11. ARCHIVI E FILE

La **rubrica dei contatti** presente nel nostro smartphone può rappresentare un buon esempio di **archivio di dati**.

L'archivio è un insieme di informazioni relative ad oggetti dello stesso tipo, memorizzato su un supporto di memoria permanente (memorie di massa).

Una gestione senza problemi di questa grande *massa* di dati dipende strettamente dalla loro corretta *organizzazione* e *memorizzazione* secondo una logica che renda la *ricerca* e la *consultazione* più efficienti possibile.

Un archivio di dati per essere considerato ben organizzato deve ovviamente contenere dati tra loro *omogenei* ossia tutte le informazioni devono avere le stesse caratteristiche.

Consideriamo dunque **la rubrica dei contatti** presente nel nostro smartphone: in essa troviamo, tra la svariata molteplicità dei tipi di informazioni possibili (l'email, l'indirizzo web, le note, l'account messenger, la suoneria, il tipo vibrazione, etc.) le seguenti tre informazioni fondamentali:

- il nominativo del contatto;
- l'indirizzo del contatto;
- un suo numero di telefono

Queste informazioni dette **campi** (in inglese "field") possono essere considerate gli **attributi** di ogni **contatto** e sono sempre raggruppati in un unico elemento detto **record**.

Un **campo** di un record può essere **composto** se è ulteriormente costituito da più informazioni semplici oppure può essere **elementare** se ciò non è.

DEF: Quindi un **RECORD (logico)** o registrazione è una **struttura di dati** composta da un insieme finito di elementi eterogenei detti **"campi"**.
I campi sono tra loro logicamente connessi e corrispondono agli **"attributi"**.

DEF: L'ARCHIVIO è una **struttura dati astratta** costituita da un insieme di **RECORD (logici)** che rappresentano **oggetti omogenei** (nel senso che tutti i record presenti devono avere i medesimi campi)
Ogni **RECORD (logico)** di un **archivio** è identificato per mezzo della sua **"posizione"** all'interno dell'archivio stesso e che ne costituisce il suo **"indirizzo logico"**

DEF: Il FILE è una **struttura fisica di memoria** in cui è possibile memorizzare informazioni sotto forma di **sequenza di byte** (file di byte) o **sequenza di record** (file di record fisici).
Il **FILE** dunque è la **struttura dati concreta** adatta a implementare **la struttura dati astratta** *archivio.*
All'interno di un FILE ogni **RECORD (logico)** è individuato tramite un **"indirizzo fisico".**

Esempio: Supponiamo che il contatto di un nostro amico compaia nella rubrica dei contatti in quarta posizione.
*Possiamo dire che **l'indirizzo logico** della sua registrazione (**RECORD logico**) all'interno **dell'archivio** è pari a 4.*
*Poiché però l'archivio ha dato origine al FILE dei contatti (ossia all'implementazione sulla memoria di massa del nostro smartphone dell'archivio stesso), la sua registrazione avrà un certo **indirizzo fisico** pari ad un numero intero espresso in esadecimale.*

IMPORTANTE

Un archivio di dati è sempre implementato mediante uno o più file, ma un file non è sempre l'implementazione di un archivio

(Esempio: basti pensare ad un file word, ad una immagine digitale, un file audio/video che contengono dati che non sono però in alcun modo organizzati e strutturati in accordo di una precisa struttura record).

CARATTERISTICHE FONDAMENTALI DI UN ARCHIVIO

Le tre caratteristiche fondamentali di un archivio di dati sono:

- la **permanenza**;
- la **razionalità**;
- la **sistematicità**

La **permanenza** è un concetto essenziale, anche se ovvio.
Una informazione registrata in modo che in seguito possa risultare incomprensibile o vada persa è una informazione potenzialmente inutilizzabile.
Occorre che i supporti dove conservare le informazioni connesse all'archivio di dati siano tali da conservarle inalterate per tutto il tempo necessario. Solo l'intervento dell'uomo potrà distruggerle o modificarle

La **razionalità** prevede che le informazioni di un archivio siano organizzate in modo da garantire un loro reperimento rapido e senza errori.

La **sistematicità** obbliga che le informazioni appartenenti ad uno stesso archivio debbano essere strutturalmente uguali tra loro.

CHIAVE PRIMARIA E CHIAVE SECONDARIA DI UN RECORD

Per rendere più efficiente (veloce) la ricerca delle informazioni in un archivio, è utile stabilire la **chiave** del record ossia un sottoinsieme dei suoi attributi i cui valori siano in grado di identificarlo univocamente.

DEF. In particolare si definisce **chiave primaria di un RECORD**, un campo o un insieme di campi (o ciò che è lo stesso un attributo o un insieme di attributi del record) **i cui valori** identificano **univocamente** un record all'interno dell'archivio.

Esempio:
Sono esempi di possibili chiavi primarie di archivi:
- il numero di matricola di uno studente delle superiori (o universitario);
- il codice di un articolo di un magazzino;
- il codice fiscale di una persona;
- il codice IBAN di un correntista.

Una caratteristica fondamentale della chiave primaria di un RECORD è la sua **non ridondanza** ossia essa non deve contenere campi (o attributi) superflui rispetto la sua funzione, potendo, se possibile, limitarsi ad un unico campo (o attributo)

Nota bene: spesso, si preferisce utilizzare chiavi primarie "parlanti" ossia i cui valori contengono sinteticamente due o più informazioni in grado di caratterizzare il record

*Primo esempio di chiave primaria "parlante": **il codice fiscale di una persona***

Supponiamo di volere conoscere il codice fiscale di un certo

BIANCHI MARIO, nato a ROMA il 20/01/1970

Esso sarà costituito da una stringa lunga 16 caratteri che risulta così suddiviso;

***Nome**: Sono necessari 3 caratteri e sono la 1a, la 3a e la 4a consonante, se il numero di consonanti è inferiore a 3 si aggiungo le vocali.*

***Cognome**: Sono necessari 3 caratteri per rappresentare il cognome e sono la 1a, la 2a e la 3a consonante, se le consonanti sono meno di tre si aggiungono le vocali nell'ordine in cui compaiono nel cognome*

***Anno**: Per l'anno vengono prese semplicemente le ultime 2 cifre.*

***Mese**: Per quanto riguarda il mese c'è una tabella di conversione in cui ad ogni mese viene associata una lettera dell'alfabeto, come riportato in seguito: (A = Gennaio) (B = Febbraio) (C = Marzo ...) D E H L M P R S (T = Dicembre). Alcune non sono usate per evitare confusioni (la I sembra un 1 o la O sembra lo 0).*

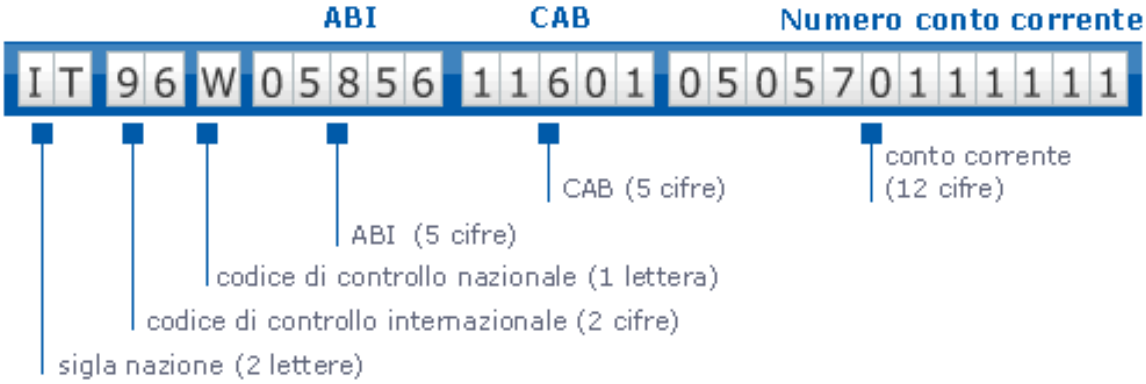
***Giorno**: Basta riportare il numero del giorno, con il particolare che per le persone di sesso femminile il numero deve essere incrementato di 40.*

***Comune di nascita**: E' composto da 4 caratteri alfanumerici e viene rilevato dai volumi dei codici dei comuni italiani oppure da vari data base (esempio F839 =NAPOLI e G964= POZZUOLI)*

***Il codice di controllo**: E' composto da 1 carattere e serve a verificare la correttezza dei precedenti caratteri in fase di digitazione.*

*Secondo esempio di chiave primaria "parlante": **il codice IBAN di un correntista***

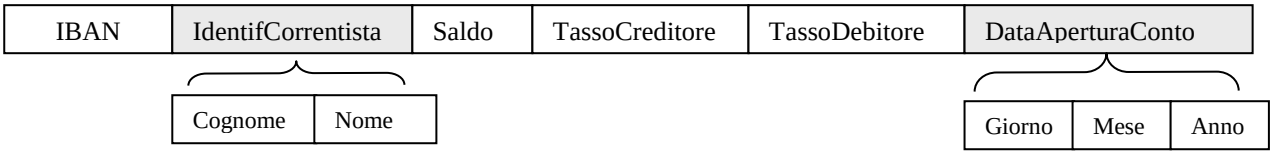
Il codice IBAN di un correntista di una qualunque banca italiana è così costituito:



N.B. Questo non è l'IBAN del professore…… :-)

DEF. Si definisce **chiave secondaria di un record**, una chiave non primaria che è in grado di individuare in genere più record presenti in un archivio.
Una **chiave secondaria** si dice poi **selettiva** se è in grado di individuare nell'archivio un numero non elevato di record.

Esempio: Consideriamo esistente il seguente tracciato record relativo ad un certo archivio di correntisti appartenenti a differenti banche:



*I campi **IdentifCorrentista** e **DataAperturaConto** sono **campi composti o aggregati** poiché costituiti da più informazioni elementari (ciò corrisponde di fatto al concetto di SOTTO RECORD), mentre tutti gli altri sono **campi elementari**.*

*Solo il campo **IBAN** ha contenuto univoco e quindi potrebbe essere utilizzato come **chiave primaria.***
(Teoricamente ci possono essere più correntisti con lo stesso IdentifCorrentista o Saldo o TassoCreditore o TassoDebitore o DataAperturaConto ma il correntista con IBAN pari à

IT 96 W 05856 11601 050570111111

è sicuramente, nel caso esista, uno ed uno solamente)

*Una **chiave secondaria** potrebbe essere considerato il campo **DataAperturaConto** poiché se applicata potrebbe in maniera selettiva individuare un numero non elevato di record (ad esempio i conti aperti prima o dopo una certa data).*

*Altra **chiave secondaria** potrebbe essere considerato il campo **IdentifCorrentista** in quanto se applicata mi permetterebbe di accedere anche questa ad un numero esiguo di record (ossia i conti correnti riconducibili ad una stessa persona)*

RECORD LOGICI E RECORD FISICI

Fino questo momento quando abbiamo parlato di **record** abbiamo inteso un insieme di informazioni relative ad una entità logica definita tenendo conto dell'applicazione che deve essere implementata.

Questa definizione coincide con quella di **record logico** ossia con la descrizione di come l'analista o il programmatore vorrà suddividere il gruppo di informazioni che caratterizzano l'oggetto osservato.

Il **record logico** ha una lunghezza in byte pari alla somma della dimensione dei campi che costituiscono il record stesso.

Sulla memoria di massa dove risiede il file che ha implementato il relativo archivio, i **record logici** devono essere raggruppati in **blocchi o record fisici** (di lunghezza prefissata)

DEF: Il RECORD fisico rappresenta **il numero di byte** che possono essere letti o scritti in memoria di massa (quindi su FILE) **con una singola operazione di lettura o scrittura**.

