ERASMUS

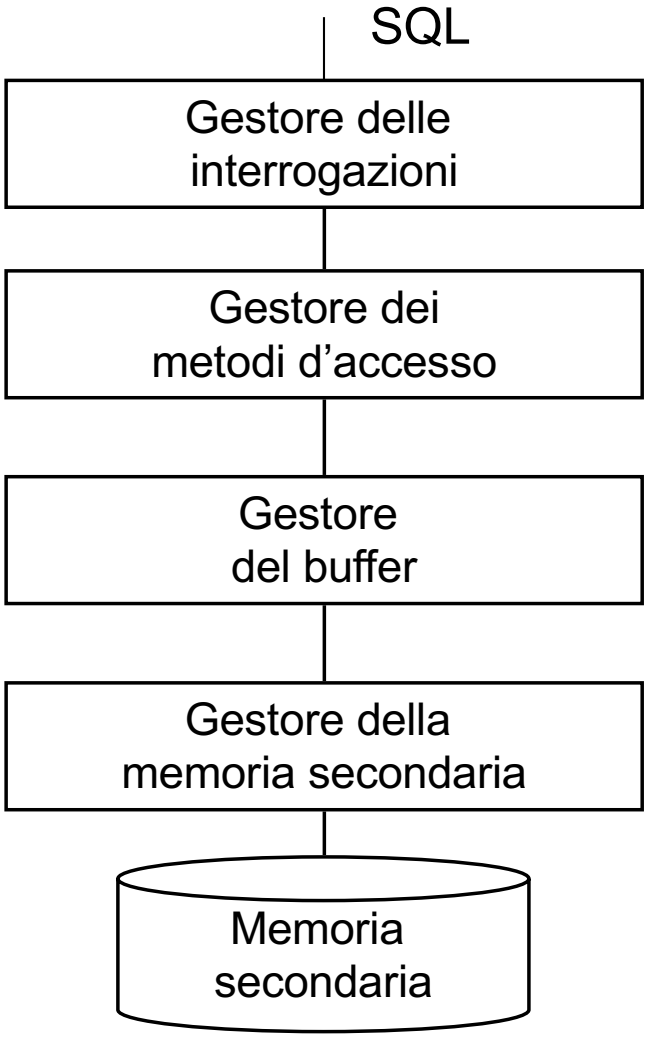
- 1 punto se lo studente ottiene la convalida di un numero di CFU compresi tra 20 e 29 CFU
- 2 punti se lo studente ottiene la convalida di almeno 30 CFU e congiuntamente la MEDIA dei voti minore o uguale a 25/30.
- 3 punti se lo studente ottiene la convalida di almeno 30 CFU e congiuntamente la MEDIA dei voti maggiore a 25/30.

Basi dati II

Organizzazione fisica dei dati

Cap. 14 -15 Sistemi di Basi di dati – Elmasri et al.
Cap. 7 Sistemi di Gestione dati- Catania et al.

Gestore degli accessi e delle interrogazioni



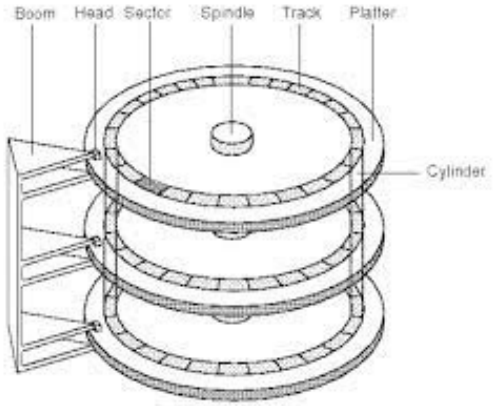
Memoria principale e secondaria

- Le basi di dati debbono essere (sostanzialmente) in memoria secondaria per due motivi:
  - dimensioni
  - persistenza
- I programmi possono fare riferimento solo a dati in memoria principale
- I dati in memoria secondaria possono essere utilizzati solo se prima trasferiti in memoria principale (questo spiega i termini "principale" e "secondaria")

Memoria principale e secondaria

- I dispositivi di memoria secondaria sono organizzati in **blocchi** di lunghezza (di solito) **fissa** (ordine di grandezza: alcuni KB)
- Le uniche operazioni sui dispositivi sono la lettura e la scrittura di un **intero blocco**
- Accesso a memoria secondaria:
  - tempo di posizionamento della testina
  - tempo di latenza
  - tempo di trasferimento

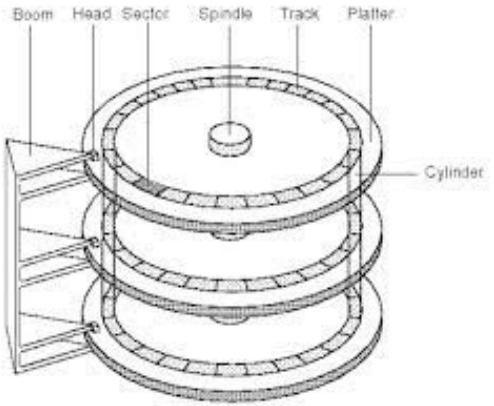
  in media 10 ms (hard disk 7200 rpm)



Il blocco rappresenta l'unità di trasferimento dati tra memoria secondaria e memoria principale

Memoria principale e secondaria

- Il costo di un accesso a memoria secondaria è quattro o più ordini di grandezza maggiore di quello per operazioni in memoria centrale
- Perciò, nelle applicazioni "I/O bound" (cioè con molti accessi a memoria secondaria e relativamente poche operazioni) il costo dipende esclusivamente dal numero di accessi a memoria secondaria
- Inoltre, accessi a blocchi “vicini” costano meno (contiguità)

