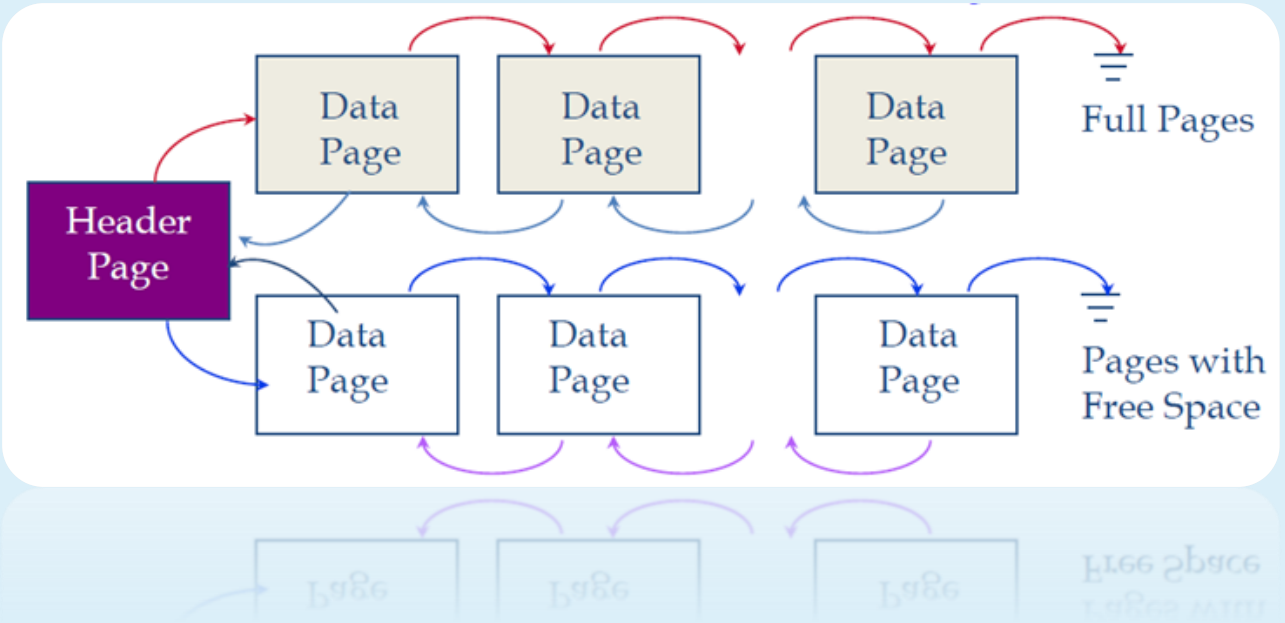


Organizzazioni primarie

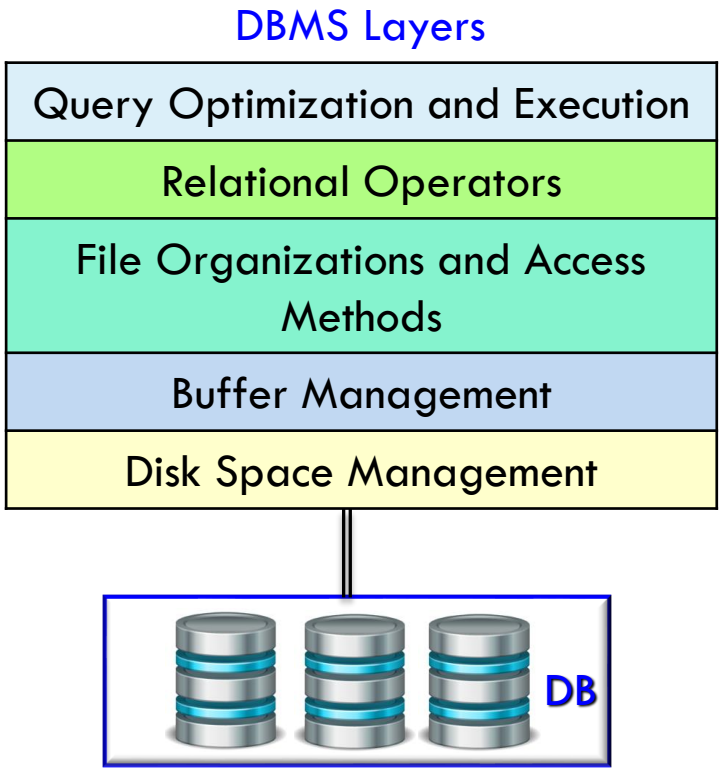
Annalisa Franco, Dario Maio

Università di Bologna

Introduzione alle organizzazioni dei file

- Sono state già discusse in precedenza le modalità di memorizzazione dei dati a livello fisico e le principali tecniche di gestione delle pagine. Un file è una collezione di pagine ciascuna delle quali è una collezione di record.

I livelli di un DBMS che intervengono nella gestione dei dati residenti sui dispositivi di memoria secondaria sono schematizzati in figura.



Tipi di organizzazioni dei dati

- Primaria vs secondaria:
  - Un'organizzazione primaria, al contrario di un'organizzazione secondaria, impone un criterio di allocazione dei dati. Un'organizzazione secondaria si può anche definire come ulteriore metodo di accesso all'organizzazione primaria.
- Statica vs dinamica:
  - Un'organizzazione dinamica si adatta alla mole effettiva dei dati. Viceversa, un'organizzazione statica prevede fasi di "riorganizzazione" globale a fronte di variazioni, più o meno consistenti, del volume di dati da gestire.
- Per chiave primaria vs chiave secondaria
  - Il valore della chiave identifica un unico record in un'organizzazione per chiave primaria, e più record nel secondo caso.

