Indice di file

- **Struttura ausiliaria** per **l'accesso** (efficiente) ai record di un file sulla base dei **valori di un campo** (o di una "concatenazione di campi") detto **chiave** (o, meglio, **pseudo-chiave**, perché non è necessariamente identificante)
- Idea fondamentale: **l'indice analitico di un libro**
  - **Lista** di coppie (termine, pagina), **ordinata** alfabeticamente sui termini, posta in fondo al libro e separabile da esso
- Un indice I di un file f è un altro file, con record a **due campi**: **chiave** e **indirizzo** (dei record di f o dei relativi blocchi), ordinato secondo i valori della chiave

Tipi di indice

- **Indice primario**:
  - su un **campo** sul cui ordinamento è **basata la memorizzazione**
  - detti anche **indici di cluster**, anche se talvolta si chiamano primari quelli su una chiave identificante e di cluster quelli su una chiave identificante
- **Indice secondario**:
  - su un **campo** con ordinamento **diverso da quello di memorizzazione**
- **Indice denso**:
  - contiene un record per ciascun valore del campo chiave
- **Indice sparso**:
  - contiene un numero di record inferiore rispetto al numero di valori diversi del campo chiave

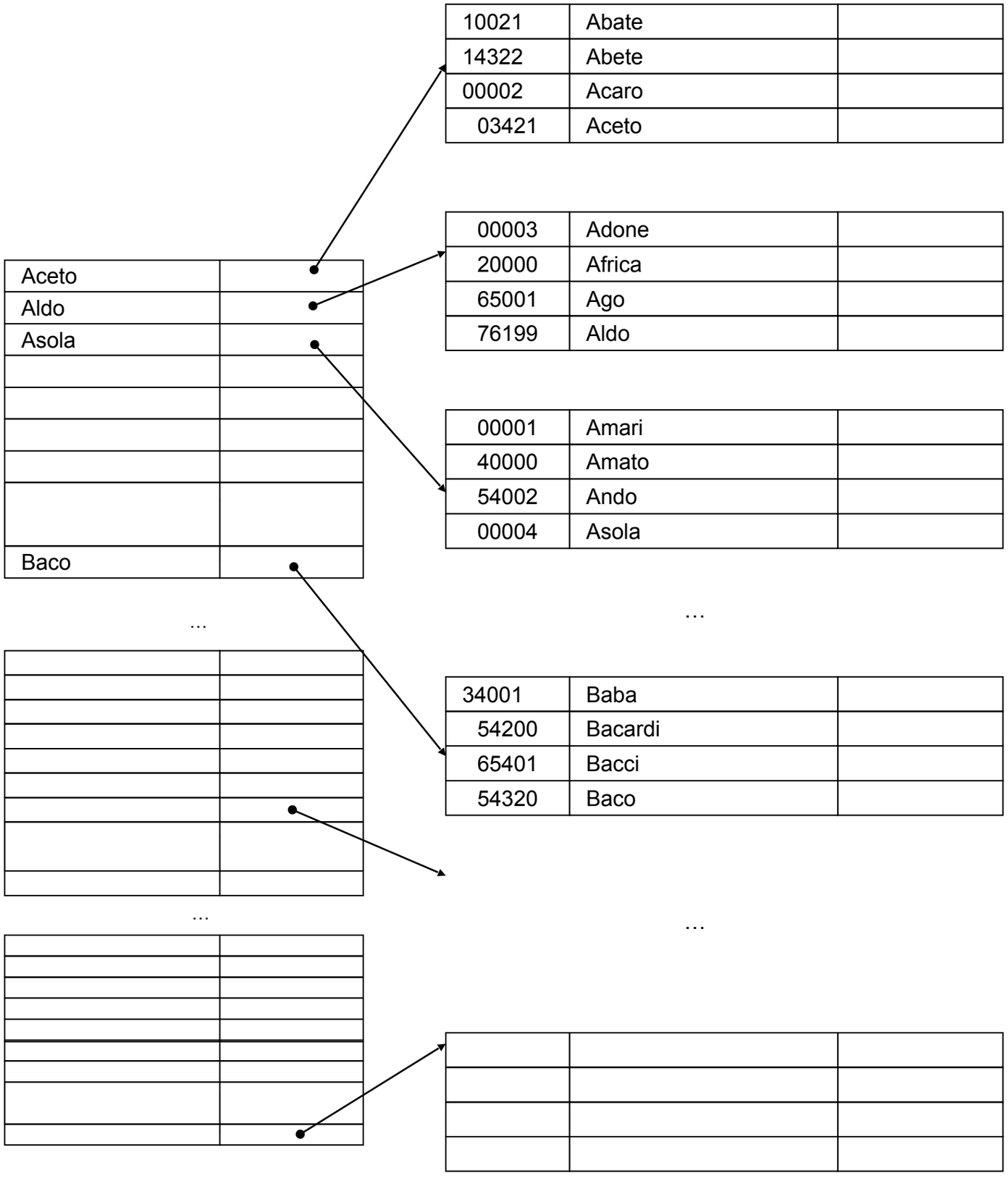
Caratteristiche degli indici

- **Accesso diretto** (sulla chiave) **efficiente**
  - sia **puntuale** sia per **intervalli**
- **Scansione sequenziale** ordinata **efficiente**
- **Modifiche** della chiave, **inserimenti**, **eliminazioni** **inefficienti** (come nei file ordinati)
- **Tecniche per alleviare i problemi**:
  - marcatura per le eliminazioni
  - riempimento parziale
  - blocchi collegati (non contigui)
  - riorganizzazioni periodiche