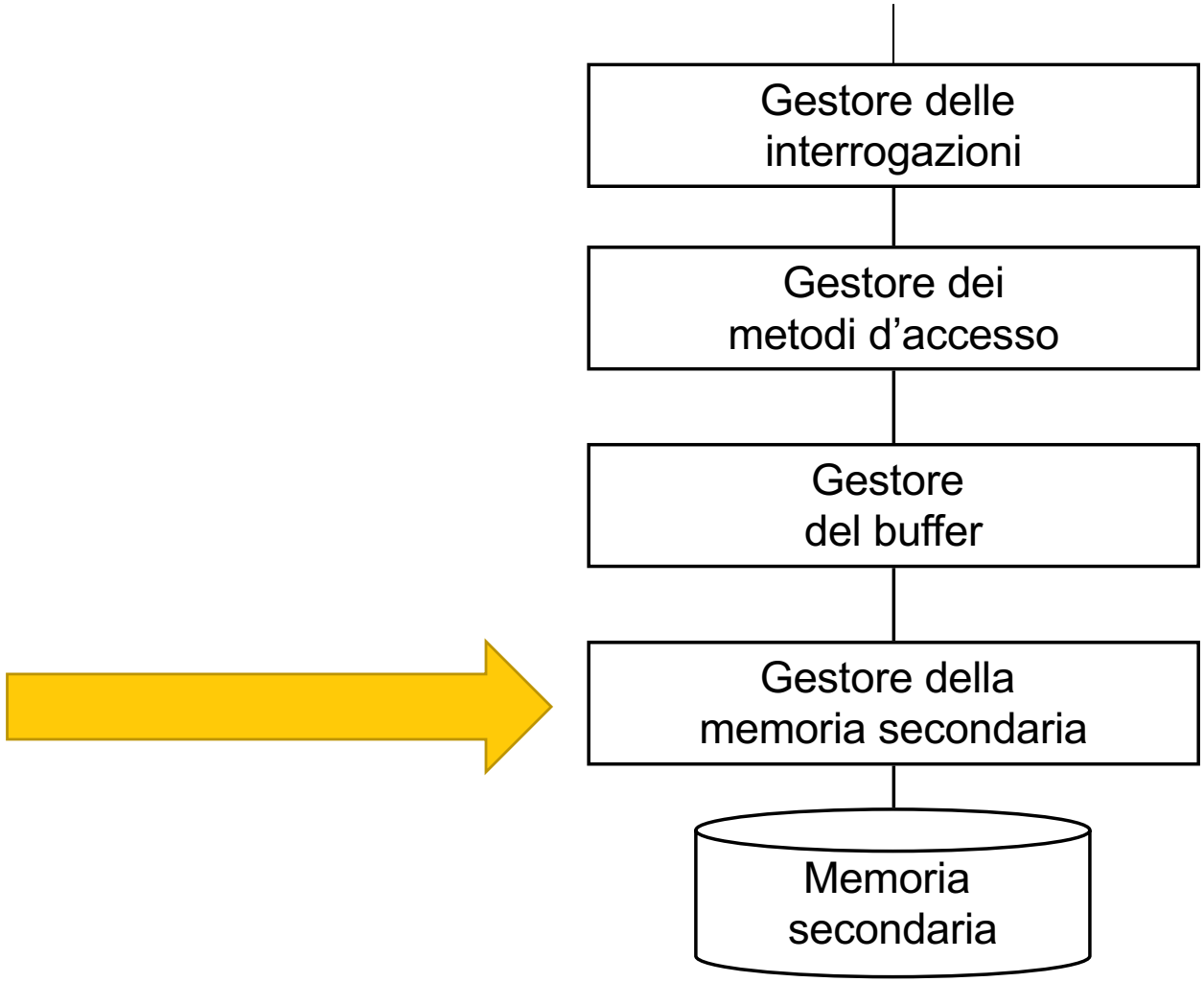


Come migliorare?

Buffer (si vedrà dopo)

Organizzazione fisica del file

Gestore degli accessi e delle interrogazioni



DBMS e file system

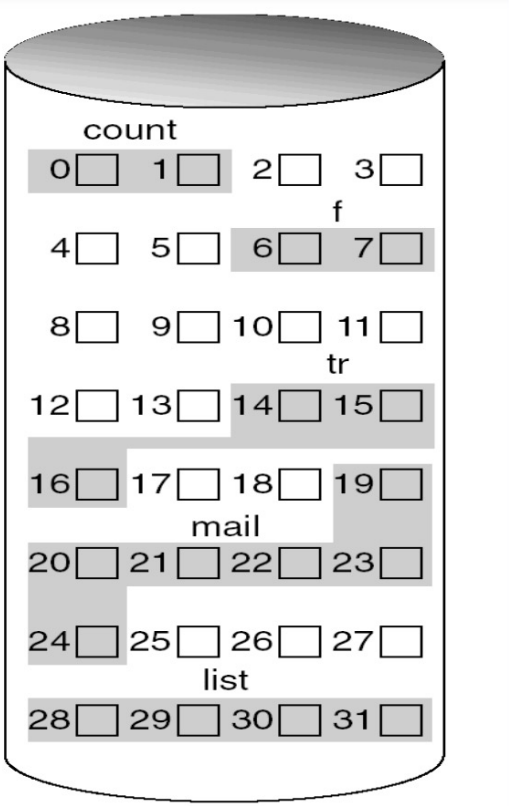
- Il file system è il componente del sistema operativo che gestisce la memoria secondaria
- I DBMS ne utilizzano le funzionalità:
  - per creare ed eliminare file
  - per leggere e scrivere singoli blocchi o sequenze di blocchi contigui

File, blocchi: allocazione

- I dispositivi di memoria secondaria sono organizzati in **blocchi** di lunghezza **fissa**
- L'**allocazione dei blocchi** di un file su disco può essere:

File, blocchi: allocazione

- L'**allocazione dei blocchi** di un file su disco può essere:
  - **contigua**: i blocchi di un file sono allocati consecutivamente;

