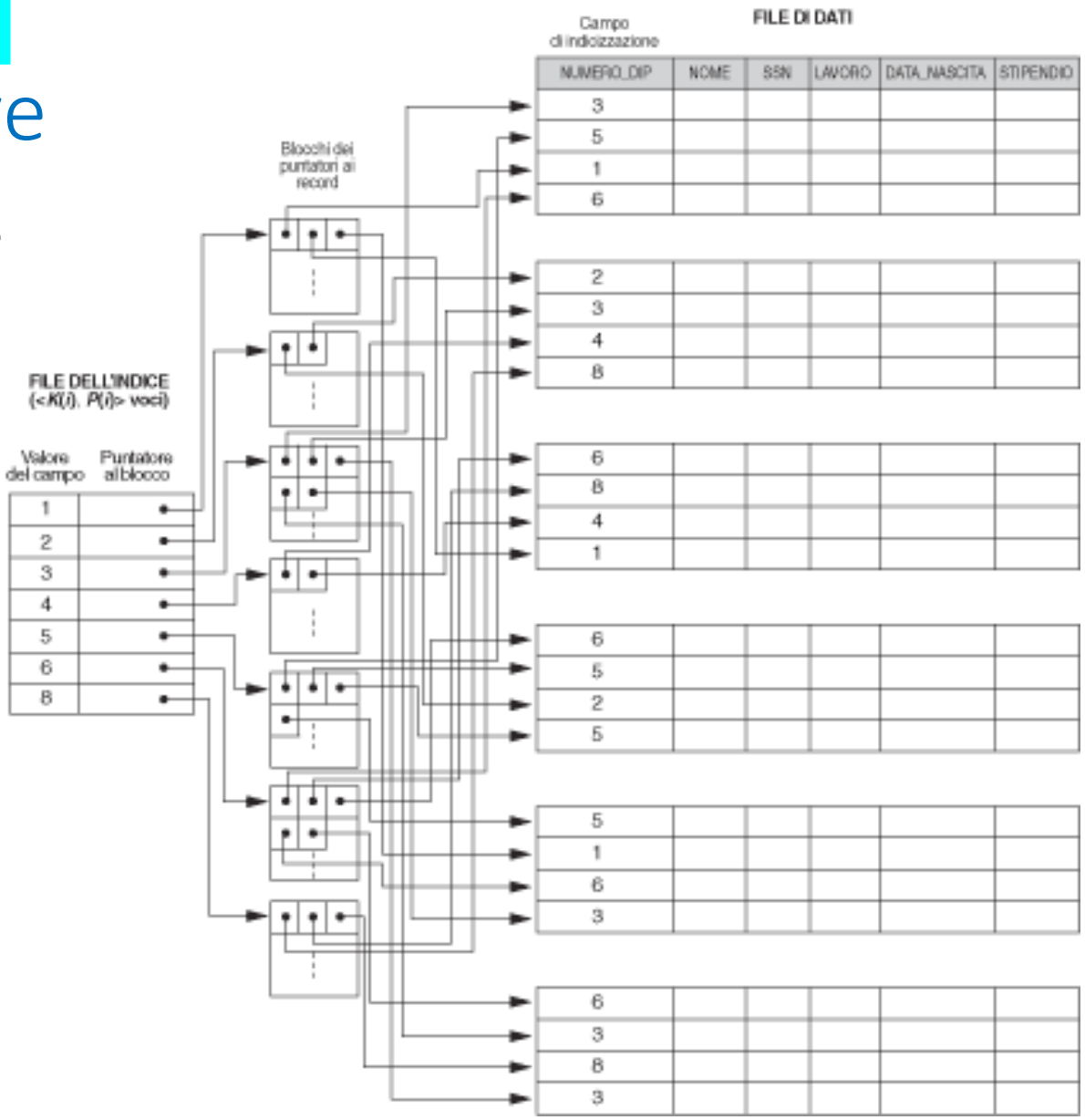
INDICE SECONDARIO
su campo NON chiave

- Numerosi record del file possono ave lo stesso valore del campo indicizzazione
- Varie soluzioni per file indice
  - Inserire più voci con stesso valore del campo di indicizzazione
  - Usare record di lunghezza variabile per ospitare tutti i puntatori ai record
  - Creare un ulteriore livello per gestire i puntatori multipli. Una voce per valore con puntatore che punta al blocco dei puntatori



INDICE SECONDARIO
su campo NON chiave

- Numerosi record del file possono ave lo stesso valore del campo indicizzazione
- Varie soluzioni per file indice
  - Inserire più voci con stesso valore del campo di indicizzazione
  - Usare record di lunghezza variabile per ospitare tutti i puntatori ai record
  - Creare un ulteriore livello per gestire i puntatori multipli. Una voce per valore con puntatore che punta al blocco dei puntatori



INDICI MULTILIVELLO

- Indici ordinati permettono di fare una ricerca binaria
  - Ogni passo della ricerca riduce la parte da ricercare di un fattore pari a 2

$\log_2$ (numero blocchi file indice da leggere)

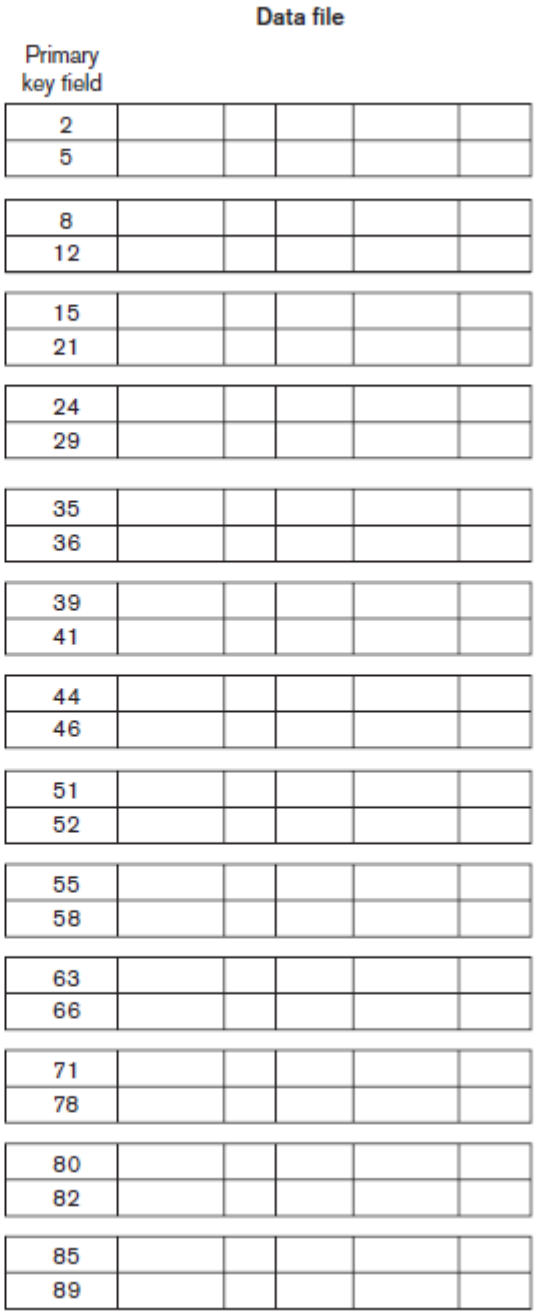
INDICI MULTILIVELLO

- Indici ordinati permettono di fare una ricerca binaria
  - Ogni passo della ricerca riduce la parte da ricercare di un fattore pari a 2