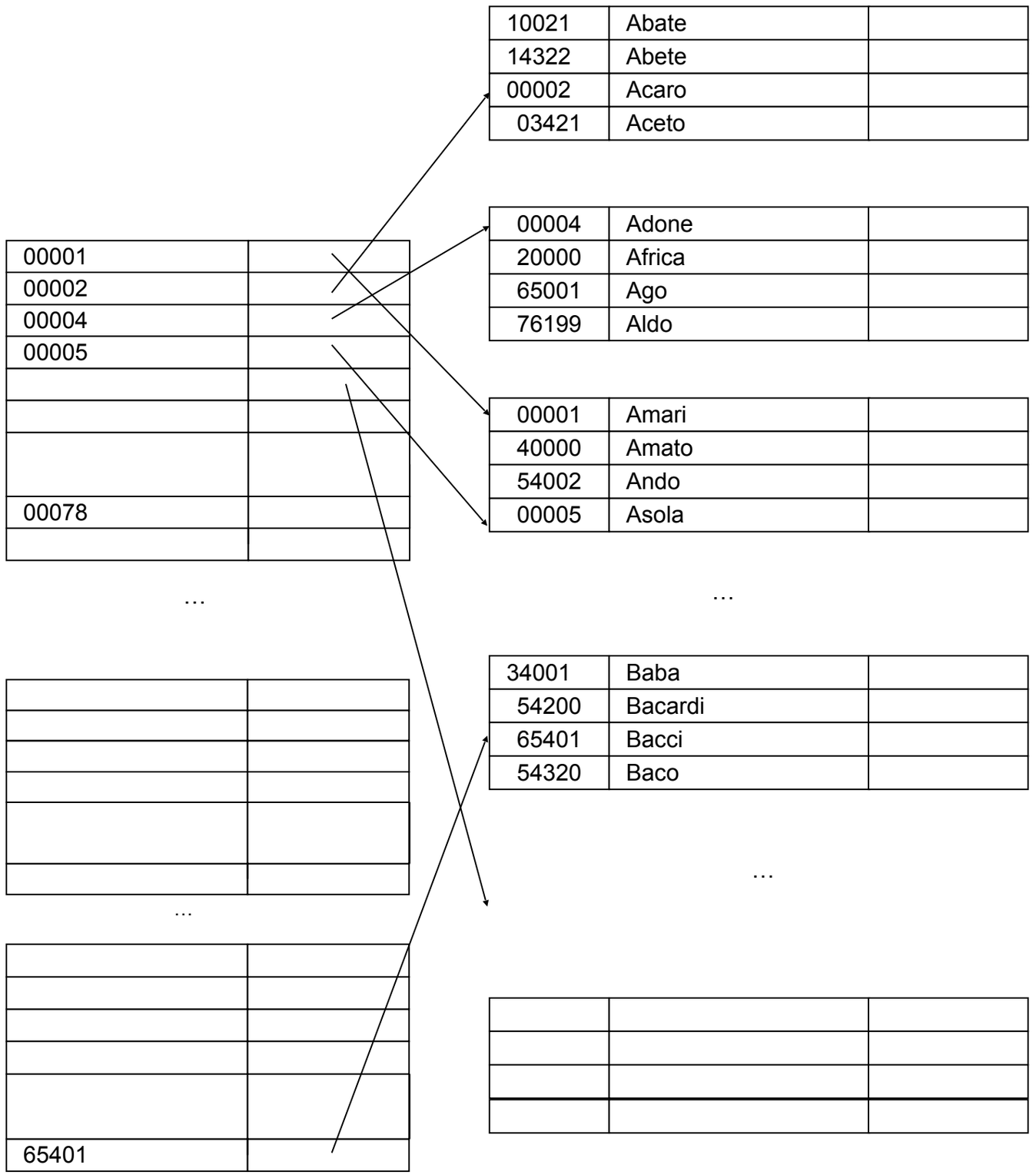
Commenti

- Un indice **primario può essere sparso**
  - **Non tutti i valori** della chiave compaiono nell'indice
  - Esempio, sempre rispetto ad un libro: indice generale
- Un indice **secondario deve essere denso**
  - **Tutti i valori** della chiave secondaria devono essere raggiungibili
  - Esempio, sempre rispetto ad un libro: indice analitico
- Ogni file può avere **al più un indice primario** e **un numero qualunque di indici secondari** (su campi diversi)
  - Esempio: una guida turistica può avere l'indice dei luoghi e quello degli artisti
- Un **file hash non può avere un indice primario**

Indice Primario



Indice Secondario



Dimensioni dell'indice

- T = numero di record nel file
- B = dimensione dei blocchi
- R = lunghezza dei record (fissa)
- K = lunghezza del campo chiave
- P = lunghezza degli indirizzi (ai blocchi)