Esempio: supponiamo di avere un **record logico** con il seguente tracciato

TIPO Persona = **RECORD**
Nominativo: **ARRAY** [14] **DI CHAR**
Classe: **ARRAY** [2] **DI CHAR**
FINE RECORD

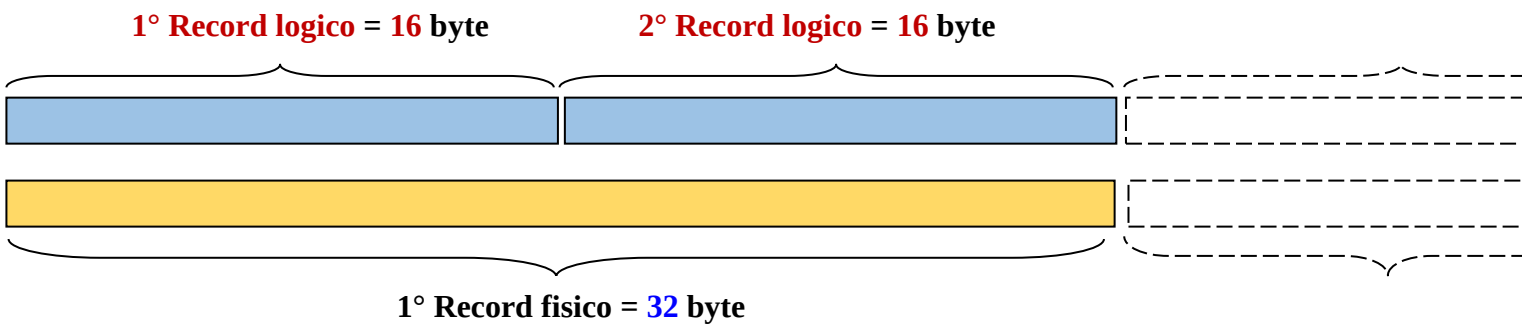
Questo **record logico** viene gestito dal programmatore come un insieme di due campi di **16 byte** di lunghezza complessiva.

Viene invece gestito dal **File System** (ossia da quel modulo del sistema operativo che si occupa della gestione di file ossia della memorizzazione e reperimento dei dati sui dispositivi di memoria di massa) **all'interno di un blocco di dimensione fissa da lui stesso stabilita**.

Supponiamo che nel nostro caso il **File System** preveda blocchi di **32 byte**
(nota bene le dimensioni tipiche dei blocchi di unità a disco rigido variano da 32 a 4096 bytes, tipicamente 512 e comunque sempre con valori pari a potenze di 2)

Allora con queste ipotesi appare chiaro che due record logici formeranno un record fisico.

Schematizzando:



DEF: Il numero di record logici presenti in un unico blocco fisico si chiama **fattore di bloccaggio** o **fattore di blocco**.
Tale numero può essere **maggiore**, **minore** oppure **uguale** ad **uno**

In particolare:

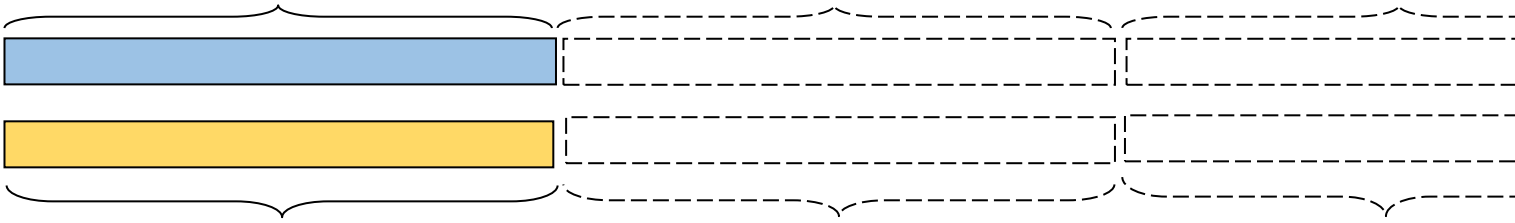
- se **è maggiore di 1** ossia se ogni record fisico contiene più record logici, i record si dicono **bloccati** *(n.b. è il caso mostrato dell'esempio precedente)*;

- se **è uguale ad 1** ossia se un record fisico corrisponde ad un solo record logico, i record si dicono **sbloccati**;

Esempio: cambiamo opportunamente il tracciato del **record logico** precedente ossia

TIPO Persona = **RECORD**
Nominativo: **ARRAY** [30] **DI CHAR**
Classe: ARRAY [2] DI **CHAR**
FINE RECORD

1° Record logico = 32 byte



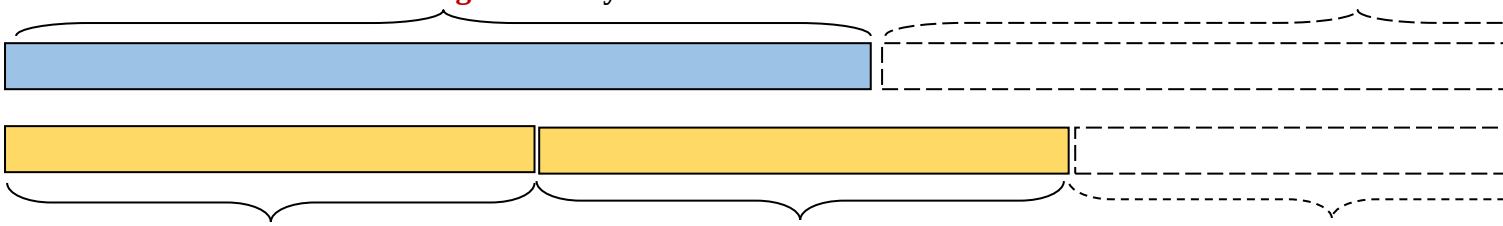
1° Record fisico = 32 byte

- se **è minore di 1** ossia se sono necessari più record fisici per memorizzare un record logico, si parla di **multiblocco**.

Esempio: cambiamo ancora opportunamente il tracciato del **record logico** precedente

TIPO Persona = **RECORD**
Nominativo: **ARRAY** [48] **DI CHAR**
Classe: **ARRAY** [2] **DI CHAR**
FINE RECORD

1° Record logico = 50 byte



1° Record fisico = 32 byte **2° Record fisico = 32 byte**

Per elaborare i dati contenuti nei record logici conservati in file presenti su memorie di massa è necessario un loro **trasferimento in memoria centrale (RAM)** poiché la CPU è in grado di operare solo su questo tipo di memoria.
Quando la CPU durante l'esecuzione di un programma richiede una operazione di lettura/scrittura di un record logico, non viene prelevato dalla memoria di massa solo il record logico interessato, bensì l'intero record fisico in cui esso è contenuto.
Per potere essere elaborato il record fisico viene posto in un area della RAM chiamata **buffer** adibita dal sistema operativo proprio a questo scopo.

Nel nostro esempio volendo leggere il 2° record logico, tale operazione comporterà il caricamento in un buffer della memoria centrale dell'intero record fisico (costituito da due record logici) all'interno del quale tale record logico è contenuto. Quindi anche 1° record logico verrà caricato all'interno di detto buffer.

ORGANIZZAZIONE DEGLI ARCHIVI

DEF: Per **organizzazione** o **implementazione** di un archivio si intende sia:
- il modo in cui l'archivio è rappresentato sul supporto fisico di memoria,
- il modo in cui viene elaborato.

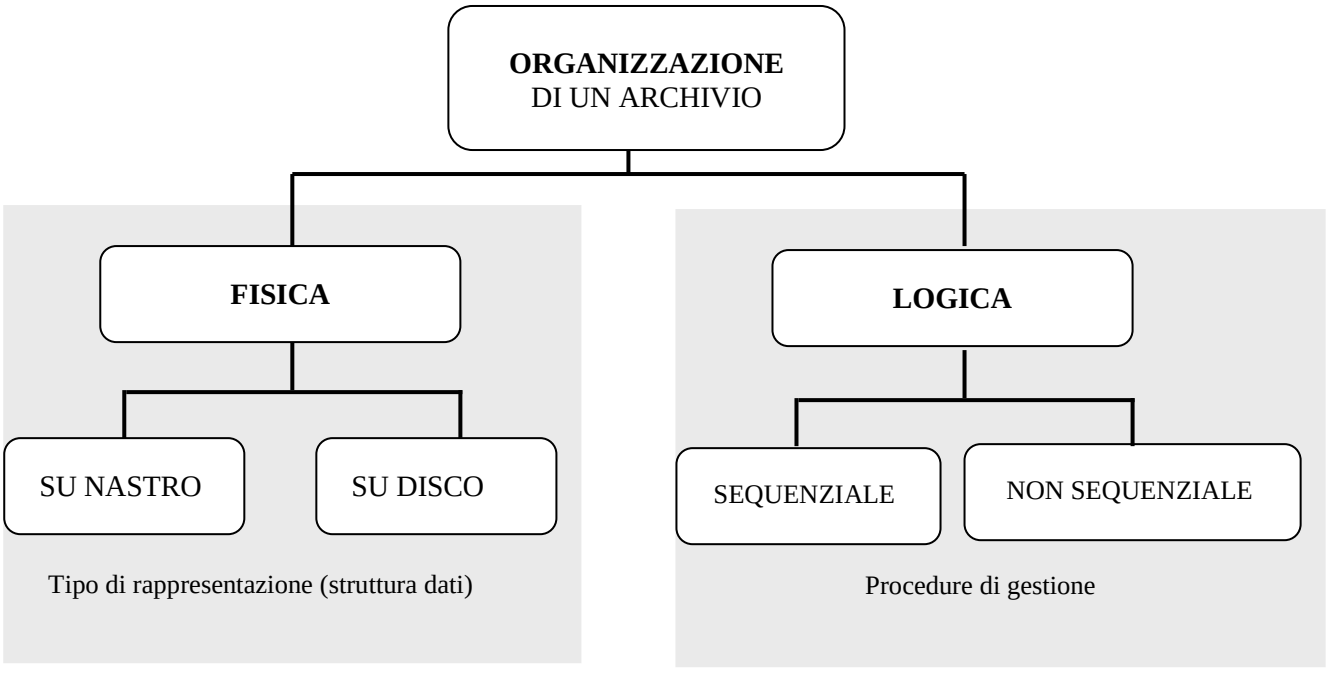
L'organizzazione pertanto si divide in:

- **fisica** ossia relativa al supporto fisico di memoria che contiene l'archivio di dati;

- **logica** ossia relativa alla modalità con cui viene elaborato l'archivio di dati.

N.B. L'aspetto fisico dell'organizzazione di un archivio riguarda la macchina, mentre l'aspetto logico riguarda il lavoro del programmatore

Organizzazione FISICA e LOGICA di un archivio



L'organizzazione fisica degli archivi

I supporti fisici per la memorizzazione dei file sono stati già trattati durante lo scorso anno scolastico (sistemi) parlando dell'architettura del computer.
Ricordiamo solo che i supporti fisici si classificano in:

- **supporti ad accesso sequenziale**: come i **nastri** nei quali per accedere ad un particolare record è necessario scorrere tutto il file record per record. La registrazione ed il reperimento di un record possono avvenire solo in modo consecutivo al record precedentemente registrato o esaminato.
In tal caso organizzazione logica ed organizzazione fisica coincidono.

- **supporti ad accesso diretto**: come i **dischi** nei quali è possibile accedere ai record sia sequenzialmente, sia in modo diretto. In questi tipi di supporto si può reperire una informazione anche in modo diverso rispetto alla loro organizzazione fisica.
Pertanto la dipendenza tra organizzazione fisica ed organizzazione logica diviene meno rilevante.

L'organizzazione logica degli archivi

L'organizzazione logica si riferisce alla gestione dell'archivio ossia stabilisce il modo in cui i record sono disposti al suo interno e di conseguenza il modo nel quale possono essere reperiti.

Essa può essere:
- **sequenziale**: quando per ogni record ne esiste uno precedente ed uno successivo ad eccezione del primo e dell'ultimo. In questo tipo di organizzazione i record sono memorizzati uno dietro all'altro.

- **non sequenziale**: se la sequenza non esiste, ossia quando i record sono memorizzati in maniera sparsa all'interno dello spazio a disposizione.

Su uno stesso supporto fisico è possibile implementare un archivio utilizzando diverse organizzazioni logiche.

Al concetto di organizzazione logica è quindi legato quello di **metodo di accesso** ai record.
Parlando dell'organizzazione fisica abbiamo già introdotto il termine "accesso" e lo abbiamo usato per caratterizzare la struttura fisica dei supporti di memorizzazione.

DEF: Con il termine **metodo di accesso** intendiamo l'operazione per mezzo della quale è possibile reperire i record contenuti nell'archivio.

