I FILE

I file servono per memorizzare i dati in modo ***permanente*** nelle memorie di massa. Siccome la gestione delle memorie di massa è effettuata dal S.O. (precisamente da un modulo del S.O. il *file system*), i programmi che devono gestire dei file devono colloquiare con questo modulo.

Le operazioni principali che vengono richieste al File System sono di:

- apertura del file (*OPEN*), insieme di operazioni preliminari per l'individuazione del file sulla memoria di massa e l'indicazione che è in uso,
- la scrittura del file dalla memoria di massa alla memoria centrale (*READ*),
- la scrittura del file dalla memoria centrale alla memoria di massa (*WRITE*),
- la chiusura del file (*CLOSE*) scarico del buffer di I/O e operazioni conclusive come l'indicazione che il file è stato rilasciato dal programma.

RECORD LOGICI.

Cominciamo col dire che i file, oggetto del nostro studio, sono un "contenitore" di ***record***, ove per record indichiamo un insieme di dati in relazione tra loro (ad esempio i dati anagrafici di una persona). Un record, quindi, è formato da un insieme di *campi* caratterizzati da una tipologia e lunghezza. Ogni campo conterrà uno specifico dato individuato da un nome (nome del campo). Ad esempio un record anagrafico sarà formato dai campi: *codice_fiscale* - campo di tipo stringa di lunghezza 16 byte, *cognome* - campo di tipo stringa di lunghezza 100 byte, *nome* - campo di tipo stringa di lunghezza 100 byte, *data_di_nascita* - campo di tipo date di lunghezza 10 byte, *indirizzo* - campo di tipo stringa di lunghezza 200 byte… e così via. Ogni record conterrà, per ogni posizione anagrafica i corrispondenti dati: nel campo *codice_fiscale* il codice fiscale di una persona, *cognome* il cognome di una persona…

I record si suddividono in record a lunghezza fissa, in cui ogni record del file ha la stessa lunghezza, e record a lunghezza variabile, in cui ogni record del file ha una lunghezza diversa e la fine del record è individuato da caratteri speciali (CRLF – Carriage Return Line Feed).

Rappresentazione grafica di un record a lunghezza fissa:

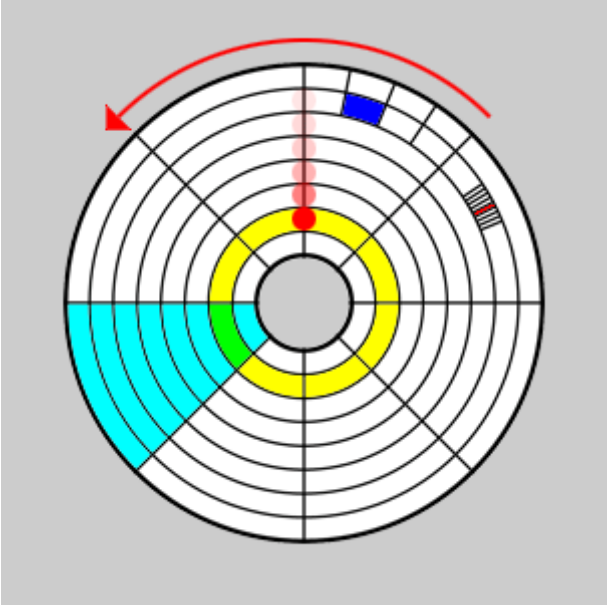
codiceFiscale	cognome	nome	eta	indirizzo	telefono

Tutti i record del file avranno tutti la stessa dimensione e struttura. Esempio:

File ***Anagrafe***:

codiceFiscale	cognome	nome	eta	indirizzo	telefono
rntovz23g88kt99r	Rossi	Antonio	22	Via dei tali	098212345
vrdlmt99rst89r	Verdi	Marco	26	p.zza Libertà	098845678
……	……	……	……	……	……

DISCO RIGIDO.



Il disco rigido è formato da un “piatto” di materiale ferromagnetico capace, cioè, di magnetizzarsi come i poli di una calamita conservando così l’informazione. I dischi rigidi detti HD o HDD (Hard Disk o Hard Disk Drive) vengono scritti e letti tramite una testina elettromagnetica che scorre in modo radiale sull’HD mentre il disco gira al di sotto della testina.