- Numero di blocchi per il file (circa):

$$N_B = T \times R/B$$

- Numero di blocchi per un indice denso:

$$N_D = T \times (K + P)/B$$

- Numero di blocchi per un indice sparso:

$$N_S = N_B \times (K + P)/B$$

Indici Secondari

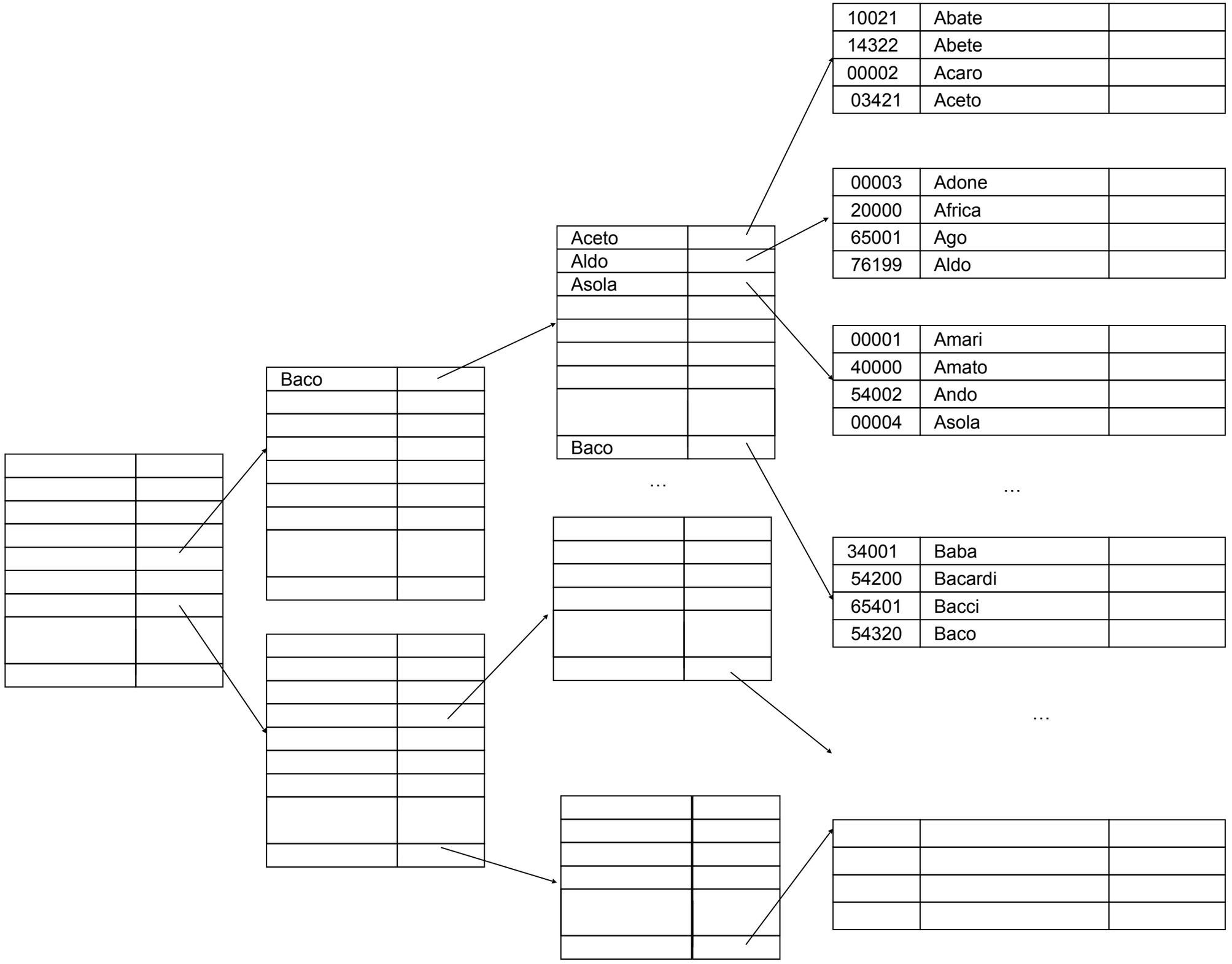
- Si possono usare, **puntatori ai blocchi** oppure **puntatori ai record**
  - I puntatori ai **blocchi** sono **più compatti**
  - I puntatori ai **record** permettono di **semplificare alcune operazioni** (effettuate solo sull'indice, senza accedere al file se non quando indispensabile)

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; i **livelli** sono di solito **abbastanza pochi**, perché:
  - l'indice è ordinato, quindi **l'indice sull'indice è sparso**
  - i record dell'indice sono piccoli
- Numero di blocchi al livello j dell'indice (circa):

