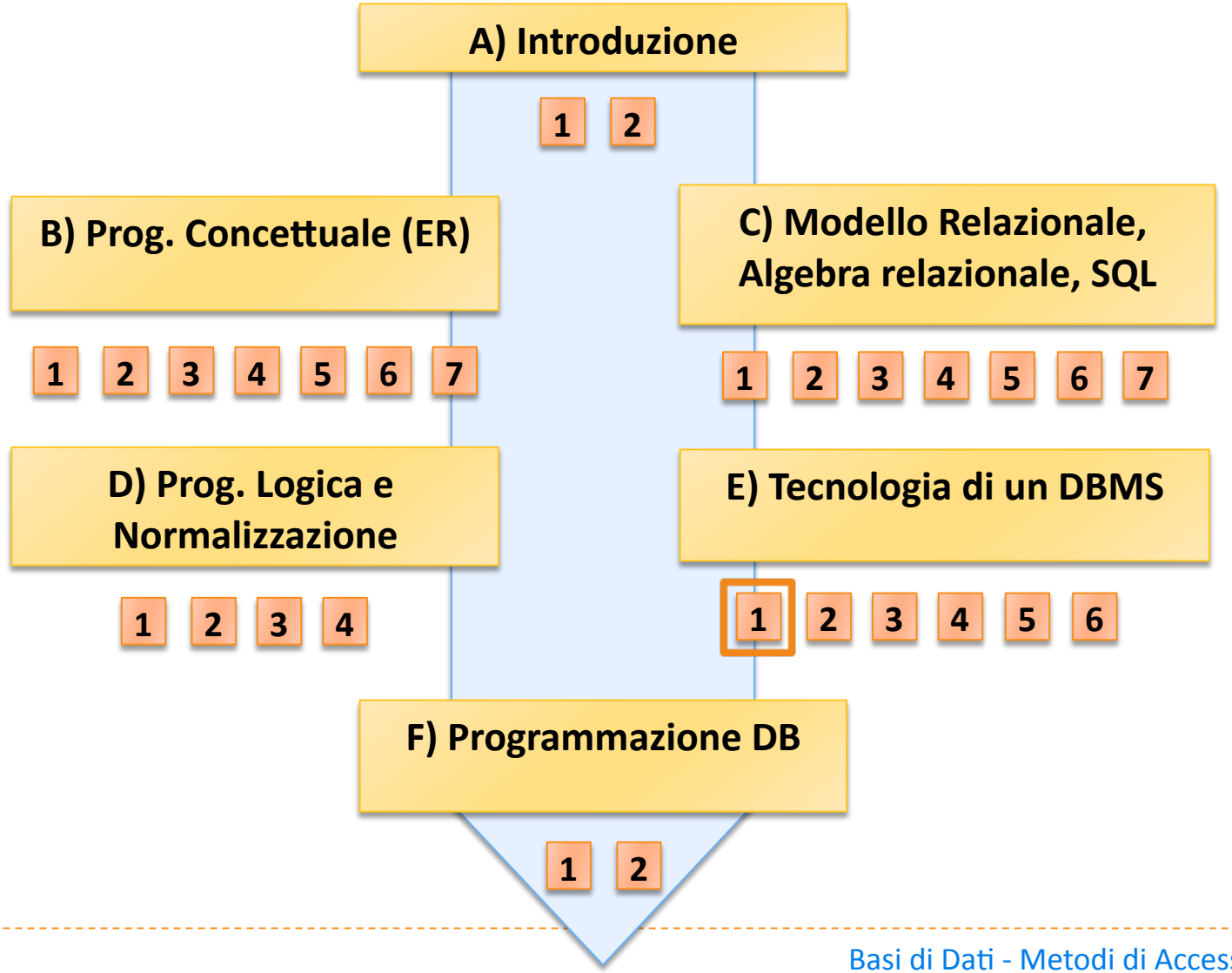
Basi di Dati

Metodi di Accesso (File)

Basi di Dati – Dove ci troviamo?



