



Basi di Dati  
Modulo Tecnologie

# Strutture fisiche e strutture di accesso ai dati (II parte)

Dr. Sara Migliorini

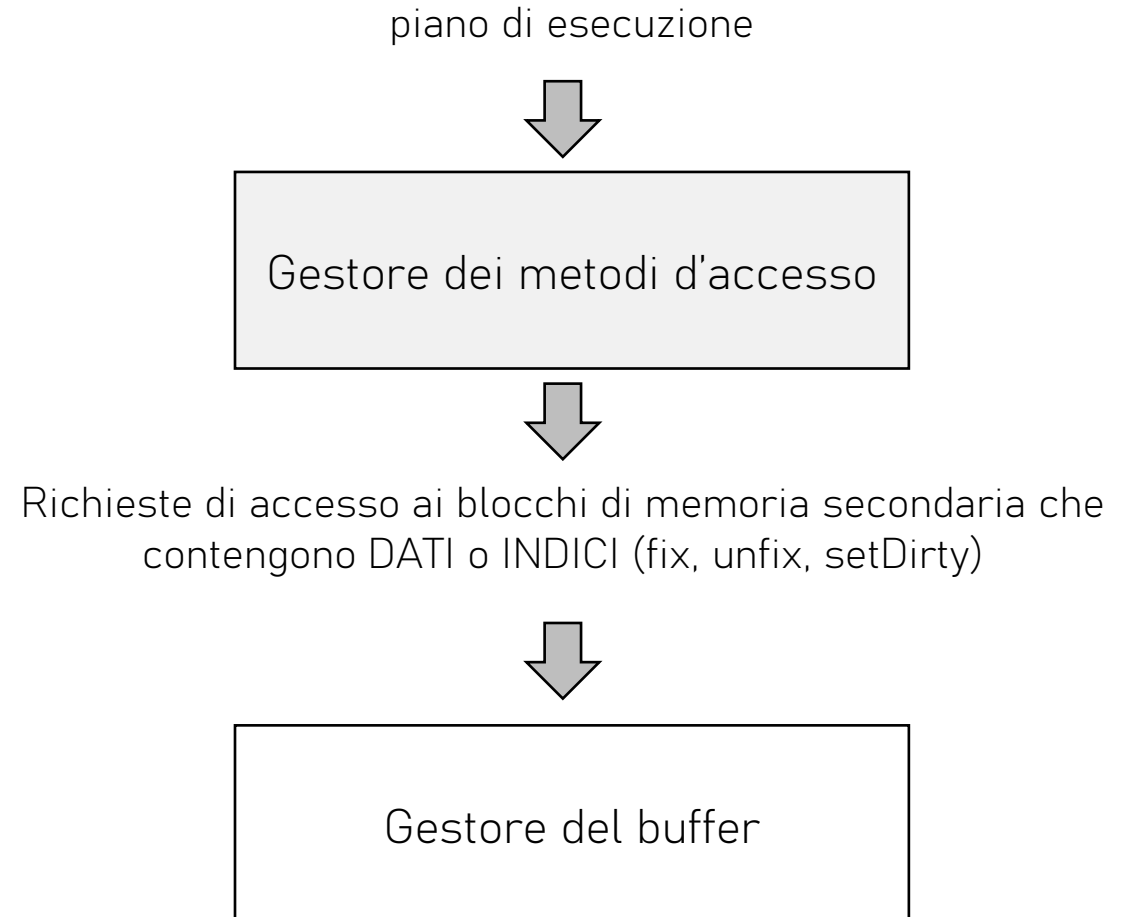
AA 2024-2025

# DBMS e File System

- Il DBMS gestisce i blocchi dei file allocati nel file system come se fossero un unico grande spazio di memoria secondaria e costruisce in tale spazio le **strutture fisiche** con cui implementa le relazioni (tabelle).
- Nel caso più frequente, ogni blocco è dedicato a tuple di un'unica relazione, ma esistono tecniche che prevedono la **memorizzazione delle tuple di più tabelle**, tra loro correlate, negli stessi blocchi.

# Gestore dei metodi di accesso (Cap. 11.2 → 11.5)

- E' il modulo del DBMS che esegue il piano di esecuzione prodotto dall'ottimizzatore e produce sequenze richieste di accessi ai blocchi della base di dati presenti in memoria secondaria.
- Le richieste vengono inviate al gestore del buffer che si occupa di caricare i blocchi necessari in pagine di memoria centrale.



# Gestore dei metodi di accesso

## Metodi d'accesso

- Sono i moduli software che implementano gli algoritmi di *accesso e manipolazione dei dati* organizzati in specifiche strutture fisiche.
- Esempio:
  - Scansione sequenziale
  - Accesso via indice
  - Ordinamento
  - Varie implementazioni del join