N.B. Il termine chiave indica una combinazione di campi che identifica univocamente un record; spesso è usato anche con il significato di chiave di ricerca, ovvero uno o più campi tramite i quali si accede ai dati.

N.B. Spesso il termine blocco è usato in questa sede come sinonimo di pagina.

Tipi di operazioni

Le operazioni, per quanto complesse, sono riconducibili, in termini di I/O, ad alcune primitive di base:

- Ricerca (esatta) che può essere per:
  - chiave primaria: restituisce al più un solo record

    es. lo studente con matricola 2106110234
  - chiave secondaria: restituisce 0 o più record

    es. gli studenti residenti a Cesena
  - intervallo: restituisce 0 o più record

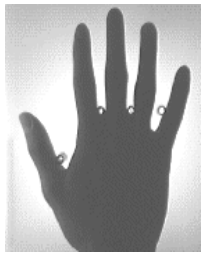
    es. i contribuenti con reddito inferiore a 40.000 €
  - varie combinazioni: restituisce 0 o più record

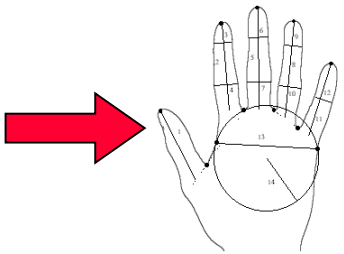
    es. i giocatori dell'Inter o della Juventus di età inferiore a 19 anni
- Inserimento di uno o più record
- Cancellazione di uno o più record
- Modifica di uno o più record
  - Può essere vista come la combinazione di cancellazione e inserimento.

N.B. Il termine transazione indica l'insieme di operazioni elementari che devono essere eseguite per soddisfare una determinata richiesta.

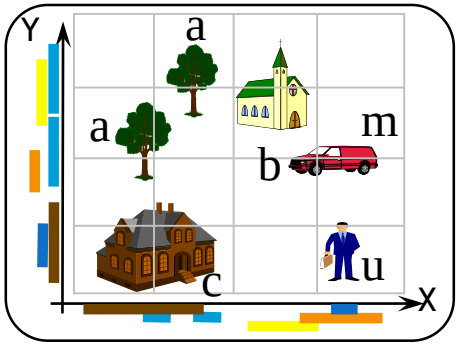
Alcune note: ricerche per similarità

- Si assume d'ora in avanti, per le operazioni di ricerca, di fare riferimento a ricerche di tipo esatto. In alcune applicazioni esiste invece la necessità di ricerche per similarità, ad esempio:

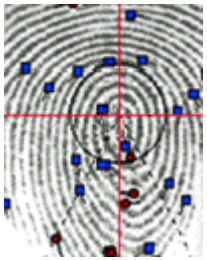


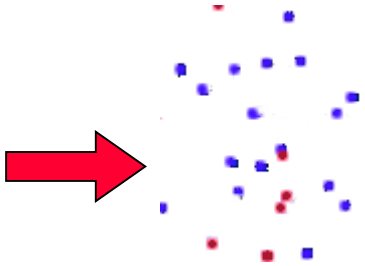


17 feature numeriche



X: a < ca < b <um
Y: cu < abm < a



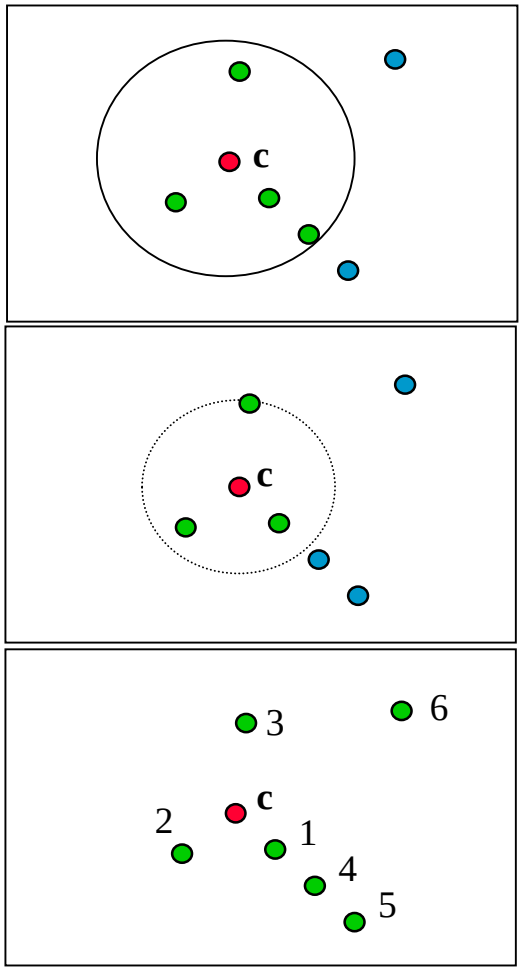


minuzie

Esempi di ricerche per similarità

Ricerche per similarità in uno spazio multidimensionale

- Dato un punto c, detto "centro della query":
  - range query: trovare tutti i punti distanti da c meno di una soglia fissata;
  - k-nearest neighbor: trovare i k punti più vicini a c;
  - ricerche nearest neighbor incrementali o distance scan: reperire incrementalmente i punti ordinati in base alla loro distanza da c.