Affinché una testina elettromagnetica possa scrivere dati e successivamente ritrovare l'"indirizzo", cioè la posizione dei dati scritti e salvati, l'intera superficie di un disco a memoria magnetica viene strutturata in aree "virtuali":

la prima suddivisione fondamentale del disco è quella in corone circolari concentriche, ciascuna delle quali viene chiamata "traccia", e la testina elettromagnetica può spostarsi avanti e dietro, verso il centro o verso la periferia del disco, da una traccia all'altra;

la seconda suddivisione fondamentale del disco è quella in spicchi, ciascuno dei quali chiamato "settore", ogni settore è diviso dall’altro tramite una zona priva d’informazione chiamato “gap”, e quindi i settori passano velocemente sotto la testina elettromagnetica alternandosi al ruotare del disco;

ora che la testina può spostarsi secondo un sistema di coordinate, ciascun settore di ciascuna traccia diventa un'area di intersezione, la quale può essere ulteriormente suddivisa in blocchi di dati a due diversi livelli;

la prima suddivisione secondaria è il "grappolo" o "gruppo" di blocchi di dati, in inglese cluster, tutti contigui nella stessa traccia e settore;

la seconda suddivisione secondaria è il singolo "blocco" di dati all'interno del cluster, ciascuno dei quali contiene un "numero" di dati insieme a "segnali" di riferimento e disincronismo.

Il singolo blocco viene detto anche “record fisico” ed è l’insieme di bytes che vengono letti e scritti i un’unica operazione dal file system in read o write.

MEMORIE A STATO SOLIDO.

Le unità di memoria a stato solido (SSD – acronimo per Solid state drive) sono dispositivi di memoria di massa che si contraddistinguono perché in grado di memorizzare grandi quantità di dati in modo non volatile senza servirsi di parti meccaniche come accade con gli hard disk tradizionali.
Essi utilizzano infatti memoria flash, ovvero flash memory, cioè appunto un tipo di memoria a stato solido, non volatile dove le informazioni sono registrate su transistor che rappresentano vere e proprie *celle* di memoria con capacità di mantenere la carica elettrica per tempi lunghi.
Proprio questa basilare differenza di funzionamento rispetto ai dischi tradizionali fa sì che la tecnologia SSD sia molto veloce nello scrivere e leggere i dati, comporti quindi un minor dispendio energetico, permetta di costruire dispositivi che non riscaldano, non emettono rumore e hanno dimensioni più piccole rispetto alle unità storage cui si era abituati.

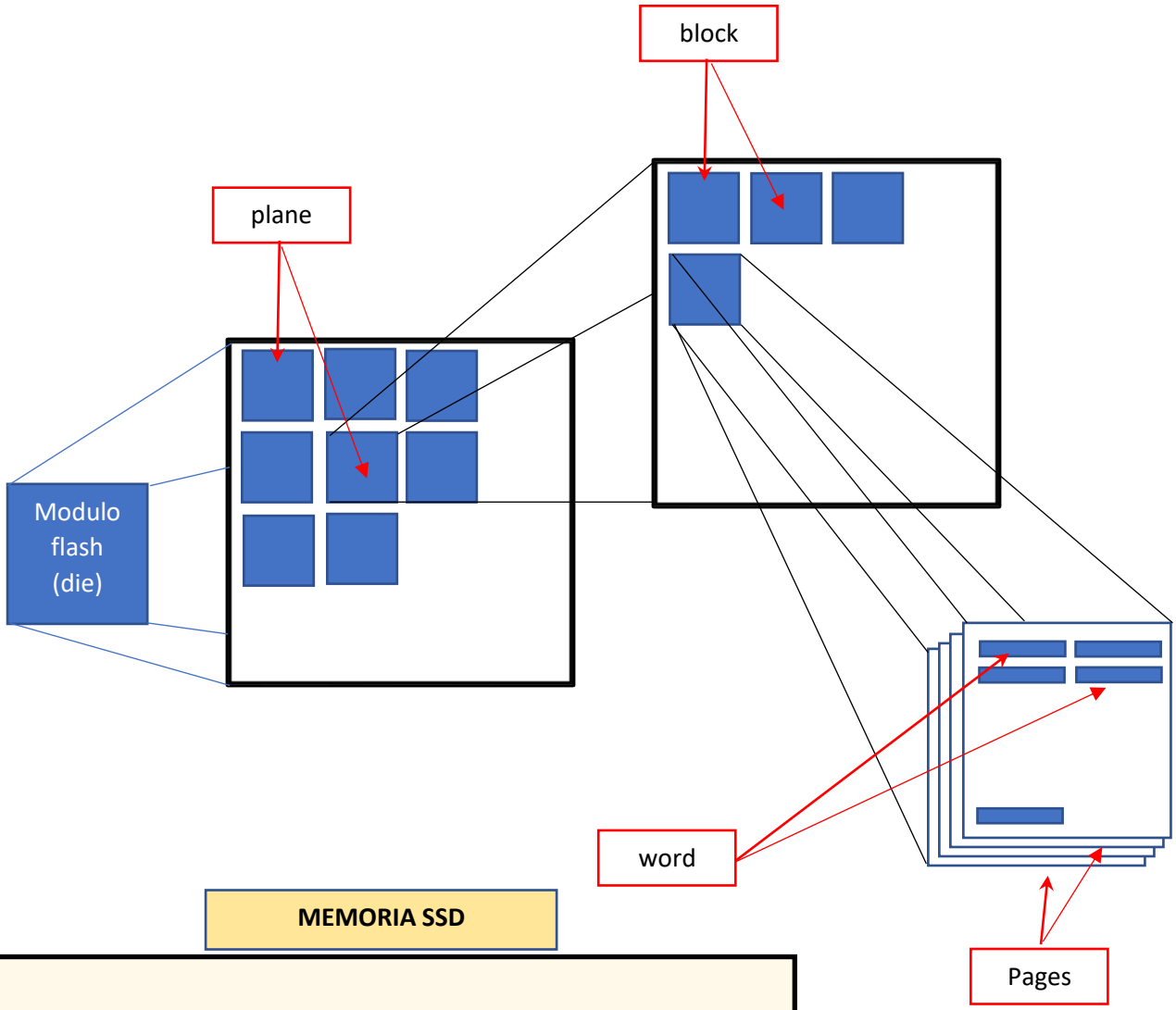
Le SSD possono essere considerate un vero e proprio sottosistema informatico composto principalmente dai componenti: ***Controller***, ***memoria cache*** e ***supercondensatore***.
Nello specifico, il ***Controller*** (microprocessore presente nel SSD), che ha un certo numero di core, coordina le operazioni di memoria di massa, controllare gli eventuali errori in lettura/scrittura, distribuisce uniformemente la scrittura su tutto il drive, gestisce la cache. La **memoria cache** è utilizzata dal processore per registrare temporaneamente le informazioni che gli servono per compiere le sue attività.
Grazie a un ***supercondensatore*** (simile a una batteria, ma caratterizzato dal fatto che può caricarsi e scaricarsi molto velocemente) le memorie a stato solido possono terminare la scrittura di dati che hanno avviata anche in mancanza di tensione.