Per quanto riguarda i metodi di accesso esso possono essere:
- **metodo di accesso sequenziale**: è il metodo di accesso che consente di accedere ad un record solo dopo avere fatto accesso a tutti i record che lo precedono. In questo caso il **tempo di accesso** al record è **proporzionale** alla posizione del record ossia dipende dal numero di record che devono essere scanditi prima di accedere al record che si vuole trovare.
E' il tipico metodo di accesso per i file memorizzati su supporti sequenziali come i nastri magnetici.
Quindi in un archivio implementato su nastro i dati potranno essere consultati nello stesso ordine con il quale sono stati inseriti.
E' evidente in questo caso la coincidenza tra organizzazione logica ed organizzazione fisica.

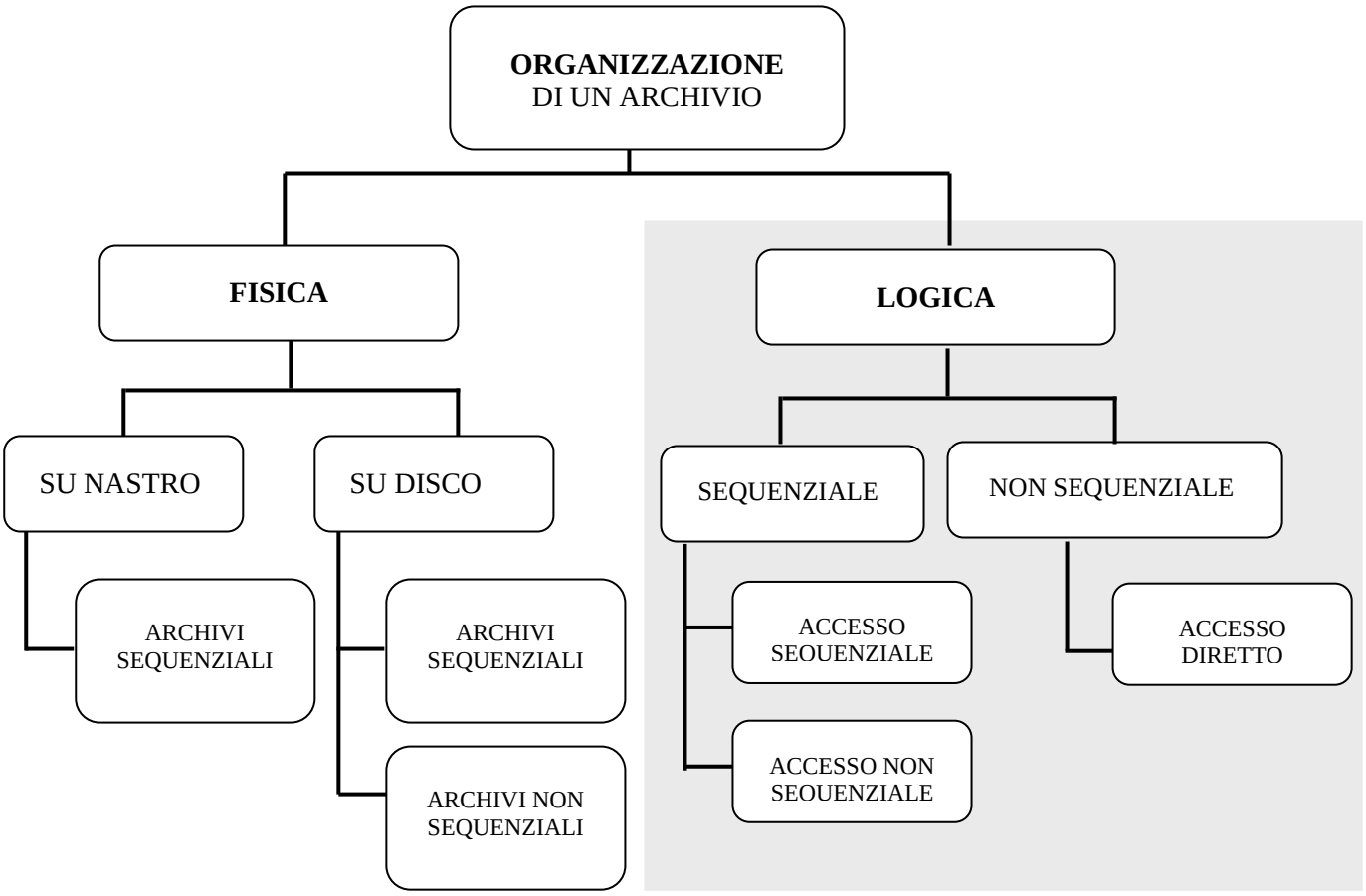
- **metodo di accesso diretto o casuale o random**: è il metodo di accesso che consente di posizionarsi direttamente sul record interessato in un tempo **indipendente** dalla posizione che esso occupa nell'archivio.

Esempio: supponiamo di voler registrare brani musicali su di un nastro oppure su un hard disk.
Le caratteristiche delle organizzazioni sono:

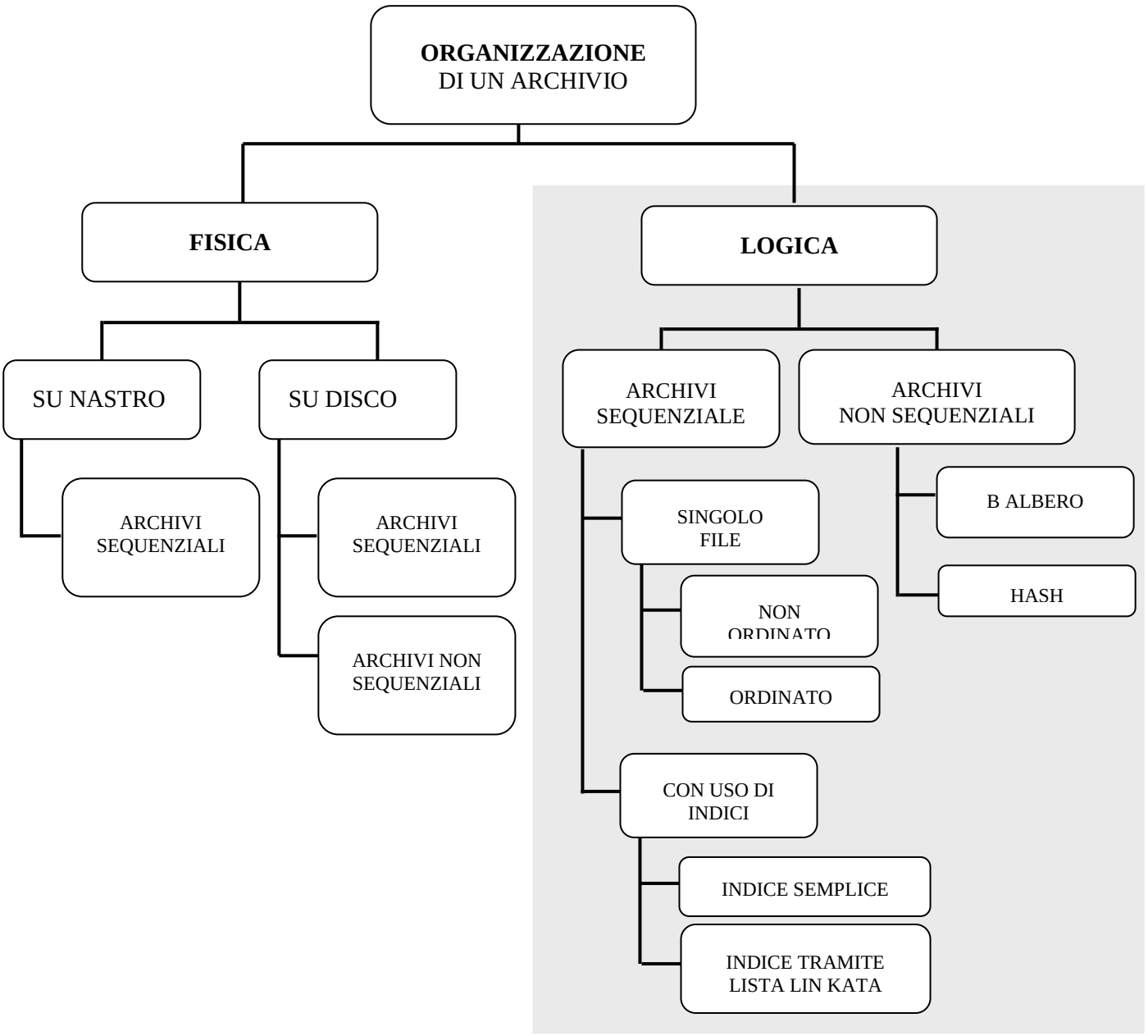
Tipo di organizzazione e metodo di accesso	NASTRO	HARD DISK
Organizzazione fisica	Sequenziale	Non sequenziale
Organizzazione logica	Sequenziale	Sequenziale
Metodo di accesso	Sequenziale	Sequenziale oppure diretto

Pertanto:
- i supporti fisici ad accesso sequenziale (come i **nastri**) possono ospitare solo archivi sequenziali ad accesso sequenziale;
- i supporti fisici ad accesso diretto (come gli **hard disk**) possono ospitare sia archivi sequenziali sia archivi non sequenziali.

L'organizzazione logica di un archivio ed i metodi di accesso



Le principali implementazioni delle organizzazioni logiche



FATTORI CHE INFLUENZANO LA SCELTA DELL'ORGANIZZAZIONE

Come abbiamo già detto, i criteri di organizzazione logica degli archivi sono condizionati dalla natura fisica dei supporti che li dovranno contenere.
Esistono però altri fattori che possono influenzare la scelta dell'organizzazione:

1. **Operazioni previste e loro frequenza d'uso:** continue operazioni di inserimento, cancellazione o aggiornamento dei dati possono influenzare notevolmente i tempi di risposta da parte del sistema;
2. **Frequenza di riferimento:** occorre rendere più accessibili i record più frequentemente utilizzati, creando opportune chiavi secondarie;
3. **Tempi e metodi di elaborazione:** se le operazioni eseguite sull'archivio avvengono in modalità interattiva (*on-line*) occorre garantire tempi di risposta immediati; se le operazioni non sono interattive (*off-line o batch*) è accettabile un tempo di risposta più lungo;
4. **Integrità e ripristino**: occorre tenete presente e valutare opportunamente le conseguenze ed i costi derivanti da spiacevoli situazioni accidentali che possono richiedere un recupero dei dati;
5. **Valutazioni del progettista:** l'organizzazione è legata alle scelte fatte dal progettista ed a volte al linguaggio di programmazione utilizzato.

LE OPERAZIONI SUGLI ARCHIVI

L'archivio è una struttura astratta di dati organizzati e sui quali si possono effettuare operazioni.

Tali operazioni possono essere classificate in:

- **operazioni fisiche**;
- **operazioni logiche**.

Le **operazioni fisiche** coinvolgono il file contenente l'archivio quale struttura fisica concreta e sono quelle che riguardano la **lettura** e la **scrittura** di record sul file.
Vengono definite *operazioni fisiche* in quanto la loro esecuzione comporta un accesso sia in memoria di massa sia in memoria centrale.
(Con l'operazione di lettura avviene un accesso alla memoria di massa ed il record fisico viene copiato nel buffer della memoria centrale. Con l'operazione di scrittura avviene dapprima l'accesso alla memoria centrale ed il record fisico viene depositato sul supporto fisico).
Anche la **creazione** di un archivio è una operazione fisica: essa consente di generare fisicamente la struttura ed il tracciato record.

Le principali **operazioni logiche** che riguardano l'archivio in quanto ADT sono:
- l'**apertura**: operazione che consente di aprire un archivio precedentemente creato;
- l'**inserimento** di un nuovo record all'interno di un archivio esistente;
- la **cancellazione** di un record ossia la sua eliminazione dall'archivio. Esistono due tipi di cancellazione: quella logica e quella fisica. La cancellazione logica non elimina fisicamente il record ma si limita a *marcarlo* valorizzando opportunamente un campo del suo tracciato precedentemente creato. La cancellazione fisica rimuove effettivamente il record dall'archivio.
- **l'aggiornamento** del contenuto dei singoli campi di un record già esistente;
- la **ricerca** dei dati relativi ad uno o più record per valori specifici della sua chiave ;
- la **scansione** che consiste nello scorrere tutti i record dell'archivio al fine di compiere una determinata operazione;
- l'**ordinamento** dei record dell'archivio in base ai valori assunti in indeterminato campo (in genere quello chiave). Questa operazione in genere ottimizza le operazioni di scansione e di ricerca.