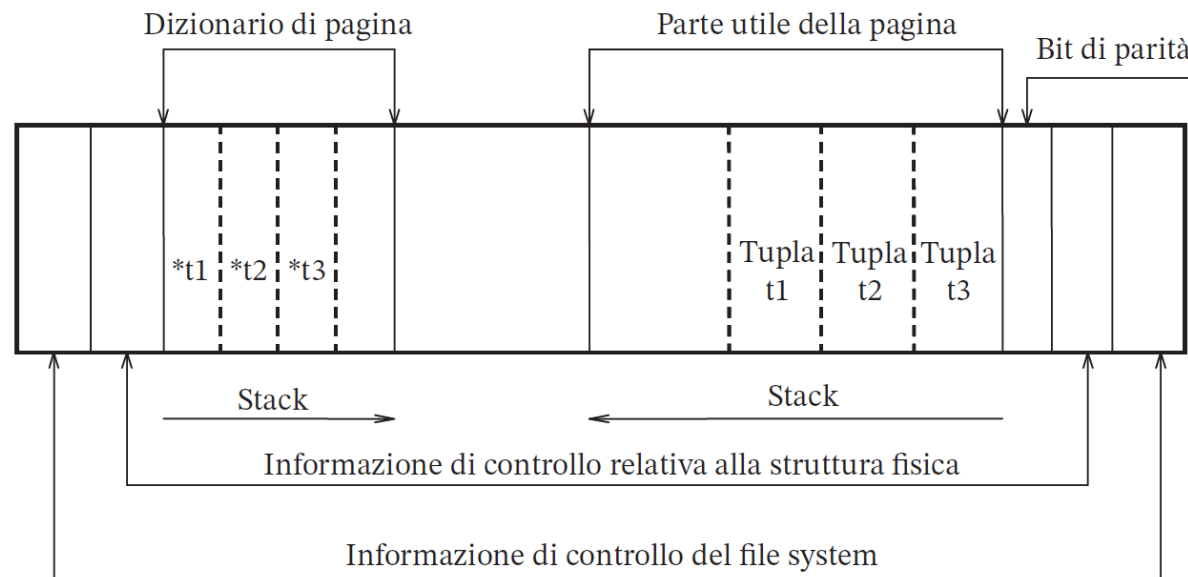
# Gestore dei metodi di accesso

Ogni metodo d'accesso ai dati conosce:

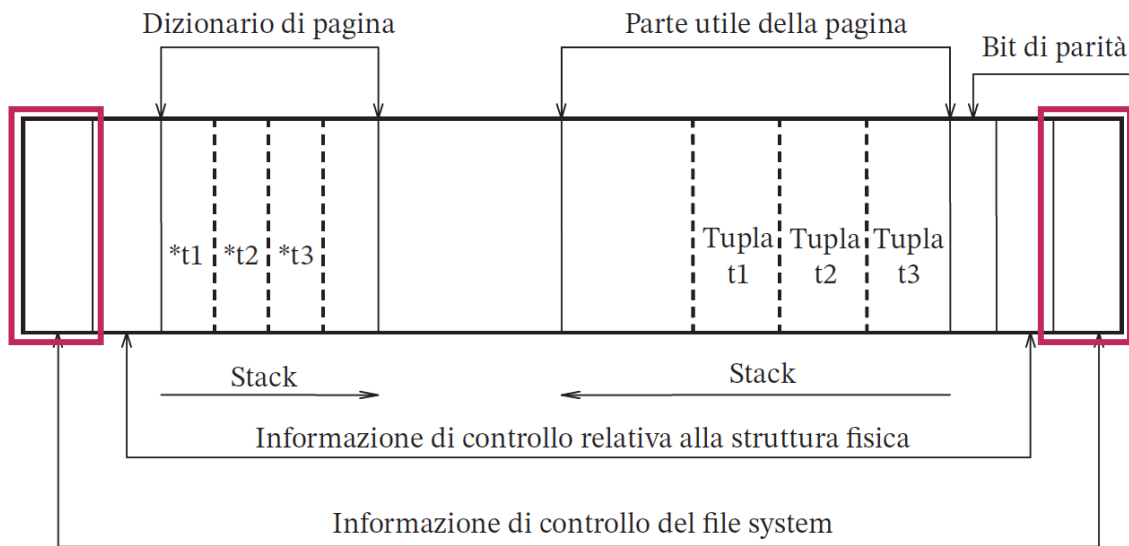
- L'organizzazione delle tuple (o dei record di indice) nei blocchi DATI (o INDICE) salvati in memoria secondaria → come una tabella (o indice) viene organizzata in pagine della memoria secondaria.
- L'organizzazione fisica interna delle pagine sia quando contengono DATI (vale a dire, tuple di una tabella) sia quando contengono strutture fisiche di accesso o INDICI (vale a dire, record di un indice).

# Organizzazione di una pagina

- In una pagina sono presenti informazioni utili e informazioni di controllo:
  - **Informazioni utili:** dati veri e propri (tuple della tabella)
  - **Informazioni di controllo:** consentono di accedere alle informazioni utili (dizionario, bit di parità, altre informazioni del file system o della specifica struttura fisica).