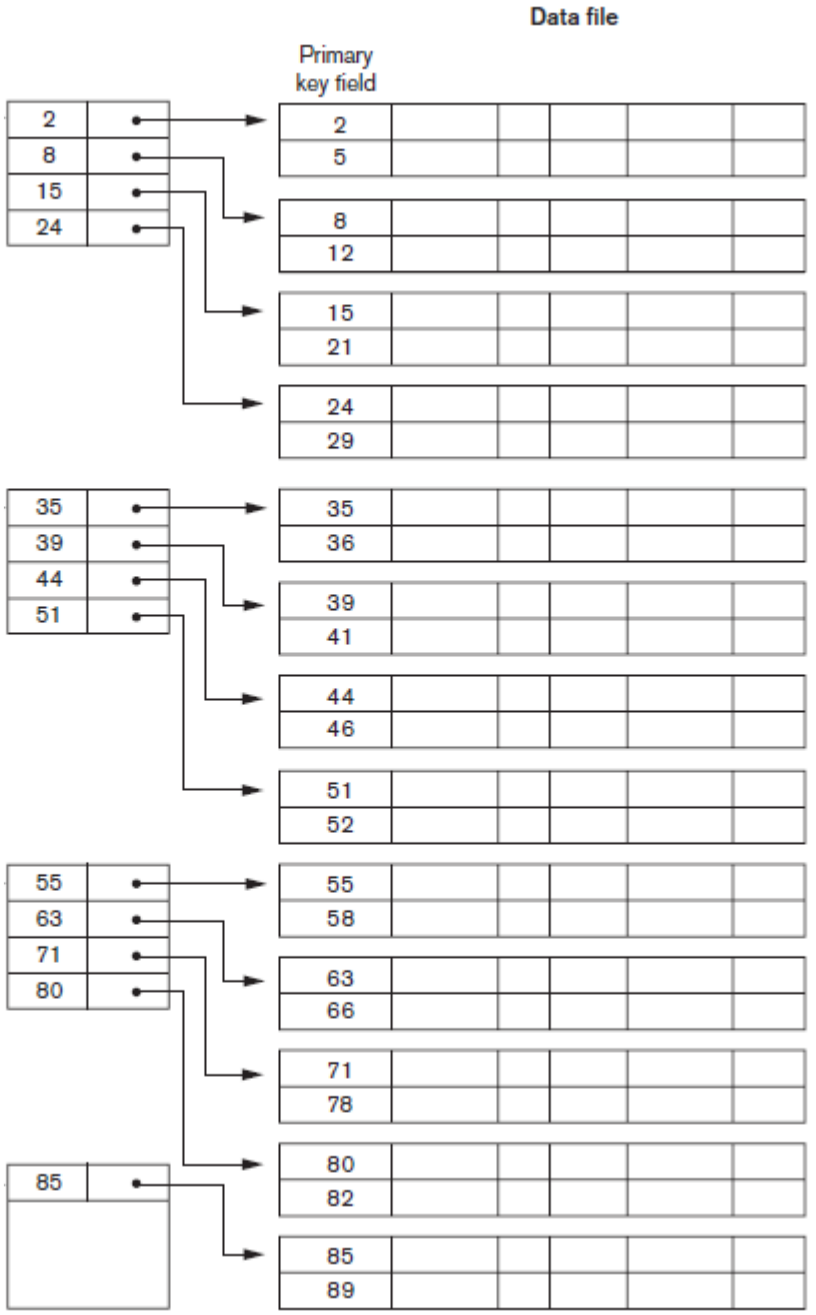
$$\log_2 \text{(numero blocchi file indice da leggere)}$$

- Obiettivo indice multilivello:
  - ridurre lo spazio da considerare ad ogni passo di un fattore >2
- IDEA: creare un indice per il file indice!

INDICI MULTILIVELLO

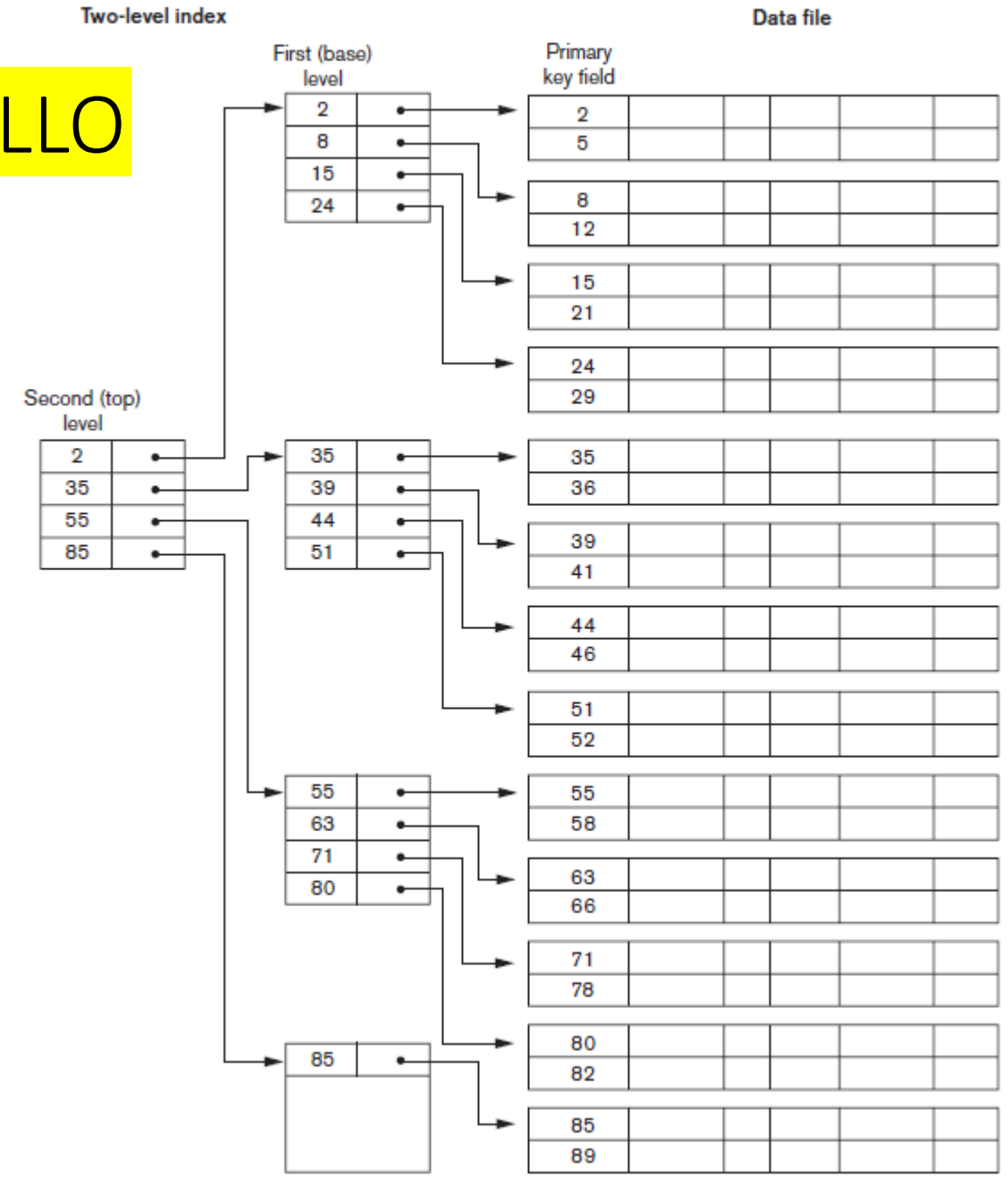


INDICI MULTILIVELLO



INDICI MULTILIVELLO

Indice primario
per il file indice
(indice di secondo livello)