Occorre tenere distinto l'ordine fisico (ossia la sequenza secondo la quale sono disposti i record sul supporto di memorizzazione) dall'ordine logico (ossia il modo in cui vengono visti i record dall'utente).
Un archivio si dirà ordinato quando ordine fisico ed ordine logico coincidono;
- la **chiusura** che consente di terminare le operazioni su di un archivio. Essa completa il trasferimento dei dati diretti alla memoria di massa.

OPERAZIONI LOGICHE SUGLI ARCHIVI E PSEUDOISTRUZIONI

PREMESSA: I metodi di organizzazione logica e le operazioni eseguibili su di un archivio sono strettamente legati al linguaggio di programmazione utilizzato.
La pseudocodifica deve essere svincolata da tali dipendenze.
Per questo motivo terremo nettamente separate le operazioni di lettura e di scrittura da quelle di posizionamento sui record della testina di lettura/scrittura.
Tutto questo per rendere la gestione degli archivi più semplice, generale ed applicabile a qualsiasi linguaggio di programmazione.

Pseudosistruzioni per il trattamento di archivi sequenziali

Assegna (<*NomeArchivio*>, <*NomeFile*>)

Questa pseudoistruzione fa corrispondere il file <*NomeFile*> all'archivio logico <*NomeArchivio*>. Implica la predisposizione di un buffer all'interno della memoria centrale.

CreaArchivio (<*NomeArchivio*>)

Questa pseudoistruzione fa in modo che il File System crei un nuovo file sul supporto fisico.
Se esso già esiste lo svuota.

ApriArchivioInScrittura (<*NomeArchivio*>)

Questa pseudoistruzione apre l'archivio <*NomeArchivio*>, consentendo di scrivere sui record.
L'apertura dell'archivio in scrittura posiziona la testina lettura/scrittura sul primo record.

ApriArchivioInLettura (<*NomeArchivio*>)

Questa pseudoistruzione apre l'archivio <*NomeArchivio*>, consentendo di leggere i record in esso contenuti. L'apertura dell'archivio in lettura posiziona la testina lettura/scrittura sul primo record.

ApriArchivio (<*NomeArchivio*>)

Questa pseudoistruzione apre l'archivio <*NomeArchivio*>, consentendo di scrivere e leggere i record. L'apertura dell'archivio in lettura/scrittura posiziona la testina lettura/scrittura sul primo record.

Leggi (<*NomeRecord*>, <*NomeArchivio*>)

Questa pseudoistruzione legge il record sul quale è posizionata la testina di lettura/scrittura nell'archivio <*NomeArchivio*> e trasferisce il contenuto in memoria centrale in <*NomeRecord*>.
Dopo la lettura la testina rimane sul record appena letto.

Scrivi (<*NomeRecord*>, <*NomeArchivio*>)

Questa pseudoistruzione scrive il record <*NomeRecord*> nell'archivio <*NomeArchivio*> sul record in cui è posizionata la testina di lettura/scrittura. Dopo la scrittura la testina rimane sul record appena scritto.

VaiSulProssimoRecord (<*NomeArchivio*>)

Questa pseudoistruzione sposta la testina di lettura/scrittura sul successivo record dell'archivio <*NomeArchivio*> rispetto la posizione corrente.

VaiAFineArchivio (<*NomeArchivio*>)

Questa pseudoistruzione sposta la testina di lettura/scrittura dopo l'ultimo record presente nell'archivio <*NomeArchivio*>.

E'FinitoArchivio (<*NomeArchivio*>) : BOOL

Questa pseudoistruzione è una funzione booleana che restituisce VERO se la testina di lettura/scrittura è posizionata sull'ultimo record presente nell'archivio <*NomeArchivio*>, FALSO in caso contrario.

EsisteArchivio (<*NomeArchivio*>) : BOOL

Questa pseudoistruzione è una funzione booleana che restituisce VERO se l'archivio <*NomeArchivio*> è già stato creato, FALSO in caso contrario.

CancellaArchivio (<*NomeArchivio*>)

Questa pseudoistruzione cancella l'archivio <*NomeArchivio*>.

RinominaArchivio (<*NomeArchivio*>, <*NuovoNomeFile*>)

Questa pseudoistruzione attribuisce al file associato all'archivio <*NomeArchivio*> il nuovo nome <*NuovoNomeFile*>

ChiudiArchivio (<*NomeArchivio*>)

Questa pseudoistruzione cancella l'archivio <*NomeArchivio*>. Dopo questa operazione non può essere effettuata alcuna operazione sull'archivio <*NomeArchivio*>.

Pseudosistruzioni per il trattamento di archivi NON sequenziali

Oltre a quelle sopra definite negli archivi ove è consentito l'accesso diretto possono essere utilizzate le seguenti pseudoistruzioni:

VaiSulRecord (<*NumeroRecord*>, <*NomeArchivio*>)

Questa pseudoistruzione sposta direttamente la testina di lettura/scrittura sul record che occupa la posizione <*NumeroRecord*> all'interno dell'archivio <*NomeArchivio*>.
Ricordiamo che i record all'interno dell'archivio sono numerati a partire da 1.

NumeroRecord (<*NomeArchivio*>): INT

Questa pseudoistruzione è una funzione che restituisce il numero di record presenti nell'archivio <*NomeArvìhivio*>.

LA DICHIARAZIONE DEI FILE

Prima di passare all'implementazione deglia rchivi, occorre definire il nuovo tipo di dato - t**ipo FILE** – attraverso il quale sarà possibile creare le variabili necessarie per contenere l'archivio logico.
Dichiarare un archivio significa:
- *dichiarare il record*;
- *dichiarare l'archivio* composto dai record del tipo precedentemente definito;

TIPO <NomeTipoArchivio> = **FILE DI** <NomeTipoRecord>

Esempio: se volessimo costruire un archivio destinato a rappresentare il listino prezzi di un grande magazzino le dichiarazioni che dovremmo fare sarebbero le seguenti:

```
TIPO prodotti = RECORD
                    codice: ARRAY[6] DI CHAR
                    descrizione: ARRAY[40] DI CHAR
                    prezzo: REAL
                FINE RECORD

TIPO listino = FILE DI prodotti          /* definizione di un tipo archivio di nome listino */

mioprodotto : prodotti                   /* dichiarazione di una variabile di tipo prodotti */
miolistino . listino                     /* dichiarazione di una variabile di tipo listino *
```

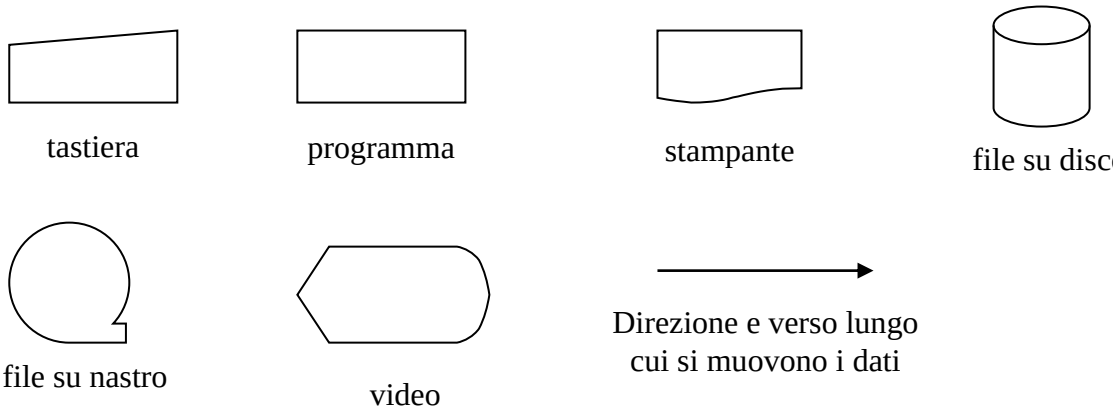
I DIAGRAMMI DI FLUSSO DEI DATI

Una grande importanza assumono i **diagrammi di flusso dei dati** nell'analisi dei dati relativi ad algoritmi che coinvolgono archivi.

Il **diagramma di flusso dei dati** o *data flow diagram* descrive ed illustra nell'ambito del sistema informativo il percorso e le trasformazioni delle informazioni, dei documenti e dei dati originali sino ai documenti ed ai dati finali.

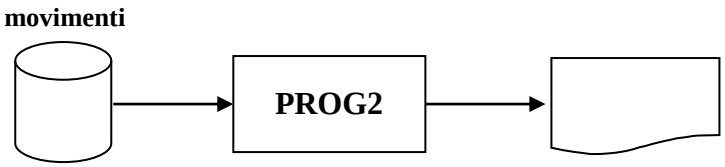
I **diagrammi di flusso dei dati** contengono pertanto riferimenti espliciti ai supporti di memorizzazione dei dati (nastro, disco) ed ai dispositivi di I/O (tastiera, video, etc.) utilizzati.

I principali simboli utilizzati nei *data flow diagram* sono:

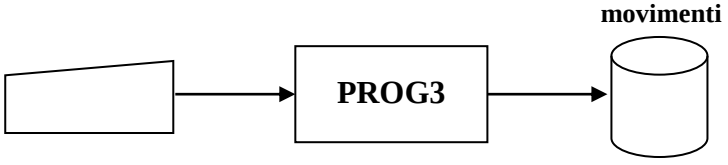


Esempi di *data flow diagram*

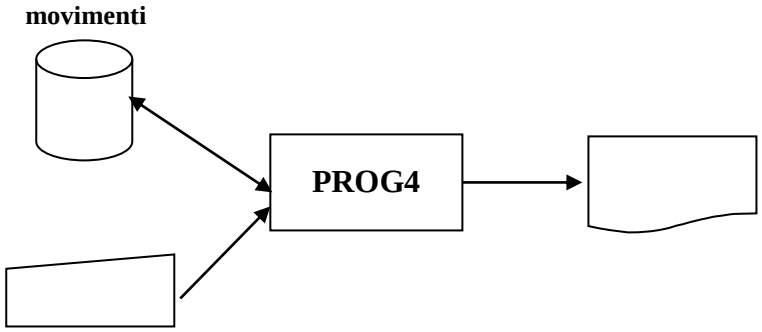
Il programma **PROG1** riceve i dati da tastiera e fornisce i risultati su monitor



Il programma **PROG2** riceve i dati attraverso la lettura su disco del file *movimenti*. I dati elaborati da PROG2 verranno poi inviati alla stampante
(N.B. All'interno dei simboli è possibile inserire i nomi dei dati coinvolti



I dati immessi da tastiera verranno manipolati dal **PROG3** ed al termine dell'elaborazione l'output memorizzato su disco nel relativo file *movimenti*



L'input è costituito sia dal file su disco movimenti sia da altri dati immessi da tastiera. Una volta elaborati dal programma **PROG4** i dati di output saranno visualizzati sulla stampante e memorizzati sul file movimenti stesso