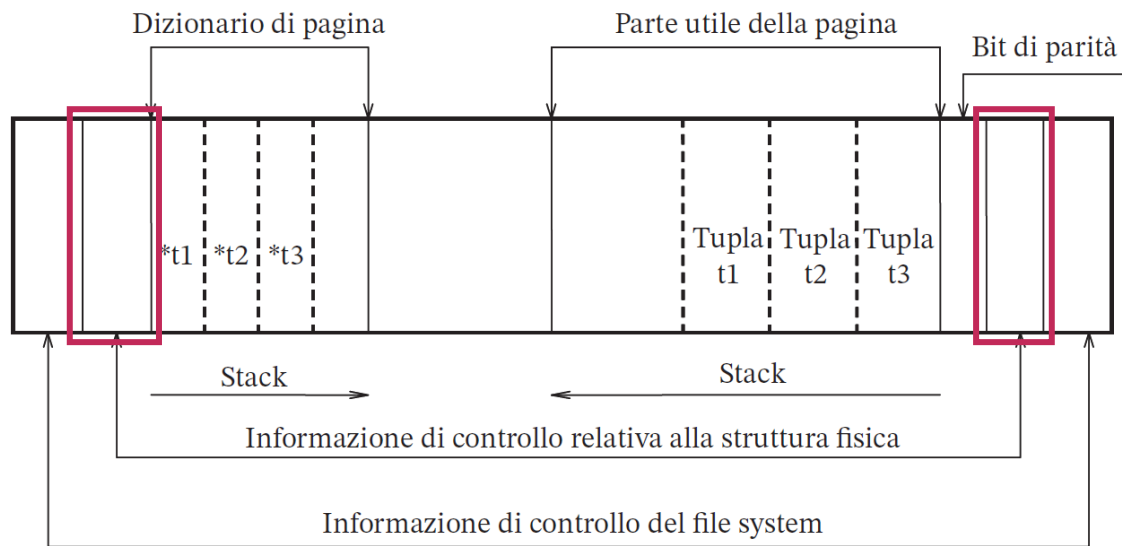
# Organizzazione di una pagina



## Block header e block trailer

Ogni pagina poichè coincide con un blocco di memoria di massa, ha una **parte iniziale** ed una **parte finale** che contengono **informazioni di controllo utilizzate dal file system**

# Organizzazione di una pagina



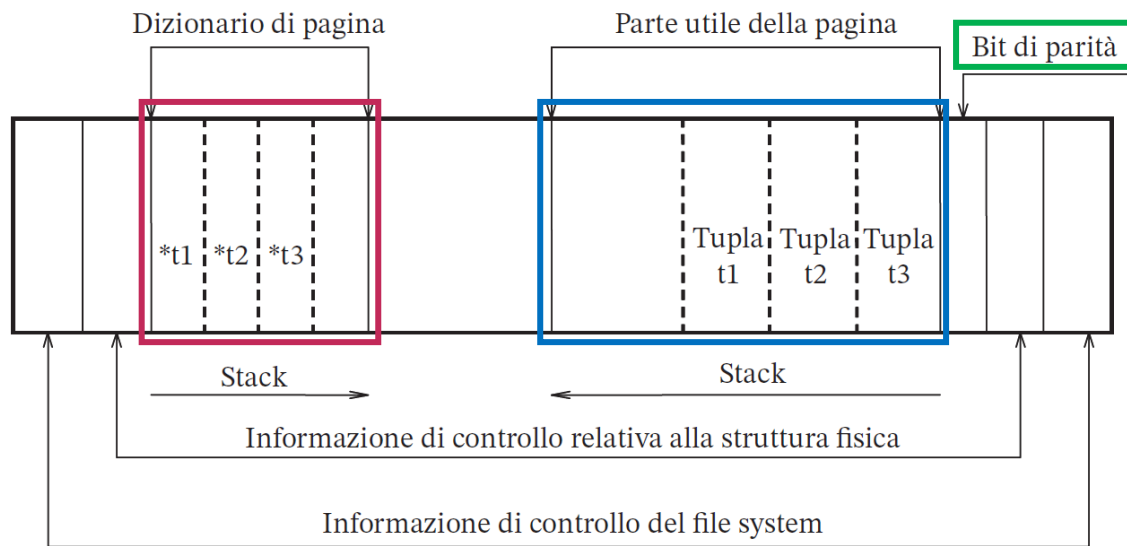
## Page header e page trailer

Ogni pagina poichè contenente dati gestiti dal DBMS, ha una parte iniziale ed una finale contenenti l'informazione di controllo relativa alla specifica struttura fisica.

Es: identificatore dell'oggetto (tabella, indice, dizionario dei dati, ecc.) contenuto nella pagina, puntatori a pagine successive o precedenti nella struttura dati, numero di dati utili elementari (tuple) contenuti nella pagina e quantità di memoria libera (contigua o non contigua) disponibile nella pagina.



# Organizzazione di una pagina



## Dizionario di pagina

Contiene **puntatori a ciascun dato utile** elementare contenuto della pagina.

## Parte utile

Contiene i **dati**.

**Dizionari** di pagina e **parte utile** crescono come **stack contrapposti**, lasciando libera la parte centrale in uno spazio contiguo.

## Bit di parità

Verifica che **l'informazione contenuta nella pagina** sia valida.

# Organizzazione di una pagina

## Struttura del dizionario

- **Tuple di lunghezza fissa**: il dizionario non è necessario, si deve solo memorizzare la dimensione delle tuple e l'offset del punto iniziale.
- **Tuple di lunghezza variabile** (valori null oppure di tipo stringa): il dizionario memorizza l'offset di ogni tupla presente nel blocco e di ogni attributo di ogni tupla.