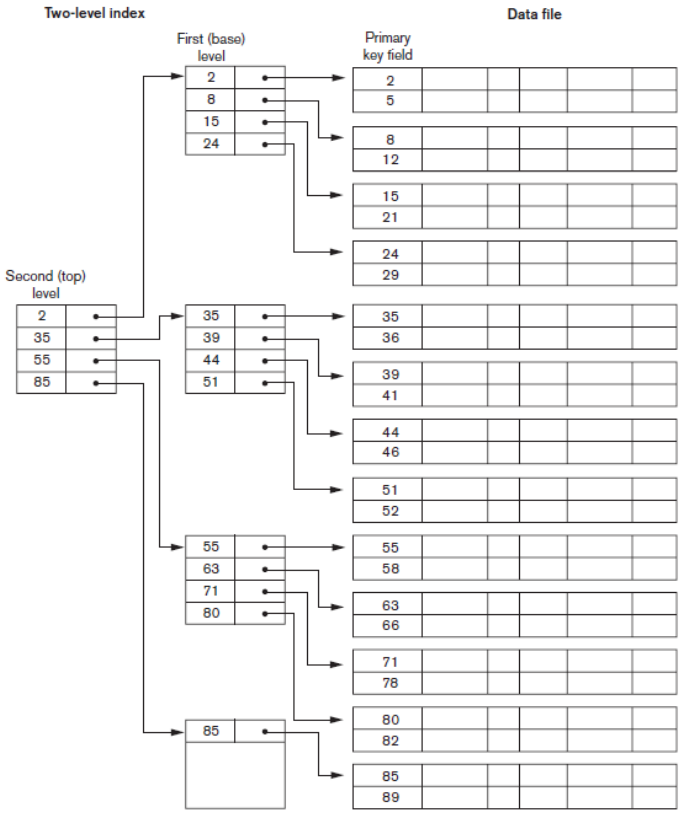


INDICI MULTILIVELLO

- Gli indici sono file essi stessi e quindi ha senso costruire indici sugli indici, per evitare di fare ricerche fra blocchi diversi
- Possono esistere più livelli fino ad avere il livello più alto con un solo blocco

Questo indice di quanto riduce

lo spazio di ricerca ad ogni passo?



INDICI MULTILIVELLO

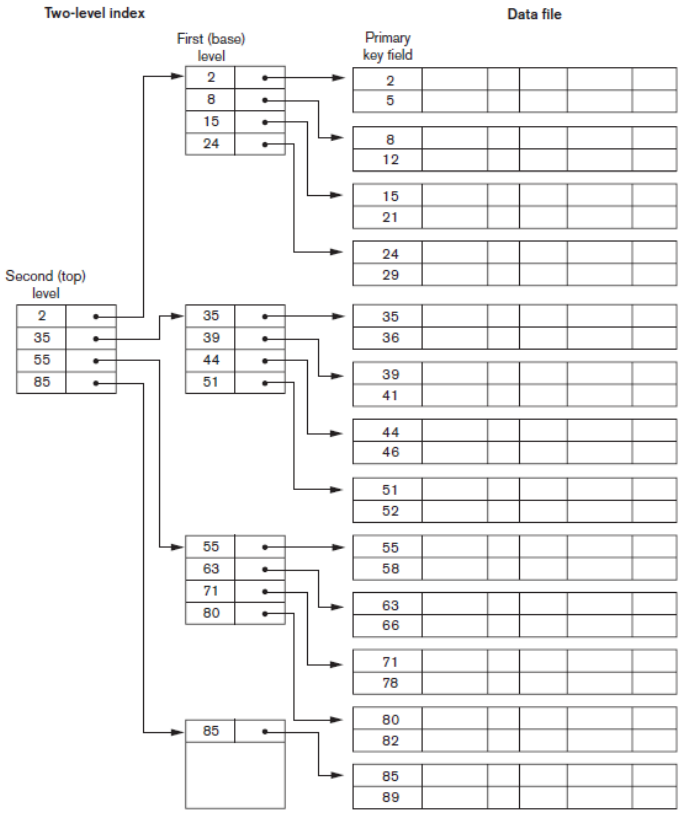
- Gli indici sono file essi stessi e quindi ha senso costruire indici sugli indici, per evitare di fare ricerche fra blocchi diversi
- Possono esistere più livelli fino ad avere il livello più alto con un solo blocco

Questo indice di quanto riduce

lo spazio di ricerca ad ogni passo?

Fan-out = 4 - Numero voci per blocco

(fattore blocco del file indice)



INDICI MULTILIVELLO

- Gli indici sono file essi stessi e quindi ha senso costruire indici sugli indici, per evitare di fare ricerche fra blocchi diversi
- Possono esistere più livelli fino ad avere il livello più alto con un solo blocco

Questo indice di quanto riduce

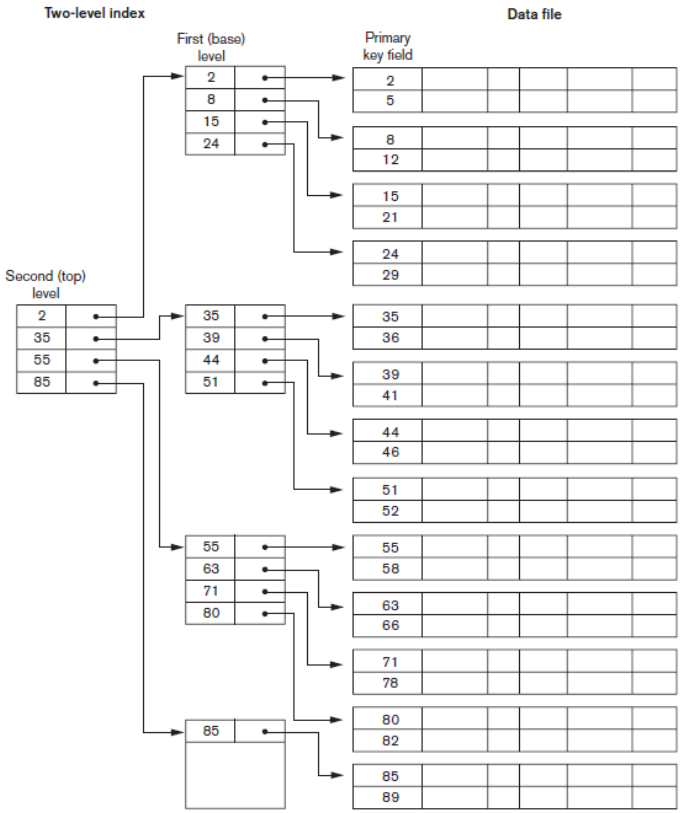
lo spazio di ricerca ad ogni passo?

Fan-out = 4 - Numero voci per blocco

(fattore blocco del file indice)

Con questo indice quanti accessi

devo fare per ricercare un record?



INDICI MULTILIVELLO

- Gli indici sono file essi stessi e quindi ha senso costruire indici sugli indici, per evitare di fare ricerche fra blocchi diversi
- Possono esistere più livelli fino ad avere il livello più alto con un solo blocco

Questo indice di quanto riduce

lo spazio di ricerca ad ogni passo?

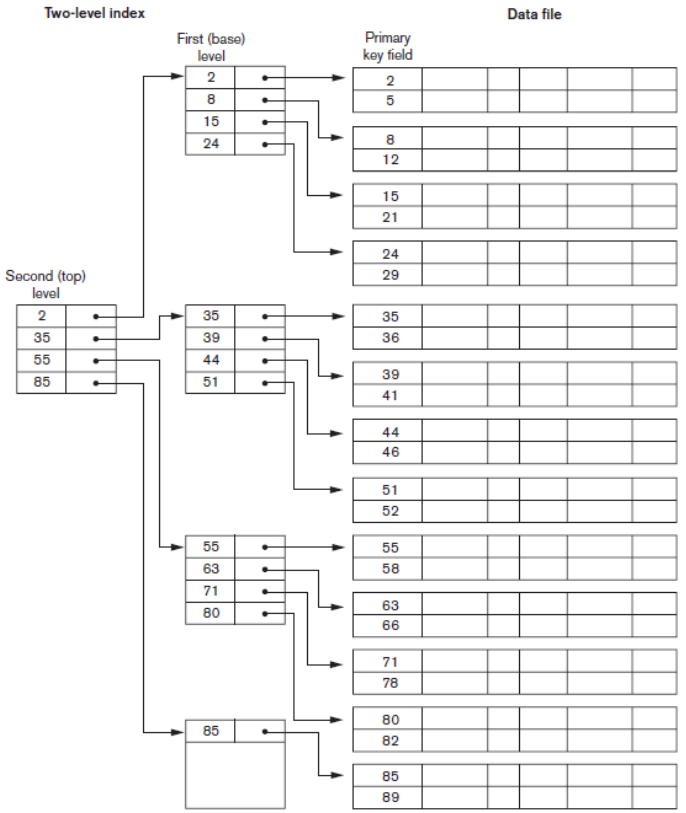
Fan-out = 4 - Numero voci per blocco

(fattore blocco del file indice)

Con questo indice quanti accessi

devo fare per ricercare un record?