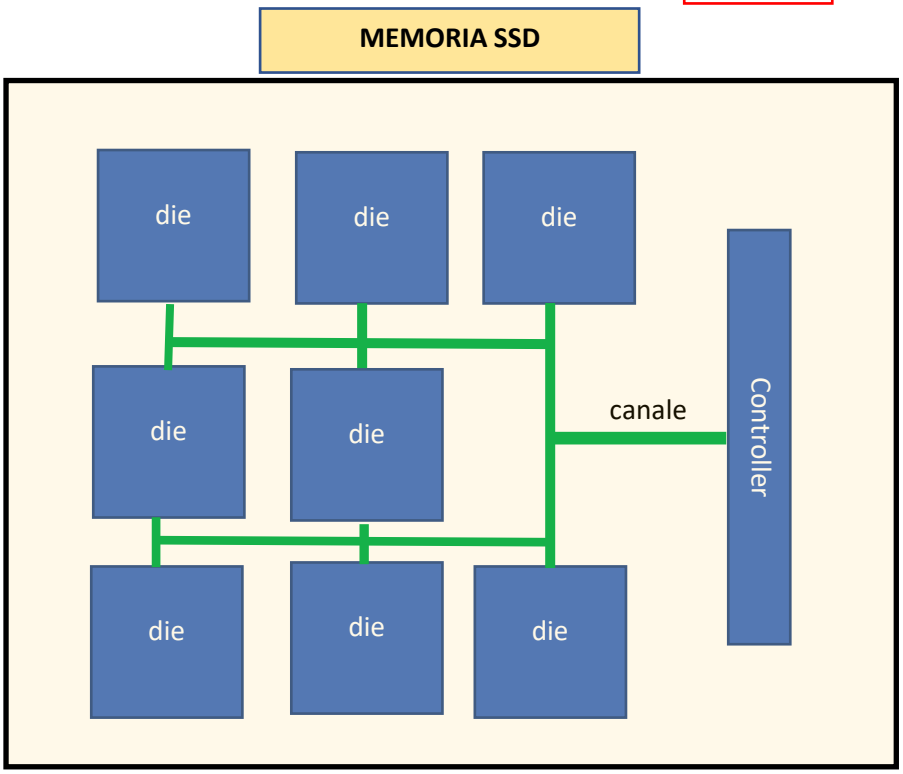
Inoltre gli SSD hanno un proprio "File System": il ***flash translation layer*** o ***FTL*** (di solito una componente software del Controller), che letteralmente significa livello di traduzione di tipo flash, è un software particolarmente complesso che ha lo scopo di tradurre un indirizzo logico usato dal sistema (File System) in un indirizzo fisico nella memoria flash.