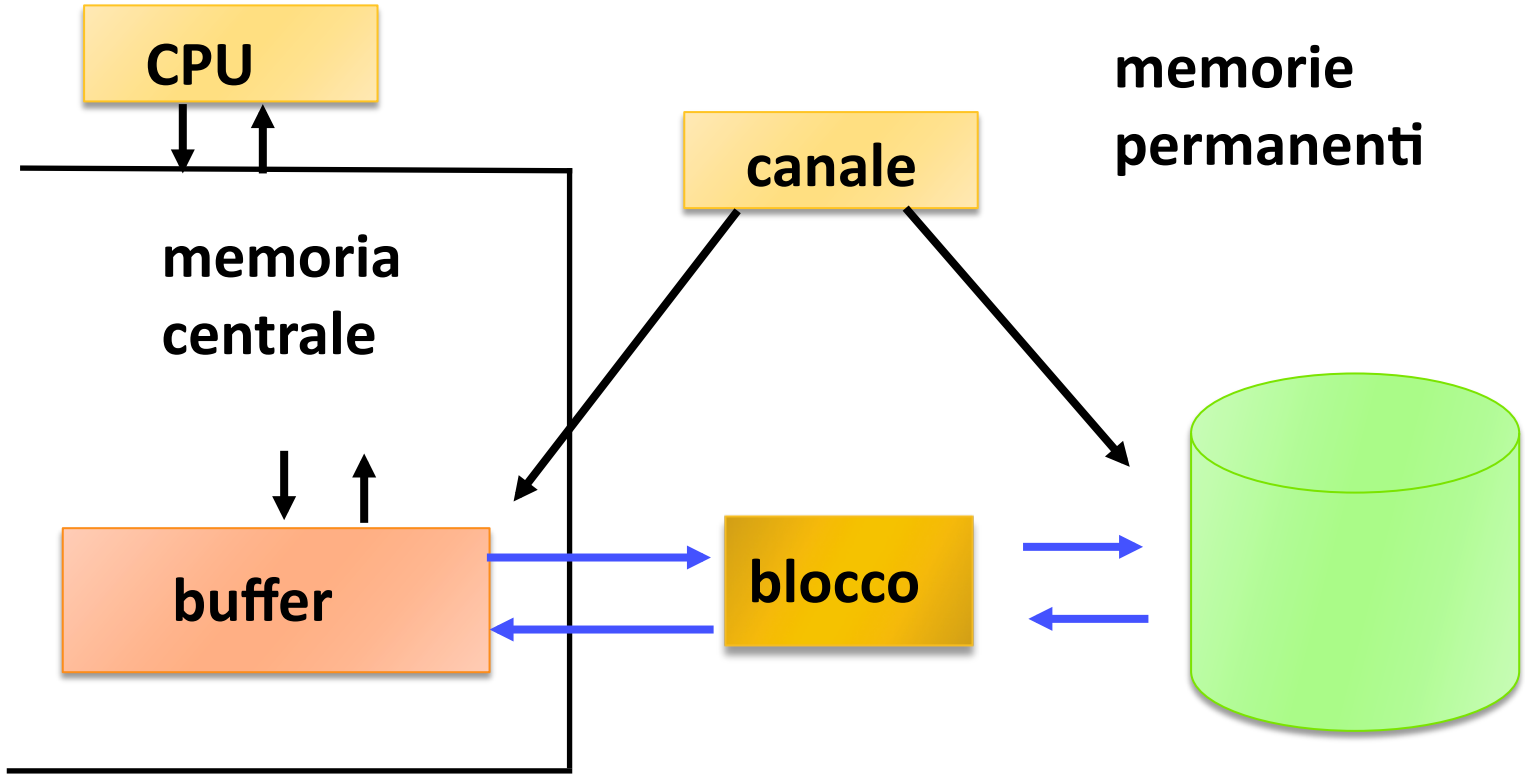
Nelle lezioni precedenti

- **Avete visto:**
  - **le caratteristiche delle principali unità di memoria permanente**