$$N_j = N_{j-1} \times (K + P)/B$$

Indice Primario Multilivello



Indice Secondario Multilivello

10021	Abate	
14322	Abete	
00002	Acaro	
03421	Aceto	

00004	Adone	
20000	Africa	
65001	Ago	
76199	Aldo	

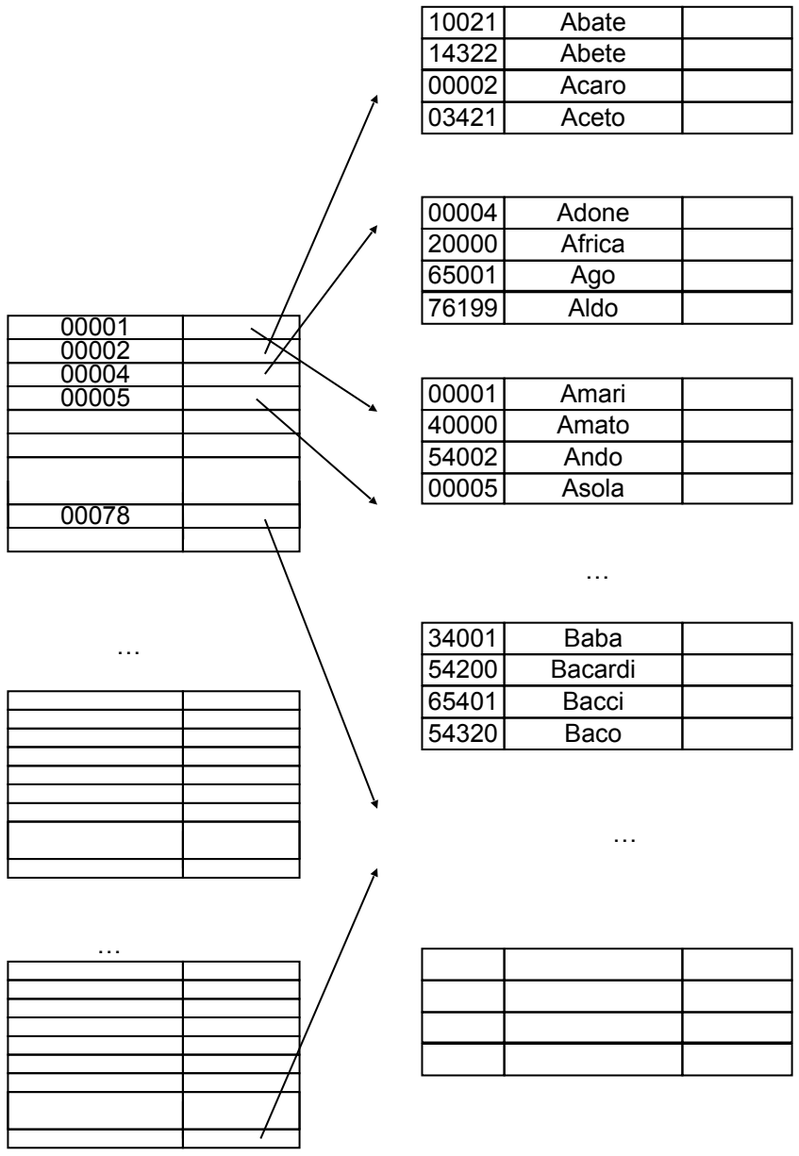
00001	Amari	
40000	Amato	
54002	Ando	
00005	Asola	

...

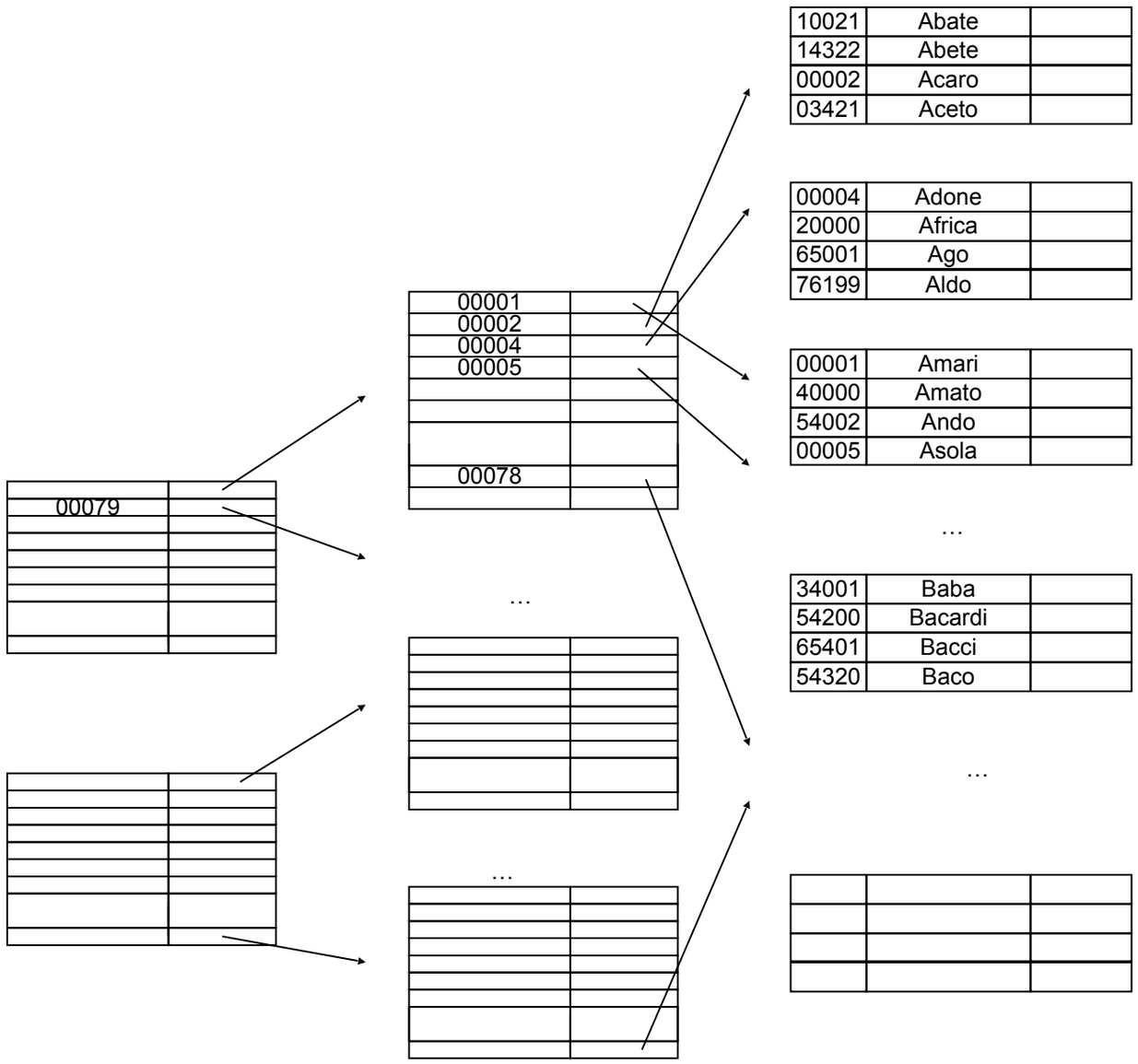
34001	Baba	
54200	Bacardi	
65401	Bacci	
54320	Baco	

...