Quando si cancella un file o un record di un file, questa cancellazione è logica. Mentre negli HD le aree di memorie (blocchi) che contengono dati cancellati vengono considerate vuote dal S.O. mentre in realtà i dati sono sempre presenti ma sovrascrivibili, ciò non è possibile negli SSD in cui le aree di memoria devono essere effettivamente vuote per essere scritte. I moderni S.O. sono stati dotati di una nuova funzione: ***TRIM*** che permette di liberare i blocchi in anticipo rispetto alle operazioni di scrittura. Questa "pulizia in anticipo", permette di evitare inutili attese nella fase di scrittura sulla memoria SSD nel momento in cui un dato deve essere salvato. Se infatti per poter scrivere un dato su un SSD si deve prima aspettare che un'area venga cancellata fisicamente e poi procedere alla scrittura, si perderebbe tempo. La funzione TRIM comunica con il controller che effettua la pulizia delle aree indicate dal S.O..

Le celle di memoria di un SSD sono raggruppate in "righe" (rows) dette ***words***, a loro volta raccolte nelle cosiddette ***pagine*** (***pages***). La pagina rappresenta il quantitativo minimo di informazioni che può essere letta o scritta sull'SSD in un determinato istante.
Le pagine sono raccolte in ***blocchi*** (***bloks***) tipicamente costituiti da 256 pagine. Infine, i blocchi sono generalmente raggruppati in 1.024 unità dette ***planes***, con diversi planes per singolo ***modulo flash*** (***die***) di memoria.

STRUTTURA DI UNA MEMORIA A STATO SOLIDO





RECORD LOGICI E FISICI - FATTORE DI BLOCCO.

I record logici, creati dal programmatore /progettista, vengono memorizzati sul disco, precisamente all'interno dei blocchi (sia degli HD che dei SSD), detti record fisici.

Un record fisico in generale può contenere un certo numero di record logici e in particolare il numero di record logici presenti in un record fisico è detto *fattore di bloccaggio o fattore di blocco* e si ha che:

- se un record fisico contiene più record logici, quei record si dicono *bloccati* e il fattore di blocco è maggiore di 1;
- se un record fisico contiene un solo record logico, quei record si dicono *sbloccati* e il fattore di blocco è 1;
- se per memorizzare un record logico sono necessari più record fisici, quei record si dicono *multiblocco* e il fattore di blocco è minore di 1.

ORGANIZZAZIONE dei FILE

I file si suddividono in due gruppi, in base alla tipologia di accesso ai loro record:

- SEQUENZIALE i record vengono letti in sequenza o fino al record cercato o fino EOF (End Of File – marcatore che indica la fine del file).

- DIRETTO

  Questi si dividono in file a:

  - Accesso ad indice
  - Accesso calcolato
  - Relative.

ACCESSO AD INDICE

Cominciamo a definire il concetto di chiave (key): una CHIAVE è uno o più campi di un record che individuano UNIVOCAMENTE un record all'interno di un FILE.

Questa organizzazione fa uso del *file degli indici*, una tabella formata da due campi, un campo contiene la ***key*** di ogni record presente nel file, un altro il *blocco* ove è contenuto il record. Questa tabella viene scaricata in Memoria Centrale. Se il file è bloccato, all'interno del blocco la ricerca viene effettuata in modo sequenziale. La tabella del file indice è una tabella ***ordinata*** sulla chiave. Se cerco un record, questa ricerca deve essere effettuata "per chiave (key)"; la chiave viene ricercata sul file indice (detto anche file delle chiavi) con la ricerca dicotomica (binaria) e se esiste ho anche l'informazione del blocco dove è presente il record.

I record vengono memorizzati nel file dati (i record organizzati nei blocchi).