In questa lezione

- **Presenteremo:**
  - **la struttura dei file**
  - **il loro ordinamento**

Struttura di un data server



blocco-min = settore

File Directory

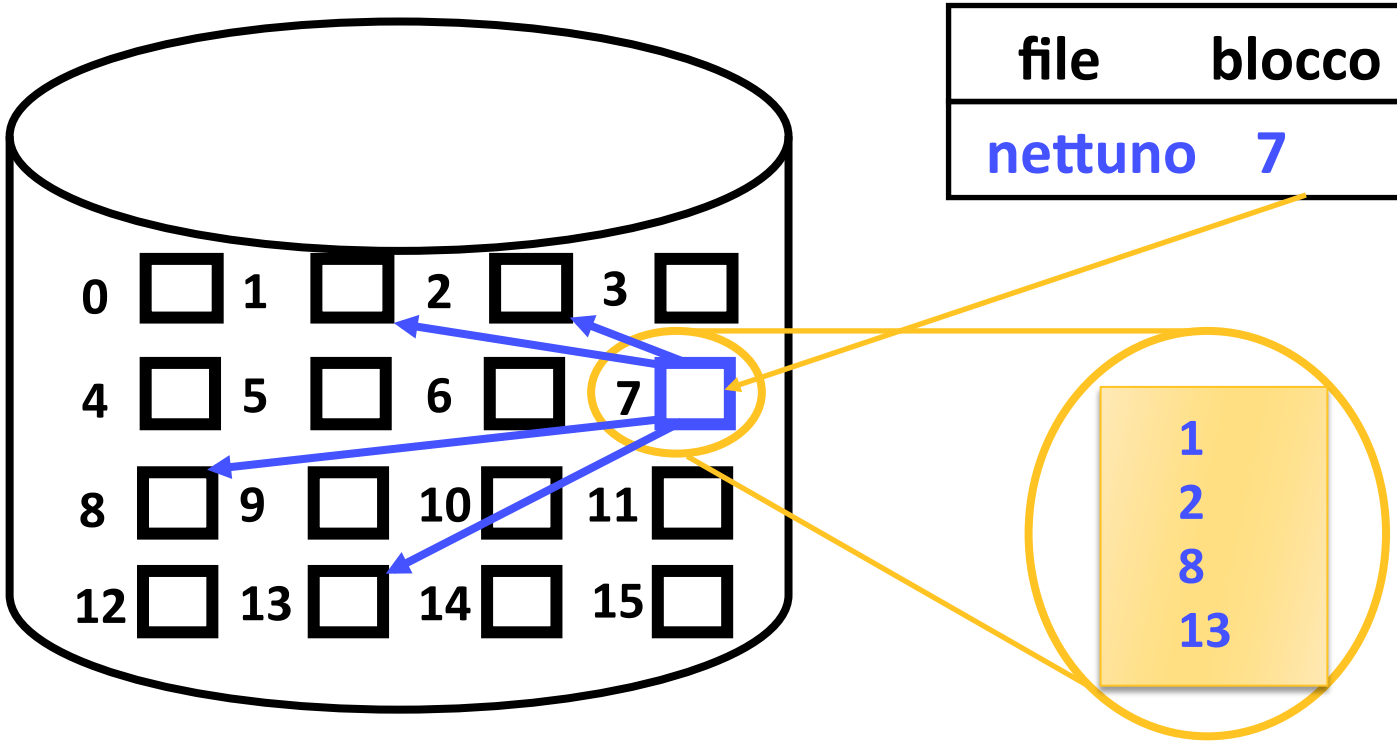
- **DIRECTORY (elenco dei file) è una tabella che viene consultata dal FILE SYSTEM (che è parte del sistema operativo) o dal DBMS e che contiene le informazioni sui file. Sta sul disco.**
- **Il contenuto è molto diverso da sistema a sistema, informazioni tipiche sono:**
  - **nome del file, proprietario,**
  - **locazione di inizio,**
  - **numero di blocchi allocati, numero di blocchi utilizzati ecc.**

Struttura del file

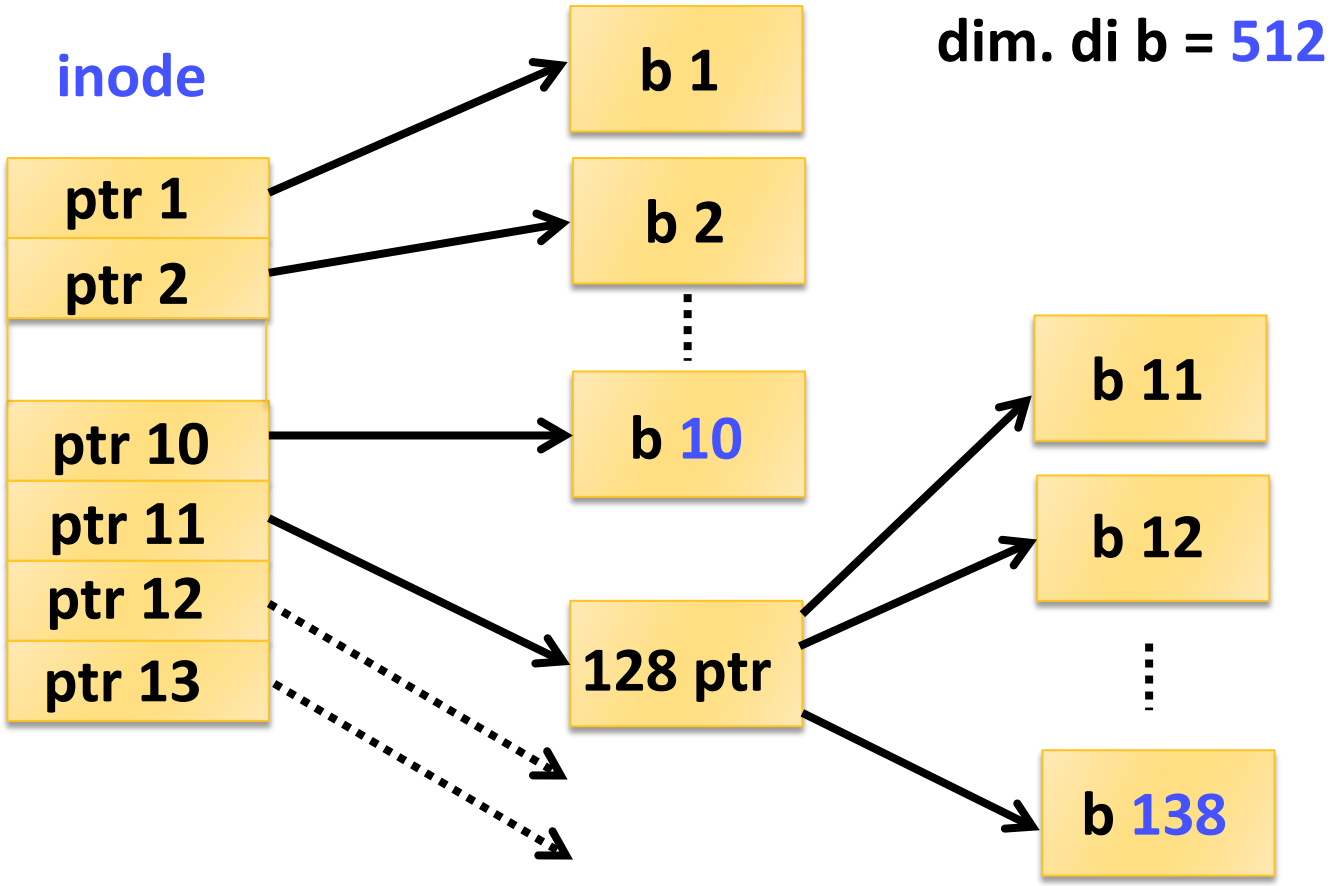
Un comune metodo di allocazione è l'allocazione con indice