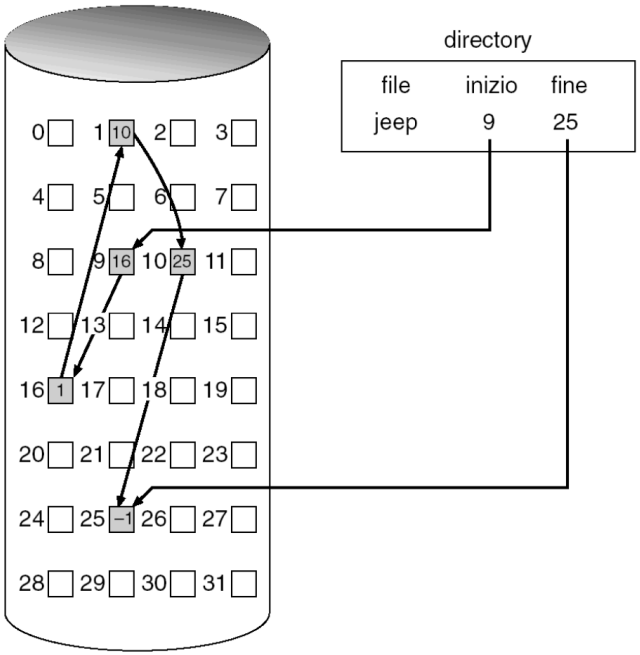


directory

file	inizio	lunghezza
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

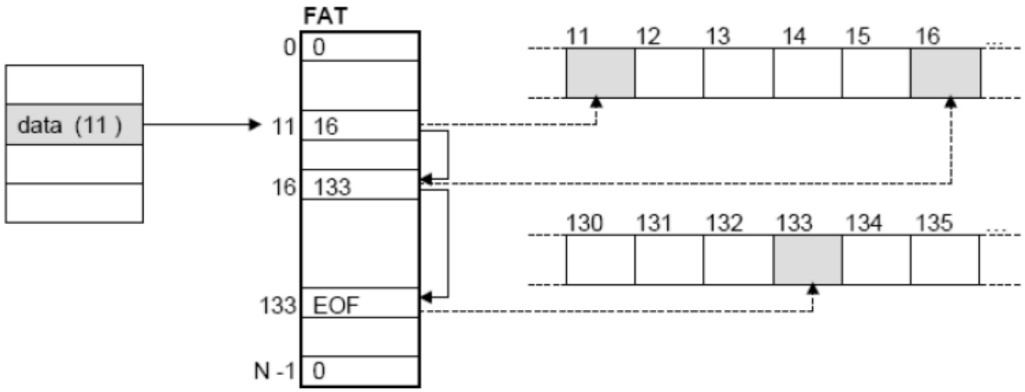
File, blocchi: allocazione

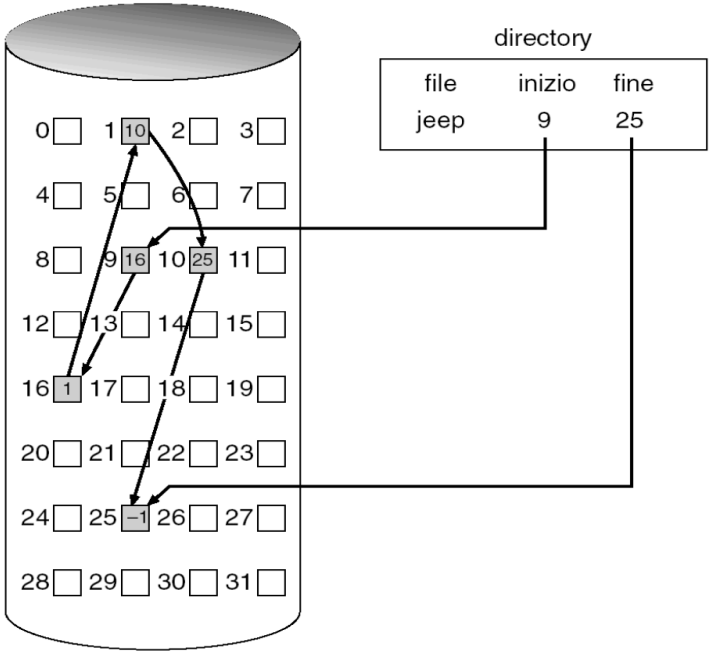
- L'**allocazione dei blocchi** di un file su disco può essere:
  - **contigua**: i blocchi di un file sono allocati consecutivamente;
  - **concatenata**: i blocchi non sono necessariamente consecutivi e sono collegati tra loro mediante puntatori;



File, blocchi: allocazione

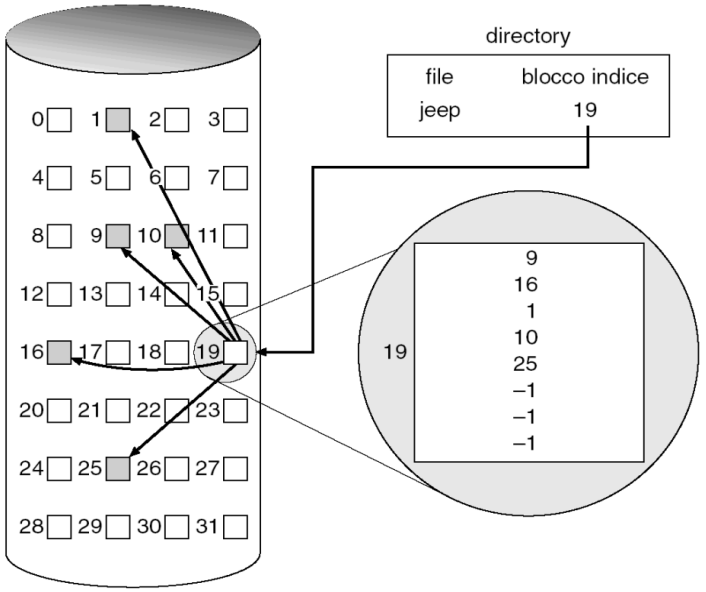
- L'**allocazione dei blocchi** di un file su disco può essere:
  - **contigua**: i blocchi di un file sono allocati consecutivamente;
  - **concatenata**: i blocchi non sono necessariamente consecutivi e sono collegati tra loro mediante puntatori;





File, blocchi: allocazione

- L'**allocazione dei blocchi** di un file su disco può essere:
  - **contigua**: i blocchi di un file sono allocati consecutivamente;
  - **concatenata**: i blocchi non sono necessariamente consecutivi e sono collegati tra loro mediante puntatori;
  - **indicizzata**: uno o più blocchi indice contengono i puntatori ai blocchi di dati del file.