Clustering e indexing

- Clustering: presenza di "addensamenti" di dati, nei blocchi del file; si noti che l'ordinamento è un caso particolare.
  - La presenza di clustering è importante per ricerche su chiave secondaria, e può essere indotto da dipendenze esistenti tra gli attributi (si pensi, ad esempio, alla dipendenza tra età e stipendio di un impiegato, oppure fra tipo merce e scaffale di allocazione in un supermercato).

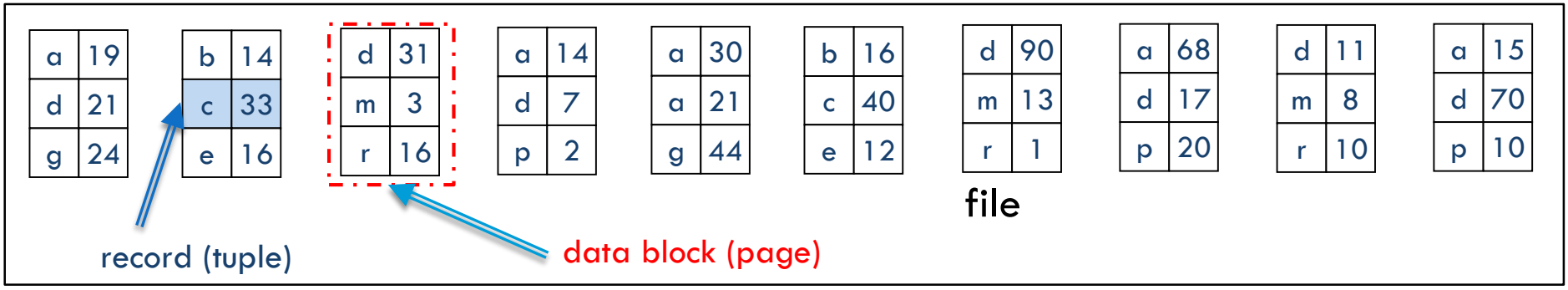
- Indexing: a differenza del clustering che riguarda come i dati sono raggruppati nei blocchi, il concetto di indexing concerne aspetti relativi all'accesso ai dati, cioè la possibilità effettiva di risolvere efficacemente operazioni di ricerca e aggiornamento tramite opportune strutture dati e metodi di accesso.

Astrazioni di file: richiami

- A livello di applicazione un file è un'organizzazione di dati astratta.
  - File "non strutturato": sequenza di byte (stream) su cui è possibile operare attraverso primitive orientate alla manipolazione del singolo byte o di blocchi di byte.
    - Esempi di uso sono: gestione di immagini bit map, pattern matching, gestione di dispositivi virtuali di I/O, operazioni che prescindono dalla struttura logica dei record (copie di backup, trasferimenti in rete, ecc.).
  - File strutturato: collezione di record, a lunghezza fissa o variabile, che appartengono a un certo tipo di dato astratto.
    - Esempi: un file di persone, un file di libri, un file di testo, …

Assunzioni di base

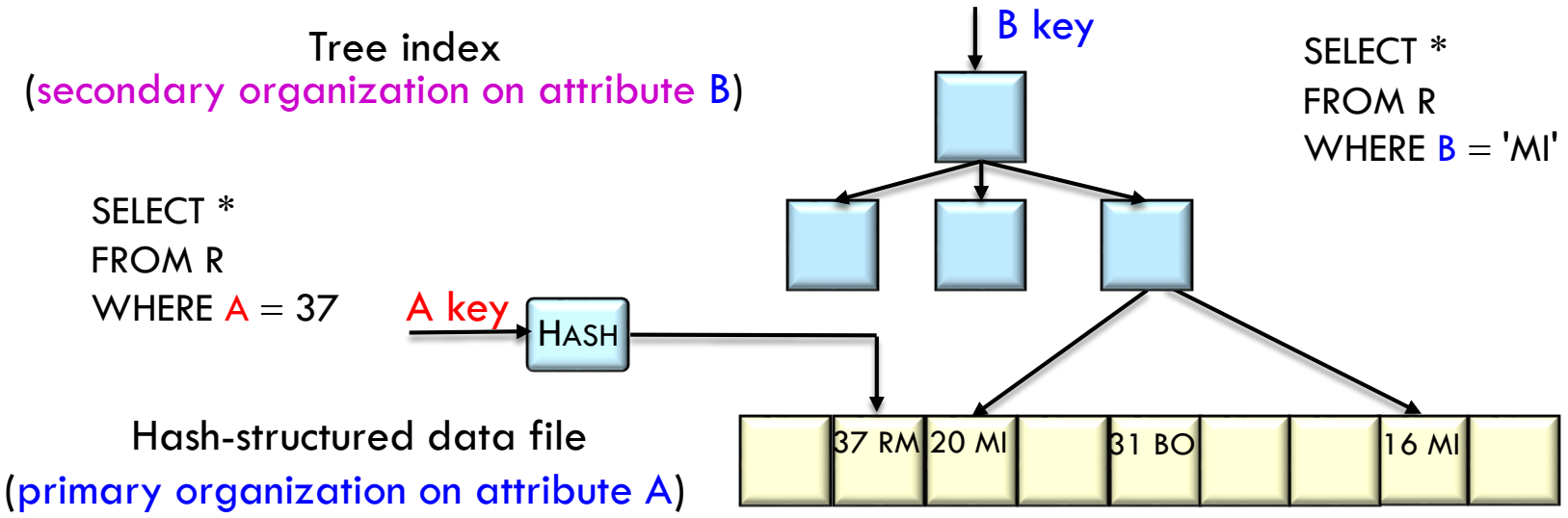
- Al fine di semplificare l'analisi della complessità degli algoritmi, valutata solo in termini di costi di I/O, e più precisamente come numero di letture e/o scritture di blocchi (pagine), i modelli di stima adottati in questa sede fanno alcune assunzioni:
  - si assume che i record del file abbiano tutti la medesima lunghezza; non sono pertanto previsti record a lunghezza variabile;
  - si assume che ogni blocco del file contenga un numero intero e costante di record (a parte eventualmente l'ultimo blocco che può contenere un numero inferiore di record); non è dunque previsto il caso di record che occupino più di un blocco;
  - per la ricerca di un record in un file si assume che ogni blocco del file abbia la stessa probabilità di ospitarlo