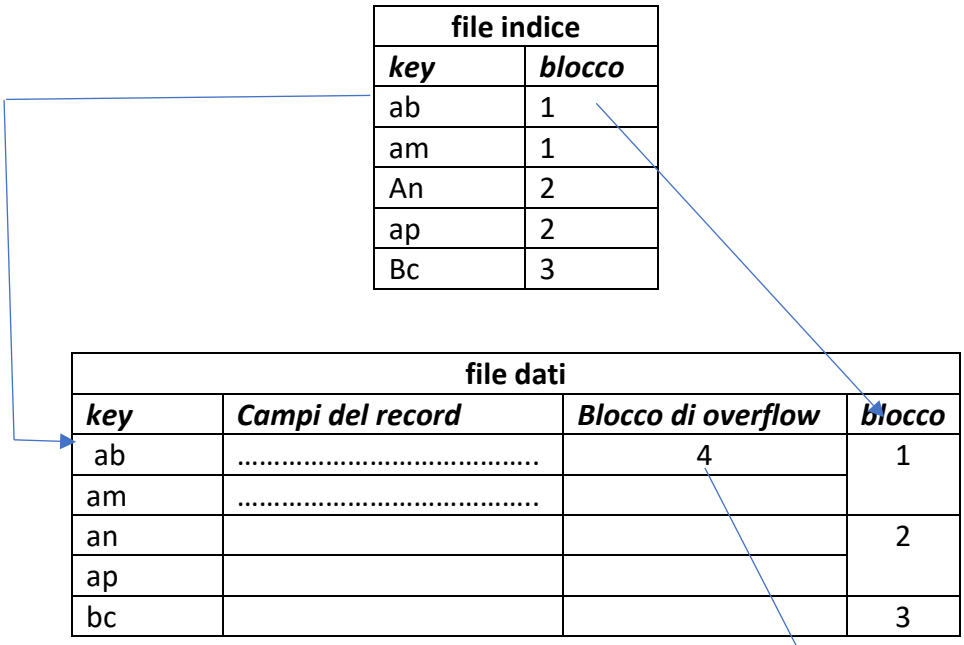
file indice	
key	***blocco***
ab	1
am	1
An	2
ap	2
Bc	3

File dati		
key	***Campi del record***	***blocco***
ab	……	1
am	……	
an		2
ap		
bc		3

Ovviamente il problema dell'ordinamento del file indice pone il problema del suo continuo ordinamento se si inserisce un nuovo record.

Può accadere che il file diventi grande e che quindi venga riempito il file degli indici. In questo caso ci sono due strategie che possono essere attuate per risolvere il problema. La prima strategia è inserire un ulteriore campo al file dei dati chiamato “aria di overflow”. Quindi si crea un’ulteriore file dei dati chiamato file di overflow.

Esempio:



file indice

key	***blocco***
ab	1
am	1
An	2
ap	2
Bc	3

file dati

key	***Campi del record***	***Blocco di overflow***	***blocco***
ab	……..	4	1
am	……..		
an			2
ap			
bc			3

(I record del file dati sono suddivisi nei blocchi)

file di overflow

key	***Campi del record***	***Blocco di overflow***	***blocco***
aa	……..		4

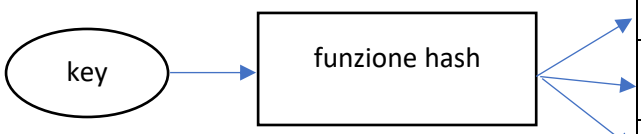
(anche I record del file di overflow sono suddivisi nei blocchi)

Supponiamo di voler inserire il record con chiave ‘aa’. Siccome il file degli indici è pieno questo record si inserisce nel file di overflow nel blocco 4. Siccome la chiave ‘aa’ è più piccola della chiave ‘ab’, nel blocco di overflow del record di chiave ‘ab’ viene inserito il numero del blocco contenente il record con chiave ‘aa’. Se devo ricercare, quindi, il record di chiave ‘aa’, siccome questa chiave non è presente nel file degli indici, ma è minore della chiave ab, il sistema si posiziona sul record di chiave ‘ab’ nel file dati, legge se nel blocco di overflow di questo record è indicato un numero di blocco e se è indicato, si posiziona sul primo record del blocco del file di overflow, che, in questo caso, contiene il record cercato.

Un’altra tecnica è lo spazio libero distribuito (si lasciano spazi vuoti nel file indice per consentire l’inserimento di nuovi record).

ACCESSO CALCOLATO.

Accesso calcolato fa uso di una funzione (funzione di *hash*) che dalla chiave cerca di determinare il blocco ove è presente il record.



File dati		
blocco	***key***	***Campi del record***
1	ab	……
	ao	……
2	an	
	ap	
3	ba	

La funzione hash però, se l'algoritmo non è ben costruito, può dar luogo alle cosiddette 'collisioni' chiave diverse producono lo stesso valore del blocco calcolato con la funzione di hash in modo però errato.

ACCESSO RELATIVE

Relative si accede al record tramite la sua posizione (come se fosse un vettore).

File relative	
Posizione	***record***
0	……
1	……
2	……
3	……