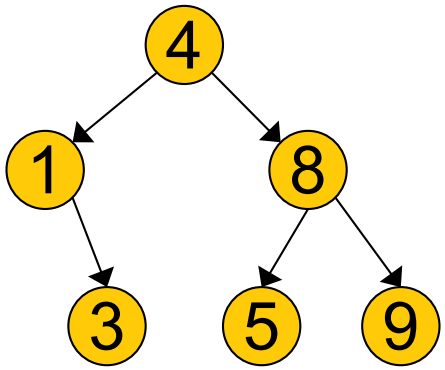
# Indici, problemi

- Tutte le strutture di indice viste finora sono basate su strutture ordinate, 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

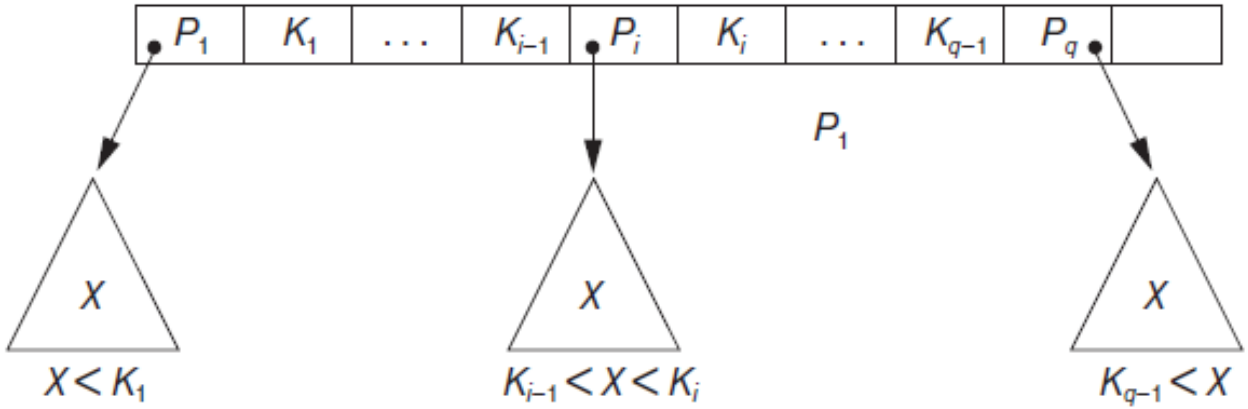



tempo di ricerca è pari alla profondità dell'albero

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

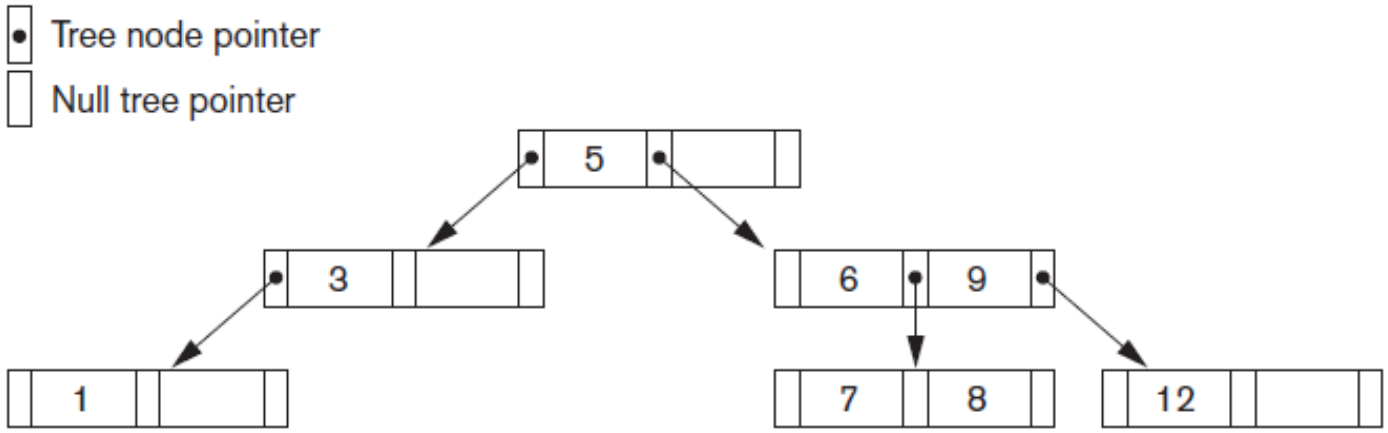
# Albero di ricerca di ordine p

- 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);
    - Un puntatore $P_i$ punta ad un sottoalbero i cui valori X 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