Organizzazioni primarie e secondarie

- Le organizzazioni primarie di base si possono suddividere in quattro principali categorie:
  - strutture sequenziali non ordinate (heap)
  - strutture sequenziali ordinate (sorted sequential file);
  - ad accesso calcolato (hash file);
  - ad albero (indexed organization).
- A ciascuna di queste organizzazioni si possono associare alcuni ulteriori metodi di accesso rispetto a quelli propri stabiliti dalle modalità di allocazione dei record nei blocchi del file. Ad esempio, per un sorted file si può costruire un indice ad albero, per facilitare ricerche sulla combinazione di attributi di ordinamento, e altri indici per ricerche su altri attributi.
- Le organizzazioni secondarie sono principalmente strutture ad albero, anche se sono possibili indici hash e altre tipologie di indici (es. bitmap index), e di fatto rappresentano ulteriori cammini d'accesso al file dati.

Organizzazioni di file: esempio



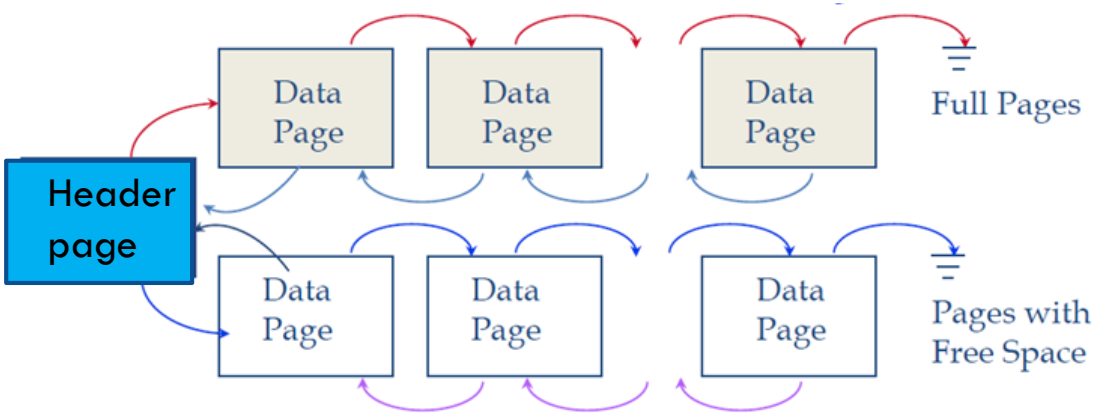
Organizzazioni sequenziali

- Indipendentemente dalla reale allocazione fisica dei blocchi, la sequenza logica di registrazione dei record può rispettare:
  - l'ordine temporale della registrazione e l'organizzazione è detta seriale o heap o pile o anche unsorted file;
  - un ordinamento in base a un campo chiave e l'organizzazione è detta sorted sequential file o clustered file.
- Un heap file è in generale disordinato rispetto ai valori della chiave primaria o ai valori di una qualunque combinazione di attributi, a meno che l'operazione di caricamento dei dati non sia effettuata rispettando un ordine che il DBMS non può comunque conoscere a priori.
- Un sorted sequential file facilita la ricerca sulla combinazione di attributi di ordinamento; tuttavia, mantenere l'ordinamento dei record è oneroso e comporta il ricorso a opportune tecniche di gestione di record in overflow.
- Per entrambe le organizzazioni si rendono necessarie operazioni di riorganizzazione a fronte di numerose modifiche, inserimenti e cancellazioni di record.
- Di solito queste organizzazioni sono in stretta associazione con l'uso di indici.
- Di norma è privilegiata un'allocazione contigua dei blocchi per velocizzare le operazioni di accesso in sequenza ai dati d'interesse.