Organizzazione interna dei file

- I dati nei file sono generalmente memorizzati in forma di record:
  - un record è costituito da un insieme di valori (campi) collegati
  - ogni tupla di una relazione corrisponde ad un record, ogni attributo ad un campo
- un file è una collezione di record:
  - file con record a lunghezza fissa se tutti i record memorizzati nel file hanno la stessa dimensione (in byte)
  - file con record a lunghezza variabile sono però necessari per:
    - memorizzazione di tipi di record diversi nello stesso file
    - memorizzazione di tipi di record con campi di lunghezza variabile (varchar)
    - memorizzazione di tipi di record opzionali
    - memorizzazione di tipi di record con campi multivalore (es. basi di dati OO o OR)

14

Organizzazione interna dei file

- Un file può essere visto come una collezione di record
- Tuttavia, poiché i dati sono trasferiti in blocchi tra la memoria secondaria e buffer, è importante assegnare i record ai blocchi in modo tale che uno stesso blocco contenga record tra loro correlati
- Se si riesce a memorizzare sullo stesso blocco record che sono spesso richiesti insieme si risparmiano accessi a disco

15

Mapping di relazioni a file

- L'organizzazione dei file, sia in termini di distribuzione dei record nei blocchi sia relativamente alla struttura all'interno dei singoli blocchi è gestita direttamente dal DBMS.
- Per DBMS di:
  - piccole dimensioni, una soluzione spesso adottata è di memorizzare ogni relazione in un file separato
  - grandi dimensioni, una strategia frequente è di allocare un unico grosso file, in cui sono memorizzate tutte le relazioni

File, blocchi, record, fattore di blocco

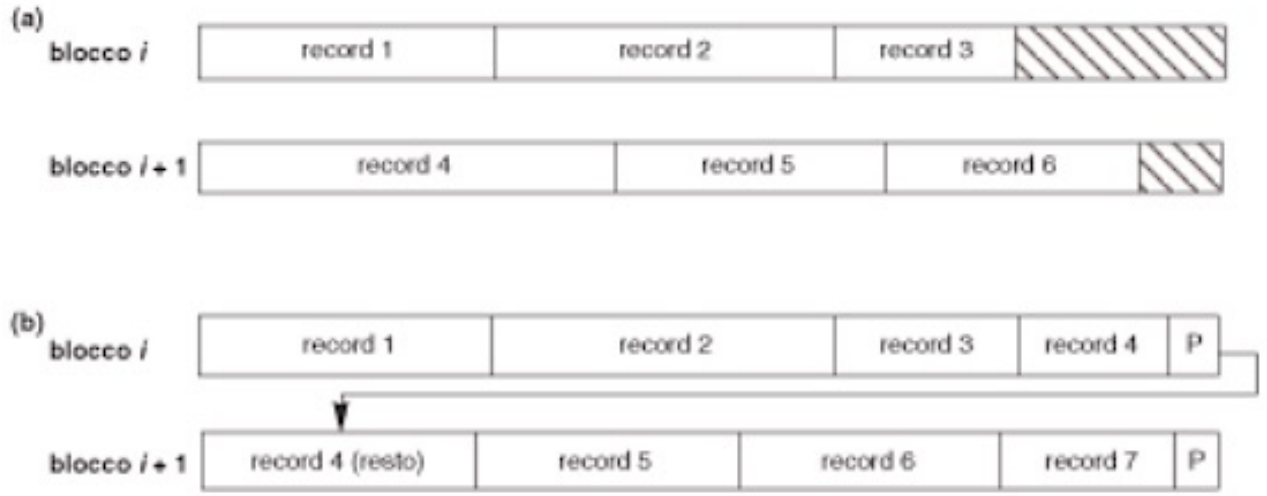
- I blocchi (componenti "fisici" di un file) e i record (componenti "logici") hanno dimensioni in generale diverse:
  - la dimensione del blocco dipende dal file system
  - la dimensione del record dipende dalle esigenze dell'applicazione (DBMS –> tuple), e può anche variare nell'ambito di un file
- Fattore di blocco: numero di record in un blocco
  - L_R: dimensione di un record (nei record a lunghezza fissa, o media della dimensione)
  - L_B: dimensione di un blocco
  - se $L_B > L_R$, possiamo avere più record in un blocco:

$$\lfloor L_B / L_R \rfloor$$

File, blocchi, record, fattore di blocco

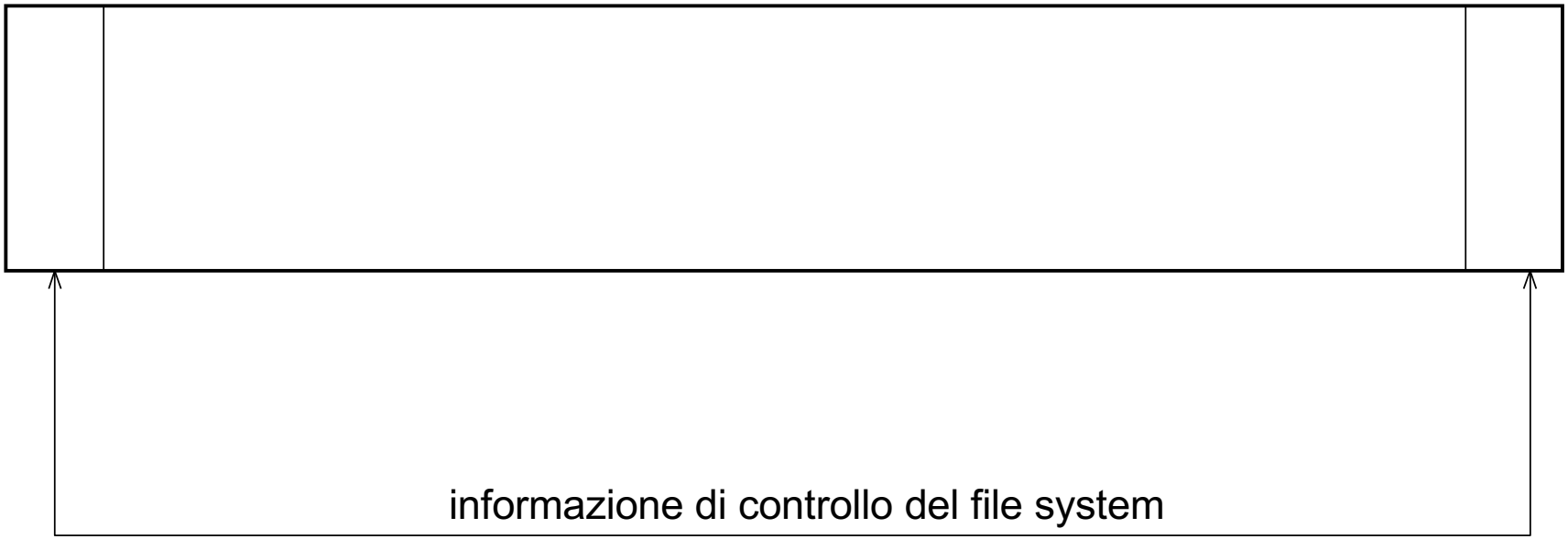
L'**allocazione dei record** nei blocchi può essere

- **unspanned**, in cui ogni record è interamente contenuto in un blocco, oppure
- **spanned**, in cui è possibile allocare un record a cavallo di più blocchi.