3 - (2 per i livelli dell'indice, 1 per il blocco dati)



INDICI MULTILIVELLO

- La ricerca di un record dipende dal numero di livelli
  - meno livelli, meno accessi
- Dato un file con n record quanti livelli servono per un indice multilivello?
  - Dipende da quanti voci/record si possono inserire in ogni blocco
    - Fattore di blocco dell'indice -> Fan out

Esercizio: Costo ricerca in termini di numero accessi

- Si supponga un file ordinato su campo ID con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.
- Si assuma un indice primario su campo ID dove:
  - Dimensione ID è 9 byte
  - Dimensione puntatore è 6 byte

Mediamente, quanti accessi sono richiesti per una ricerca sul campo ID?

Esercizio: Costo ricerca in termini di numero accessi

- Si supponga un file ordinato su campo ID con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.
- Si assuma un indice primario su campo ID dove:
  - Dimensione ID è 9 byte
  - Dimensione puntatore è 6 byte

Mediamente, quanti accessi sono richiesti per una ricerca sul campo ID con indice multilivello?

- Fattore di blocco per il file dati $\lfloor 4096/100 \rfloor = 40$
- Numero blocchi per memorizzare i record: $\lceil 300.000/40 \rceil = 7500$
- Fattore di blocco per il file indice: $\lfloor 4096/15 \rfloor = 273$ FAN OUT
- Indice primo livello per 7500 blocchi di dati
  - Numero blocchi per memorizzare indice primo livello: $\lceil 7500/273 \rceil = 28$
- Indice secondo livello per 28 blocchi dell'indice primo livello
  - Numero blocchi per memorizzare indice secondo livello $\lceil 28/273 \rceil = 1 <=$ blocco radice
- N. Accessi per ricercare record 2+1

Esercizio: Costo ricerca in termini di numero accessi

- Si supponga un file NON ordinato su campo ID con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.
- Si assuma un indice primario su campo ID dove:
  - Dimensione ID è 9 byte
  - Dimensione puntatore è 6 byte

Mediamente, quanti accessi sono richiesti per una ricerca sul campo ID con indice multilivello?

Esercizio: Costo ricerca in termini di numero accessi

- Si supponga un file NON ordinato su campo ID con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.
- Si assuma un indice primario su campo ID dove:
  - Dimensione ID è 9 byte
  - Dimensione puntatore è 6 byte

Mediamente, quanti accessi sono richiesti per una ricerca sul campo ID con indice multilivello?

- Fattore di blocco per il file dati $\lfloor 4096/100 \rfloor = 40$
- Numero blocchi per memorizzare i record: $\lceil 300.000/40 \rceil = 7500$
- Fattore di blocco per il file indice: $\lfloor 4096/15 \rfloor = 273$ FAN OUT
- Indice primo livello per 300.000 record
  - Numero blocchi per memorizzare indice primo livello: $\lceil 300.000/273 \rceil = 1099$
- Indice secondo livello per 1099 blocchi del primo livello
  - Numero blocchi per memorizzare indice secondo livello $\lceil 1099/273 \rceil = 4$
- Indice terzo livello per 4 blocchi del secondo livello
  - Numero blocchi per memorizzare indice terzo livello $\lceil 4/273 \rceil = 1 <=$ blocco radice
- N. Accessi per ricercare record 3+1

Indici, problemi

- Tutte le strutture di indice viste finora sono basate su strutture ordinate e quindi sono poco flessibili in presenza di elevata dinamicità
  - Inserimenti e cancellazioni richiedono riordino degli indici
- Gli indici utilizzati dai DBMS sono più sofisticati:
  - indici dinamici multilivello: B-tree (intuitivamente: alberi di ricerca bilanciati)

Esercizio-1

Si consideri un disco con dimensione di blocco *B* = 512 byte. Un puntatore a blocco è lungo *P* = 6 byte. Il file ha *r* = 30.000 record della tabella IMPIEGATO di *lunghezza fissa*. Ciascun record ha i seguenti campi:

- NOME (30 byte),
- CF (16 byte),
- NUMERO_DIPARTIMENTO (9 byte),
- TELEFONO (9 byte),
- DATA_NASCITA (8 byte),
- CODICE_LAVORO (4 byte).
- Un ulteriore byte viene utilizzato come indicatore di cancellazione.

a) Calcolare la dimensione del record *R* in byte.

b) Calcolare il fattore di blocco *bfr* e il numero di blocchi *b* del file supponendo un'organizzazione unspanned

c) Si supponga che il file sia *ordinato* con il campo chiave CF e che si voglia creare un *indice primario* su CF. Calcolare:

   i. il fattore di blocco dell'indice *bfri*

   ii. il numero delle voci e il numero di blocchi dell'indice primario;

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

   iv. il numero totale di blocchi richiesti dall'indice multilivello;

   v. il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di CF, usando l'indice primario;

   vi. il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di CF, usando l'indice multilivello;

d) Si supponga che il file *non sia ordinato* tramite la chiave CF e si voglia creare un *indice secondario* su CF. Ripetere l'esercizio precedente (c) per l'indice secondario e confrontarlo con l'indice primario.

Esercizio-1

Si consideri un disco con dimensione di blocco *B* = 512 byte. Un puntatore a blocco è lungo *P* = 6 byte. Il file ha *r* = 30.000 record della tabella IMPIEGATO di *lunghezza fissa*. Ciascun record ha i seguenti campi:

- NOME (30 byte),
- CF (16 byte),
- NUMERO_DIPARTIMENTO (9 byte),
- TELEFONO (9 byte),
- DATA_NASCITA (8 byte),
- CODICE_LAVORO (4 byte).
- Un ulteriore byte viene utilizzato come indicatore di cancellazione.

a) Calcolare la dimensione del record *R* in byte.

77 byte

b) Calcolare il fattore di blocco *bfr* e il numero di blocchi *b* del file supponendo un'organizzazione unspanned

Bfr= $\lfloor 512/77 \rfloor = 6$

numero blocchi $\lceil 30.000/6 \rceil = 5000$

Esercizio -1

B = 512 byte | *r* = 30.000 record | Fattore di blocco del record =6

P = 6 byte | Dimensione del record *R* in byte= 77 byte | numero blocchi file di dati=5000

c) Si supponga che il file sia *ordinato* con il campo chiave CF (16 byte) e che si voglia creare un *indice primario* su CF. Calcolare:

i. il fattore di blocco dell'indice *bfri*

Dimensione voce= Dimensione CF + dimensione puntatore= 16+6= 22

Fattore di blocco dell'indice bfri= $\lfloor 512/22 \rfloor = 23$

ii. il numero delle voci e il numero di blocchi dell'indice primario

Il file è ordinato sul campo CF (campo dell'indice primario). E' necessario una voce per ogni blocco dati (indice sparso)

Numero voci indice primario= 5000

Numero blocchi dell'indice primario: $\lceil 5.000/23 \rceil = 218$

Esercizio-1

B = 512 byte | *r* = 30.000 record | Fattore di blocco del record =6 | fattore di blocco dell'indice =23

P = 6 byte | Dimensione del record *R* in byte= 77 byte | numero blocchi file di dati=5000 | numero blocchi dell'indice primario:= 218

c) Si supponga che il file sia *ordinato* con il campo chiave CF (16 byte) e che si voglia creare un *indice primario* su CF. Calcolare:

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

2° livello: indice per un file ordinato (indice primario)

numero voci secondo livello:

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

numero blocchi per secondo livello:

⌈218/23⌉= 10

serve un altro livello?