Molti gestori delle pagine non consentono la separazione di una tupla su più pagine → la dimensione massima di una tupla = dimensione massima dell'area disponibile su un blocco

Alcuni gestori delle pagine consentono di distribuire una tupla su più pagine, altrimenti va gestito il caso di tuple memorizzate su più pagine → in PostgreSQL si veda la soluzione TOAST - The Oversized-Attribute Storage Technique.

# Operazioni sulle pagine

- Inserimento ed aggiornamento di una tupla:
  - Se esiste spazio contiguo sufficiente: inserimento semplice.
  - Se non esiste spazio contiguo ma esiste spazio sufficiente: riorganizzare lo spazio ed eseguire un inserimento semplice (operazione in memoria centrale).
    - Operazione aggiuntiva: riorganizzazione della pagina.
  - Se non esiste spazio sufficiente: interazione col file system per allocare nuovi blocchi per il file.
- Cancellazione: sempre possibile anche senza riorganizzare il contenuto nella pagina.
- Accesso ad una tupla particolare tramite il valore di chiave oppure l'offset nel dizionario.
  - Accesso sequenziale a più tuple.
- Accesso ad un attributo di una tupla: identificato in base all'offset e alla lunghezza del campo dopo aver identificato la tupla tramite la chiave o il suo offset.

# Strutture primarie per l'organizzazione di file

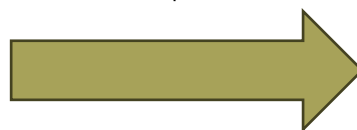
- La struttura primaria di un file stabilisce il criterio secondo il quale sono disposte le tuple all'interno del file presente in memoria di massa.
  - Sequenziali**: disposizione consecutiva delle tuple che si basa su un criterio specifico, come l'ordine di inserimento.
  - Ad accesso calcolato (hash)**: collocano le tuple in posizioni determinate sulla base dell'esecuzione di un algoritmo di hash.
  - Ad albero**: utilizzate soprattutto come strutture secondarie (indici)



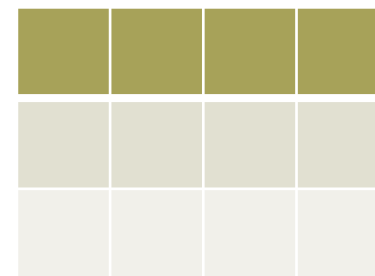
(memoria secondaria)

Tuple memorizzate  
nella parte utile nel blocco

struttura primaria



Livello logico



Modello dei dati relazionale  
(tabella)

# Struttura Sequenziale

- Un file è costituito da blocchi “logicamente” consecutivi.
- Le tuple vengono inserite nei blocchi rispettando una sequenza:
  - Struttura **sequenziale disordinata** (o seriale): le tuple sono disposte in base al loro **ordine di inserimento**.
  - Struttura **sequenziale ad array**: le tuple sono disposte come in un array, e la **loro posizione dipende dal** valore assunto in ciascuna tupla da un campo detto **indice**.
  - Struttura **sequenziale ordinata**: la sequenza delle tuple **dipende dal valore assunto** in ciascuna tupla da un campo del file.

# Struttura Sequenziale Seriale o Disordinata

- È la struttura sequenziale più semplice e più diffusa.
- Le tuple vengono aggiunte nel file nell'ordine con cui si sono state inserite.
- **Inserimento: operazione molto efficiente**, è sufficiente mantenere un riferimento all'ultimo blocco e richiede un solo accesso in memoria secondaria.
  - Costo maggiore in caso di verifica dei vincoli di integrità (vedi ricerca).
- **Ricerca:** operazione non efficiente, **richiede ogni volta una scansione sequenziale che consideri tutti i record.**
  - Costo lineare nel numero di blocchi.
  - Uso assieme a strutture secondarie (indici).
- **Eliminazione e modifica: facili una volta individuate le tuple coinvolte.**
  - Eliminazione: si marcano le tuple come «cancellate», senza riorganizzazione locale.
  - Modifiche: si procede in loco se possibile (nessuna modifica alla dimensione della tupla), oppure cancellazione ed inserimento in fondo al file.
- Eliminazioni e modifiche possono portare ad un uso non ottimale dello spazio → **riorganizzazioni periodiche.**

# Struttura Sequenziale ad Array

- È possibile solo quando le tuple di una tabella hanno una dimensione fissa.
  - Questa struttura non è quasi mai usata nei DBMS reali perché difficilmente sono soddisfatte le condizioni per la loro applicabilità.
- Ad ogni file viene assegnato un numero  $n$  di blocchi contigui e ciascun blocco è dotato di un numero  $m$  di «posizioni» disponibili per le tuple → array di  $n \times m$  posizioni
- Ciascuna tupla è dotata di un valore numerico  $i$  che funge da indice → posto nella  $i$ -esima posizione dell'array.
- Inserimento:
  - Iniziale → l'indice  $i$  viene ottenuto incrementando un contatore.
  - Successivo → in una posizione libera oppure alla fine del file.
- Cancellazioni: creano delle posizioni libere

# Struttura Sequenziale Ordinata

- Una struttura sequenziale ordinata è un **file sequenziale** dove le tuple sono **ordinate** secondo una **chiave di ordinamento** (diversa dalla chiave primaria!).