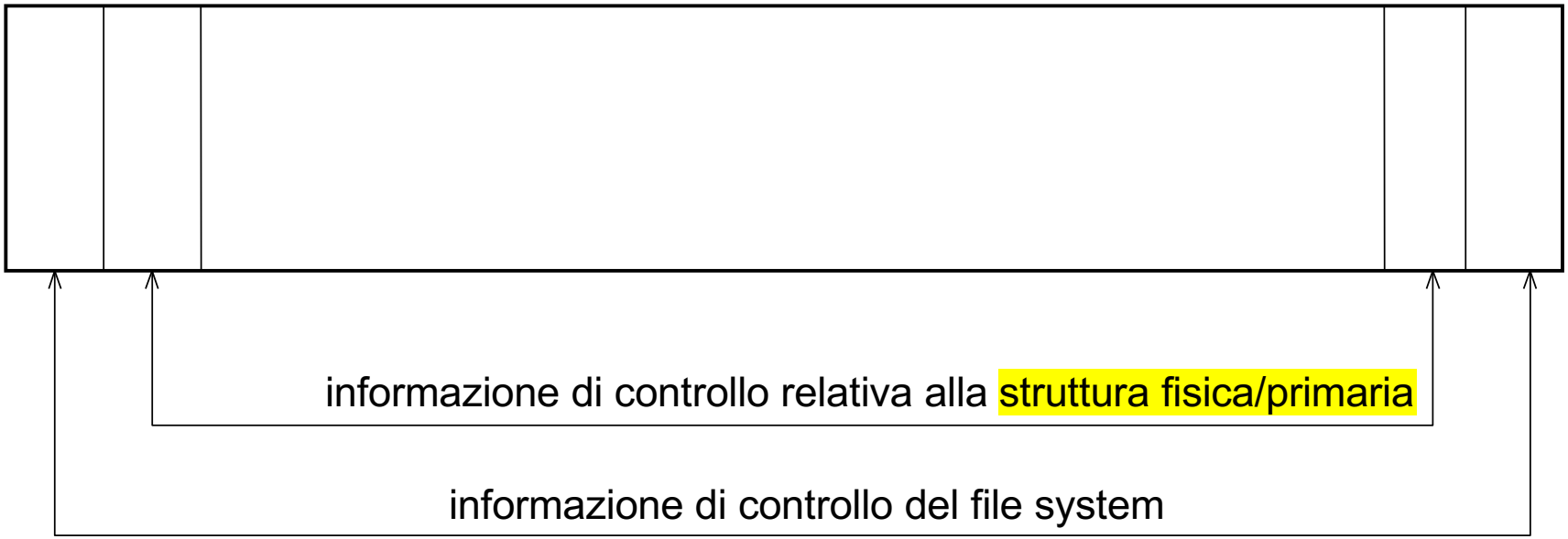
Organizzazione dei record all'intero di un blocco

- Ci sono varie alternative per organizzare i record (tuple) nei blocchi, vediamo una possibilità



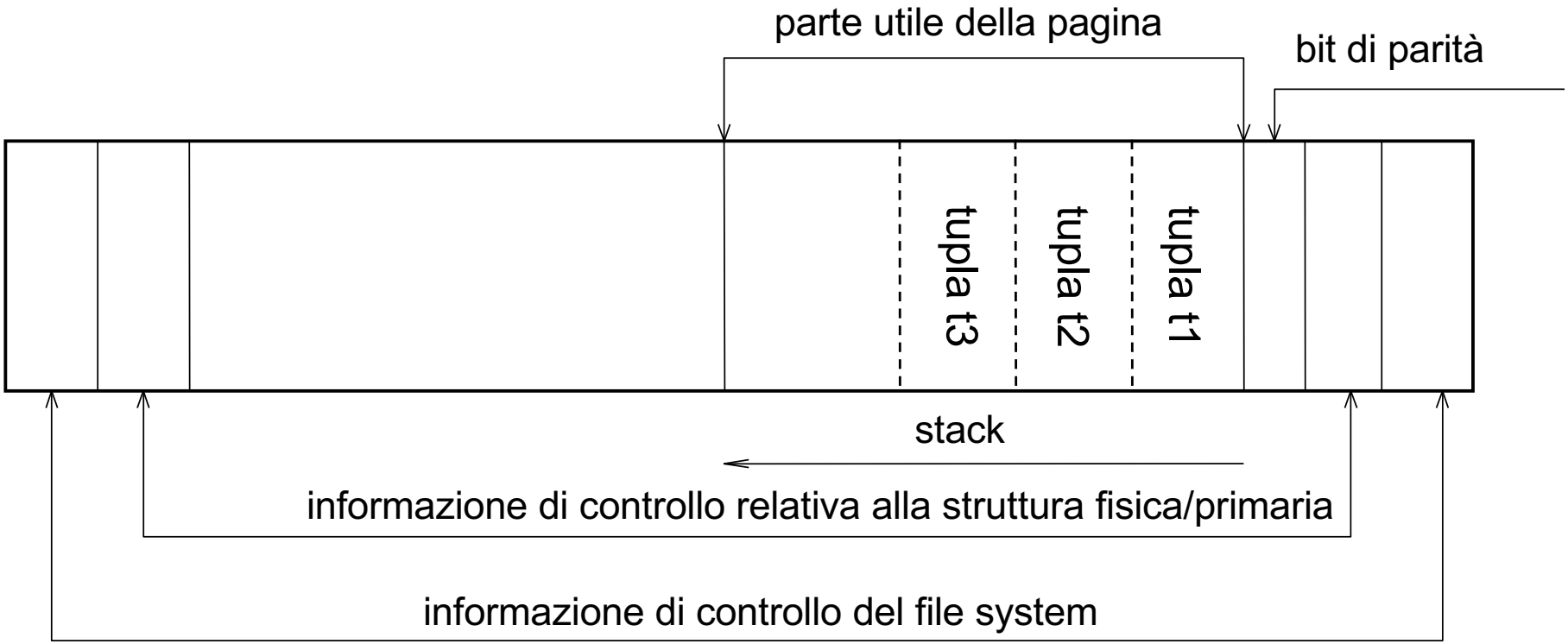
Organizzazione dei record all'intero di un blocco

- Ci sono varie alternative per organizzare i record (tuple) nei blocchi, vediamo una possibilità