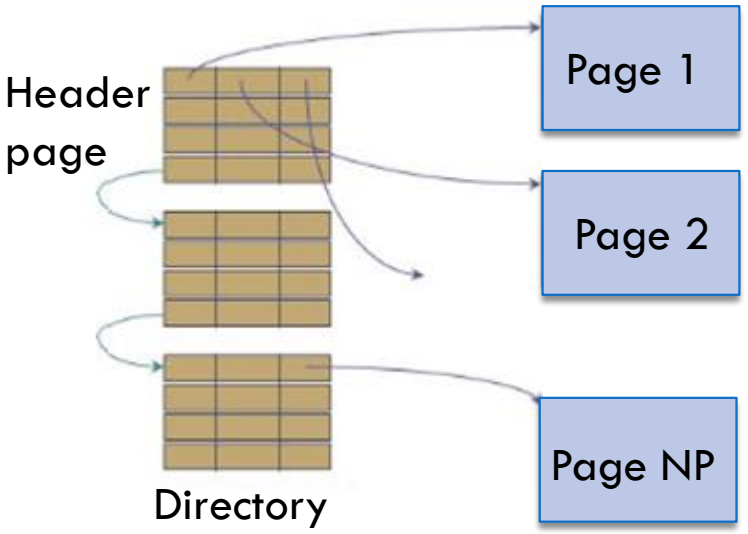
Heap file

- Nella versione più semplice:
  - l'inserimento di un nuovo record è sempre effettuato appendendolo alla fine del file, pertanto è sufficiente mantenere un riferimento all'ultimo blocco del file;
  - l'eliminazione comporta semplicemente marcare come "non valido" il record interessato.
- La struttura permette di gestire record a lunghezza fissa o variabile, nonché record che occupano più pagine.
- Per scandire il file in sequenza il DBMS deve tenere traccia dei blocchi (pagine) che contengono i vari record.
- Nel caso di record a lunghezza fissa è semplice implementare un accesso logico basato su un numero d'ordine del record (relative file).
- I DBMS implementano opportuni accorgimenti per gestire le eliminazioni e gli inserimenti di record in posizioni marcate non più valide, cercando così di non degradare l'efficienza delle operazioni di ricerca e di minimizzare lo spreco di spazio. Con questi metodi, tuttavia, l'inserimento di un record può comportare l'accesso a molte pagine dati fino a trovarne una atta ad ospitarlo e si rende, inoltre, necessario tracciare le pagine con record marcati "non validi".
- La modifica di un record a lunghezza variabile può richiedere la cancellazione del vecchio record e l'inserimento in altra pagina che ha spazio sufficiente.

Heap file: organizzazione delle pagine



Questa soluzione prevede due doubly linked list. Cercare uno spazio per un nuovo record può comportare la visita di svariate pagine nella lista di quelle con spazi liberi.



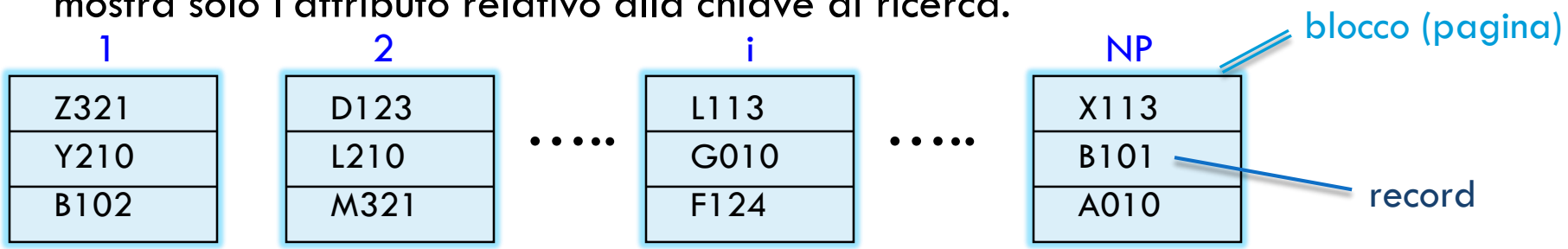
Questa soluzione gestisce invece una directory per tener traccia delle pagine con spazi liberi. In tal modo è più semplice la ricerca di spazio per allocare un nuovo record.

Heap file: utilità

- La struttura non è efficiente per memorizzare record a lunghezza variabile, se soggetta a frequenti aggiornamenti.
- Periodicamente deve essere effettuata una riorganizzazione del file compattando i record ed eliminando quelli marcati come non più validi.
- Queste organizzazioni sono utili in particolari situazioni:
  - Piccoli volumi di dati;
  - operazioni che interessano tutti o gran parte dei record, ad esempio:
    - lettura file di configurazione;
    - calcolo della media dei valori di un campo;
    - interrogazioni poco selettive;
  - aggiornamenti poco frequenti.
- Spesso queste organizzazioni (di base nei DBMS) sono usate in combinazioni con indici come percorsi di accesso alternativi.

Heap file: ricerca di un valore di chiave

File disordinato rispetto alla chiave di ricerca. Per semplicità l'esempio assume che il file contenga valori unici per la chiave di ricerca. La figura mostra solo l'attributo relativo alla chiave di ricerca.



Il modello assume che ogni blocco abbia probabilità $1/\mathrm{NP}$ di ospitare il record cercato:

- caso di esistenza del record cercato:
  - in media si accede a $\sum_{i=1}^{\mathrm{NP}} i \times \frac{1}{\mathrm{NP}} = \frac{\mathrm{NP}+1}{2} \approx \frac{\mathrm{NP}}{2}$ blocchi per $\mathrm{NP} >> 1$;
  - nel caso peggiore si accede a NP blocchi;
- caso di non esistenza del record cercato:
  - si visitano NP blocchi.

N.B. Nel caso di valori ripetuti il costo della ricerca è pari a NP.

