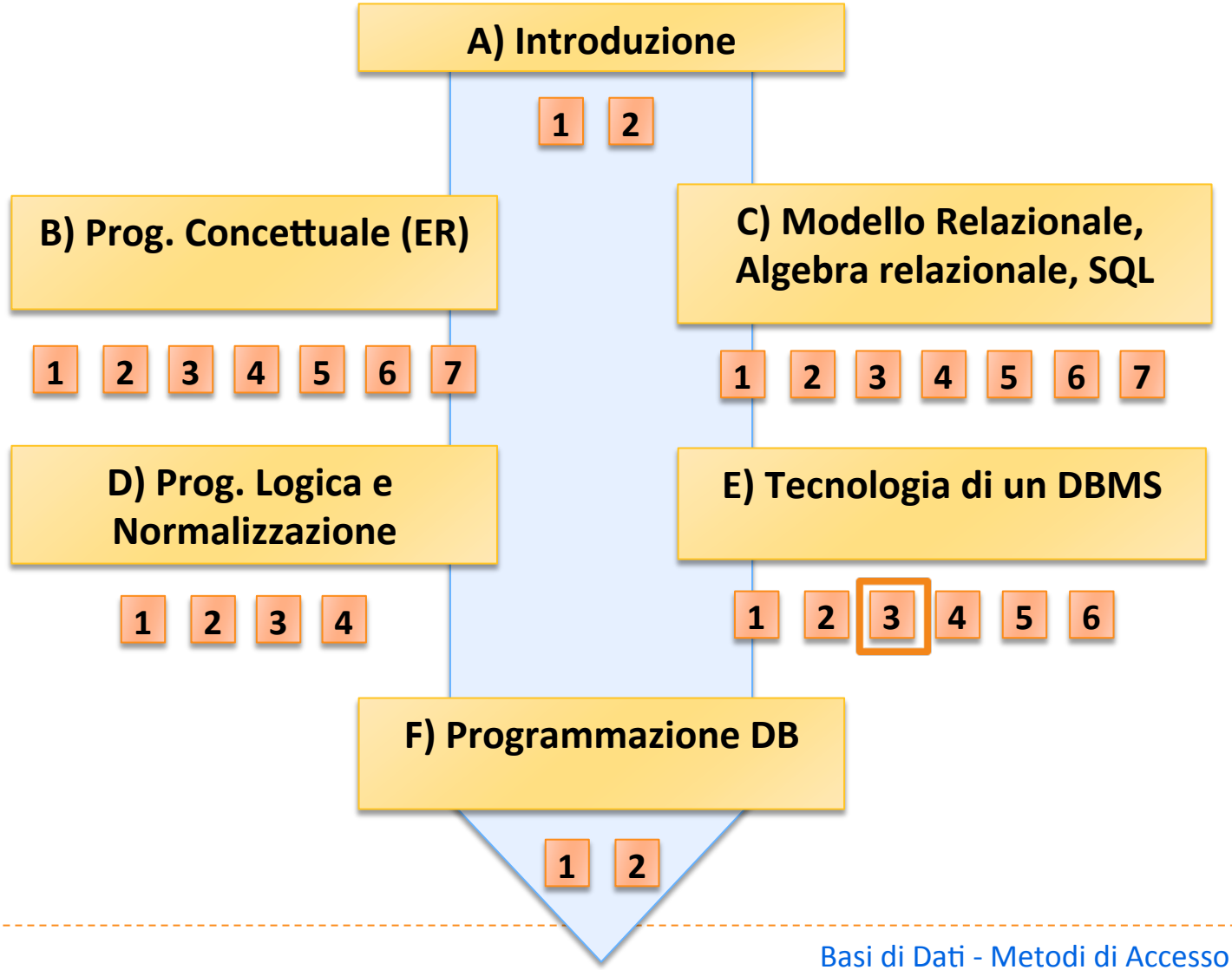
Basi di Dati

Metodi di Accesso (Indici)

Basi di Dati – Dove ci troviamo?