IL TRATTAMENTO DEGLI ARCHIVI SEQUENZIALI

NOZIONI DI BASE

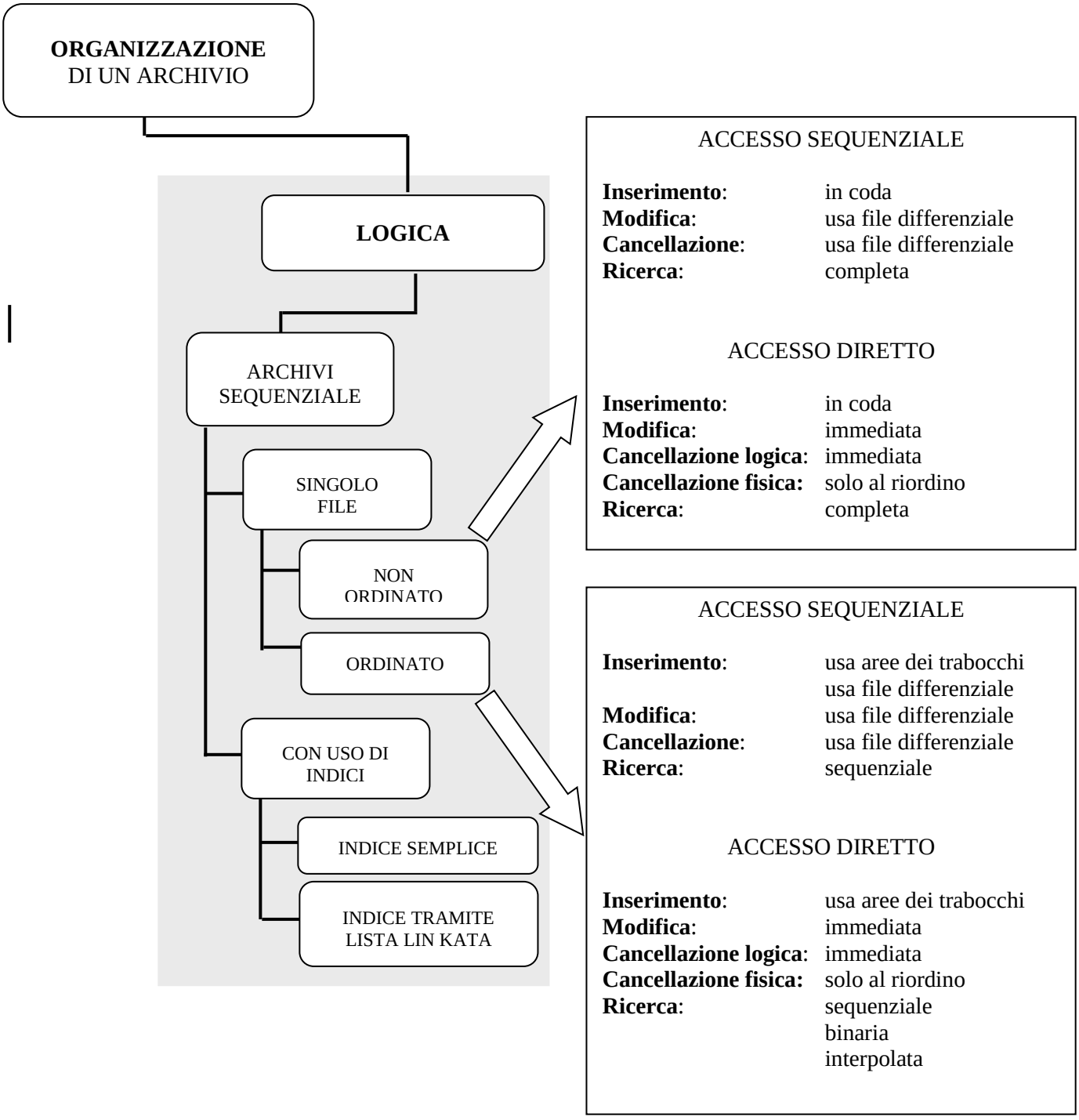
Con **archivio sequenziale** intendiamo un archivio in cui i record sono registrati in posizioni **contigue** a partire dalla prima.
Quando su uno o più campi del record è possibile stabilire un ordinamento ossia è definibile un *ordinamento logico* tra le registrazioni ed esso coincide con l'*ordinamento fisico,* allora l'archivio viene detto archivio **sequenziale ordinato** altrimenti archivio **sequenziale seriale**.

N.B. Uno dei principali **vantaggi** nell'utilizzo di archivi sequenziali ordinati è che molte elaborazioni possono essere eseguite più velocemente rispetto a quelli ordinati. Pertanto spesso è necessario prima **ordinare** l'archivio nel caso non lo fosse prima di sottoporlo ad altre elaborazioni.
Senza scendere nei dettagli molti metodi di ordinamento descritti l'anno scorso possono essere applicati tranquillamente anche agli archivi sequenziali ad accesso diretto, mentre ciò non è valido per gli archivi memorizzati su dispositivi ad esclusivo accesso sequenziale (*ad esempio i nastri*).

Tra le caratteristiche fondamentali dell'**organizzazione sequenziale** ricordiamo:
- la possibilità di **immissione solo in coda** (e quindi non viene perso l'ordine di immissione);
- i **vantaggi** per le operazioni **batch** (ossia *off-line*);
- l'**inefficienza** per elaborazioni di tipo **interattivo** (*on-line*) a causa degli elevati tempi di risposta;
- la possibilità di utilizzare esclusivamente il **metodo di accesso sequenziale** puro.

OPERAZIONI LOGICHE SUGLI ARCHIVI SEQUENZIALI IMPLEMENTATI CON SINGOLO FILE

Le operazioni consentite su di un **archivio ad organizzazione sequenziale implementato su di un unico file** sono riassunte nel seguente diagramma:



INSERIMENTO
Occorre distinguere tra *archivi non ordinati* ed *archivi ordinati*.

Su un *archivio non ordinato* **indipendentemente dal metodo di accesso** l'inserimento non comporta problemi: il record viene aggiunto in coda oppure, se nell'archivio esistono posizioni libere, il record può essere inserito nella prima di esse.

Su un *archivio ordinato* **indipendentemente dal metodo di accesso** l'operazione crea qualche problema: occorre individuare l'esatta posizione in cui inserire il record ed una volta individuata, occorrerà traslare tutti i record successivi in modo da creare lo spazio necessario.
Appare evidente che se nell'archivio esistono posizioni libere l'inserimento diverrebbe meno oneroso in quanto si potrebbe limitare la traslazione soltanto fino al primo spazio libero.
Allora perché non creare questi spazi liberi appositamente in fase di progettazione dell'archivio?
Occorrerà quindi in base alla dimensione del problema ed alle risorse a disposizione distribuire un certo numero di aree libere lungo l'archivio.
Queste certamente agevoleranno l'operazione di inserimento ostacolando però l'operazione di *ricerca*.
Vi sono tuttavia alcune soluzioni efficienti che possono essere applicate solo su archivi ordinati.

1) Si possono registrare i nuovi record in un *archivio temporaneo o differenziale* e si procede poi ad una *fusione* periodica tra i due archivi (principale e temporaneo) per ricostruirne uno solo.
Tale soluzione appare molto svantaggiosa quando si deve effettuare una ricerca perché i tempi di risposta diverrebbero la somma dei tempi di ricerca sui due archivi (principale e temporaneo)

2) E' possibile predisporre nell'archivio delle aree libere dette **aree di overflow** (o **aree dei trabocchi**) gestite tramite liste concatenate destinate a raccogliere i nuovi record inseriti.
Tali aree possono essere sistemate *in più punti* nell'archivio principale oppure in un altro archivio differenziale (**aree di overflow distribuito**) oppure essere allocate *in un unico punto* dell'archivio principale oppure di un altro archivio differenziale (**aree di overflow concentrato**).
Questa è la soluzione ottimale.

MODIFICA (O AGGIORNAMENTO O RISCRITTURA)
La fase di aggiornamento ossia l'operazione che consente di modificare i campi del record (ad eccezione della chiave) può essere svolta in due modi distinti che dipendono dal metodo di accesso.

Se l'accesso è sequenziale, l'aggiornamento di un record può essere svolto solo riscrivendo l'intero archivio. Le modifiche vengono raccolte in un altro archivio (differenziale) e successivamente si provvede ad aggiornare l'archivio principale.

Se l'accesso è diretto, l'aggiornamento di un record risulta essre una delle operazioni più semplici e si realizza in due fasi:

a) **ricerca** del record con chiave K da modificare
b) **riscrittura** del record modificato nella medesima posizione.

CANCELLAZIONE
Anche per la cancellazione vale quanto detto per l'aggiornamento.
Se l'accesso è sequenziale, l'unico modo per poter cancellare un record (*e mantenere la corrispondenza tra struttura logica e struttura fisica*) è quello di riscrivere l'intero archivio privato del record stesso su di un archivio differenziale.

Se l'accesso è diretto, è possibile evitare la riscrittura dell'intero archivio effettuando la **cancellazione logica** (*ossia predisponendo un apposito campo in grado di contenere un valore che indica la cancellazione del record*) e poi periodicamente provvedendo alla compattazione dell'archivio eliminando fisicamente dall'archivio i record marcati in precedenza (**cancellazione fisica)**

RICERCA
La ricerca dei record registrati appare una delle operazioni più importanti e più frequenti da eseguire su di un archivio. E' evidente che essa debba essere effettuata utilizzando una tecnica che consenta di ottenere **il miglior risultato nel minor tempo possibile**.
La scelta di tale metodo varierà in base al tipo di supporto di memoria utilizzato ed alla presenza o meno di un ordinamento.

Ricerca applicata ad archivi sequenziali memorizzati su supporti ad accesso sequenziale
In passato gli archivi erano memorizzati su nastri magnetici e su di essi l'unico metodo di ricerca applicabile era quello della **ricerca completa** dell'archivio fino al reperimento del record avente la chiave desiderata.
Il numero medio di accessi di tale metodo è il seguente:

- **(N+1) / 2** in caso di successo
- **N** in caso di insuccesso

dove **N** è il numero di record presenti nell'archivio.

Quando vennero introdotti archivi in cui solo su alcuni record si interveniva con frequenza, le cose migliorarono di molto collocando tali record nelle prime posizioni dell'archivio.
Ciò aumentò l'efficienza solo in caso di successo, lasciando tutto com'era in caso di insuccesso.
Il numero medio di accessi necessari nel caso in cui i record con chiavi di frequente movimento venivano posizionati nelle prime posizioni:

- **In funzione della probabilità della chiave** in caso di successo
- **N** in caso di insuccesso

dove **N** è il numero di record presenti nell'archivio.

Con gli archivi ordinati fu possibile ricorrere ad un altro metodo di ricerca, la **ricerca sequenziale**, ottenuta mediante l'ottimizzazione dell'operazione di scansione *(la ricerca infatti si interrompe quando si trova la chiave cercata oppure quando ci si posiziona su un record con chiave maggiore di quella cercata).*
In questo caso il numero medio di accessi diveniva:

- **(N+1) / 2** in caso di successo
- **(N+1) / 2** in caso di insuccesso

dove **N** è il numero di record presenti nell'archivio.

Ricerca applicata ad archivi sequenziali memorizzati su supporti ad accesso diretto
In questo caso possono verificarsi le seguenti situazioni:

a) si conosce l'indirizzo del record da ricercare: allora il problema non sussiste perché mediante un accesso ci si posiziona direttamente sul record cercato;
b) NON si conosce l'indirizzo del record da ricercare e l'archivio è DISORDINATO: allora il problema viene affrontato come nel caso dell'accesso sequenziale;
c) NON si conosce l'indirizzo del record da ricercare e l'archivio è ORDINATO: allora il problema viene risolto con tecniche di ricerca molto efficienti quali ad esempio la ricerca binaria.
   In questo caso il numero medio (ma anche riferito al caso pessimo) di accessi diveniva:
   - **log (N)** in caso di successo
   - **log(N)** in caso di insuccesso

   dove **N** è il numero di record presenti nell'archivio.

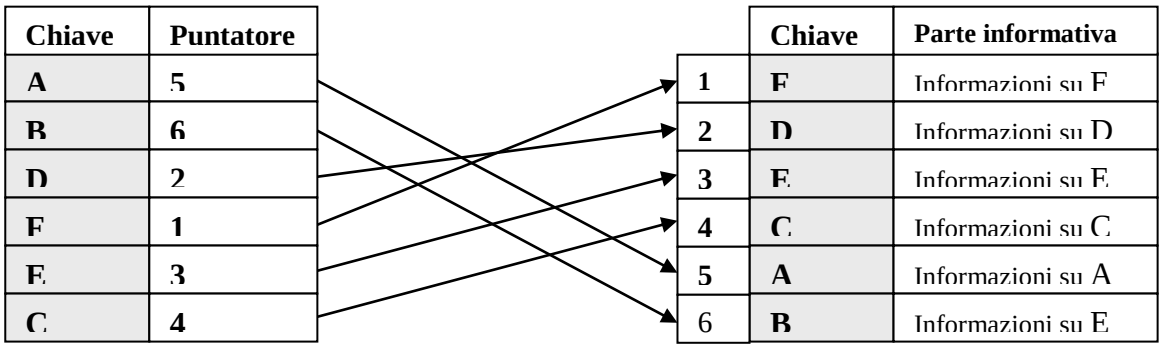
L'ORGANIZZAZIONE SEQUENZIALE CON INDICE

Struttura

Gli archivi sequenziali registrati su supporti fisici che permettono l'accesso diretto (come i dischi), possono essere gestiti anche con una **variante** dell'organizzazione sequenziale, caratterizzata dalla possibilità di *velocizzare la ricerca di un record* utilizzando uno o più **indici**.