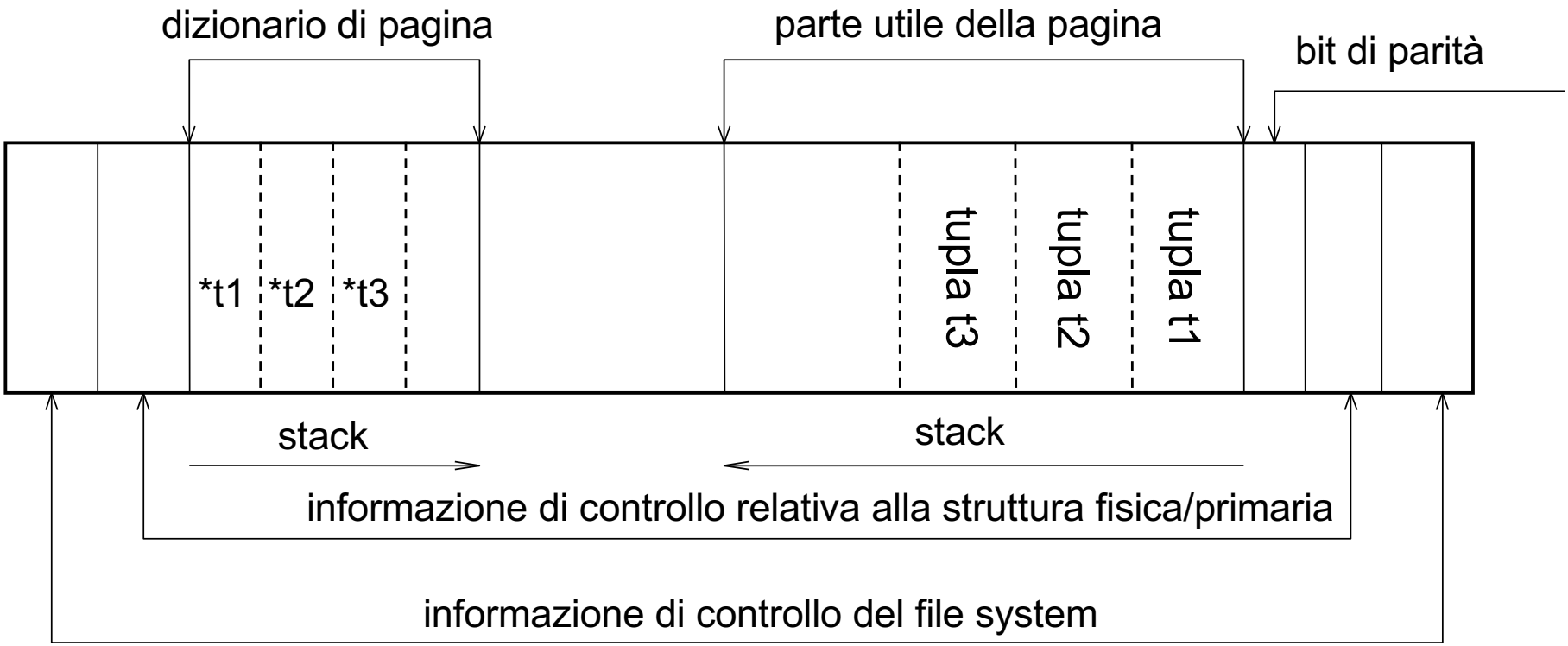
Organizzazione dei record all'intero di un blocco

- Ci sono varie alternative per organizzare i record (tuple) nei blocchi, vediamo una possibilità



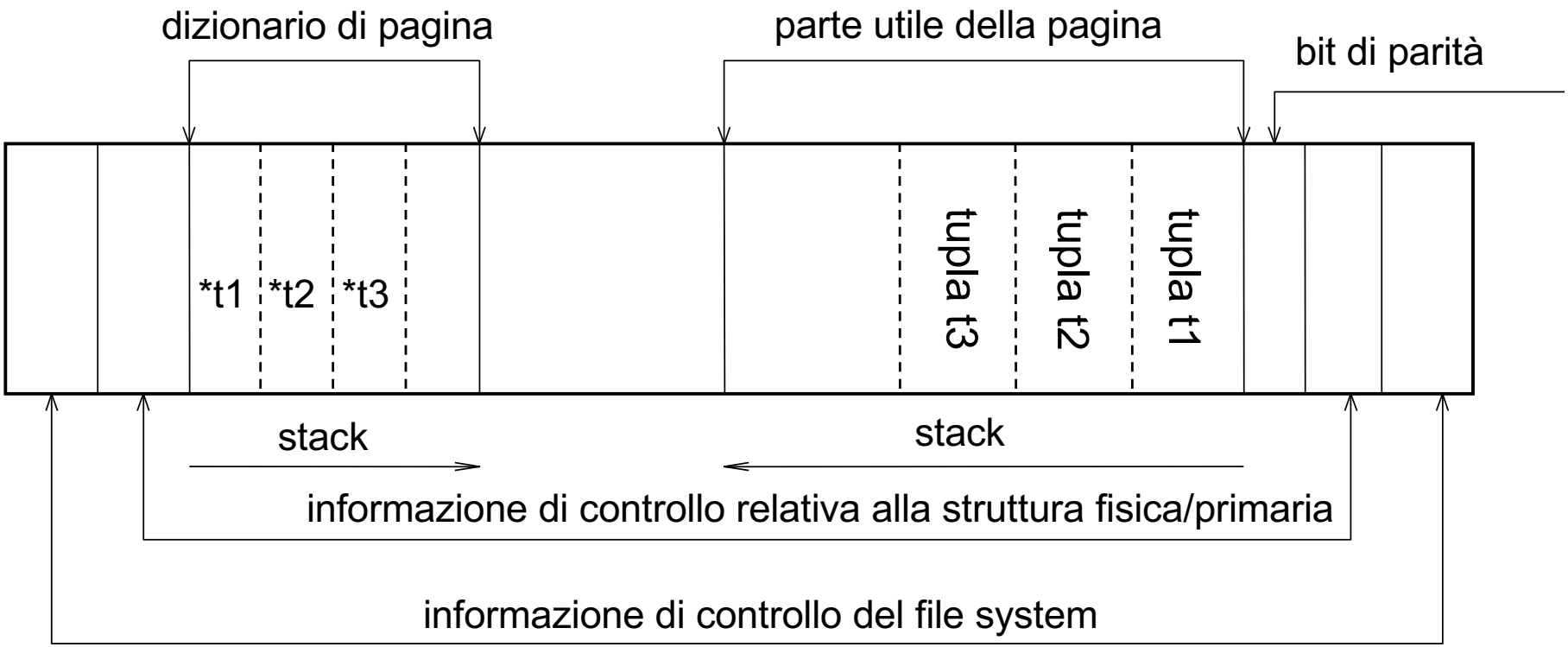
Organizzazione dei record all'intero di un blocco