Si, per indicizzare i 218 voci nei 10 blocchi

3° livello: indice per un file ordinato (anche il secondo livello è ordinato)

numero voci terzo livello:

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

numero blocchi per terzo livello:

⌈10/23⌉= 1

serve un altro livello?

No. 10 voci si indicizzano con un blocco solo

iv. il numero totale di blocchi richiesti dall'indice multilivello;

Blocchi 1° livello + blocchi 2° livello + blocchi 3° livello= 218+ 10+1 =229

v. il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di CF, usando l'indice primario;

ricerca binaria su indice primario + accesso al blocco dati 1 => : $\log_2 218$= 8 +1 => 8+1=9

vi. il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di CF, usando l'indice multilivello;

uno accesso al blocco per livello + tempo accesso al blocco dati: 3+1=4

Esercizio-1

B = 512 byte | *r* = 30.000 record | Fattore di blocco del record =6 | fattore di blocco dell'indice =23

P = 6 byte | Dimensione del record *R* in byte= 77 byte | numero blocchi file di dati=5000 | numero blocchi dell'indice primario:= 218

d) Si supponga che il file *non sia ordinato* tramite la chiave CF e si voglia creare un *indice secondario* su CF. Ripetere l'esercizio precedente (c) per l'indice secondario e confrontarlo con l'indice primario.

i. il fattore di blocco dell'indice *bfri* -- non cambia

ii. Indice secondario: il numero delle voci e il numero di blocchi dell'indice secondario

Il file NON è ordinato sul campo CF. E' necessario una voce per ogni record (indice denso)
Numero voci indice secondario= 30000
Numero blocchi dell'indice secondario: ⌈30.000/23⌉= 1305

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

2° livello: indice per un file ordinato (indice secondario è ordinato)

numero voci secondo livello: 1305 (una per blocco dell'indice secondario – indice sparso)
numero blocchi per secondo livello: ⌈1305/23⌉= 57
serve un altro livello?
Si, per indicizzare i 1305 voci nei 57 blocchi

3° livello: indice per un file ordinato (anche il secondo livello è ordinato)

numero voci terzo livello: 57 (una per blocco per l'indice di secondo livello – indice sparso)
numero blocchi per terzo livello: ⌈57/23⌉=3
serve un altro livello?
Si. 57 voci richiedono 3 blocchi

4° livello: indice per un file ordinato

numero voci terzo livello: 3 (una per blocco per l'indice di secondo livello – indice sparso)
numero blocchi per terzo livello: ⌈3/23⌉=1
serve un altro livello?
No. 3 voci richiedono1 blocco solo

iv. il numero totale di blocchi richiesti dall'indice multilivello;

Blocchi 1° livello + blocchi 2° livello + blocchi 3° livello + blocchi 4° livello = 1305+ 57+ 3+1 =1366

v. il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di CF, usando l'indice secondario;

ricerca binaria su indice primario: $\log_2 1305 = 11$ + accesso al blocco dati 1 => 11+1=12

vi. il numero di accessi necessari per cercare e reperire un record dal file, dato il valore di CF, usando l'indice multilivello;

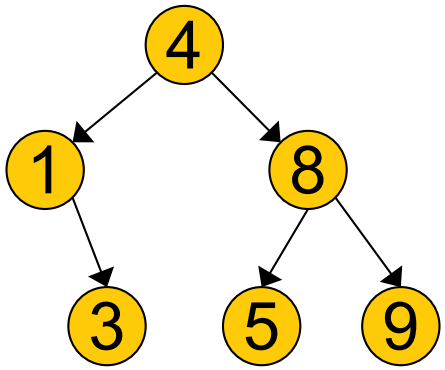
uno accesso al blocco per livello + tempo accesso al blocco dati: 4+1=5

Indici, problemi

- Tutte le strutture di indice viste finora sono basate su strutture ordinate e quindi sono poco flessibili in presenza di elevata dinamicità
  - Inserimenti e cancellazioni richiedono riordino degli indici
- Gli indici utilizzati dai DBMS sono più sofisticati:
  - indici dinamici multilivello: B-tree (intuitivamente: alberi di ricerca bilanciati)

Albero di ricerca (binario)

- Un albero di ricerca è un albero che viene usato per guidare la ricerca di un record dato il valore di un suo campo.
- Esempio:
  - Albero binario di ricerca: per ogni nodo n
    - il sottoalbero sinistro contiene solo valori minori del valore di n
    - il sottoalbero destro etichette maggiori