Nell'**organizzazione sequenziale ad indici** (o *indexed sequential*) si possono distinguere due elementi fondamentali:
- un **archivio primario** caratterizzato da record consecutivi che possono anche essere ordinati;
- un **archivio secondario** o **indice ordinato** (o *dizionario*) o una gerarchia di indici i cui elementi sono composti generalmente da due campi:
  - un **campo chiave** contenente la chiave del record;
  - un **campo puntatore** contenente la posizione del record all'interno dell'archivio primario.

L'indice consente quindi di stabilire *una corrispondenza tra ogni chiave e la posizione* del relativo record memorizzato nell'archivio.
L'indice ovviamente deve contenere **tutte** le chiavi presenti nell'archivio primario.



Archivio secondario o Indice ordinato (o dizionario)

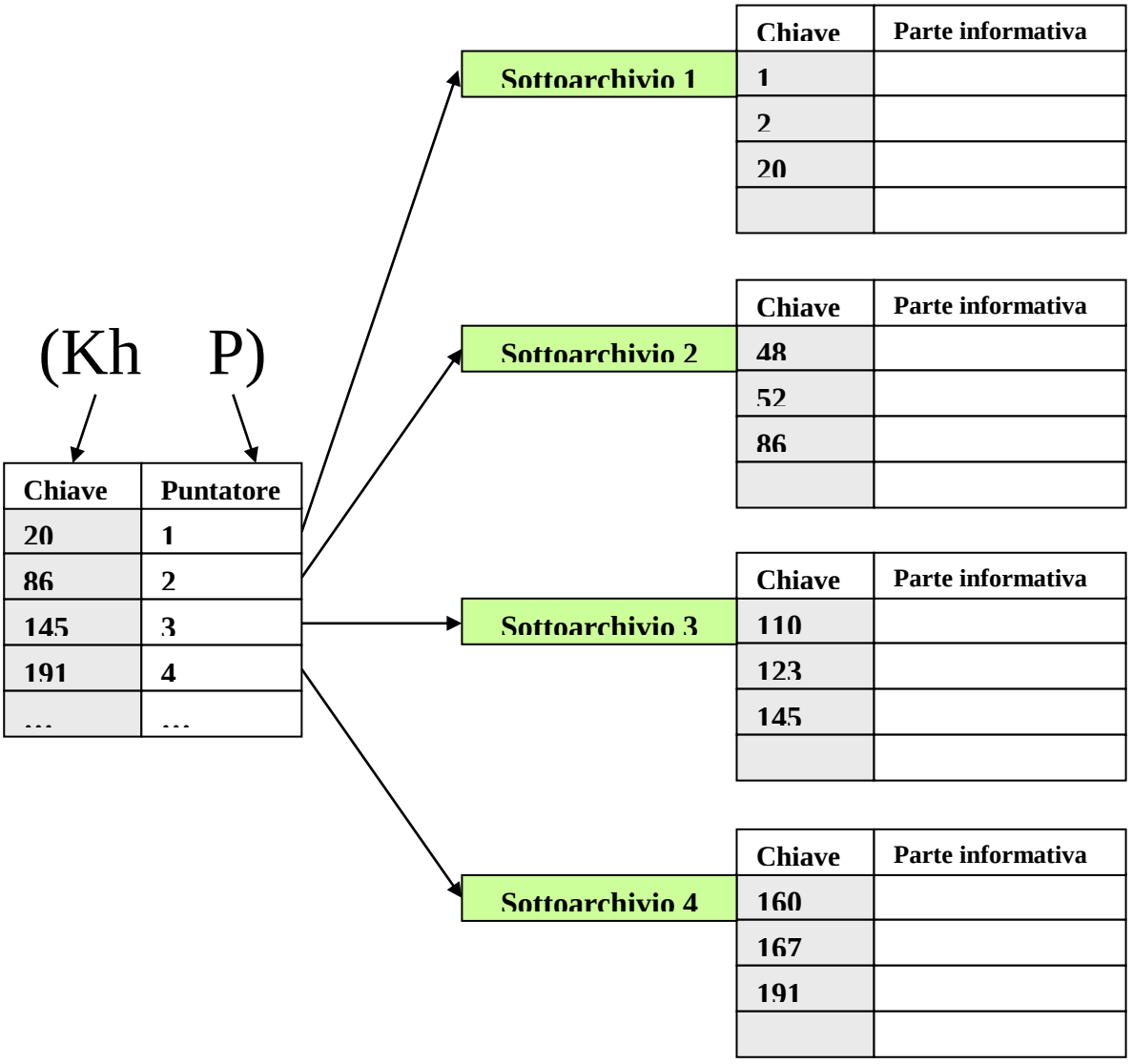
Archivio primario

Per poter potenziare le operazioni di ricerca dei record bisogna innanzitutto distinguere tra:
- **strutture sequenziali con indice ORDINATE**;
- **strutture sequenziali con indice NON ORDINATE**.

Nelle **strutture sequenziali con indice ORDINATE**, l'archivio primario viene implementato in una **struttura a pagine** (*ossia **sottoarchivi** costituiti da blocchi contigui di ugual numero di record*) e l'indirizzo della chiave più alta di ciascun sottoarchivio è contenuto nell'indice i cui record sono del tipo

(Kh, P)

dove ***Kh*** indica la chiave più alta e ***P*** il numero di sottoarchivio.
In questo modo l'indice viene utilizzato esclusivamente per la ricerca all'interno dei vari sottoarchivi.



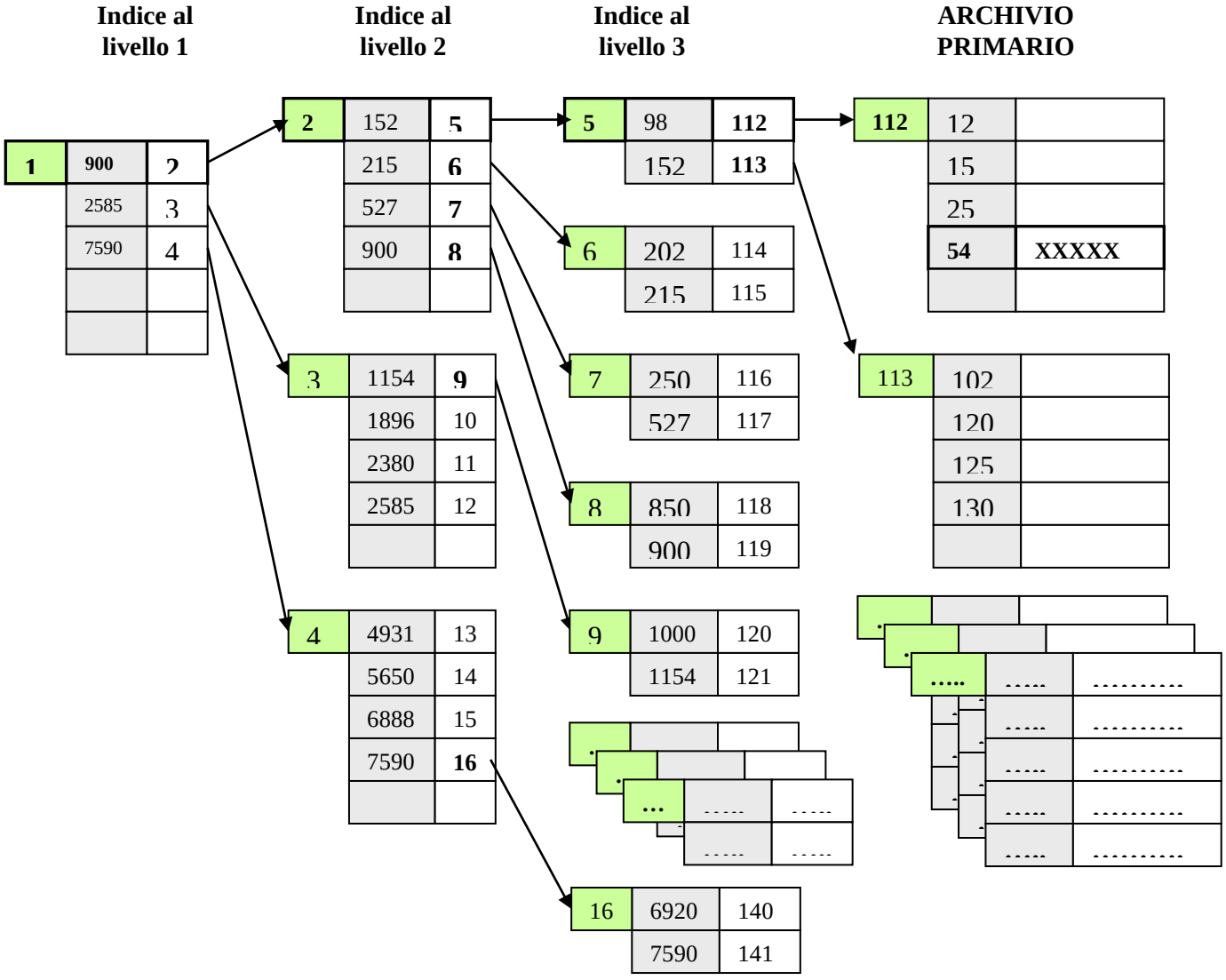
In pratica il valore della chiave nell'indice indica esplicitamente la chiave di valore più alto contenuta all'interno del sottoarchivio mentre il puntatore indica implicitamente la posizione della chiave più bassa poiché rappresenta l'indirizzo del sottoarchivio ossia del primo record presente ossia quello con chiave più bassa.

Generalmente la ricerca di una chiave **K** avviene rispettando le seguenti procedure:

Ricerca nell'indice la prima *Kh* ≥ K
Accedi al sottoarchivio P associato alla chiave *Kh*
Ricerca la chiave K nel sottoarchivio P

La ricerca all'interno dell'archivio indice (che ricordiamo è ordinato) può avvenire con un qualunque metodo di ricerca (sequenziale, binaria, etc.)

Nel caso in cui il numero si sottoarchivi diventi rilevante e di conseguenza il numero di record presenti nell'indice cominci a divenire considerevole (appesantendo così la ricerca), è possibile organizzare a sua volta l'indice come un archivio sequenziale con indice.
Si creano in questo modo **sottoindici** di diverso livello che permettono una diminuzione nel tempo di scansione dell'indice stesso.
In presenza di **indici a più livelli (o indici multipli)**, in fase di ricerca, la gerarchia viene utilizzata per poter individuare, partendo da un indice a livello k, quale indice a livello k+1 debba essere esaminato al fine di selezionare il sottoarchivio all'interno del quale si trova il record cercato



Supponiamo di voler ricercare il record con chiave K = **54.**
Procediamo nel seguente modo:
1) Si accede all'indice di livello 1 e si cerca *sequenzialmente* la prima chiave maggiore o uguale a 54. La chiave è **900;**
2) Si accede all'indice 2 di livello 2 tramite il puntatore associato e si cerca la prima chiave maggiore o uguale a 54. La chiave è **152;**
3) Si accede all'indice 5 di livello 3 tramite il puntatore associato e si cerca la prima chiave maggiore o uguale a 54. La chiave è **98;**
4) Si accede al sottoarchivio **112** che fa parte dell'archivio primario tramite il puntatore associato e si cerca la chiave 54 utilizzando una *ricerca sequenziale*.

IL TRATTAMENTO DEGLI ARCHIVI NON SEQUENZIALI

L'**organizzazione non sequenziale** è una tecnica di organizzazione che consente di accedere ad un record senza dover attraversare tutti quelli che lo precedono, in quanto è possibile ricavare direttamente *l'indirizzo* di una registrazione tramite la *chiave* del record stesso.

Da ciò si deduce che:
- l'**organizzazione non sequenziale** non necessita che i record siano memorizzati in maniera contigua all'interno del dispositivo di memoria ausiliaria: pertanto ogni record **NON E'** caratterizzato dalla presenza di un predecessore e di un successore;
- è necessario che un campo del record abbia il ruolo di **chiave primaria** in modo che sia possibile individuare univocamente il record al quale accedere, creando così ***una relazione tra il valore della chiave e l'indirizzo del record.***

Su archivi non sequenziali sono consentite operazioni di *lettura, scrittura, riscrittura, cancellazione* e *posizionamento.*
L'organizzazione non sequenziale viene implementata nei seguenti modi che si differenziano per il ruolo che la chiave primaria assume nell'archivio e per i passi da seguire per accedere ad un record.

a) organizzazione **relative**;
b) organizzazione **hash**;
c) organizzazione a **B-alberi**.