directory

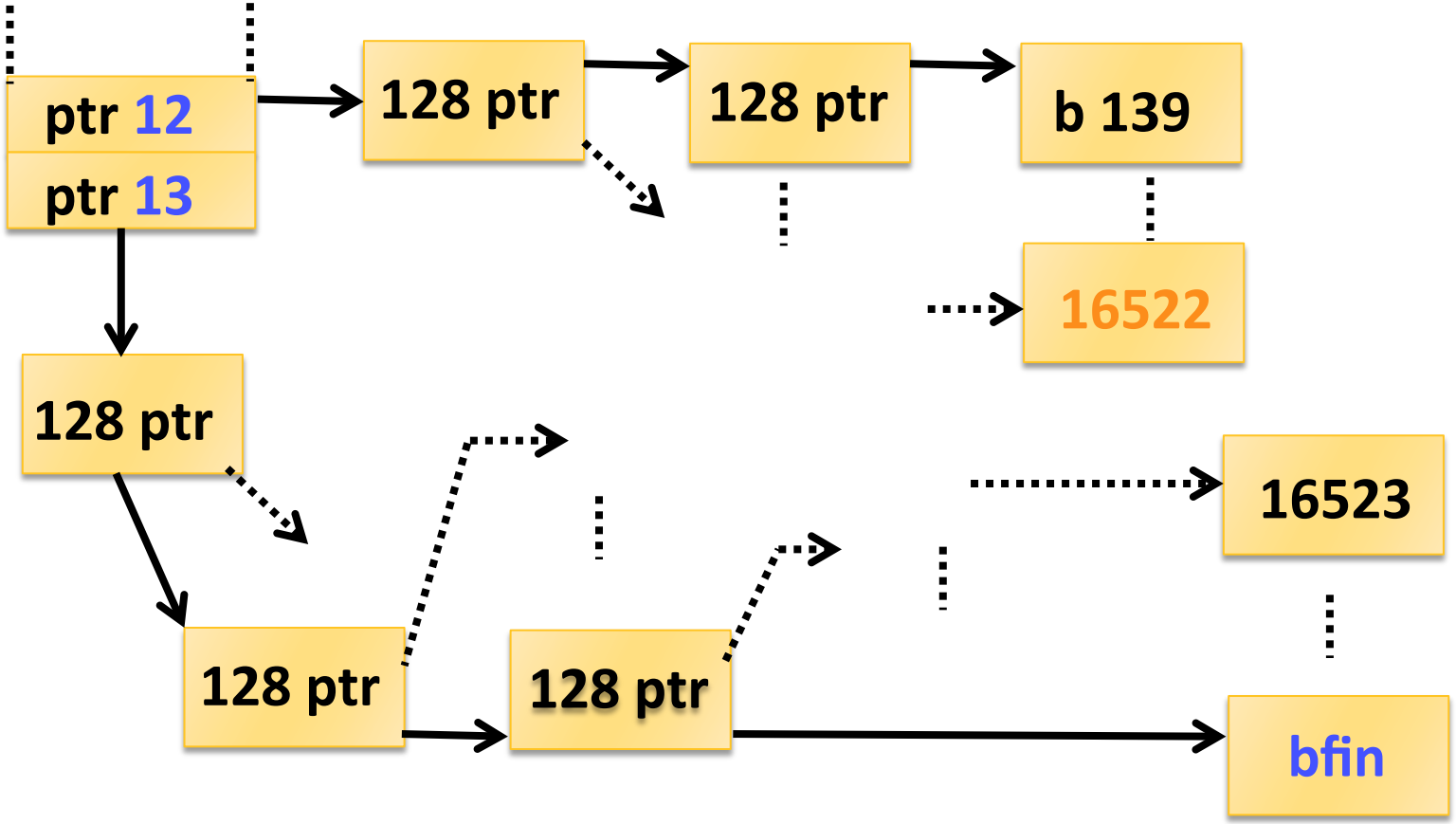
file	blocco
nettuno	7



Struttura del file UNIX



Struttura del file UNIX



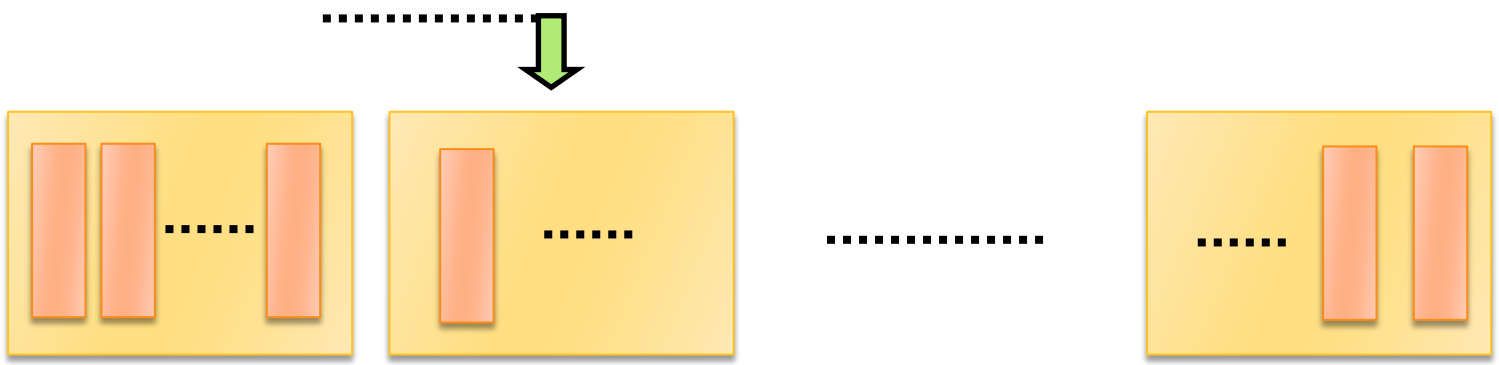
bfin: 2 113 674 tot.: 1 GB

Struttura dei file

- La dimensione dei blocchi può essere superiore, fino ad oltre 8 ÷ 16 kB
- Anche se ci sono più livelli di indirizzamento e i file system ottimizzano la gestione del buffer e il trasferimento, le prestazioni generali per l'accesso ad un file vengono valutate come somma del numero di blocchi di dati che vengono scritti o letti da un'operazione

Accesso sequenziale al file

Select * from cittadini where codice = 'cf'



- cittadini è una relazione di NT tuple contenuta in un file di NB blocchi
- la relazione non è in ordine di codice
- vengono visitati i blocchi fino a che si trova 'cf'

Accesso sequenziale al file

- Se il 'cf' non c'è il numero di accessi è = NB
- Se il 'cf' c'è, assumendo un uguale numero di tuple per blocco e le tuple equiprobabili, ogni blocco ha la stessa probailità 1/NB di contenere 'cf' e il numero di accessi è =

$$\Sigma_{NB}\, j/NB \approx NB/2$$

(j è la lunghezza di ricerca per ognuno degli NB blocchi)

assumendo NB = 10000 e il tempo di accesso ad un blocco su disco = 20 ms (T_D), si ha:

$$T_A = NB \times T_D / 2 = 100 \text{ sec (sono molti)}$$

Ordinamento del file

- Ordinare un file è utile non solo per la presentazione del contenuto (elenchi, listini anagrafi ecc.), ma anche per velocizzare la ricerca
- L'ordinamento di un file molto grande è un'operazione molto lenta che viene di regola effettuata con il metodo Sort/Merge (a M vie),
  supponiamo di avere un file di NB blocchi che non può essere contenuto in memoria di lavoro
  il file viene ordinato in due fasi:
  - la fase di sort,
  - la fase di merge