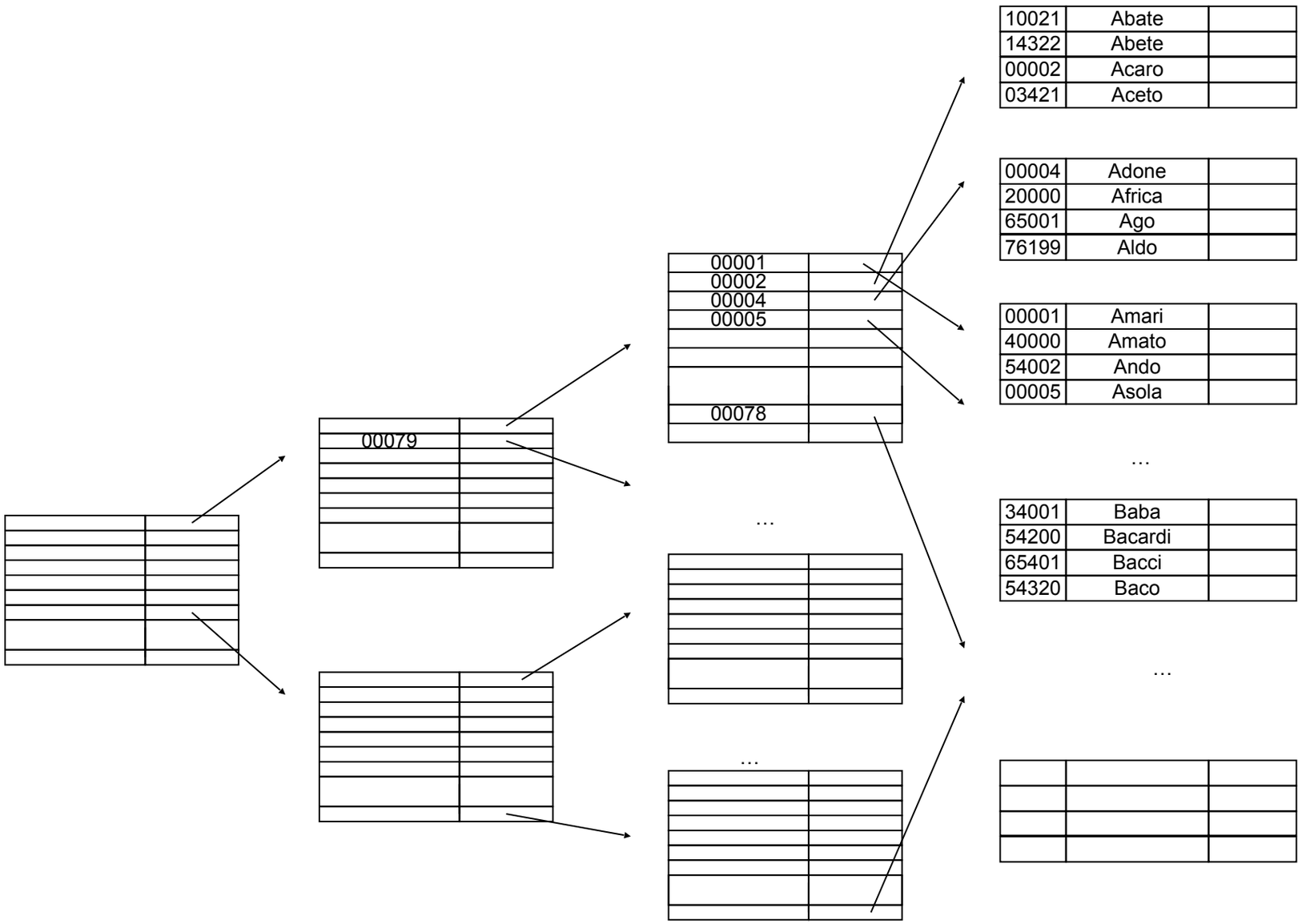
Indice Secondario Multilivello



Indice Secondario Multilivello



Indice Secondario Multilivello

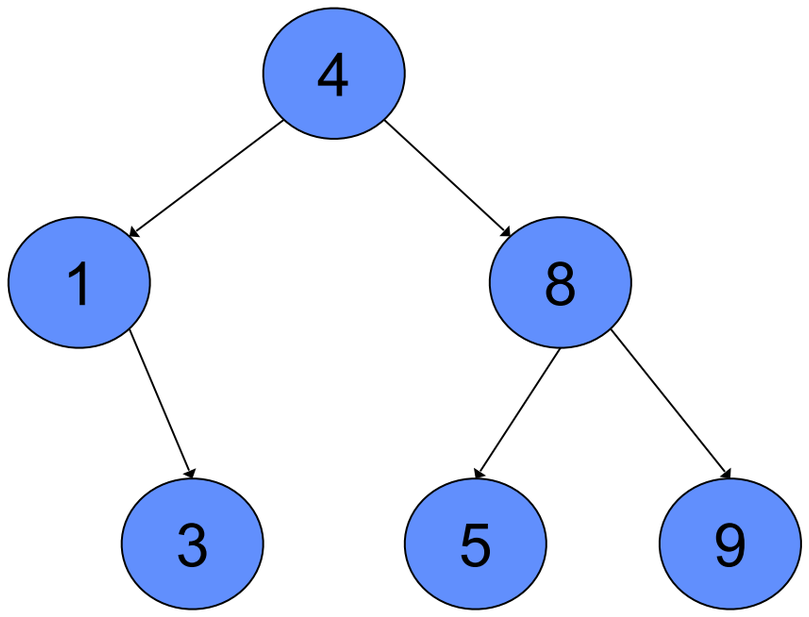


Problemi degli indici

- Le strutture di indice basate su strutture ordinate sono poco flessibili in presenza di elevata dinamicità
- Gli indici utilizzati dai DBMS sono in generale
  - indici dinamici multilivello efficienti anche in caso di aggiornamenti
- Vengono memorizzati e gestiti come B-tree (intuitivamente: alberi di ricerca bilanciati)
  - Alberi binari di ricerca
  - Alberi n-ari di ricerca
  - Alberi n-ari di ricerca bilanciati

Albero binario di ricerca

- Albero binario **etichettato** in cui per ogni nodo il **sotto-albero sinistro** contiene solo **etichette minori** di quella del nodo e il **sotto-albero destro** solo **etichette maggiori o uguali** di quella del nodo
- **Tempo di ricerca** (e inserimento) **pari alla profondità**:
- **logaritmico** nel caso "medio" (assumendo un ordine di inserimento casuale)