Heap file: prestazioni

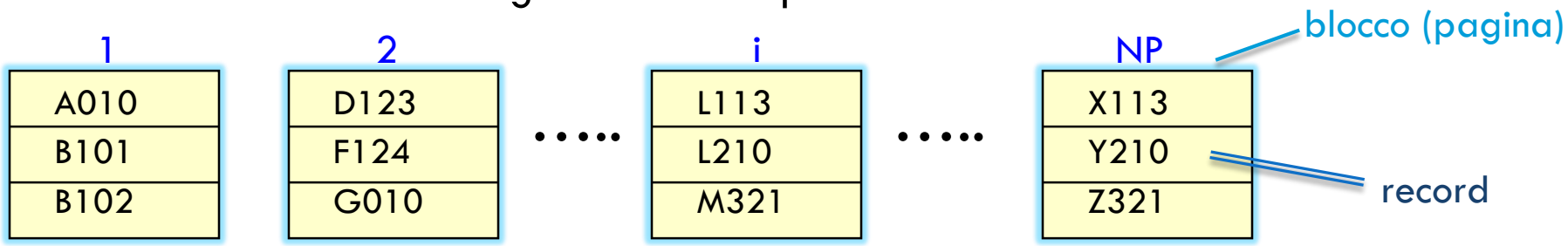
Operazione	Descrizione	Costo num. op. I/O
Ricerca di un valore di chiave (la prima occorrenza)	Scansione sequenziale delle pagine	(NP+1)/2 caso di esistenza in media NP caso peggiore NP caso di non esistenza
Ricerca di valori all'interno di un intervallo	Scansione sequenziale di tutte le pagine	NP
Ricerca di multiple occorrenze di un valore	Scansione sequenziale di tutte le pagine	NP
Inserimento di un record	Si appende in fondo al file	2 (lettura di una pagina e riscrittura)
Eliminazione di un solo record	Ricerca del record e marcatura (si assume che esista)	C(ricerca) + 1 (riscrittura pagina) (ipotesi nessun compattamento) C(ricerca) = 1 se è noto il RID del record da eliminare
Modifica di un solo record	Ricerca del record e aggiornamento (si assume che esista)	C(ricerca) + 1 (riscrittura pagina) C(ricerca) = 1 se è noto il RID del record da modificare

Sorted sequential file

- Questo tipo di organizzazione può dare vantaggi nel caso in cui si debba ricercare record secondo l'ordine dei valori di chiave di ordinamento (caso tipico chiave primaria):
  - ricerca di un singolo record;
  - ricerche di intervallo;
  - ricerca del successore di un record.
- È possibile far ricorso a un algoritmo di ricerca dicotomica.
- Rispetto a un heap file non si ottiene nessun vantaggio per ricerche su altri attributi non di ordinamento.
- Mantenere l'ordinamento puntuale a fronte di inserimenti, cancellazioni e modifiche può essere molto dispendioso. Per rendere più efficiente l'inserimento si prevedono di solito spazi liberi nei blocchi e l'allocazione di ulteriori blocchi in un overflow file non ordinato distinto dal file master. Le ricerche coinvolgono a questo punto anche il file di overflow in modalità di scansione lineare essendo il file mantenuto non ordinato. Periodicamente i due file vengono fusi per produrre un unico file ordinato.

Sorted sequential file: ricerca valore di chiave

File ordinato su un attributo o combinazione di attributi. Per semplicità si assume che il file contenga valori unici per la chiave di ricerca.



- Ricerca sequenziale: sia in caso di esistenza sia in caso di non esistenza si accede in media a (NP+1)/2 blocchi.
- Ricerca dicotomica (assumendo NP $= 2^M - 1$, con M intero > 0)
  - caso di esistenza del record cercato (con NP>>1):
    - in media si accede a $\lfloor \log_2 \mathrm{NP} \rfloor$ blocchi;
    - nel caso peggiore si accede a $\lfloor \log_2 \mathrm{NP} \rfloor + 1$ blocchi;
  - caso di non esistenza del record cercato: si visitano al più $\lfloor \log_2 \mathrm{NP} \rfloor + 1$ blocchi.

Sorted sequential file: prestazioni

Operazione	Descrizione	Costo num. op. I/O
Ricerca di un valore di chiave	Ricerca dicotomica	$\lceil \log_2 \mathrm{NP} \rceil$ caso di esistenza
Ricerca di valori all'interno di un intervallo (attributo di tipo numerico)	Si leggono solo le pagine con valori di chiave in [L,H]; si assume distribuzione uniforme dei valori di chiave	$\lceil \log_2 \mathrm{NP} \rceil + \frac{\mathrm{H} - \mathrm{L}}{\mathrm{HK} - \mathrm{LK}} \times \mathrm{NP} - 1$ N.B. HK e LK rappresentano rispettivamente il valore massimo e il valore minimo della chiave
Inserimento di un record	Si suppone vi sia spazio per l'inserimento nella pagina	$\lceil \log_2 \mathrm{NP} \rceil$+1(ricerca + riscrittura pagina)
Eliminazione di un record	Ricerca del record e marcatura (si assume che esista)	$\lceil \log_2 \mathrm{NP} \rceil$+1 (ricerca + riscrittura pagina) C(ricerca) = 1 se è noto il RID del record
Modifica di un record su attributi diversi dalla chiave.	Ricerca del record e aggiornamento (si assume che esista)	$\lceil \log_2 \mathrm{NP} \rceil$ +1 (ricerca + riscrittura pagina) se ricerca sulla chiave C(ricerca) = 1 se è noto il RID del record C(ricerca) = (NP+1)/2 se ricerca su altri campi