- Esempio

chiave ordinamento

↓

	Filiale	Conto	Cliente	Saldo	
Blocco 1	A	102	Rossi	1000	
	B	110	Rossi	3020	
	B	198	Bianchi	500	
Blocco 2	E	17	Neri	345	
	E	102	Verdi	1200	
	E	113	Bianchi	200	
	H	53	Neri	120	
	F	78	Verdi	3400	





# Struttura Sequenziale Ordinata

- Una struttura sequenziale ordinata è un **file sequenziale** dove le tuple sono **ordinate** secondo una **chiave di ordinamento** (diversa dalla chiave primaria!).
- Rendono efficienti** le operazioni che hanno bisogno dell'ordinamento utilizzato: elenco ordinato, range query, operazioni aggregate.
- Inconvenienti in caso di aggiornamento**: necessità di mantenere l'ordinamento.
  - Periodiche riorganizzazioni**.

chiave primaria				
	Filiale	Conto	Cliente	Saldo
Blocco 1	A	102	Rossi	1000
	B	110	Rossi	3020
	B	198	Bianchi	500
Blocco 2	E	17	Neri	345
	E	102	Verdi	1200
	E	113	Bianchi	200
	H	53	Neri	120
	F	78	Verdi	3400



# Struttura sequenziale ordinata: Operazioni

- Inserimento di una tupla
  - Individuare il blocco B che contiene la **tupla che precede**, nell'ordine della chiave, la tupla da inserire.
  - Se B contiene spazio sufficiente per la nuova tupla: inserire la nuova tupla in B.
  - Altrimenti si aggiunge un nuovo blocco (detto, overflow page) alla struttura e si inserisce la tupla nel nuovo blocco e si aggiusta la catena di puntatori.
- Scansione sequenziale ordinata secondo la chiave (seguendo i puntatori)
  -

# File Sequenziale: Operazioni

- Cancellazione di una tupla
  - Individuare il blocco B che contiene la tupla da cancellare.
  - Cancellare la tupla da B.
  - Aggiustare la catena di puntatori.
- Riorganizzazione: si assegnano le tuple ai blocchi in base ad opportuni coefficienti di riempimento, riaggiustando i puntatori.

# Indici

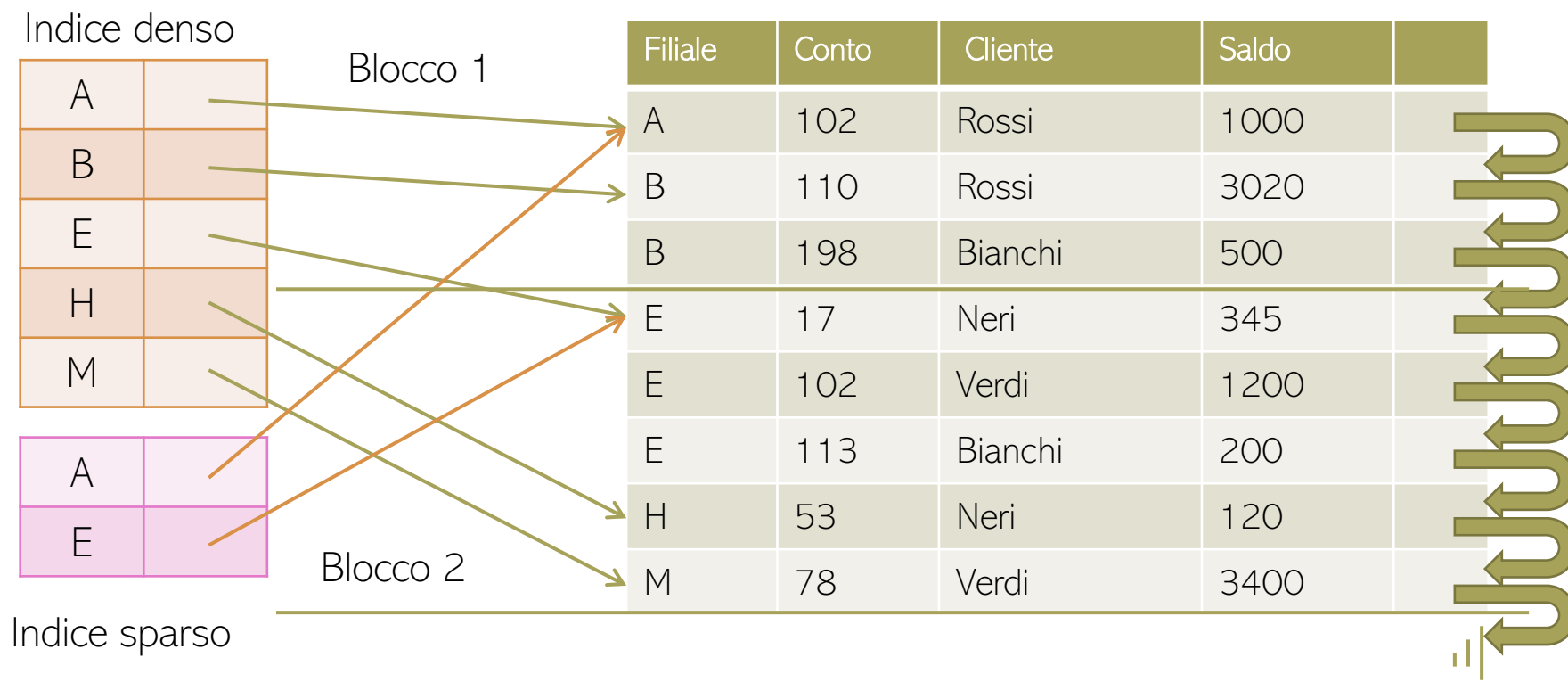
- Per aumentare le prestazioni degli accessi alle tuple memorizzate nelle strutture fisiche (**file sequenziale**), si introducono strutture ausiliarie (dette strutture di accesso ai dati o **INDICI**).
- Tali strutture velocizzano l'accesso casuale via chiave di ricerca. La chiave di ricerca è un insieme di attributi utilizzati dall'indice nella ricerca.
- Indici su file sequenziali
  - **INDICE PRIMARIO**: in questo caso la **chiave di ordinamento** del file sequenziale **coincide con la chiave di ricerca** dell'indice.
  - **INDICE SECONDARIO**: in questo caso invece la **chiave di ordinamento** e la **chiave di ricerca sono diverse**.