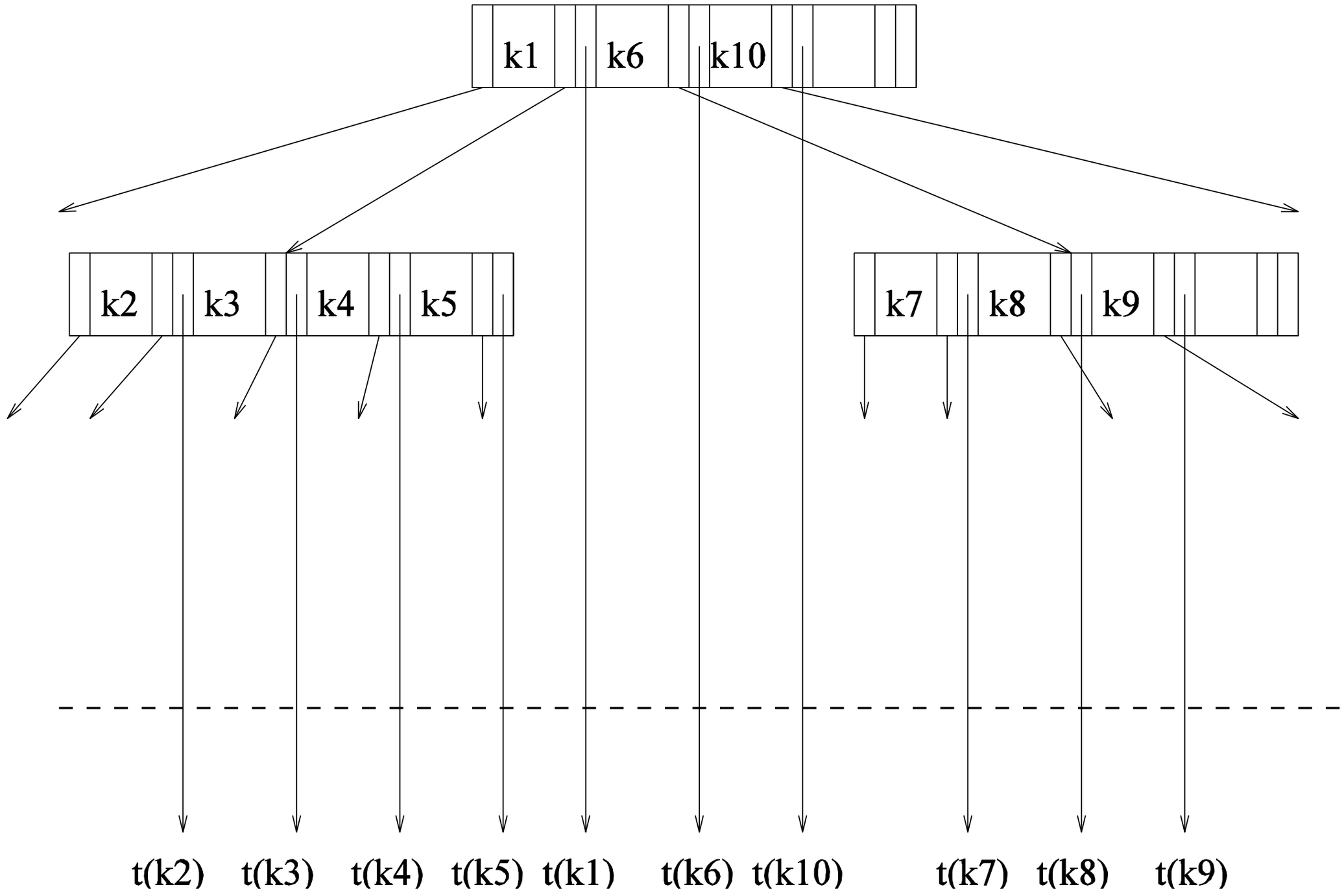
Albero di ricerca di ordine P

- Ogni nodo ha **(fino a)** P **figli** e **(fino a)** P **etichette**, **ordinate**
- Nell'i-esimo sotto-albero abbiamo tutte etichette maggiori o uguali della $(i-1)$-esima etichetta e minori della i-esima
- Ogni ricerca o modifica comporta la visita di un **cammino radice-foglia**
- In **strutture fisiche**, un **nodo** può corrispondere ad un **blocco**
- Il **legame** tra nodi è dato da **puntatori** che collegano le **pagine**
- Ogni nodo ha numero di discendenti abbastanza grande per cui gli alberi hanno un **numero limitato di livelli**; la maggior parte delle pagine è nei nodi foglia