Ordinamento del file

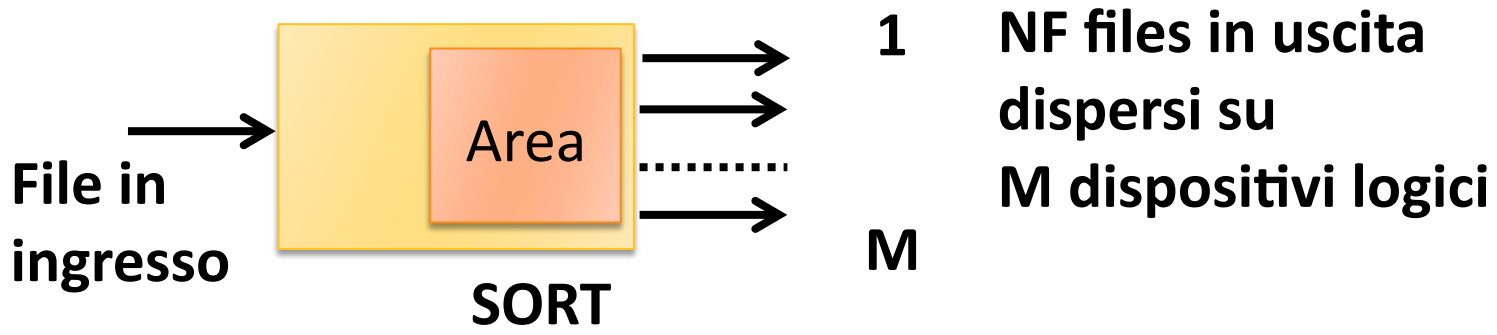
FASE DI SORT:

- **vengono portati in memoria NM blocchi (NM: disponibilità memoria centrale) per volta e le tuple ordinate con un algoritmo di sort (es. Quicksort),**
- **ogni gruppo di NM blocchi viene memorizzato in un file distinto (NF = ⌈ NB/NM ⌉ file)**

Ordinamento del file

FASE DI SORT:

Il primo passo è un SORT, dove i blocchi costituenti il file (NB) vengono raggruppati in NF file (a gruppi di NM) ed ordinati, e successivamente dispersi su M dispositivi logici.