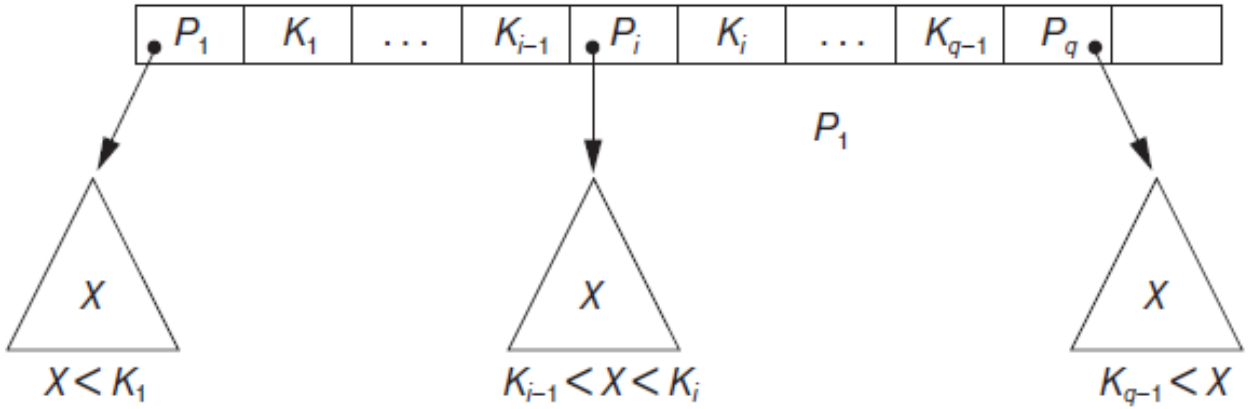


tempo di ricerca è pari alla profondità dell'albero

Se i record sono tanti, un albero binario non è adatto

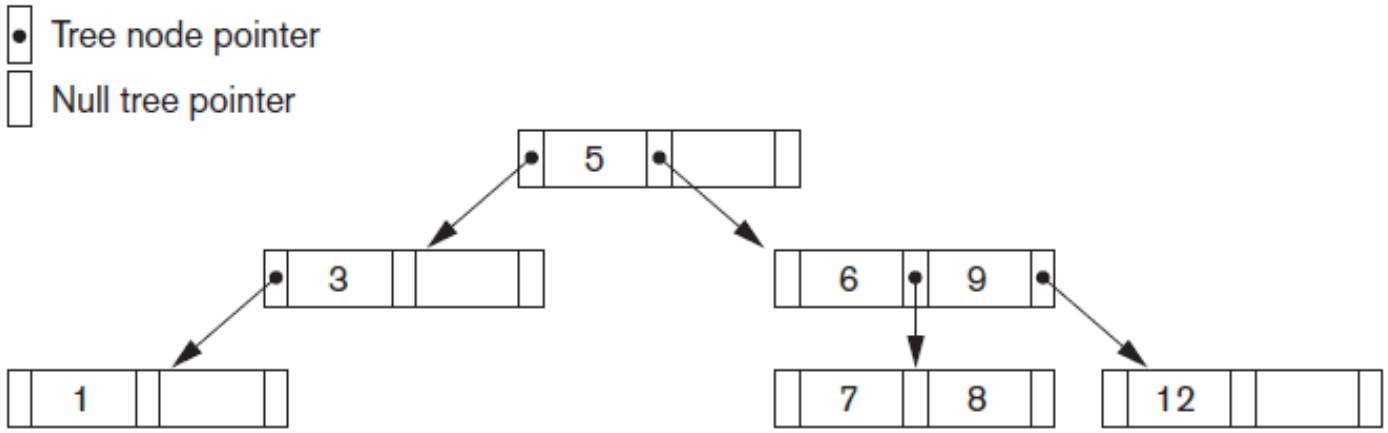
Albero di ricerca di ordine p

- Un albero di ricerca di ordine p sul valore di ricerca K, definito come campo unique/chiave:
  - ogni nodo contiene al massimo p-1 valori di ricerca e p puntatori a sottoalberi
    - all'interno del nodo i valori di ricerca sono ordinai (i.e., $K_1 < K_2 <\ldots K_{q-1}$ con q<= p);
    - i valori X del sottoalbero puntato da P_i sono tali che: $K_{i-1}<X<K_i$



Albero di ricerca di ordine p

- Esempio di albero di ricerca di ordine 3:
  - ogni nodo contiene al massimo 2 valori di ricerca e 3 puntatori a sottoalberi



NOTA: Per ricercare i record su un file, l'albero contiene per ogni valore di ricercare K_j anche il puntatore al corrispondente record/blocco sul disco (non rappresentato nella figura)

Albero di ricerca di ordine p - osservazioni

- Inserimenti e cancellazioni di record richiedono la ristrutturazione dell'albero. Esistono vari algoritmi, si basano su due operazioni:
  - SPLIT – inserimento di un valore k: se il nodo a cui si aggiunge il valore diventa troppo grande (supera p-1) viene diviso (split)
  - MERGE – cancellazione di un valore k: se il nodo dopo la cancellazione rimane con pochi valori, viene unito (MERGE) ad altri nodi attigui

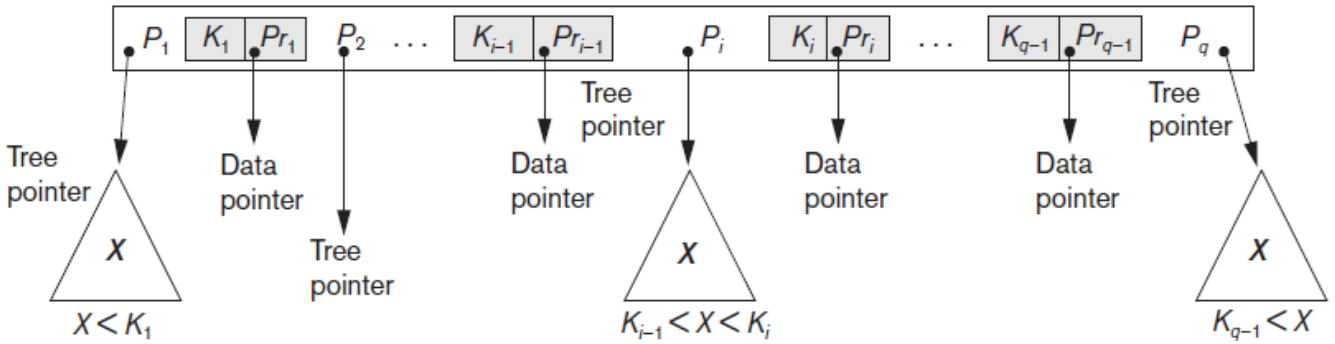
- Un albero bilanciato (nodi foglia tutti allo stesso livello) garantisce un tempo di ricerca uniforme

- B-tree garantisce che l'albero sia sempre bilanciato e minimizza lo spazio sprecato a seguito di cancellazioni

B-tree di ordine p

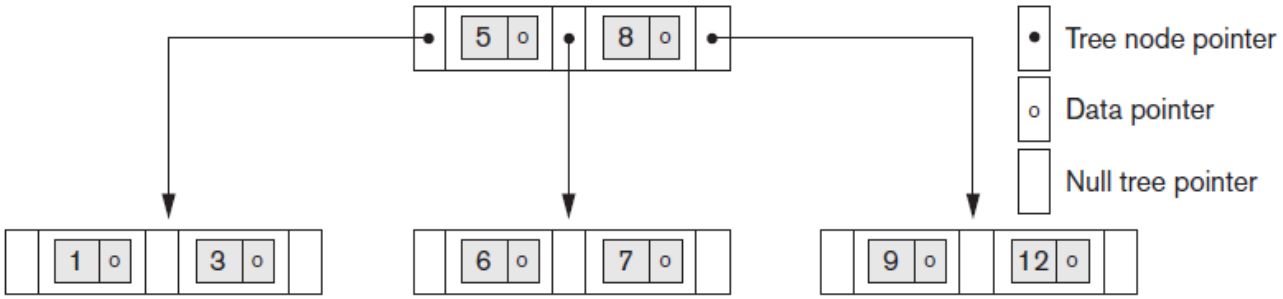
Un B-tree di ordine p su campo chiave K

- ogni nodo interno deve contenere almeno $\lceil p/2 \rceil$ puntatori ed ha la seguente forma:



- I nodi foglia sono tutti allo stesso livello, hanno una struttura simile al nodo interno, tolti i puntatori ai sottoalberi

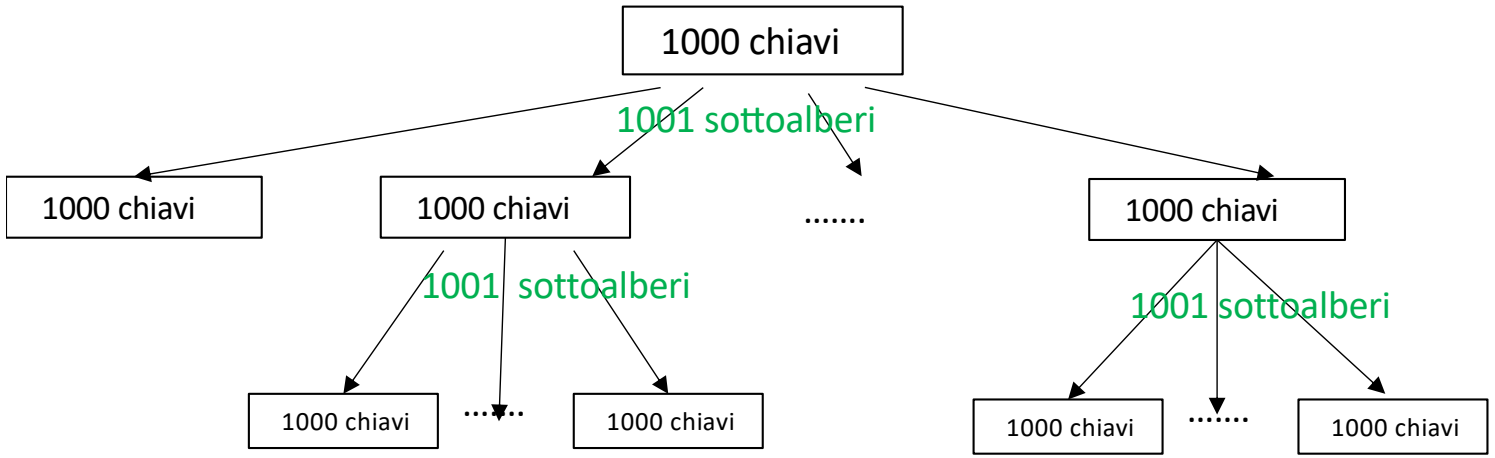
Esempio



B-tree - ordine

- L'ordine del B-Tree incide sulla quantità di chiavi rappresentabili e l'altezza (tempo ricerca)