- Gli **alberi** sono **bilanciati**

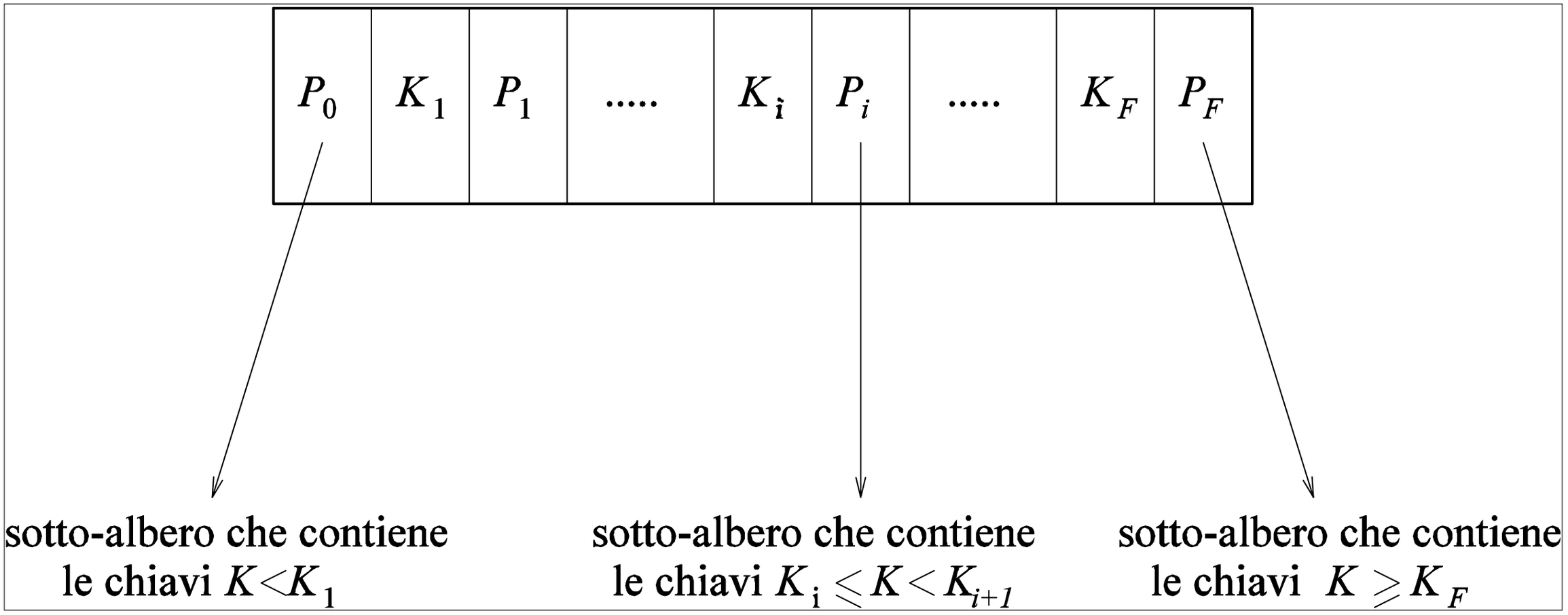
B-Tree

- Un B-tree è un albero di ricerca che viene mantenuto bilanciato, grazie a:
  - Riempimento parziale (mediamente 70%)
  - Riorganizzazioni (locali) in caso di sbilanciamento

Esempio di B Tree



Organizzazione dei nodi nel B-Tree



- Sequenza di F valori ordinati di chiave
- Ogni etichetta K_i seguita da un puntatore P_i
- F dipende dall'ampiezza della pagina e dalla dimensione occupata dai valori di chiave e di puntatore