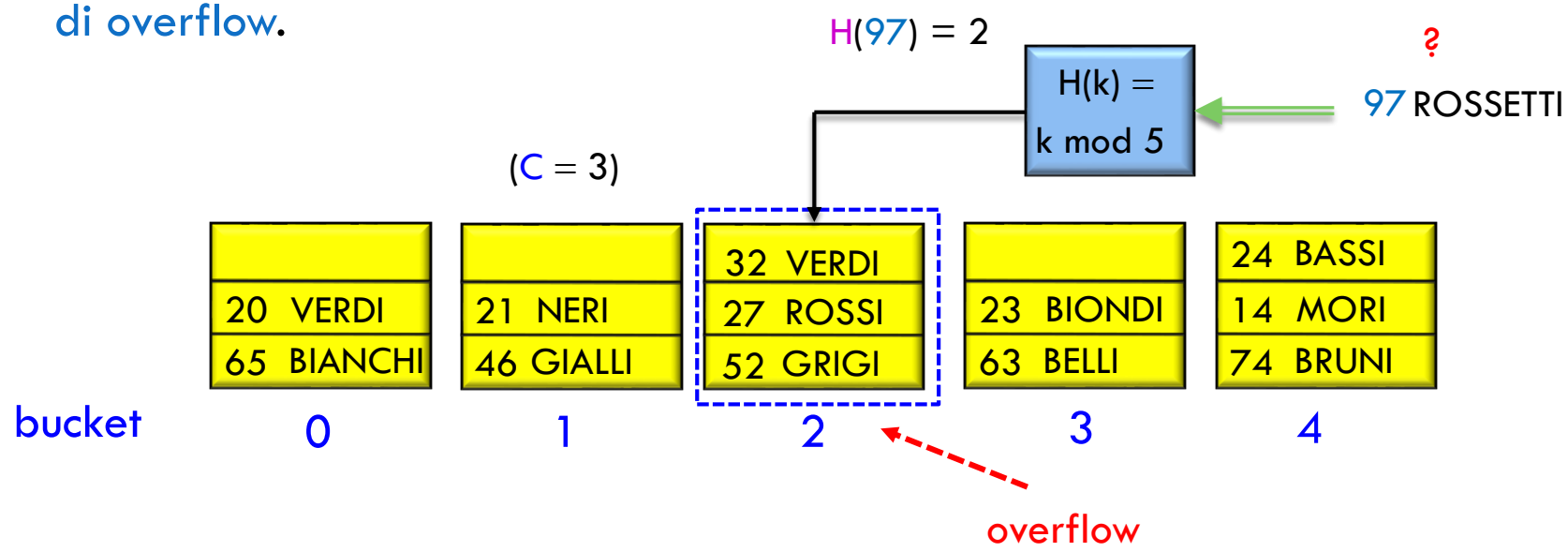
Hash file

- Un altro tipo di organizzazione primaria prevede l'uso di funzioni hash per allocare i record nei blocchi, di norma sulla base di un attributo chiave.

- Ogni indirizzo generato dalla funzione hash H individua una pagina logica (denominata usualmente bucket).
- Salvo casi particolari, le funzioni hash non sono iniettive, cioè non rispettano la proprietà $k1 \neq k2 \Rightarrow H(k1) \neq H(k2)$, e quindi possono verificarsi collisioni: k1 e k2 collidono $\Leftrightarrow H(k1) = H(k2)$.
- Se una chiave viene assegnata a un bucket che non ha spazio, si verifica un overflow (trabocco).

Bucket overflow: esempio

- L'area di memoria costituita dai bucket indirizzabili dalla funzione hash è detta area primaria.
- La presenza di overflow può richiedere, dipendentemente dalla specifica organizzazione, l'uso di un'area di memoria separata, detta appunto area di overflow.



- Pur in presenza di collisioni, le organizzazioni hash sono in genere efficienti per il reperimento di un singolo record dato il suo valore di chiave. Non sono affatto utili per altre tipi di ricerca (es. intervallo) in quanto di norma non preservano l'ordinamento rispetto al valore di chiave.