Es. B-tree con ordine 1001: ogni nodo può rappresenta al max 1000 chiavi



Livello #chiavi	#nodi	
Root	1	1.000
1 liv.	1.001	1.001.000
2 liv.	1001^2	1.002.001.000

- B-tree con ordine 1001 a due livelli rappresento fino a

1000 + 1.001.000 + 1.002.001.000= 1.003.003.000 chiavi con tempo d'accesso 3 (3 letture dei blocchi dei nodi)

B-tree Inserimento e cancellazioni

- Non vedremo nel dettaglio gli algoritmi, ma potete trovare un simulatore qui

https://dichchankinh.com/~galles/visualization/BTree.html

Nota: nel simulatore, si può cambiare il max degree di ogni nodo, ovvero l'ordine

Esercizio: B-tree

- Si supponga un file su campo chiave ID (6byte) con r=300.000 record memorizzati su disco.
- Si supponga di costruire un B-tree di ordine 100:
  - Quanti nodi sono necessari?
- Si supponga di costruire un B-tree di ordine 10:
  - Quanti nodi sono necessari?

Esercizio: B-tree

- Si supponga un file su campo chiave ID (6byte) con r=300.000 record memorizzati su disco.
- Si supponga di costruire un B-tree di ordine 100:
  - Quanti nodi sono necessari?

Livello	#nodi	#valori di chiavi	#puntatori nodi
Root	1	99	100
1 liv.	100	99x100= 9.900	100^2
2 liv.	100^2	$99x100^2$ = 990.000	--

#nodi: 10.101

- Si supponga di costruire un B-tree di ordine 10:
  - Quanti nodi sono necessari?

Livello	#nodi	#valori di chiavi	#puntatori nodi
Root	1	9	10
1 liv.	10	9 x 10= 90	10^2
2 liv.	10^2	9 x 10^2 = 900	10^3
3 liv.	10^3	9 x 10^3 = 9.000	10^4
4 liv.	10^4	9 X 10^4 = 90.000	10^5
5 liv.	10^5	9 X 10^5 = 900.000	---

#nodi: 111.111

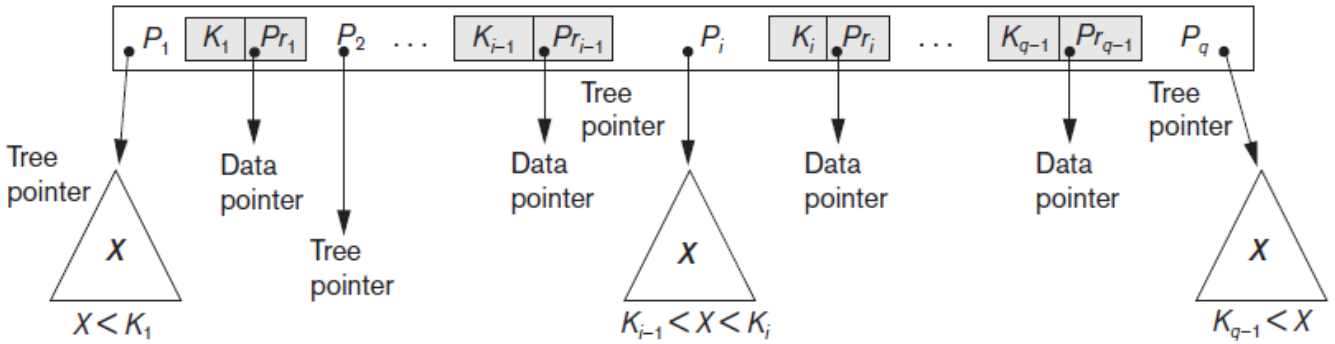
B^+ -tree

- B^+ -tree è una variante del B-tree
  - B-tree
    - i puntatori ai record sono memorizzati nel nodi (interni e foglia) insieme al corrispondente valore del campo ricerca

B-tree di ordine p

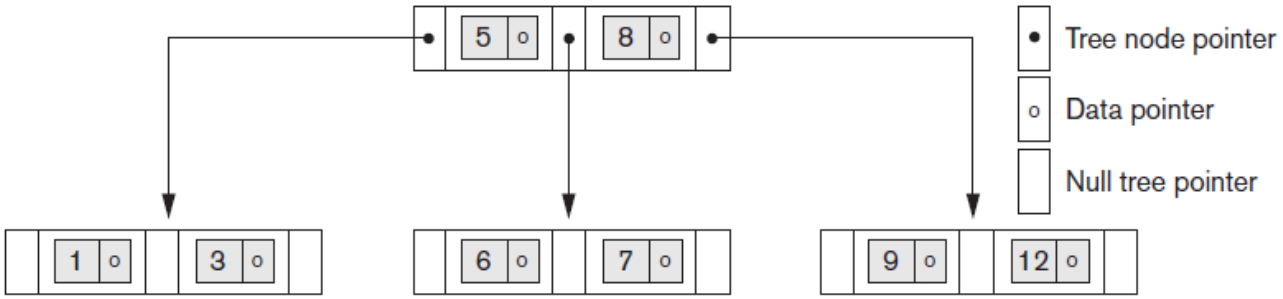
Un B-tree di ordine p su campo chiave K

- ogni nodo interno deve contenere almeno $\lceil p/2 \rceil$ puntatori ed ha la seguente forma:



- I nodi foglia sono tutti allo stesso livello, hanno una struttura simile al nodo interno, tolti i puntatori ai sottoalberi

Esempio



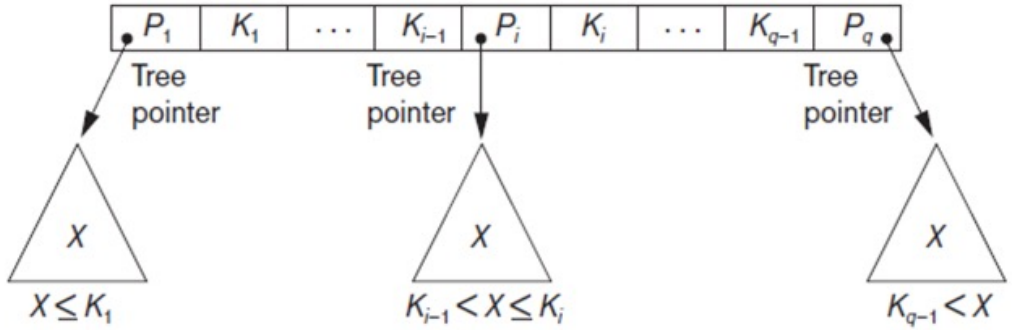
B^+ -tree

- B^+ -tree è una variante del B-tree
  - B-tree
    - i puntatori ai record sono memorizzati nel nodi (interni e foglia) insieme al corrispondente valore del campo ricerca
  - B^+-tree
    - i puntatori ai record sono memorizzati solo sui nodi foglia
      - i nodi interni contengono solo puntatori ai nodi figli
    - I nodi foglia contengono un puntatore per ogni valore del campo ricerca
    - I nodi foglia sono collegati tra loro per permettere un accesso ordinato al campo ricerca