# Indice PRIMARIO

- Usa una **chiave di ricerca** che coincide con la chiave di ordinamento del file sequenziale.
- Ogni record dell'indice primario contiene una coppia  $\langle v_i, p_i \rangle$ :
  - $v_i$ : **valore della chiave di ricerca;**
  - $p_i$ : **puntatore al primo record** nel file sequenziale con chiave  $v_i$
- Esistono due varianti dell'indice primario:
  - **Indice denso:** per **ogni occorrenza della chiave** presente nel file **esiste un corrispondente record nell'indice.**
  - **Indice sparso:** **solo per alcune occorrenze della chiave** presenti nel file esiste un corrispondente record nell'indice, **tipicamente una per blocco.**

# Indice PRIMARIO

Esempio

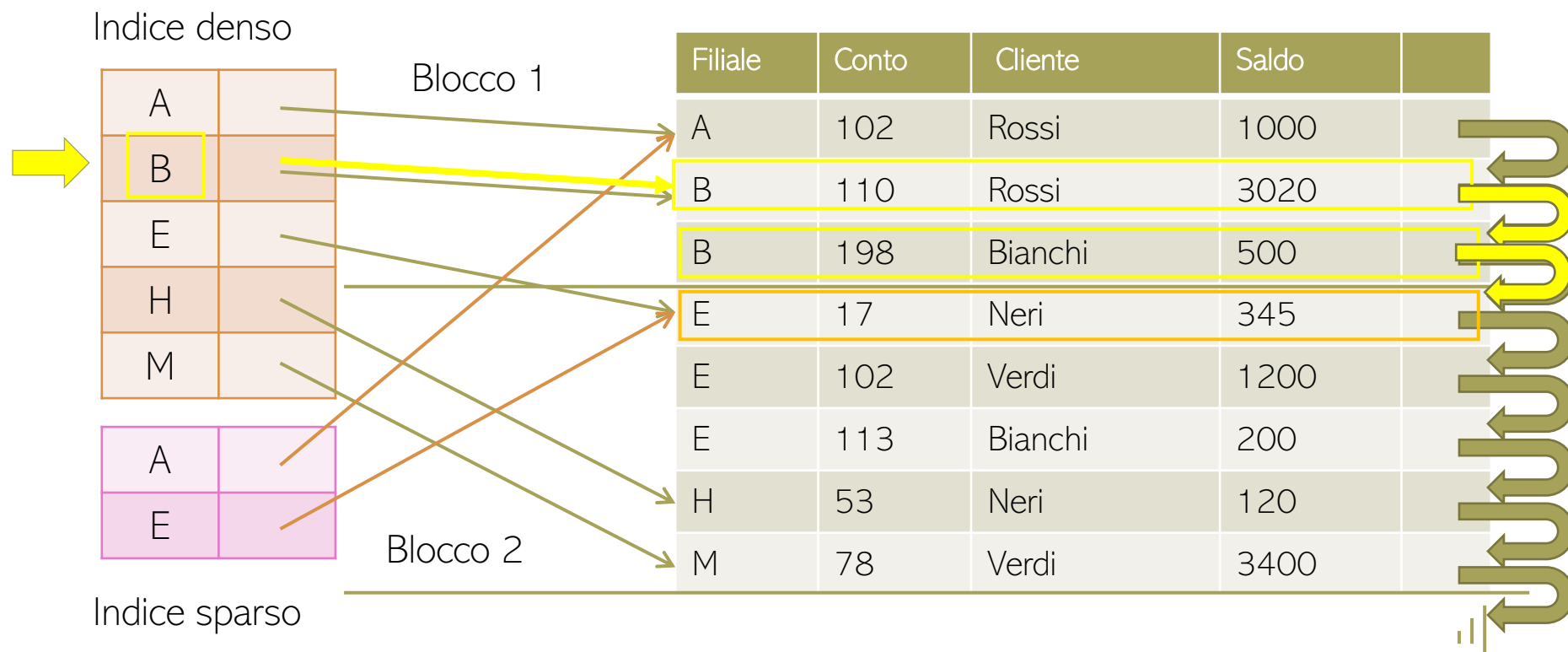


# Indice PRIMARIO: Operazioni - Ricerca

- [Ricerca](#) di una tupla con chiave di ricerca  $K$ .
  - **DENSO** ( $\Rightarrow K$  è presente nell'indice)
    - Scansione sequenziale dell'indice per trovare il record  $(K, p_k)$
    - Accesso al file attraverso il puntatore  $p_k$
  - Costo: 1 accesso indice + 1 accesso blocco dati