Ricerca nel B-Tree

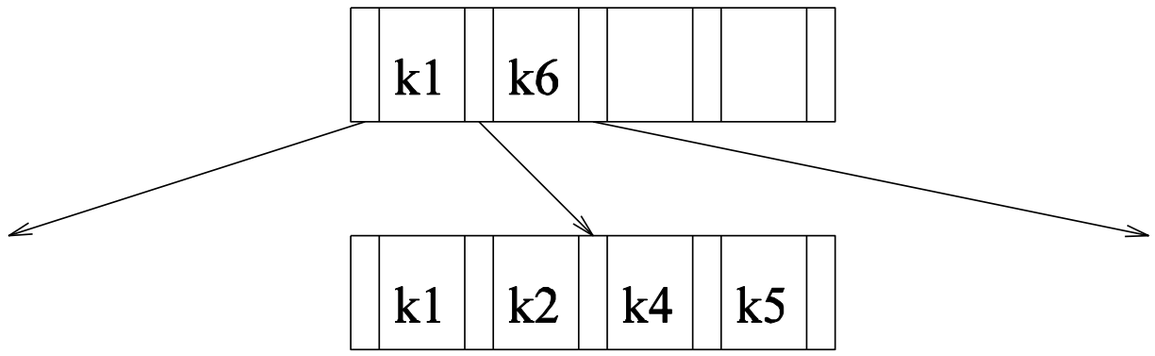
- Dato un valore V
  - Si seguono i puntatori partendo dalla radice
  - Ad ogni nodo intermedio
    - Se $V < K_1$ si segue il puntatore P_0
    - Se $V \geq K_F$ si segue il puntatore P_F
- Altrimenti si segue il puntatore P_j tale che $K_j \leq V < K_{j+1}$
- La ricerca prosegue fino ai nodi foglia dell'albero

Inserimenti ed eliminazione nel B-Tree

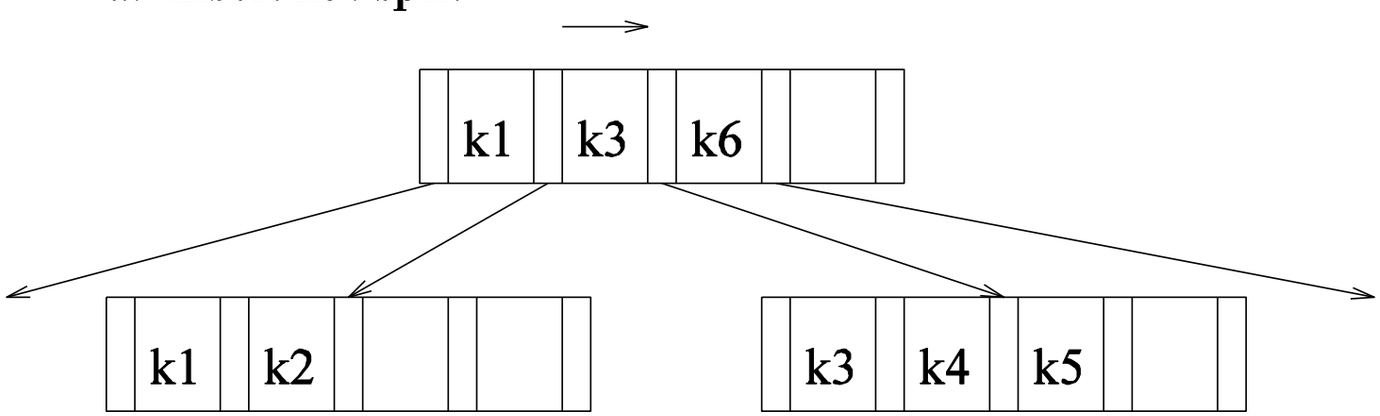
- Inserimenti ed eliminazioni provocano aggiornamento degli indici e sono precedute da una ricerca fino ad una foglia
- Per gli inserimenti, se c'è posto nella foglia, ok, altrimenti il nodo va suddiviso (*split*), con necessità di un puntatore in più per il nodo genitore; se non c'è posto, si sale ancora, eventualmente fino alla radice. Il riempimento rimane sempre superiore al $50\,\%$
- Per le eliminazioni, è possibile avere una riduzioni (*merge*) di nodi
- Modifiche del campo chiave vanno trattate come eliminazioni seguite da inserimenti

Esempio

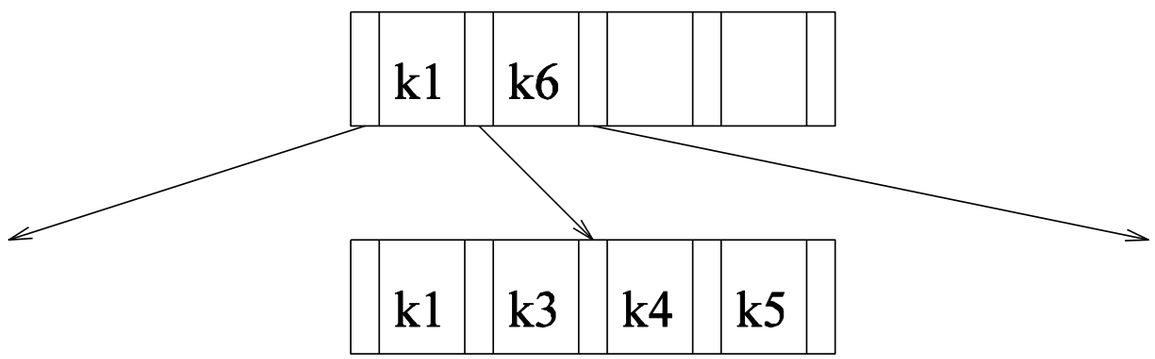
situazione iniziale



a. insert k3: split



b. delete k2: merge



B Tree e B+ Tree

- B+ tree:
  - le foglie sono collegate in una lista
  - ottimi per le ricerche su intervalli
  - molto usati nei DBMS
- B tree:
  - I nodi intermedi possono avere puntatori direttamente ai dati

Esempio di B+ Tree