Nelle lezioni precedenti

- **Avete visto:**
  - **le caratteristiche delle principali unità di memoria permanente**
  - **la struttura dei file**
  - **il loro ordinamento**

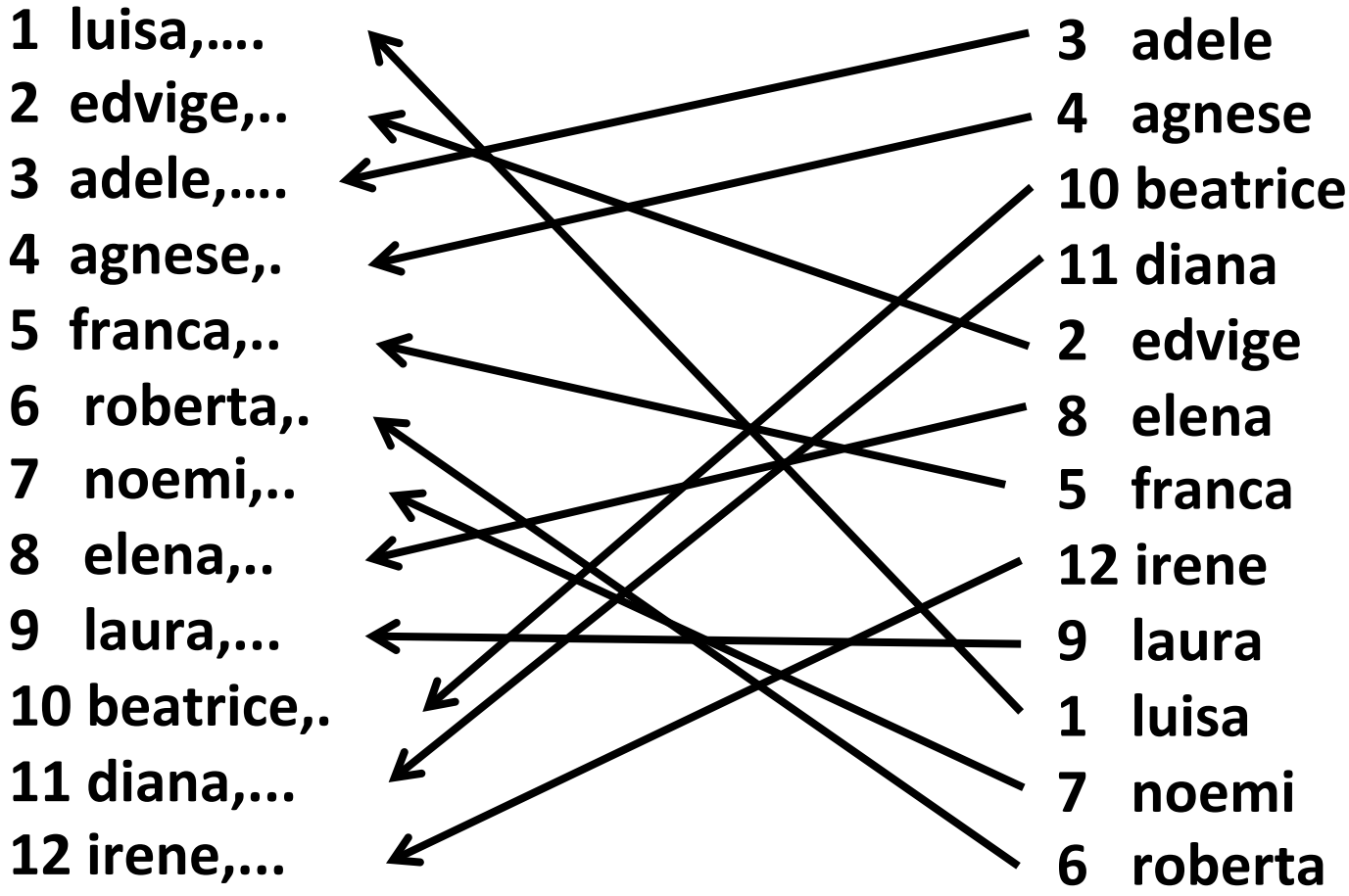
In questa lezione

- **Presenteremo:**
  - **le caratteristiche principali degli indici per l'accesso alle tuple (record)**

Organizzazioni con indice

- **IDEA DI BASE : associare ad un file una tabella nella quale l'entrata i-esima memorizza una coppia del tipo (k_i,p_i) dove :**
  - **k_i è un valore di chiave**
  - **p_i è un riferimento al record (tupla) con valore di chiave k_i**
  - **l'indice è ordinato: le coppie (k_i ,p_i) sono ordinate in base ai valori di k**