# Indice PRIMARIO: Operazioni - Ricerca

Esempio ricerca dei conti della filiale B



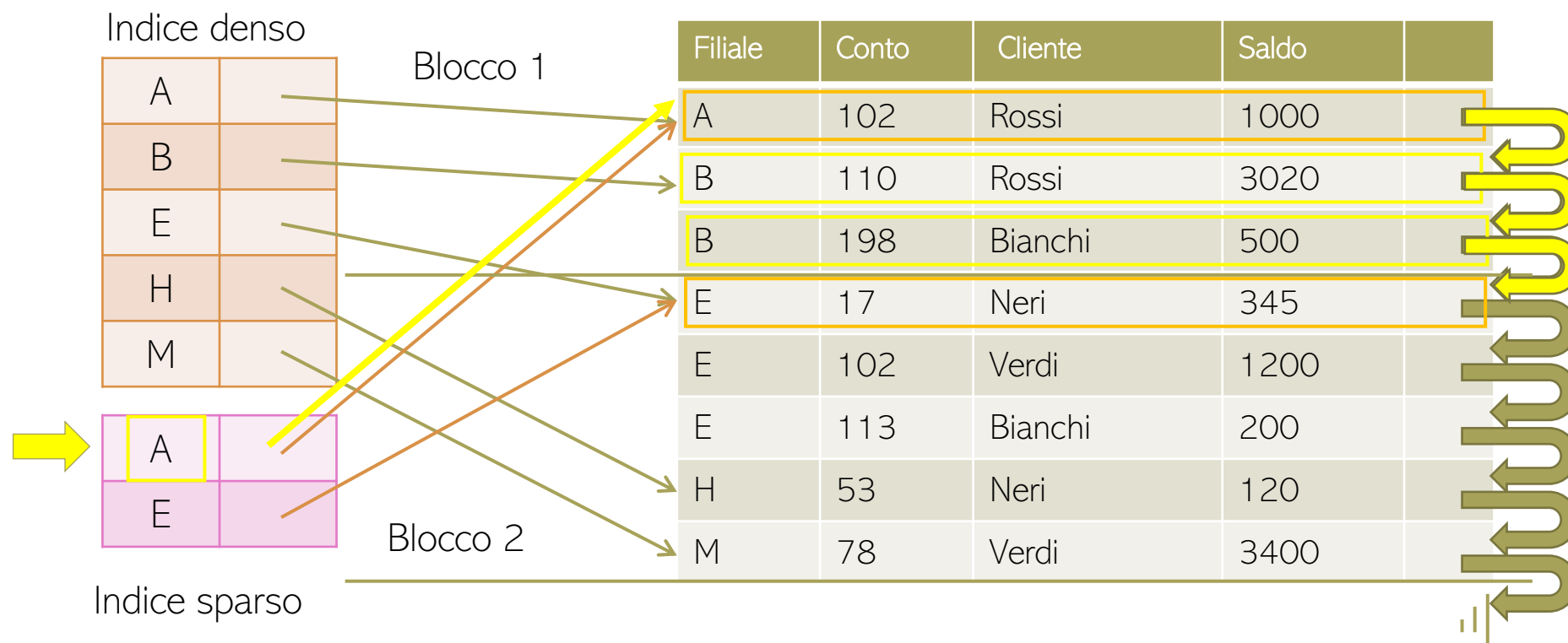


# Indice PRIMARIO: Operazioni - Ricerca

- [Ricerca](#) di una tupla con chiave di ricerca  $K$ .
  - SPARSO ( $\Rightarrow K$  potrebbe non essere presente nell'indice)
    - Scansione sequenziale dell'indice fino al record  $(K', p_{K'})$  dove  $K'$  è il valore più grande che sia minore o uguale a  $K$
    - Accesso al file attraverso il puntatore  $p_{K'}$  e scansione del file (blocco corrente) per trovare le tuple con chiave  $K$ .
  - Costo: 1 accesso indice + 1 accesso blocco dati

# Indice PRIMARIO: Operazioni - Ricerca

Esempio ricerca dei conti della filiale B



# Indice PRIMARIO: Inserimento

- **Inserimento** di un record nell'indice
  - Come inserimento nel FILE SEQUENZIALE (nel blocco della memoria secondaria invece di tuple ci sono record dell'indice)
  - **DENSO**
    - L'inserimento nell'indice avviene solo se la tupla inserita nel file ha un valore di chiave  $K$  che non è già presente.
  - **SPARSO**
    - L'inserimento avviene solo quando, per effetto dell'inserimento di una nuova tupla, si aggiunge un blocco dati alla struttura; in tutti gli altri casi l'indice rimane invariato.