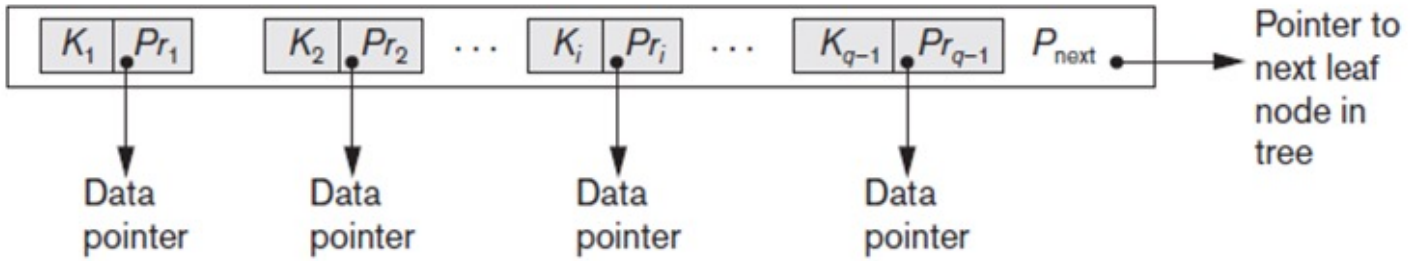
B^+-tree di ordine p

Un B^+-tree di ordine p su campo chiave K

- ogni nodo interno deve contenere almeno $\lceil p/2 \rceil$ puntatori ed ha la seguente forma:



- I nodi foglia sono tutti allo stesso livello, per ogni valore hanno il puntatore al dato (record), più un puntatore al nodo foglia successivo



B^+-tree Inserimento e cancellazioni

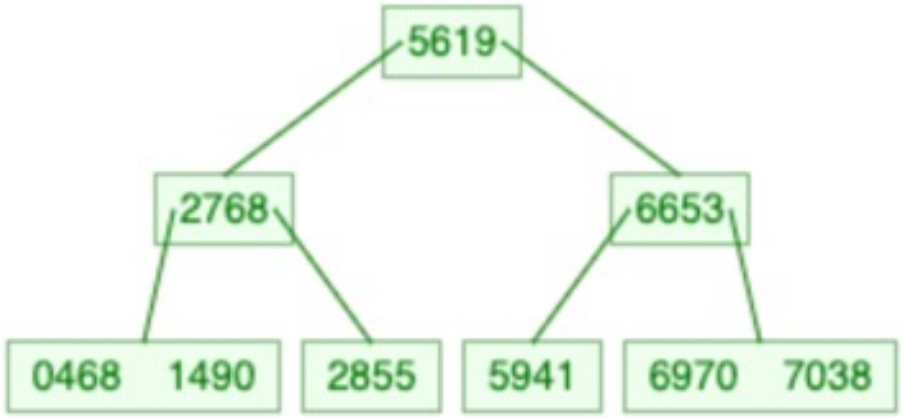
- Non vedremo nel dettaglio gli algoritmi, ma potete trovare un simulatore qui

https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

Nota: nel simulatore, si può cambiare il max degree di ogni nodo, ovvero l'ordine

B-Tree vs B^+-tree

B-tree



B^+-tree

ricerca di campi con valori nell'intervallo 1000-6000 (range query)?

B-tree/B$^+$-tree osservazioni

- Ogni nodo B-tree/B$^+$-tree è memorizzato in un blocco dati
  - Per ottimizzare split/merge: il blocco è riempito solo parzialmente, in base ad un fattore di riempimento.
- Nei B$^+$-tree, le voci dei nodi interni occupano meno spazio:
  - A parità di dimensione, l'ordine p sarà più grande per un B$^+$-tree rispetto B-tree: B$^+$-tree ha meno livelli di un B-tree
- Dato che nei B$^+$-tree le strutture dei nodi interni e foglia sono diverse, possiamo avere un ordine diverso: $p_{interno}$ e p_{foglia}
- B$^+$-tree costruiti su campi non chiave/unique richiedono un livello aggiuntivo:
  - Un puntatori al dato nel B$^+$-tree punta a un blocco che contiene i puntatori ai record con quel valore di ricerca

Esercizio: B^+-tree

- Si supponga di costruire un B^+-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6 byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Quanti record possono essere indicizzati da un B^+-tree a 3 livelli?
  - Calcolo l'ordine del B^+-tree: $p_{interno}$, p_{foglia}
  - Stimo numero dei valori di ricerca con fattore riempimento

Esercizio: calcolo dell'ordine $p_{interno}$ per B^+-tree

- Si supponga di costruire un B^+-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6 byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Calcolo l'ordine massimo per il nodo interno ($p_{interno}$)

- Un nodo interno di B^+-tree può avere fino a p puntatori ai nodi e p-1 valori del campo

$$p*P + (p-1)*V <= 512$$

$$p*6 + (p-1)*9 <= 512$$

$$6p +9p -9 <=512$$

$$15p<= 521$$

Nel B^+-tree il valore massimo $p_{interno}=34$

Esercizio: calcolo dell'ordine p_{foglia} per B^+-tree

- Si supponga di costruire un B^+-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6 byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Calcolo l'ordine massimo per il nodo foglia (p_{foglia})

- Un nodo foglia di B^+-tree contiene fino a p valori del campo e p puntatori ai record, più un puntatore al nodo successivo

$$p*Pr + p*V + P <= 512$$

$$p*7 + p*9 + 6 <= 512$$

$$16p <= 506$$

Nel B^+-tree il valore massimo $p_{foglia}=31$

Esercizio: calcolo record indicizzati B^+-tree

- Si supponga di costruire un B^+-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6 byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Quanti record possono essere indicizzati da un B^+-tree a 3 livelli?
- Con riempimento 70%: $p_{interno}$=34*0.70= 23 p_{foglia}=31*0.70= 21

Livello	#nodi	#valori di chiavi	#puntatori record
Root	1	(23-1)=22	-
1 liv.	23	23 x 22= 506	-
2 liv.	23^2=529	529 x 22 = 11.638	-
3 liv./foglie	23^3 =12.167	12.167 x 21= 255.507	255.507

Totale nodi: 12720

Puntatori record: 255.507

Esercizio: B-tree

- Si supponga che si costruisca un B-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Quanti record possono essere indicizzati da un B-tree a 3 livelli?
  - Calcolo l'ordine del B-tree: $p_{interno}$, p_{foglia}
  - Stimo numero dei valori di ricerca con fattore riempimento

Esercizio: B-tree

- Si supponga che si costruisca un B-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6 byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Calcolo l'ordine massimo per nodo interno?

- Un nodo interno di B-tree contiene fino a p-1 valori del campo, p-1 puntatori ai record, p puntatori ai nodi

$$(p-1)*V +(p-1)*Pr + p*P <= 512$$

$$(p-1)*9 +(p-1)*7 + p*6 <= 512$$

$$9p -9 +7p -7 +6p <= 512$$

$$22p <= 528$$

Nel B-tree il valore massimo $p_{interno}=24$

Esercizio: B-tree

- Si supponga che si costruisca un B-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6 byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Calcolo l'ordine massimo per nodo foglia?

- Un nodo foglia di B-tree contiene fino a p-1 valori del campo e p-1 puntatori ai record

$$(p-1)*Pr + (p-1)*V <= 512$$

$$(p-1)*7 + (p-1)*9 <= 512$$

$$16p <= 528$$

Nel B-tree il valore massimo $p_{foglia}=33$

Esercizio: calcolo record indicizzati B-tree

- Si supponga di costruire un B-tree su un campo di ricerca V (9 byte), usando blocchi B (512 byte), con puntatore al blocco P (6 byte) e puntatori ai record Pr (7 byte). Si ipotizzi, inoltre, che ogni nodo sia completo al 70% (fattore riempimento)
- Quanti record possono essere indicizzati da un B-tree a 3 livelli?
- Con riempimento 70%: $p_{interno}$=24*0.70= 16 p_{foglia}=33*0.70= 23

Livello	#nodi	#valori di chiavi	# puntatori record
Root	1	(16-1)=15	15
1 liv.	16	16 x 15= 240	240
2 liv.	16^2= 256	256 x 15 = 3840	3.840
3 liv./foglie	16^3 =4096	4096 x 23= 94208	94.208

Totale nodi: 4369 (B^+-tree, 12720)

Puntatori record : 98303 (B^+-tree, 255.507)