- Ci sono varie alternative per organizzare i record (tuple) nei blocchi, vediamo una possibilità

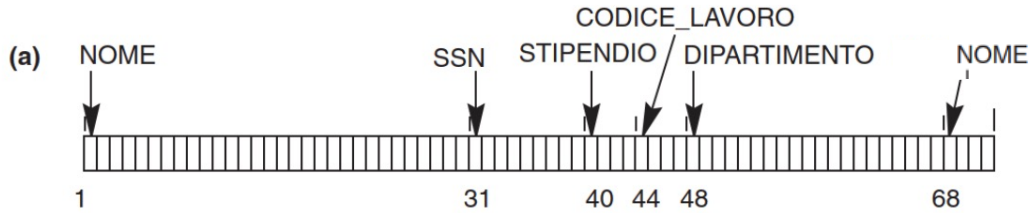


Organizzazione record all'intero di un blocco

- Dizionario di pagina contiene puntatori ai record/tuple
  - Record dim. fissa: …..
  - Record dim. variabile: ….



Organizzazione record all'intero di un blocco: Dizionario di pagina – record lunghezza fissa

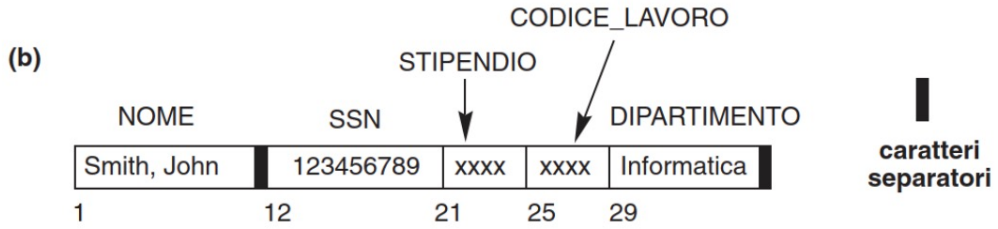



```
struct impiegato {
char nome[30];
char ssn[9];
int stipendio;
int codice_lavoro;
char dipartimento[20];
} ;
```

Puntatore campo nome del 3° record?
Puntatore campo ssn del 5° record?

Organizzazione record all'intero di un blocco:
Dizionario di pagina – record lunghezza variabile

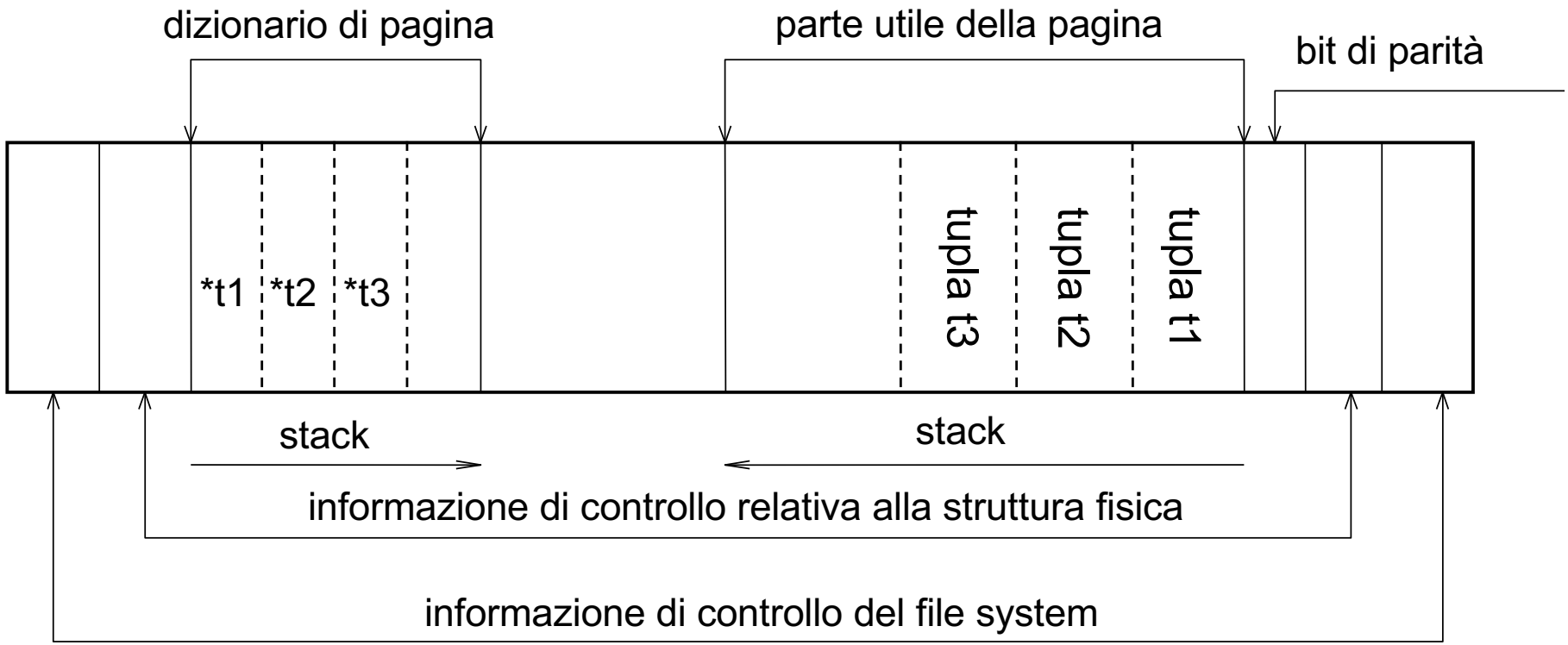
```
struct impiegato {
varchar nome[30];
char ssn[9];
int stipendio;
int codice_lavoro;
char dipartimento[20];
} ;
```



Per ogni record memorizzo l'offset da inizio parte dati
Per ogni campo, memorizzo offset da inizio record