Organizzazioni con indice

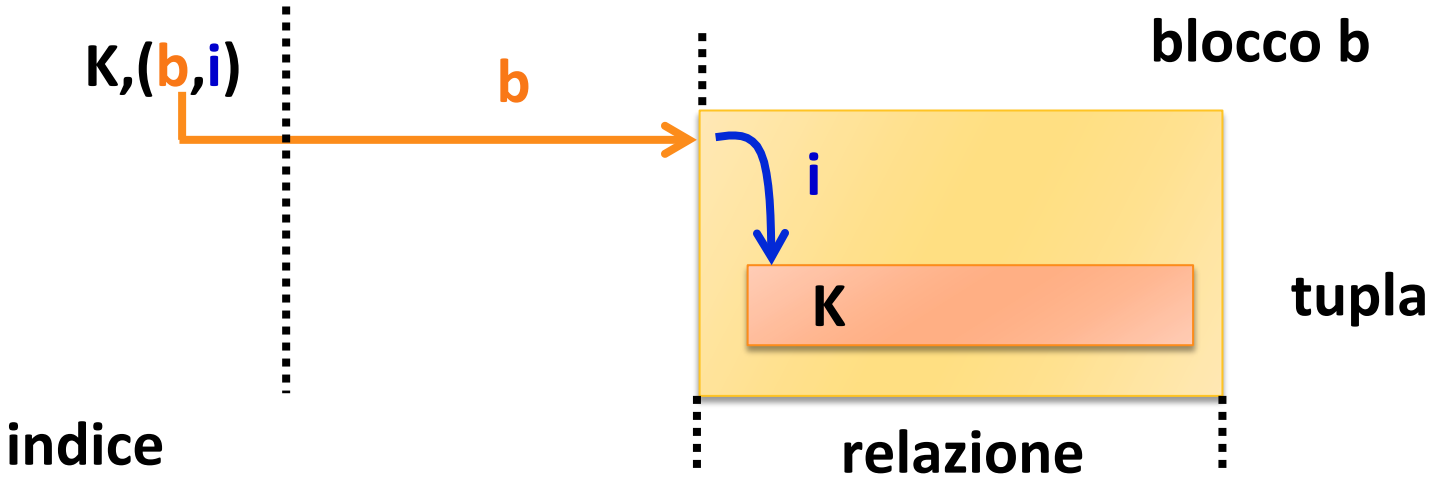


Organizzazioni con indice

- Nell'esempio abbiamo mostrato solamente la colonna chiave e la posizione della tupla, mentre l'indice è un file costituito da mini-tuple ed occupa generalmente uno spazio di memoria di circa 5÷20% della relazione
- ordinare il file indice è quindi più veloce
- l'indice può essere costruito su attributi qualsiasi (anche non chiave nel senso relazionale)
- l'indice fornisce una visione ordinata della relazione anche su colonne non di ordinamento

Organizzazioni con indice

- I riferimenti (4÷8 byte) vengono denominati: indirizzi, puntatori o, più comunemente, TID (tuple identifier)
- sono costituiti da una coppia: id. di blocco (pagina), indirizzo nel blocco:



L'indirizzo nel blocco è a sua volta indiretto

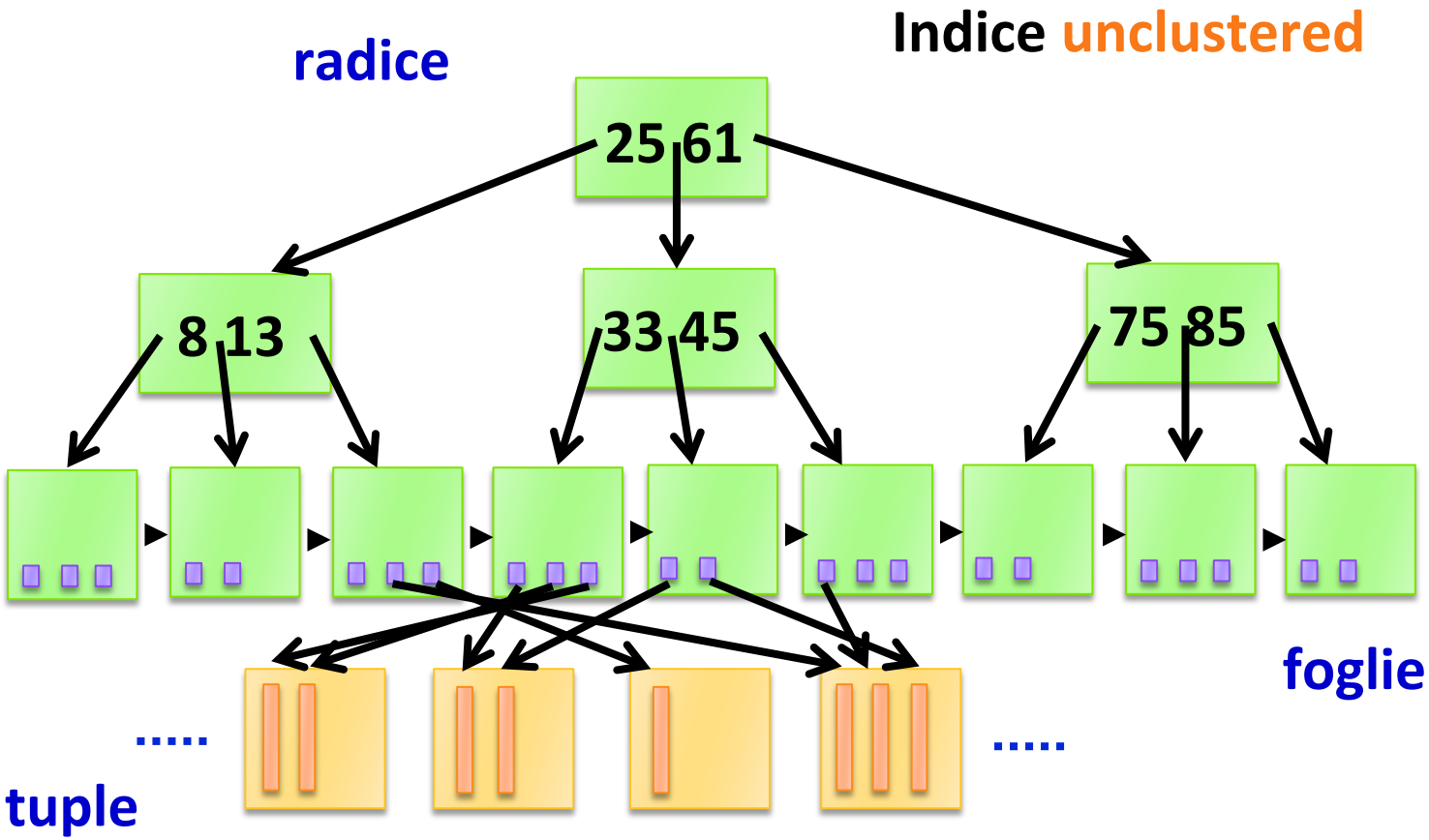
Organizzazioni con indice

- **Metodo di accesso**
  data una chiave K :
  1. accesso all'indice
  2. ricerca della coppia (K,(b,i))
  3. accesso al blocco dati b
  4. accesso alla tupla contenente K (indirizzo i)
- **le coppie (K,(b,i)) vengono collezionate in blocchi ed ordinate, questi blocchi si chiamano FOGLIE dell'indice e sono tra loro collegati da puntatori.**