# Indice PRIMARIO: Cancellazione

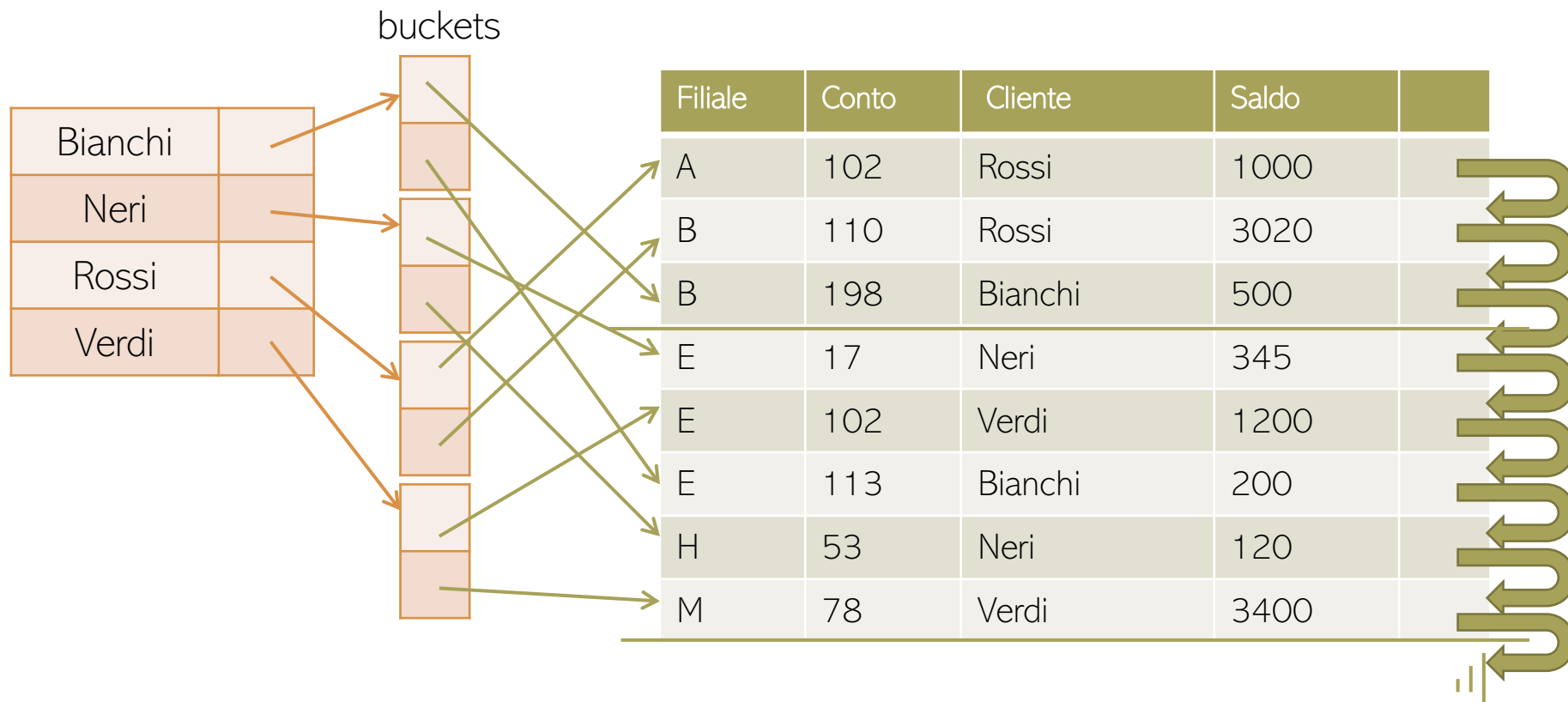
- Cancellazione di un record nell'indice
  - Come cancellazione nel FILE SEQUENZIALE
  - DENSO
    - La cancellazione nell'indice avviene solo se la tupla cancellata nel file è l'ultima tupla con valore di chiave  $K$ .
  - SPARSO
    - La cancellazione nell'indice avviene solo quando  $K$  è presente nell'indice e il corrispondente blocco viene eliminato; altrimenti, se il blocco sopravvive, va sostituito  $K$  nel record dell'indice con il primo valore  $K'$  presente nel blocco.

# Indice SECONDARIO

- Usa una chiave di ricerca che NON coincide con la chiave di ordinamento del file sequenziale.
- Ogni record dell'indice secondario contiene una coppia  $\langle v_i, p_i \rangle$ :
  - $v_i$ : valore della chiave di ricerca;
  - $p_i$ : puntatore al bucket di puntatori che individuano nel file sequenziale tutte le tuple con valore di chiave  $v_i$ .
- Gli indici secondari sono sempre DENSE.

# Indice SECONDARIO

Esempio



# Indice SECONDARIO: Operazioni - Ricerca

- Ricerca di una tupla con chiave di ricerca  $K$ .
  - Scansione sequenziale dell'indice per trovare il record  $(K, p_k)$
  - Accesso al bucket  $B$  di puntatori attraverso il puntatore  $p_k$
  - Accesso al file attraverso i puntatori del bucket  $B$ .
- Costo: 1 accesso indice + 1 accesso al bucket +  $n$  accessi pagine dati
- Inserimento e cancellazione: come indice primario denso con in più l'aggiornamento dei bucket.

# Indice SECONDARIO: Operazioni - Ricerca

Esempio di ricerca dei conti di Verdi