Organizzazione record all'intero di un blocco

- Dizionario di pagina contiene puntatori ai record/tuple
  - Record dim. fissa: facile
  - Record dim. variabile: contiene offset di ogni tupla dall'inizio parte utile, e di ciascun campo da inizio tupla



Organizzazione dei record all'interno del file: strutture primarie

- **Strutture primaria**: Criterio con cui si memorizzano le tuple all'interno del file
- Tecniche principali
  - **Disordinata/seriale/heap***:* ordinamento fisico ma non logico
  - **ordinata**: l'ordinamento delle tuple coerente con quello di un campo
  - **Hash /accesso calcolato***:* posizioni individuate attraverso indici

Organizzazione dei record all'interno del file: strutture primarie: disordinata (heap)

- I record sono inseriti nel file nell'ordine in cui si presentano (ordine di inserimento nella relazione)
- Operazioni:
  - Inserimento (molto efficiente):
    - Un nuovo record viene inserito nell'ultimo blocco (richiesto solo un rif. all'ultimo blocco)
    - ATTENZIONE: se record hanno vincoli di chiave, è richiesta verificare tutti i recordi inseriti
  - Ricerca (poco efficiente):
    - Ricerca sequenziale. Costo lineare al numero di blocchi (mediamente si deve leggere metà dei blocchi)
  - Cancellazione:
    - Richiedere una ricerca per trovare il record da cancellare
    - Il record da cancellare può essere marcato come cancellato, senza ulteriori modifiche su altri record
      - Potrebbe portare a spreco di memoria e richiede periodica riorganizzazione dei file
  - Modifica:
    - Richiede una ricerca per trovare il record da modificare
    - Modifica
      - Si sovrascrive, se la modifica non cambia la lunghezza del record (aka record lunghezza fissa)
      - Altrimenti, si cancella il vecchio record e si inserisce il nuovo record con la modifica
- È molto diffusa nelle basi di dati relazionali, associata a indici secondari (discussi dopo)

Costo ricerca in termini di numero accessi

- Si supponga un file heap con r=300.000 record memorizzati su disco con blocco B=4096 di lunghezza. I record hanno lunghezza fissa R=100 byte e sono memorizzati in modo unspanned.

Mediamente, quanti accessi sono richiesti per ricerca un record?

- Fattore di blocco:

$$\lfloor 4096/100 \rfloor = 40$$

- Numero blocchi per memorizzare il file:

$$\lceil 300.000/40 \rceil = 7500$$

- Numero di accessi ai blocchi:

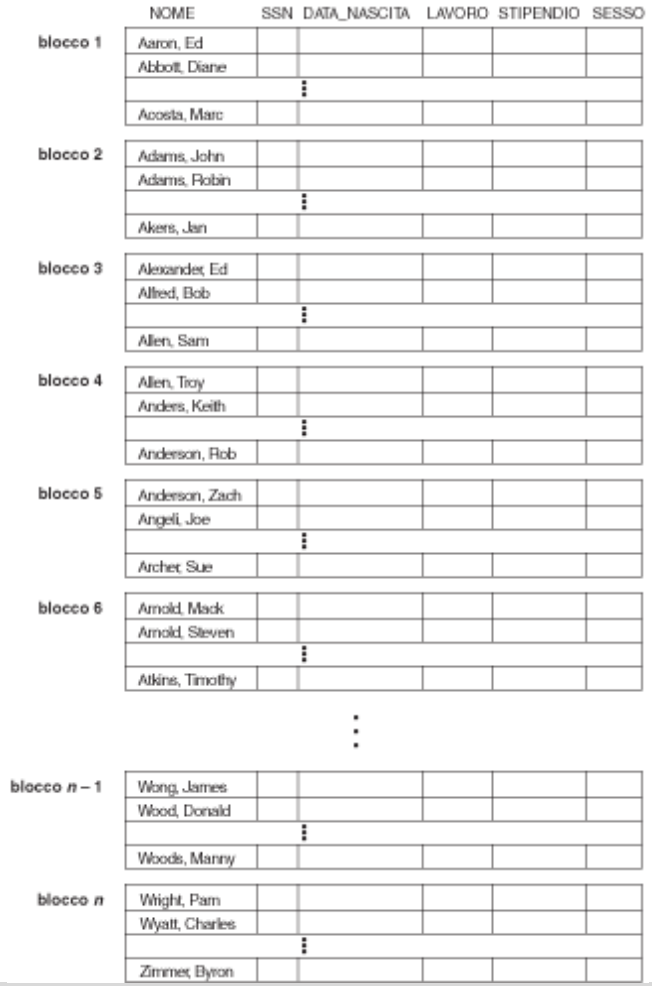
$$7500/2 = 3750$$

Organizzazione dei record all'interno del file: strutture primarie: ordinata (sorted)

- I record sono inseriti rispettando l'ordinamento di un campo (detto campo di ordinamento o chiave di ordinamento)

Organizzazione dei record all'interno del file: strutture primarie: ordinata (sorted)

- I record sono inseriti rispettando l'ordinamento di un campo (detto campo di ordinamento o chiave di ordinamento)



Organizzazione dei record all'interno del file: strutture primarie: ordinata (sorted)

- I record sono inseriti rispettando l'ordinamento di un campo (detto campo di ordinamento o chiave di ordinamento)
- Operazioni:
  - Inserimento (poco efficiente):
    - L'inserimento di un nuovo record deve rispettare l'ordinamento del campo. Questo può richiedere di spostare diversi record per inserirne uno nuovo
  - Ricerca (efficiente su campo ordinato):
    - La lettura dei record secondo l'ordine del campo d'ordinamento è estremamente facile (es. stampo impiegato ordinati per cognome)
    - Ricerca sul campo d'ordinamento ottimizzate
      - Ricerco singolo record -> ricerca binaria
      - Ricerco gruppi record con valori inclusi in un intervallo (range query) -> trovo il primo valore e leggo sequenzialmente
  - Cancellazione
    - Come per heap, con necessità di riorganizzare file
  - Modifica (poco efficiente)
    - Se non è possibile sovrascrivere il record modificato (aka cambia la lunghezza), è necessario riorganizzare il file per garantire l'ordinamento

Esercizio: ricerca binaria sul file ordinati

- Si assuma un file ordinato sul campo ID di B blocchi
- Si scriva il pseudocodice per eseguire una ricerca binaria del record con campo di ordinamento = k