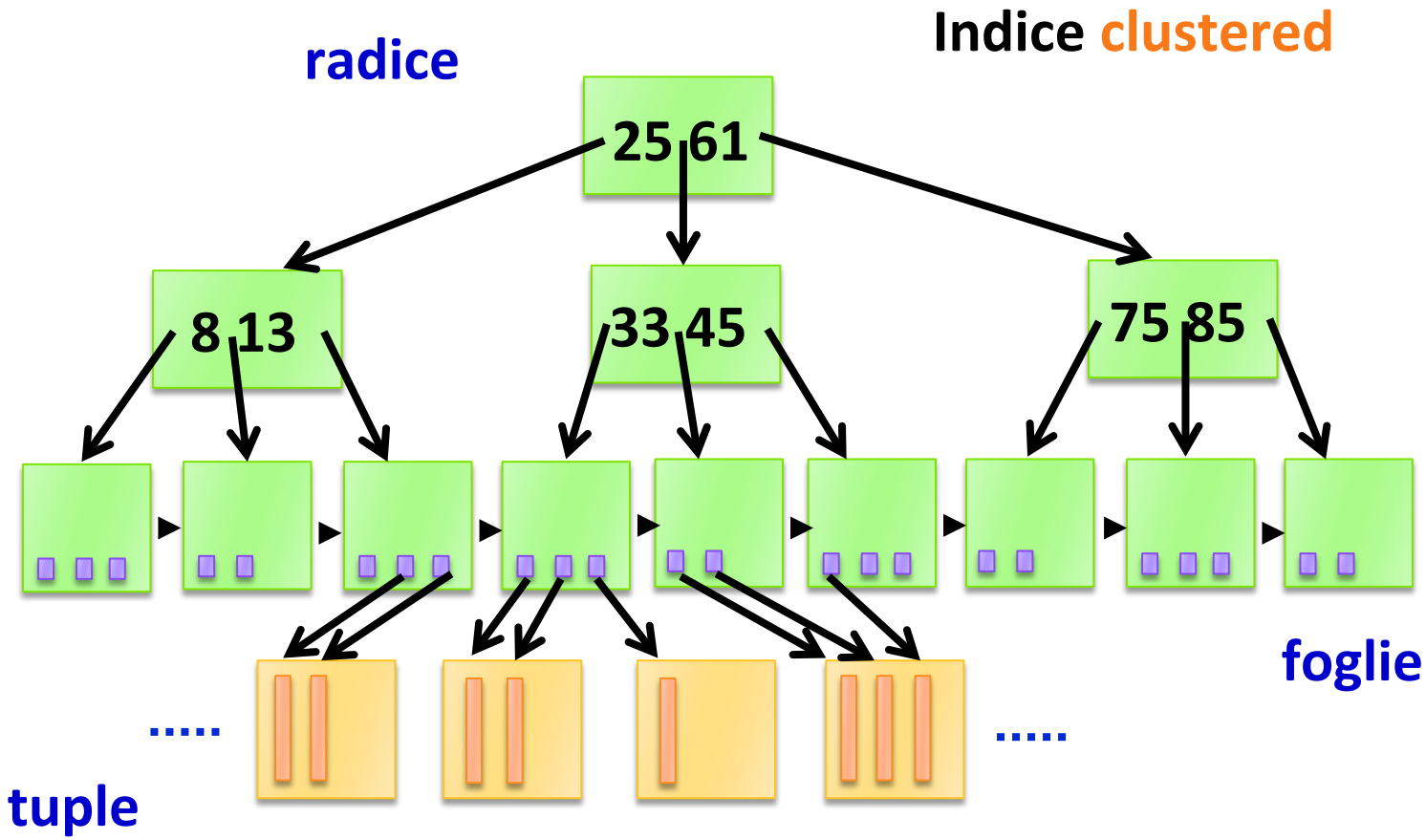
Organizzazioni con indice

- Al di sopra delle foglie si costruisce un'organizzazione ad albero a più livelli di indirizzamento:
  - ogni foglia è rappresentata da una coppia (K,b) nel livello immediatamente superiore (K è il valore minore),
  - i blocchi del livello sopra le foglie sono a loro volta rappresentati a livello superiore , e così via fino ad avere un solo blocco chiamato radice

Organizzazioni con indice



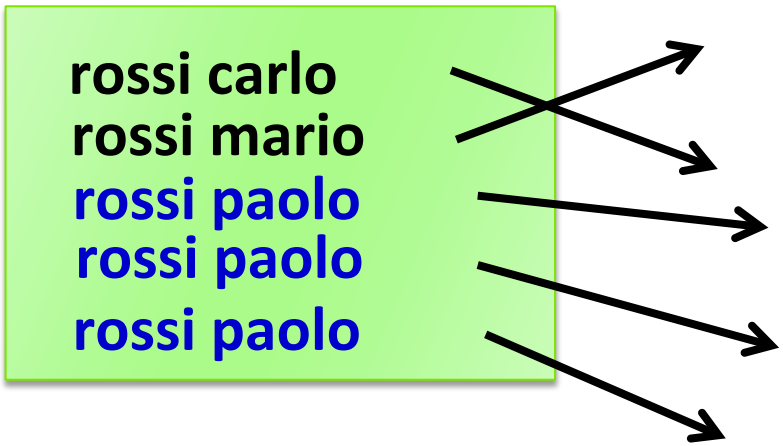
Organizzazioni con indice



Indici

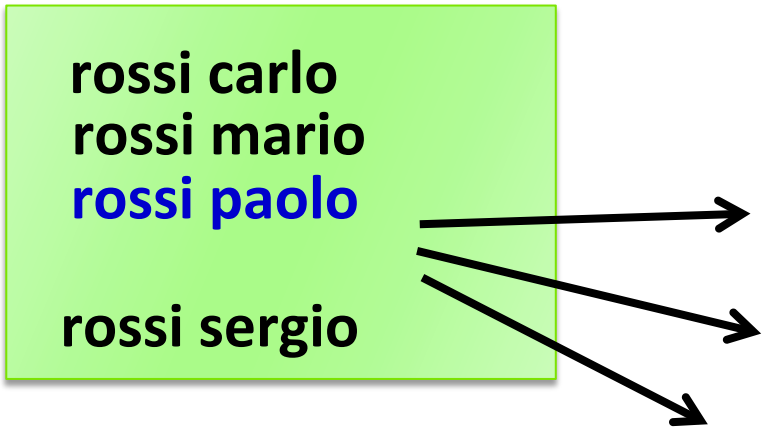
- L'indice si dice clustered se è costruito su un attributo di ordinamento, altrimenti unclustered
- su una relazione si possono costruire un indice clustered e più di un indice unclustered
- l'indice può essere anche multiattributo (multicolonna), ad es. K :<cognome,nome>
- nel caso di clausola UNIQUE l'indice garantisce la non ripetizione dei valori
- l'indice può essere costruito su attributi con valori ripetuti, in tal caso la coppia (K, p) diventa (K(p1, p2,…. pk)):