A) L'ORGANIZZAZIONE RELATIVE

Nell'**organizzazione RELATIVE** la chiave primaria di ogni record identifica univocamente sia il record stesso sia l'**indirizzo logico** in cui il record è registrato..

Esempio:
*Un foglio di calendario è strutturato in modo che ad ogni giorno **k** corrisponda la **k-esima** riga dello stesso foglio. Quindi il giorno 5 si trova alla quinta riga, il giorno 28 in 2ventottesima riga.*

Per determinare l'indirizzo del k-esimo record il sistema operativo moltiplica la dimensione del record per (k-1) e successivamente somma tale valore a quello dell'indirizzo fisico del primo record registrato nell'archivio.

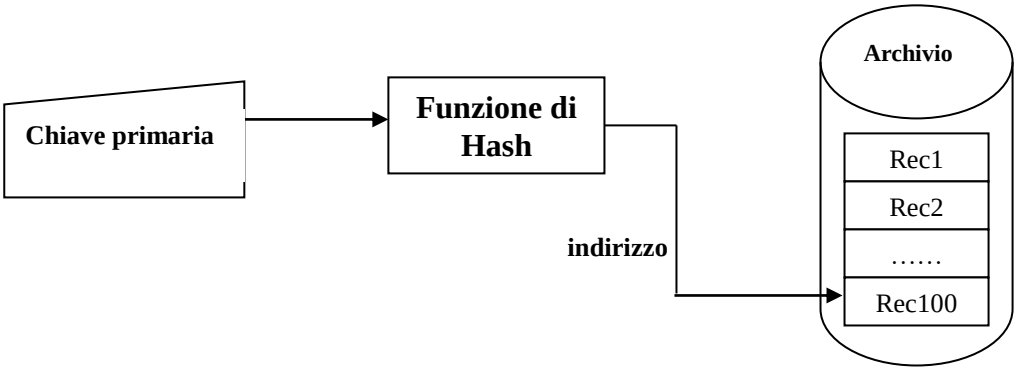
B) L'ORGANIZZAZIONE HASH

La tecnica **HASH** si basa sull'utilizzo di funzioni che, partendo dalla chiave primaria di ciascun record, trasformano quest'ultima in un **numero intero** che rappresenta l'**indirizzo logico** detto **indirizzo hash** del record stesso.

In generale applicare la tecnica Hash significa definire una funzione di randomizzazione H (funzione di Hash) che associ ad ogni record l'indirizzo di un record logico in cui è possibile memorizzarlo, attraverso la trasformazione della chiave **K** in un intero **x compreso tra 1 ed N** dove **N** è il numero di indirizzi logici a disposizione.

In altre parole **H(K) = x**

L'indirizzo x deve essere calcolato tutte le colte che occorre accedere al record per effettuare operazioni di I/O.



La funzione Hash

La scelta della *funzione Hash* è importantissima perché una funzione di trasformazione non idonea può rendere inefficiente o completamente invalida tutta l'organizzazione stessa.

Una funzione Hash **ottimale** deve:

1) **essere facilmente calcolabile** ossia essere composta da calcoli facili in modo da non appesantire il tempo di accesso;
2) produrre sempre lo stesso indirizzo a partire dalla stessa chiave (**deterministica**);
3) **generare indirizzi uniformemente distribuiti** nell'ambito dell'archivio;
4) **generare indirizzi casualmente distribuiti** nell'ambito dell'archivio, relativamente alle chiavi simili (Es. di chiavi simili X1, X2, X3 oppure DARE, MARE; CARE);
5) cercare di **coprire l'intero intervallo degli indirizzi** evitando, se possibile, che vi siano indirizzi mai generati;
6) **generare indirizzi diversi se le chiavi sono diverse**;

   Quindi secondo questo punto la funzione Hash ideale è quella che *ad ogni valore della chiave associ sempre un indirizzo diverso* in modo che ogni record possa essere raggiunto tramite un solo accesso (**trasformazione perfetta**)

   Questo è ben lontano dalla realtà in quanto è molto difficile generare funzioni Hash che consentano una corrispondenza **uno-a-uno** tra chiave ed indirizzo.

   Infatti questo tipo di associazioni:

   - richiedono che il numero di record dell'archivio fisico sia uguale al numero dei possibili valori che la chiave può assumere;
   - comportano un notevole spreco di memoria e quindi una inefficienza generale del sistema perché in generale il numero di chiavi effettive da gestire è sensibilmente minore rispetto al numero di chiavi ipotetiche.

Per questo caso si opta per funzioni che garantiscono una associazione **molti-a-uno** ossia può accadere che due o più chiavi in fase di trasformazione possano generare lo stesso indirizzo. Si generano delle **collisioni** che devono essere risolte predisponendo opportune procedure che collochino le registrazioni in conflitto dette **sinonimi** in record aventi indirizzo diverso.
Quindi possiamo formulare un ulteriore requisito che afferma che una funzione Hash ottimale deve:

7) **generare il minor numero possibile di collisioni** e definisca apposite procedure per l'allocazione dei sinonimi (**gestione delle collisioni**).

Avendo capito che è quasi impossibile realizzare funzioni Hash che realizzino la trasformazione perfetta e che quindi esisterà sempre un problema di gestione delle collisioni, occorre concentrarsi sullo sviluppo di **funzioni Hash efficienti e non perfette.**

Metodi di calcolo degli indirizzi
Esistono molti metodi per calcolare gli indirizzi. Esaminiamone alcuni:

a) Metodo del troncamento
Viene utilizzato quando la chiave numerica è composta da un elevato numero di cifre ed il numero di record che possono essere memorizzati è abbastanza limitato.
Esso consiste nell'estrarre dalla chiave un sottoinsieme di cifre che rappresentano l'indirizzo.

Esempio:
*Supponiamo di avere un archivio contenente i tesserati di una palestra e sul quale possono essere inseriti **1000** record.*
*La **chiave primaria** è il numero di tessera che è un **numero di 7 cifre**. Costruiamo la nostra funzione Hash in questo modo: detto **x** il numero di tessera (chiave primaria originale) costituito da 7 cifre, la sua trasformazione prevede l'associazione di un numero **da 000 a 999** (indirizzo logico del record nell'archivio) costituito dalle ultime **3** cifre di detta chiave aumentato però di una unità*
H (1543423) + 1 = 424
H (2325675) + 1 = 676
H (2320001) + 1 = 002
H (6885675) + 1 = 676

Vantaggi: è un metodo molto efficiente in cui le chiavi numeriche siano principalmente consecutive e con poche interruzioni

Svantaggi: è un metodo che produce sinonimi ogni volta che si incontrano chiavi aventi le ultime 3 cifre uguali. Inoltre richiede spesso che l'archivio debba essere composto da un numero di record superiore a quanto necessario.

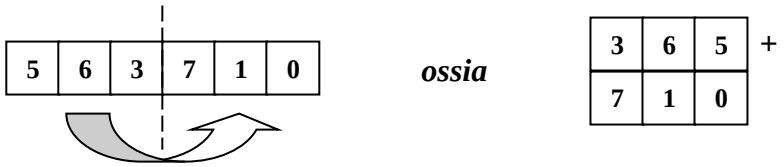
Applichiamo questo metodo ad una ***chiave alfanumerica***.
Esempio:
Supponiamo che la chiave da trasformare debba essere composta da 4 caratteri estratti dagli ultimi 4 della chiave primaria
Per procedere alla trasformazione si può procedere come segue: poniamo in corrispondenza le 26 lettere dell'alfabeto con i primi 26 numeri naturali seguendo l'ordine alfabetico (A=0, B=1; … Z=25). A questo punto come nei sistemi di numerazione pesati si converte la sottostringa formata dagli ultimi 4 caratteri applicando una sorta di conversione in base 26.
*H (HELLOCIAO) = $(CIAO)_{26}$ = C x 26^3 + I x 26^2 + A x 26^1 + O x 26^0 = **40574***

A questo punto per ottenere il reale indirizzo si può applicare il metodo del troncamento avendo però stabilito a priori il numero massimo di record dell'archivio.

Svantaggi: è un metodo che produce sinonimi ogni volta che si incontrano chiavi che terminano con le stesse 4 lettere. Inoltre anche in questo caso si richiede che l'archivio sia composto da un numero di record superiore a quanto necessario.

b) Metodo di folding (dall'inglese "piegare")
Questo metodo consiste nel dividere la chiave numerica in due o più parti.
Successivamente tali parti vengono sommate tra loro ed il valore ottenuto (o una sua parte) viene utilizzato come indirizzo.

Esempio:
Supponiamo di volere trasformare la chiave 563710 secondo questo metodo.



H (563710) = 365 + 710 = ***1075***

Anche questo metodo genera collisioni (H(349132) e presenta gli stessi svantaggi del metodo di troncamento.

c) Metodo della divisione modulo n
Questo metodo consiste nell'assegnare come indirizzo del record il resto della divisione della chiave numerica per il numero massimo di record memorizzabili nell'archivio.

In generale detta **k** la chiave e detto **N** il numero massimo di record contenuti nell'archivio, l'indirizzo del record sarà dato dalla trasformazione:

$$H(k) = (k \bmod N) + 1 \ ^{(*)}$$

(*) Il " più 1" è giustificato dal fatto che come sappiamo l'operatore *modulo* fornisce sempre valori compresi tra 0 ed N-1 e quindi anche gli indirizzi sarebbero compresi in tale intervallo. Con l'aggiunta dell'unità tale intervallo si trasforma nell'intervallo [1, N].

Anche questo metodo genera sinonimi e per ridurre la loro presenza è fondamentale la scelta dell'N. Tale metodo ha però il vantaggio di poter essere applicato anche in presenza di un numero massimo di record che non sia multiplo di 10.
Inoltre gli indirizzi generati, a causa del significato dell'operatore *modulo*, non necessitano di troncamento.

La gestione delle collisioni

Come abbiamo visto, anche facendo la massima attenzione nella scelta della funzione Hash, **è praticamente impossibile evitare le collisioni.**
Pertanto il **prevedere opportune tecniche** per la gestione delle collisioni è parte integrante dell'intera tecnica Hash.

DEF: Le tecniche di gestione delle collisioni sono metodi che consentono di individuare un indirizzo libero per la chiave **k** nel caso in cui la funzione Hash **H (k)** restituisca un valore già ottenuto per un'altra chiave.

N.B. La gestione corretta di questa specifica situazione è di grande importanza sia in fase di **inserimento dei record** sia in fase di **ricerca di una chiave.**
(*) In caso di **inserimento** si applicherà ad ogni chiave uno dei metodi per il calcolo degli indirizzi sopra specificati (o altri ancora). Ottenuto il valore si eseguirà un test per controllare se la posizione indicata è occupata (collisione) o libera. Se è libera il record può essere inserito ma se è occupata dovrà essere ricercata una nuova posizione, fino a che non se ne incontrerà una libera;
(*) In caso di **ricerca** si applicherà alla chiave da ricercare uno dei metodi per il calcolo degli indirizzi sopra specificati (o altri ancora) e si accederà al record. Se la chiave in esso contenuta coincide con quella cercata, la ricerca termina altrimenti occorrerà scandire l'archivio andando su altre posizioni sino a quando non si troverà la chiave cercata.

Esistono vari metodi per la gestione delle collisioni. Esaminiamone alcuni:

a) Scansione lineare o indirizzamento aperto
Tale tecnica consiste nel ricercare il primo posto libero in cui sistemare la chiave che ha generato la collisione, spostandosi all'interno dell'archivio di un certo numero *p* di posizioni, detto passo di scansione, fino a quando non si incontra una posizione libera.

Se in particolare **p = 1** si parla di **scansione lineare con passo unitario.**
La scelta del *passo unitario* assicura il controllo su tutti gli indirizzi fino al completamento delle posizioni disponibili nell'archivio.

Questo metodo ha l'inconveniente di creare lunghe catene di record consecutivi, rallentando il tempo di accesso ai record.
Inoltre provoca il fenomeno detto **agglomerazione o addensamento primario delle chiavi (clustering)** ossia la presenza di aree di memoria - *agglomerati* o *cluster* – nelle quali confluiscono varie sequenze di sinonimi che presentano indirizzi coincidenti da un certo punto in poi.