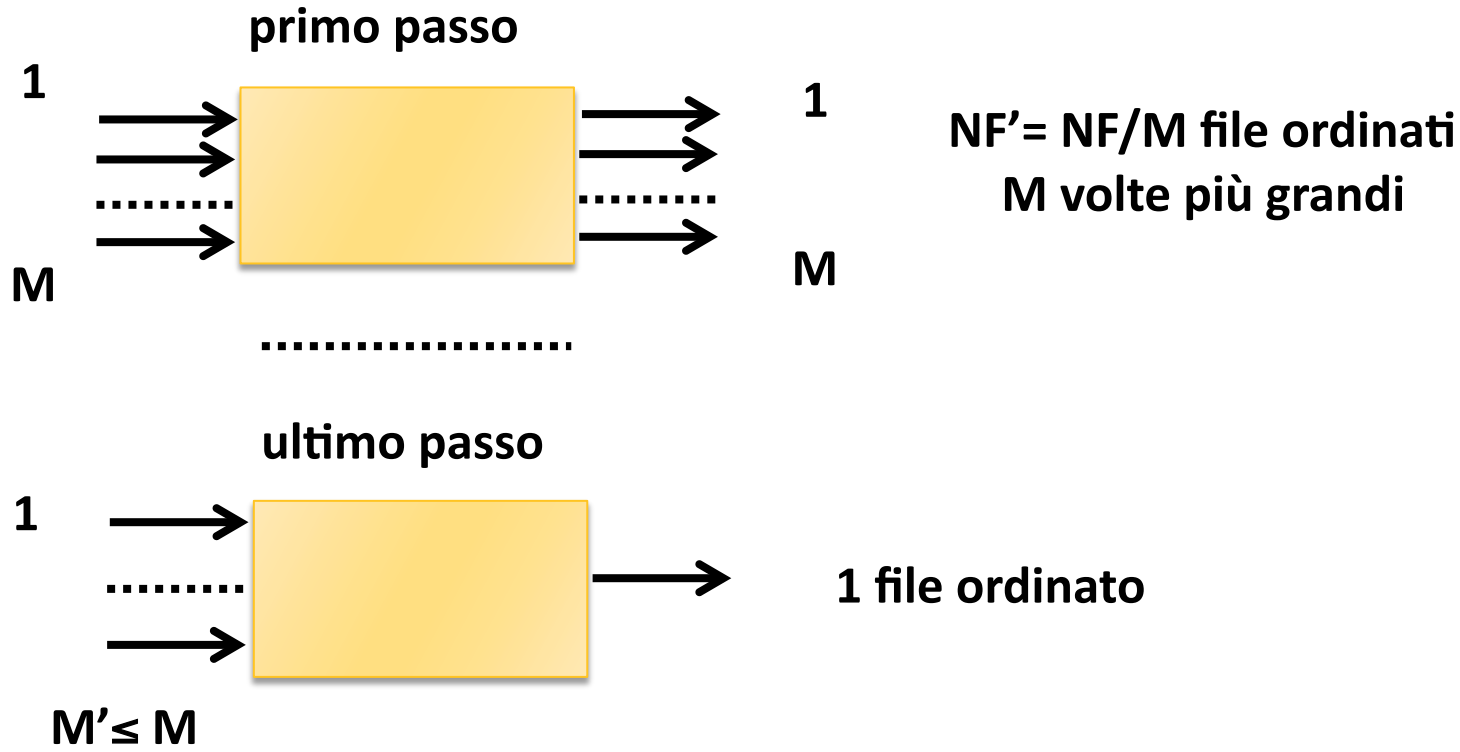
Ordinamento del file

FASE DI MERGE

- la fase è costituita da più passi:
  - passo: vengono portati in memoria gradualmente i blocchi di M (parametro del merge) file , si opera una fusione ordinata delle tuple contenute ottenendo un file M volte più grande (ordinato)
  - l'operazione si ripete fino ad esaurire gli NF file (ad ogni passo NF diminuisce)
- i passi si ripetono fondendo file sempre più grandi fino ad ottenere un unico file ordinato

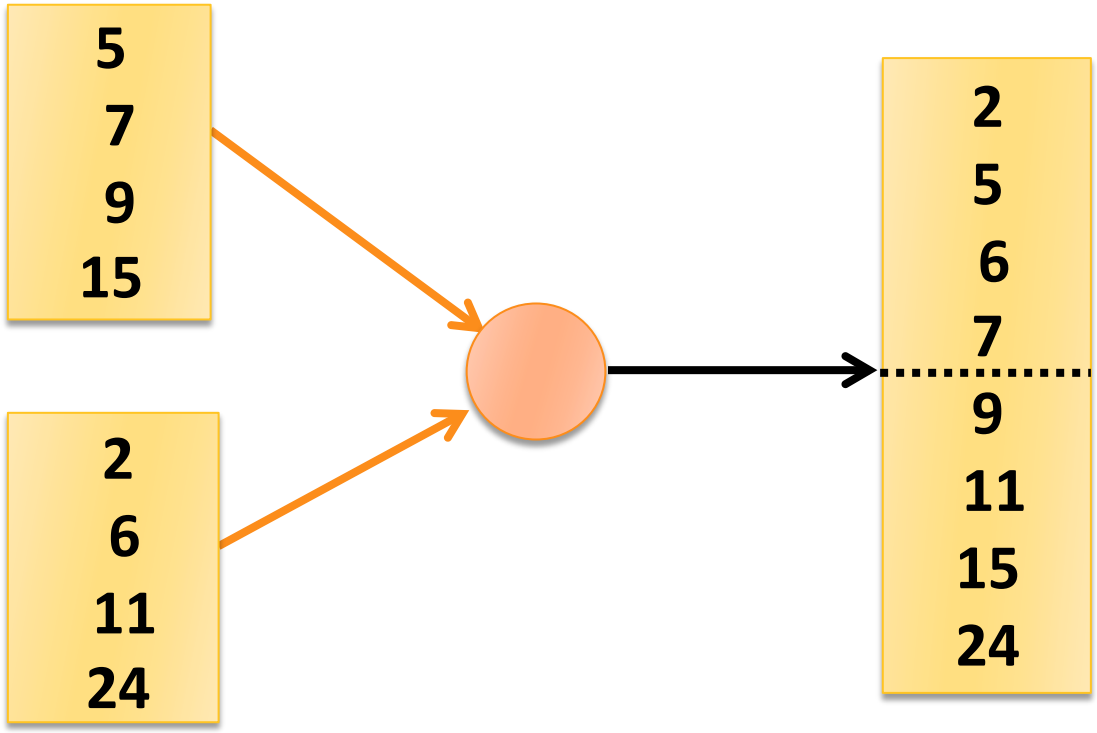
Ordinamento del file

FASE DI MERGE



Ordinamento del file

Esempio di fusione (merge) con M = 2



Ordinamento del file

- **Vediamo i passi di MERGE per un file di 81 blocchi con M = 3 , il sort produce 27 file di NM = 3 blocchi:**

F4 (9 file di 3 blocchi)
F5 (9 file di 3 blocchi)
F6 (9 file di 3 blocchi)

passo 1

F1 (3 file di 9 blocchi)
F2 (3 file di 9 blocchi)
F3 (3 file di 9 blocchi)

passo 2

F4 (1 file di 27 blocchi)
F5 (1 file di 27 blocchi)
F6 (1 file di 27 blocchi)

passo 3

F1 (1 file di 81 blocchi)

Ordinamento del file

- I passi di merge PM sono stati $\text{Log}_3(27) = 3$

  dove 27 è il numero di file che escono dal passo di sort
- In generale $PM = \text{Log}_M(NB/NM)$

  se NB/NM è una potenza di M il merge è bilanciato,

  altrimenti: $PM = \lceil \text{Log}_M(NB/NM) \rceil$
- Ad ogni passo di fusione la lunghezza di ogni file intermedio di uscita diventa:

  $NM \times M,\ NM \times M^2,\ NM \times M^3, \ldots$

  l'algoritmo termina alla k-esima fusione quando:

  $NM \times M^k \geq NB$

Ordinamento del file

- **Ogni passo comporta una lettura e una scrittura per ogni blocco (NB).**
- **Considerando che la fase di sort interno iniziale comporta un passo preventivo, otteniamo:**

$$C_{sort} = 2 \times NB \times (PM + 1)$$

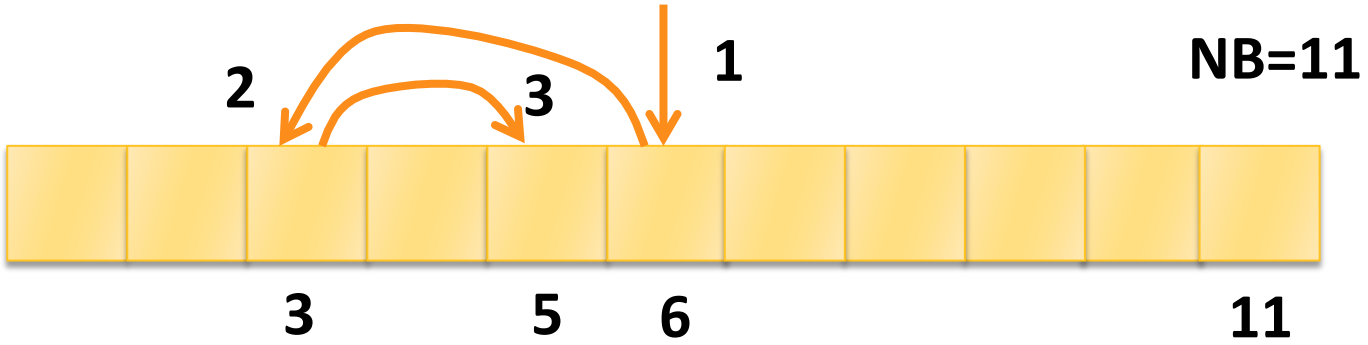
- **Quindi:**

$$C_{sort} = 2 \times NB \times (\lceil Log_M (NB/NM) \rceil + 1)$$

(sono comunque possibili ottimizzazioni)

Ricerca binaria

Sul file ordinato si può effettuare la ricerca binaria



la ricerca binaria ha un costo C_{bin} :

$C_{bin} = \lceil \text{Log}_2 NB \rceil - 1$ **(costo medio con successo)**

$C_{bin} = \lfloor \text{Log}_2 NB \rfloor + 1$ **(caso peggiore, insuccesso)**

con i dati dell'esercizio e $NM = 16$, $M = 8$ si ha

$C_{sort} = 100000$ **e** $C_{bin} = 13 \div 15 << NB/2 = 5000$

Calcolo del costo di lettura

Esempio di calcolo del costo di lettura di un record, quando i record sono ordinati rispetto alla chiave di ricerca.

Consideriamo come valori caratteristici del dispositivo quelli di un disco dove il costo della lettura di un blocco di dimensioni 512 byte è dato da:

$t_{read} = t_s + t_r + t_b$

con $t_s = 16$ ms; $t_r = 8.3$ ms;

$t_b = L_b / t(\text{data rate}) = 512/3k \text{ sec} = 0.17$

$t_{read} = 24.47$ ms

Se supponiamo di avere un archivio con

NB = 100000, $L_b = 512$,

Calcolo del costo di lettura

- con ricerca sequenziale abbiamo un numero di accessi medio = NB/2 e quindi un costo :
  - con blocchi non contigui
  - NL = NB/2 × t_{read} = 50000 × 24.47ms = 1223.5 s = ≈20 min
  - con blocchi contigui sullo stesso cilindro e trascurando il tempo di cambio di cilindro
  - NL = NB/2 × t_{read} = 50000 × 8.47ms = ≈7 min
- con ricerca binaria abbiamo un numero di accessi medi = $\lceil \log_2 NB \rceil$ -1= 16 accessi:

  NL = 16 × 24.47 = ≈392 ms cioè meno di un secondo!

Metodi di organizzazione

- In un file può esistere un solo ordinamento (su una sola colonna o su un gruppo)
- l'ordinamento è costoso da ottenere e da mantenere(a seguito di inserimenti di nuove tuple)
- l'ordinamento su un attributo favorisce solo alcune query e non ne favorisce altre
- l'ordinamento è un metodo di organizzazione 'primario', vedremo altre organizzazioni che possono essere utilizzate sia come primarie che come secondarie

Metodi di organizzazione

I tipi di organizzazione sono sostanzialmente due:

- **le organizzazioni ad INDICE che utilizzano file di supporto che riportano per ogni valore della chiave l'indirizzo nel file di dove è localizzata la tupla**
- **le organizzazioni HASH (indirizzo calcolato) che per allocare una tupla in un file sottopongono la chiave ad una trasformazione con una funzione detta funzione hash che trasforma il valore della chiave in un valore numerico che corrisponde all'indirizzo nel file**