Indici

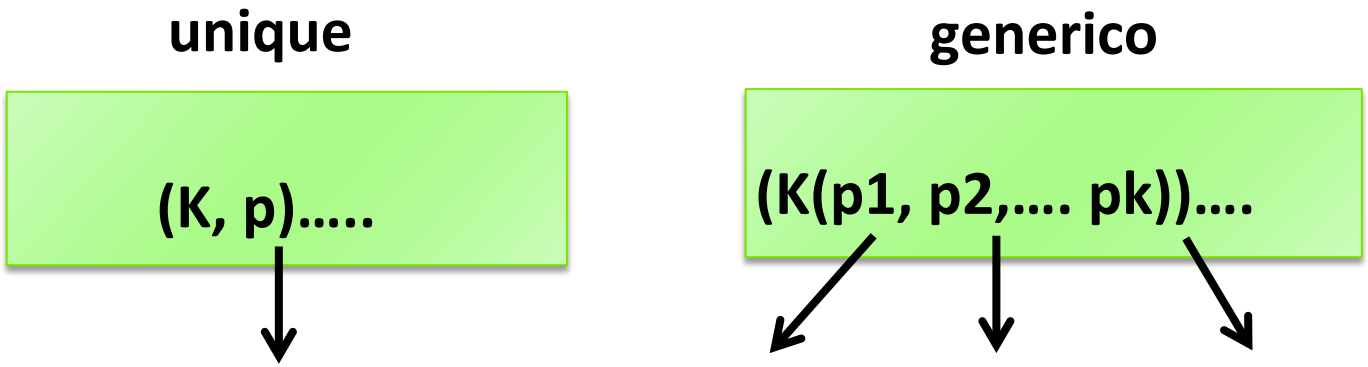


valori ripetuti

eliminazione dei valori ripetuti e riduzione di spazio



Indici



In SQL:

- CREATE {UNIQUE} INDEX nome-indice
  ON nome-relazione
  (nome-attributo { ASC | DESC })
- DROP INDEX nome-indice

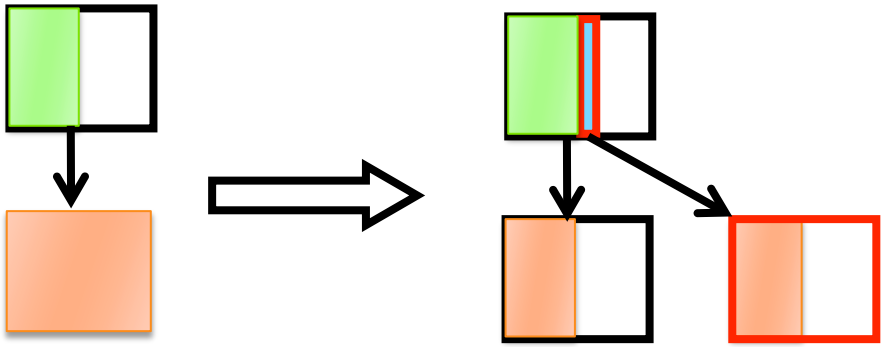
Indici B^+ tree

- **B^+ tree: l'albero è associato ad un SW di gestione che lo mantiene sempre equilibrato: ogni percorso dalla radice ad una foglia ha sempre la stessa lunghezza h, chiamata altezza del B^+ tree**
- **ogni blocco (nodo) intermedio ha sempre almeno O+1 e non più di 2×O+1 figli, contiene almeno O e non più di 2×O valori dell'attributo**
- **la radice può avere da 2 a non più di 2×O +1 figli, contiene da 1 a non più di 2×O valori**
- **O si chiama ordine del B^+ tree**
- **Le foglie sono riempite dal 50 al 100 %**

  (valore tipico: 69%)

Indici B+ tree

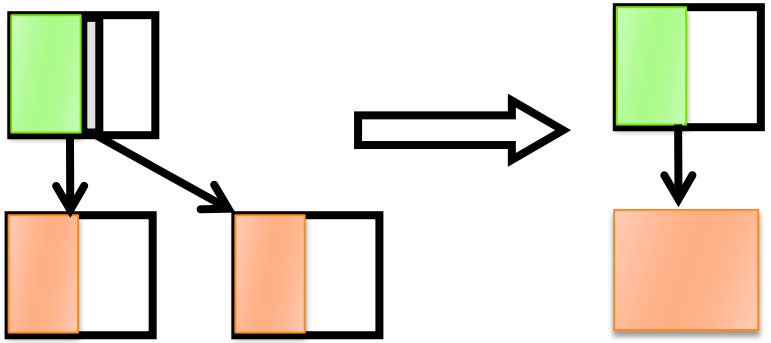
inserimento



block split
(si può propagare fino alla radice)

eliminazione

underflow
(si può propagare fino alla radice)