Esempio
Supponiamo di dover memorizzare le seguenti chiavi in un archivio composto al massimo da 9 record:
9 28 29 35 24 48 71 64
*Per generare gli indirizzi ci serviamo del **metodo della divisione modulo 9**. Abbiamo*
k= 9 ⇨ H (k) = (9 MOD 9) +1= 0 + 1 = 1
k= 28 ⇨ H (k) = (28 MOD 9) +1= 1 + 1 = 2
k= 29 ⇨ H (k) = (29 MOD 9) +1= 2 + 1 = 3
k= 35 ⇨ H (k) = (35 MOD 9) +1= 8 + 1 = 9
k= 24 ⇨ H (k) = (24 MOD 9) +1= 6 + 1 = 7
k= 48 ⇨ H (k) = (48 MOD 9) +1= 3+ 1 = 4
k= 71 ⇨ H (k) = (71 MOD 9) +1= 8 + 1 = 9 ***prima collisione***

*A questo punto in base alla tecnica della **scansione lineare con passo unitario** dobbiamo spostarci sul record successivo per vedere se la posizione è libera: dobbiamo andare sulla posizione 10. Ovviamente ciò non è possibile avendo supposto che l'archivio sia composta da 9 record.*
Occorre pertanto applicare il metodo della divisione modulo 9 anche per il risultato prodotto.
*k= 71 H (k) = (71 MOD 9) +1= 8 + 1 = **9** **collisione con la chiave 35. Si continua usando 9***
*(9 MOD 9) = 0 + 1 = **1** collisione con la chiave 9. Si continua usando 1*
*(1 MOD 9) = 1 + 1 = **2** collisione con la chiave 28. Si continua usando 2*
*(2 MOD 9) = 2 + 1 = **3** collisione con la chiave 29. Si continua usando 3*
*(3 MOD 9) = 3 + 1 = **4** collisione con la chiave 48. Si continua usando 4*
*(4 MOD 9) = 4 + 1 = **5** Non c'è più collisione . **La chiave 35 viene memorizzata in posizione 5.***
Ora analizziamo l'ultima chiave:
*k= 64 ⟹ H (k) = (64 MOD 9) +1= 1 + 1 = 2 **collisione con la chiave 28. Si continua usando 2***
*(2 MOD 9) = 2 + 1 = **3** collisione con la chiave 29. Si continua usando 3*
*(3 MOD 9) = 3 + 1 = **4** collisione con la chiave 48. Si continua usando 4*
*(4 MOD 9) = 4 + 1 = **5** collisione con la chiave 71. Si continua usando 5*
*(5 MOD 9) = 5 + 1 = **6** Non c'è più collisione . **La chiave 64 viene memorizzata in posizione 6.***
Come si può notare la chiace 64 ha dovuto attraversare una parte della catena precedente pur sapendo che i record individuati da questi indirizzi erano occupati.

b) Scansione non lineare o pseudo-random
Tale tecnica è una variante dell'indirizzamento aperto.
Questo metodo ricorre ad una tecnica di memorizzazione che non prevede una scansione sequenziale dell'archivio poiché al posto di utilizzare il passo di scansione genera un numero casuale (o meglio pseudo-casuale) ogni qualvolta si verifica una collisione.
Tale metodo si definisce **scansione non lineare o pseudo-random.**

In altri termini si calcola l'indirizzo per iniziare la scansione: se su tale indirizzo si verifica una **collisione**, non ci si sposta sulla posizione immediatamente successiva, bensì si ricalcala un nuovo indirizzo, detto **indirizzo di rehash** servendosi di una nuova funzione detta **funzione di rehash.**
L'**indirizzo di rehash** può essere calcolato con una qualsiasi funzione che generi numeri pseudo-casuali e che, per ogni passo, definisca un incremento differente per l'indirizzo.

N.B. Ribadiamo che l'indirizzo di rehash deve essere necessariamente **pseudo-casuale** in quanto in fase di ricerca di una chiave devono essere ricalcolati gli stessi indirizzi prodotti per la memorizzazione, ossia deve essere ripetuta la stessa sequenza di indirizzi.

C) L'ORGANIZZAZIONE A B-ALBERI

Abbiamo visto in precedenza come l'utilizzo degli indici possa essere di valido aiuto per implementare archivi dinamici.
La gestione di tali indici viene resa molto efficiente se si utilizzano gli **alberi binari per le chiavi** e può essere ancora ottimizzata se si mantiene l'albero **bilanciato.** (N.B. L'operazione di ricerca diventa più rapida).

I **B-alberi** (in inglese *Balanced trees*) sono strutture dati nate nei primi anni 70 proprio per venire incontro all'esigenza di creare modelli implementativi efficienti per la gestione dinamica ed ordinata di archivi di dati.
Sono **alberi binari ordinati secondo la chiave** e perfettamente **bilanciati in altezza** (ossia con tutte le foglie allo stesso livello).

I vantaggi nell'uso dei B.alberi possono essere così riassunte:

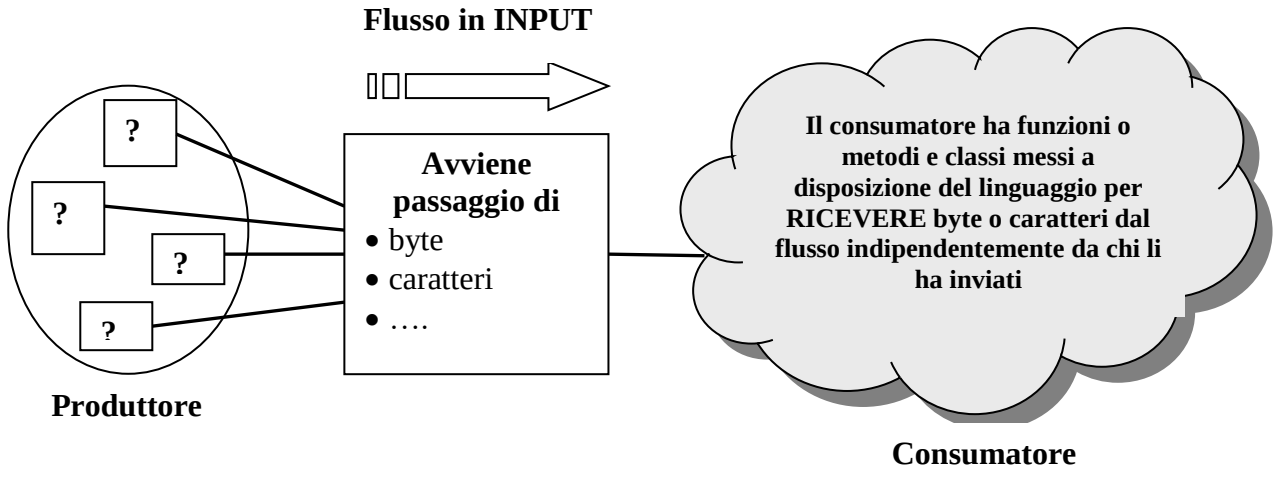
1) garantiscono ***ottime prestazioni*** sia per la ricerca sia per l'aggiornamento poiché entrambe le operazioni possono avvenire con algoritmi di complessità logaritmica e con procedure abbastanza semplici;
2) è possibile effettuare su di essi ***elaborazioni di tipo sequenziale dell'archivio primario senza doverlo riorganizzare***.

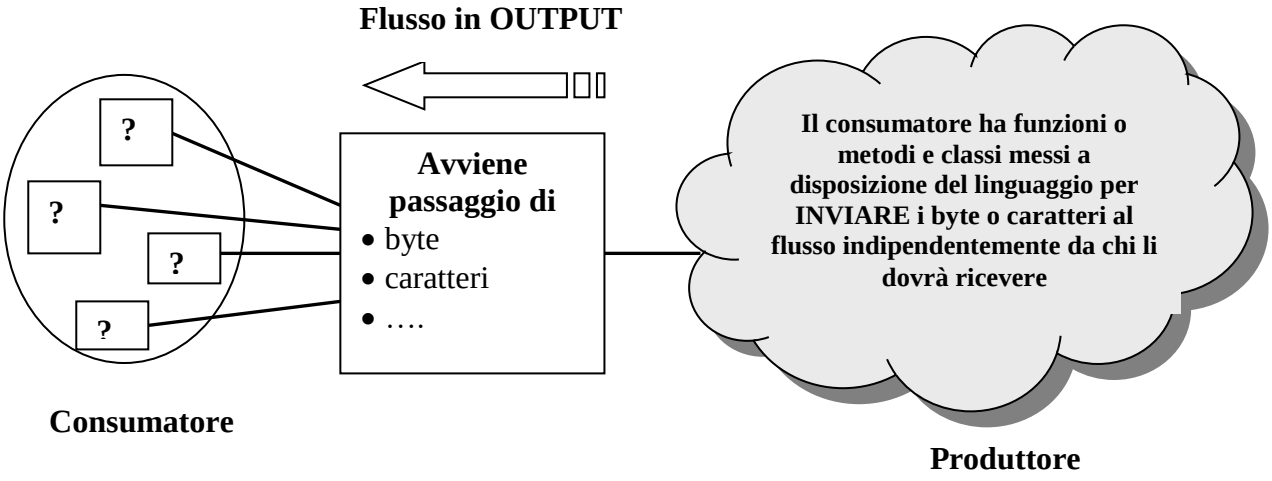
IL FLUSSO

Molti linguaggi di programmazione ad oggetti e imperativi (tra questi anche il C) si servono per gestire le funzioni relative alla gestione dei file di un concetto astratto: quello di **flusso** o **stream.**
Il flusso può essere immaginato come un *canale* tra la sorgente di una certa informazione e la sua destinazione.

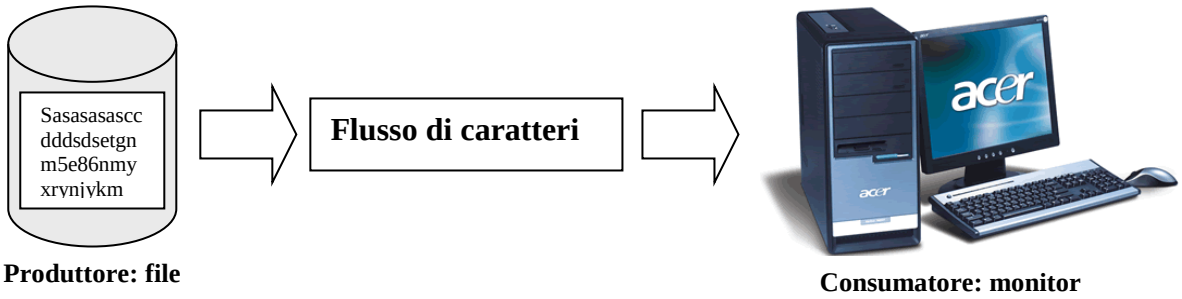
I flussi sono classificati in:
- **flusso di input**: se il punto di vista è quello del *consumatore* dell'informazione (ossia la *destinazione*) ossia di chi riceve l'informazione attraverso il flusso;
- **flusso di output:** se il punto di vista è quello del *produttore* dell'informazione (ossia la *sorgente*) ossia di chi deve inviare le informazioni in un flusso;

L'astrazione permette di non porre attenzione sulla sorgente delle informazioni quando si legge un flusso di input né tanto meno porre attenzione sulla destinazione quando si scrive un flusso di output.





Esempi di flusso:
a) ***flusso di caratteri***
produttore*: è un file di caratteri*
consumatore*: è il monitor*



Produttore: file **Consumatore: monitor**

b) ***flusso di interi***
produttore*: è la tastiera*
consumatore*: è un file di interi*

Produttore:tastiera **Consumatore: file**

In un flusso, come visto, possiamo considerare una qualsiasi sorgente ed una qualsiasi destinazione dei dati. Noi siamo interessati a quei flussi in cui la sorgente o la destinazione dei dati (oppure entrambe) sono **file**.
Ci occuperemo in particolare di:
- **file binari**;
- **file di testo**
- file di tipi primitivi;
- file di oggetti.

L'accesso a questi tipi di file è unicamente di tipo **sequenziale** ossia per accedere ad una registrazione memorizzata in uno di questi file è necessario accedere a tutte le registrazioni che la precedono.