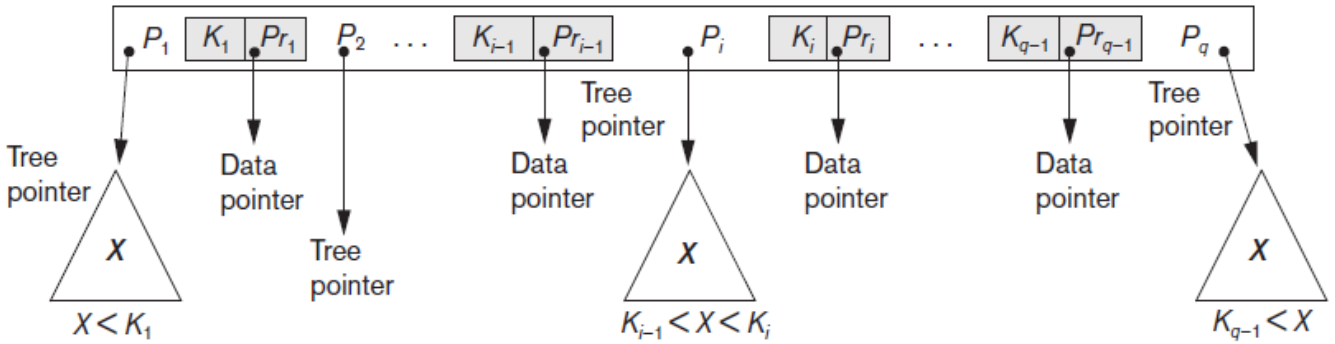
- Tempo di ricerca, dipende dalla profondità dell'albero
  - Un albero bilanciato (nodi foglia tutti allo stesso livello) garantisce un tempo di ricerca uniforme

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

# B-tree di ordine p

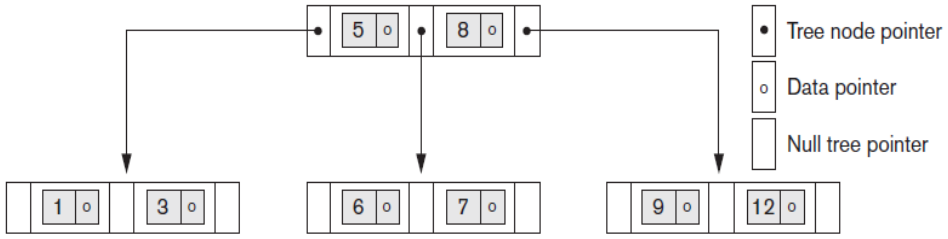
Un B-tree di ordine p su campo chiave K

- ogni nodo interno deve contenere almeno ⌈p/2⌉ puntatori ed ha la seguente forma:




  - $P_i$ puntatore ad un sottoalbero
  - $Pr_i$ puntatore ad al record (e.g., puntatore al blocco che contiene il record)
  - $K_i$ valore del campo chiave
- Tutti i nodi foglia sono tutti allo stesso livello, hanno una struttura simile al nodo interno, tolti i puntatori ai sottoalberi