Prestazioni del B^+ tree

Facciamo riferimento a:

- **dimensione del blocco D (es. 4096)**
- **Dimensione del puntatore L(p) ai nodi intermedi (es. 4 per entrambi)**
- **dimensione L(k) del valore K**

Calcoliamo :

- **l'ordine O**
- **il numero di foglie NF**
- **l'altezza H**

Prestazioni del B^+ tree

Ordine O:

consideriamo il vincolo

$$2 \times O \times L(k) + (2 \times O + 1) \times L(p) \leq D$$

da cui

$$O = \lfloor (D - L(p)) / (2 \times (L(k) + L(p)) \rfloor$$

- **con una dimensione L(k) di 10 si ha**

  O = 146 e 4 byte inutilizzati per nodo

- **con una dimensione L(k) di 40 si ha**

  O = 46 e 44 byte inutilizzati per nodo

Prestazioni del B^+ tree

Numero di foglie NF:

- dipende dal numero di tuple NT e dal fattore di riempimento medio dei blocchi = ln 2 ≈ 69 %
- ricordiamo che nelle foglie si avranno un numero di p = NT ed un numero di valori NK da cui

$$NF = \lceil (NK \times L(k) + NT \times L(p)) / (D \times \ln 2) \rceil$$

- con una dimensione $L(k) = 10$, $NT = 10^6$,

  $NK = 10^4$ si ha: $NF = 5776$

Prestazioni del B^+ tree

Altezza minima:

- **l'altezza minima Hmin si può ottenere quando tutti i nodi sono pieni,**

  poiché al penultimo livello sono necessari NF puntatori alle foglie, si ha:

$$NF \leq (2 \times O + 1)^{Hmin-1}$$ **e quindi**

$$Hmin = 1 + \lceil \text{Log}_{(2 \times O + 1)} NF \rceil$$

Prestazioni del B^+ tree

Altezza massima:

- **l'altezza massima Hmax si può ottenere quando tutti i nodi sono pieni per metà e la radice contiene due soli puntatori, poiché al penultimo livello sono necessari NF puntatori alle foglie, si ha:**

$$NF \leq 2 \times (O + 1)^{Hmax-2}$$

e quindi

$$Hmax = 2 + \lfloor \text{Log}_{(O+1)} (NF / 2) \rfloor$$

Prestazioni del B^+ tree

- Esempio

$D = 1024$, $L(p) = 4$, $NT = 10^6$

	$L(k) = 10$			$L(k) = 20$		
NK	NF	Hmax	Hmin	NF	Hmax	Hmin
10^3	5650	4	4	5664	4	4
10^4	5776	4	4	5918	4	4
10^5	7045	4	4	8454	4	4
10^6	19725	4	4	33814	5	4

si tratta in generale di alberi abbastanza bassi

Prestazioni del B^+ tree

- Continuando l'esempio precedente, con NB= 10^5, 10^6 tuple, 10 per blocco L(k) = 10
  - $C_{seq} = 5 \times 10^4$
  - $C_{sort} = 1.2 \times 10^6$ e $C_{bin} = 17$
  - $C_{ind} = 4 + 1 = 5$, bisogna sommare l'accesso al blocco che contiene la tupla cercata (+1)
- si può anche ipotizzare che se le richieste sono molte, durante l'esercizio, la radice ed i livelli superiori siano già in memoria:
  - $C_{ind} = 1 + 1 = 2$

Prestazioni del B^+ tree

- **costo di costruzione dell'indice**:
- lettura della relazione,
- estrazione delle coppie, costruzione delle foglie
- sort/merge delle stesse
  - $C_{indice} = NB + NF + 2 \times NF \times (PM+1)$

  continuando l'esempio precedente, con M e NM = 8 ed inglobando il passo di sort nella estrazione delle coppie:

  $C_{indice} = NB + NF + 2 \times NF \times PM$

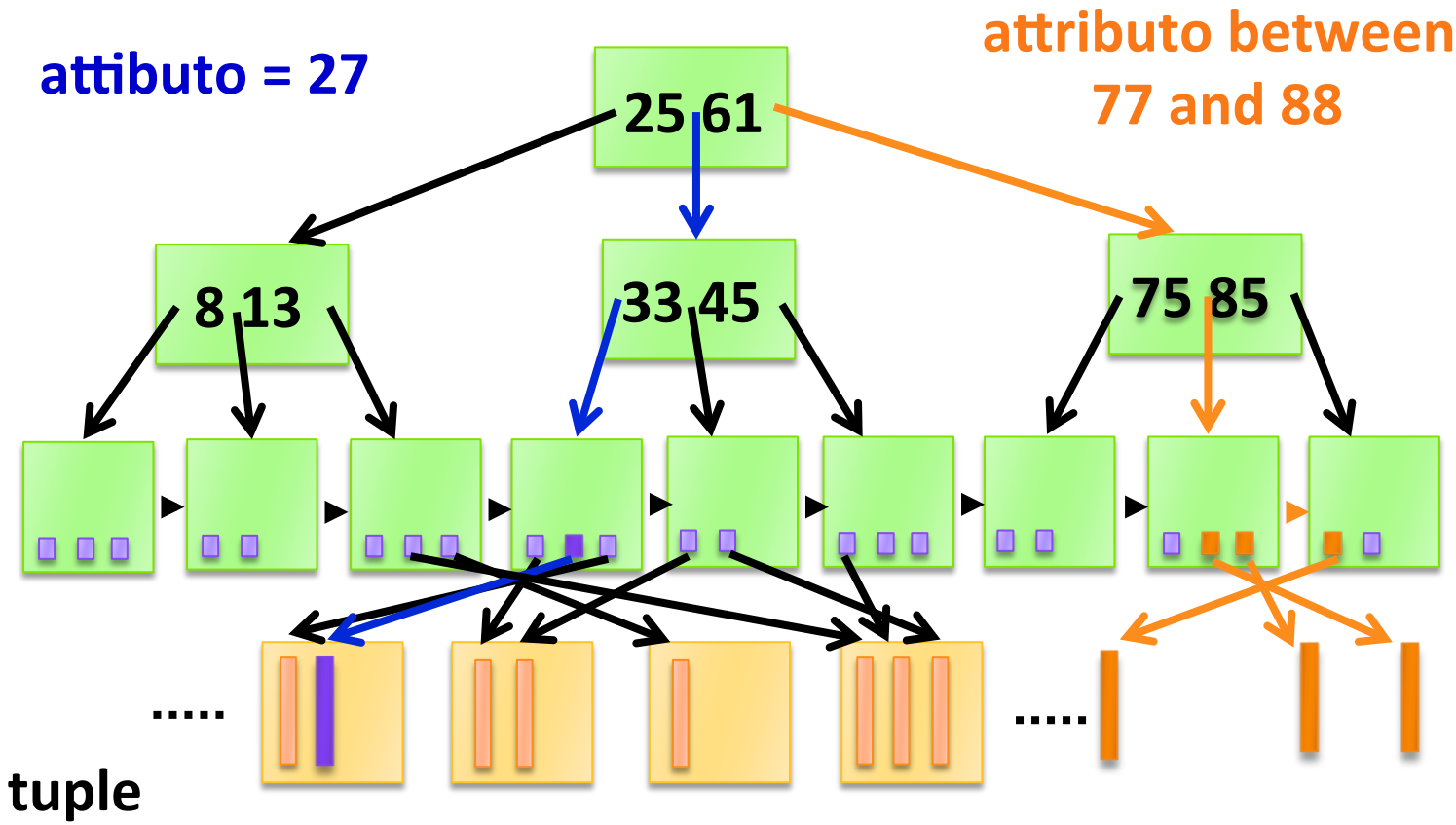
  $C_{indice} = 10^5 + 1.98 \times 10^5 \approx 3 \times 10^5$

Prestazioni del B^+ tree

Utilizzo dell'indice:

- **accesso veloce alla tupla per K unique**
- **accesso veloce alle tuple per K replicata**
- **accesso veloce alle tuple per range di K**
  (K < valore, K BETWEEN k1 AND k2)
- **facilita ORDER BY e GROUP BY**

Organizzazioni con indice



Compressione della chiave

- **FORWARD COMPRESSION**
  - **Si registrano: il numero di caratteri uguali ed i caratteri finali che differenziano dal valore precedente.**
  - **Es:**

JOHNSON	0 JOHNSON
JOHNSTON	5 TON
JOHNSTOWN	7 WN
JOHNTON	4 TON

- **REAR COMPRESSION (Riporta i caratteri iniziali) è inefficace**