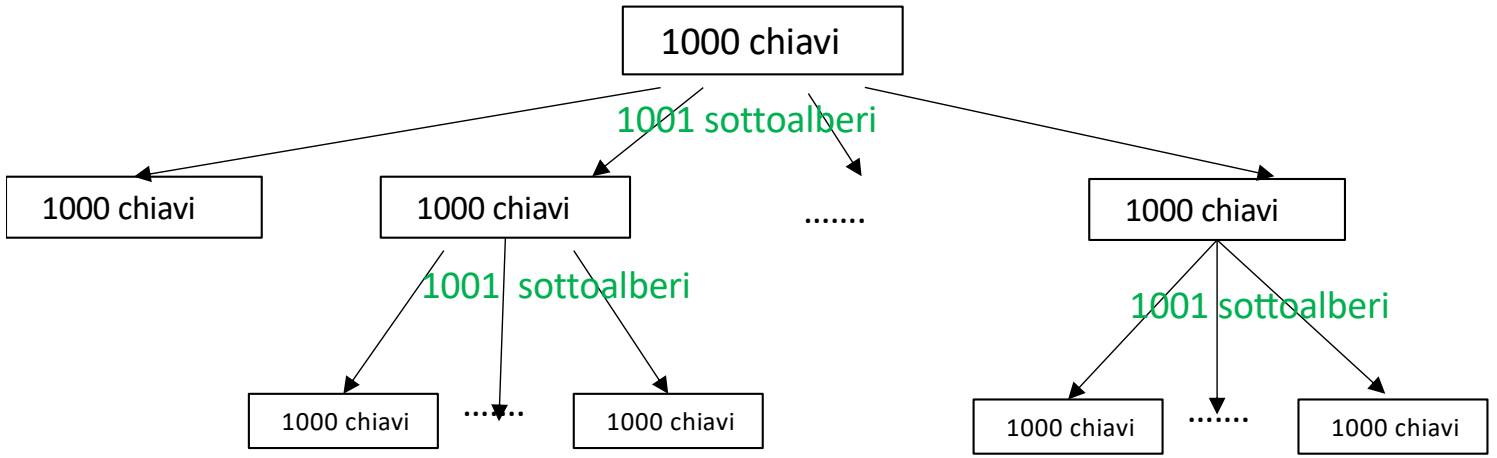
Esempio



# 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 | #nodi | #chiavi |
|---|---|---|
| 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 rappresenta 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://www.cs.usfca.edu/~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. | $100^2$ | 99x$100^2$ = 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

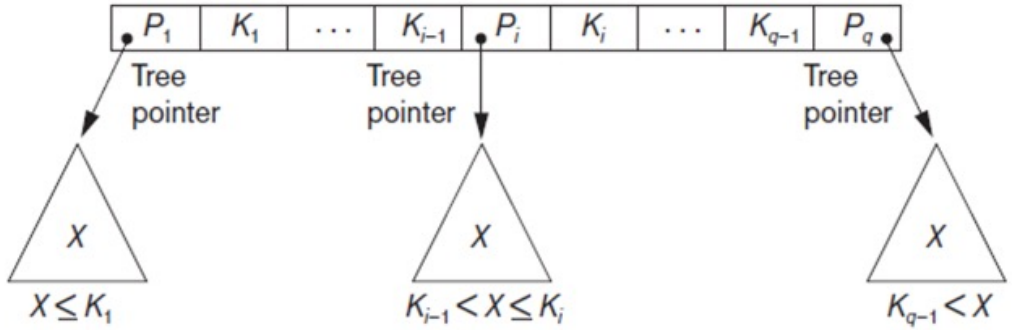
# $B^+$ -tree

- $B^+$ -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 sottoalberi
    - 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

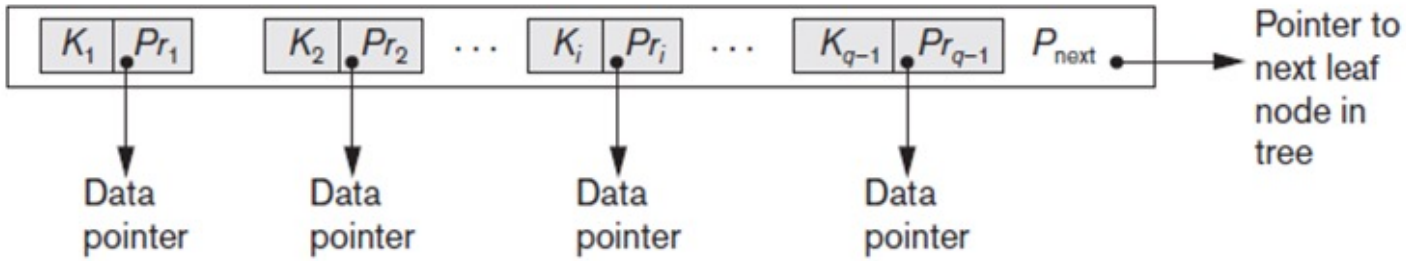
# $B^+$-tree di ordine p

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

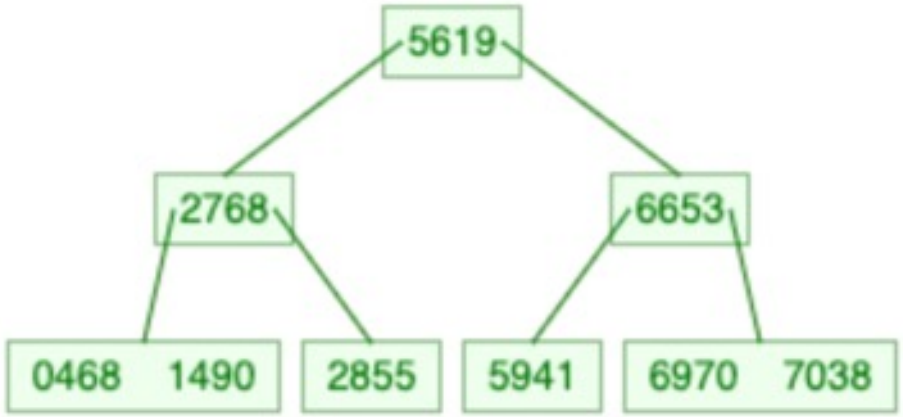
- Non vedremo nel dettaglio gli algoritmi, ma potete trovare un simulatore qui

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