Organizzazioni statiche e dinamiche

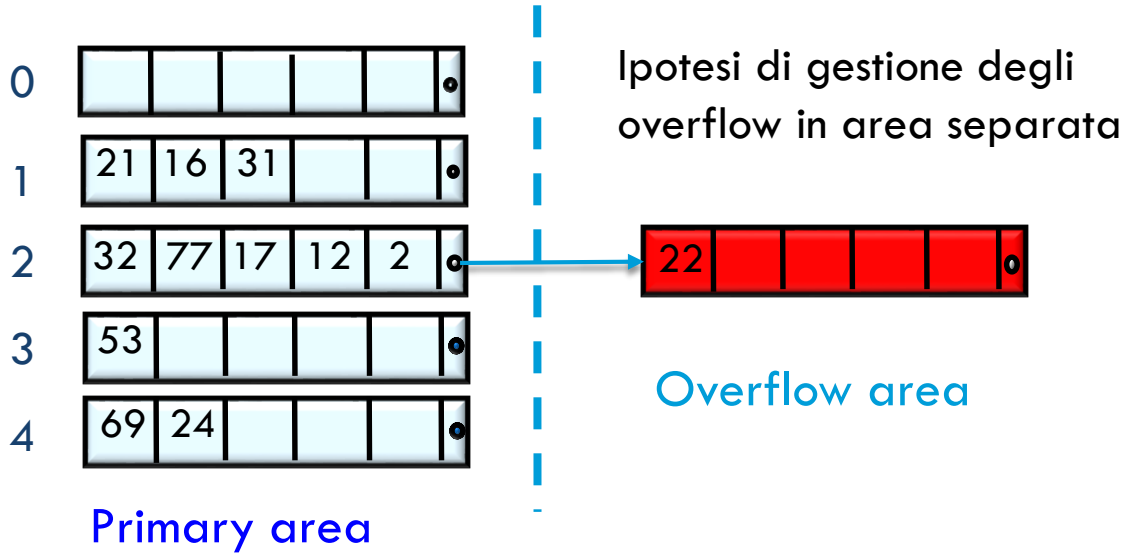
- Una funzione hash deve essere suriettiva, e quindi generare NP indirizzi (0,1,...,NP−1), tanti quanti sono i bucket dell'area primaria.
- Organizzazione statica: il valore di NP è fissato all'atto della creazione della struttura, e mantenuto costante.
  - In questo caso, il dimensionamento dell'area primaria è parte integrante del progetto dell'organizzazione. In presenza di degrado di prestazioni causato da eccessivi overflow si rende necessario provvedere a una riorganizzazione dell'intera struttura.
- Organizzazione dinamica: il valore di NP è variabile, ovvero l'area primaria può espandersi e contrarsi, per adattarsi meglio al volume effettivo dei dati da gestire.
  - In questo scenario si rendono necessarie più funzioni hash.

Parametri di progetto

- Per entrambi i tipi di organizzazione statica e dinamica, vi sono aspetti comuni che meritano considerazione:
  - scelta della funzione hash H;
  - politica di gestione degli overflow;
  - capacità C dei bucket dell'area primaria;
  - Capacità C_{ov} dei bucket dell'eventuale area di overflow (non necessariamente uguale a C);
  - utilizzazione della memoria allocata.

Organizzazioni hash statiche

- Un semplice esempio in cui:
  - le chiavi sono numeri naturali;
  - l'area primaria consiste di NP = 5 bucket di capacità C = 5;
  - la funzione hash è: $H(k_i) = k_i \bmod 5$;
  - gli overflow sono gestiti allocando, per ogni bucket dell'area primaria, uno o più bucket di overflow, di capacità $C_{ov} = 5$, collegati a lista.



Organizzazioni hash dinamiche

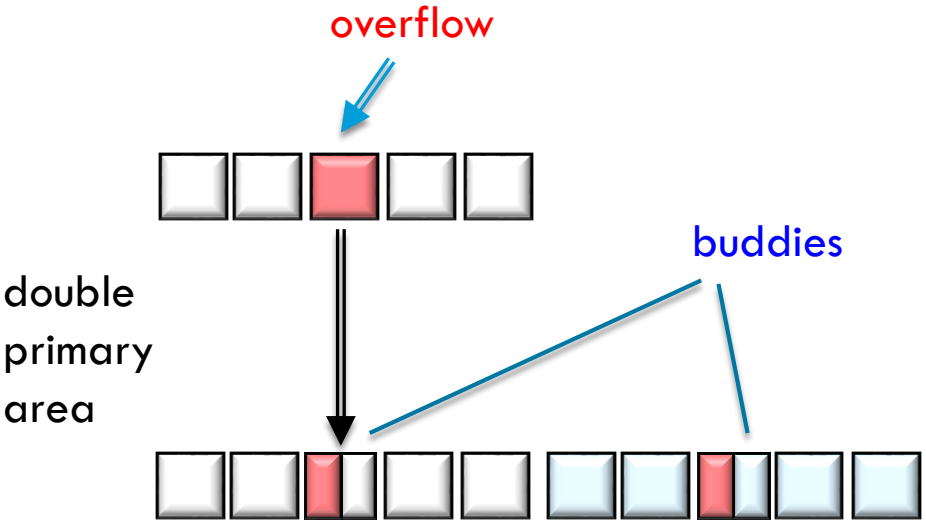
- Il limite principale delle organizzazioni hash statiche riguarda l'allocazione (statica) dell'area primaria. Nel caso di file (fortemente) dinamici un'allocazione statica è inadeguata, a causa o dell'eccessivo spreco di memoria (bassa utilizzazione) o del deterioramento delle prestazioni (alta utilizzazione). Inoltre, nel caso di overflow gestiti in area primaria, si ha il vincolo $d \leq 1$.
- Le organizzazioni hash dinamiche non presentano questi problemi, in quanto adattano l'allocazione dell'area primaria alla dimensione corrente del file.
- Esistono due grandi famiglie di organizzazioni hash dinamiche:
  - con directory (struttura ausiliaria), tra cui:

    Virtual hashing, Dynamic Hashing, Extendible Hashing
  - senza directory, tra cui:

    Linear Hashing, Spiral Hashing

Virtual hashing

- L'idea su cui si basa il Virtual hashing (Litwin 1978) consiste nel raddoppiare l'area primaria quando si verifica un overflow in un bucket, e ridistribuire i record tra il bucket saturo e il suo "buddy", facendo uso di una nuova funzione hash. In pratica si esegue lo "split" del bucket saturo.



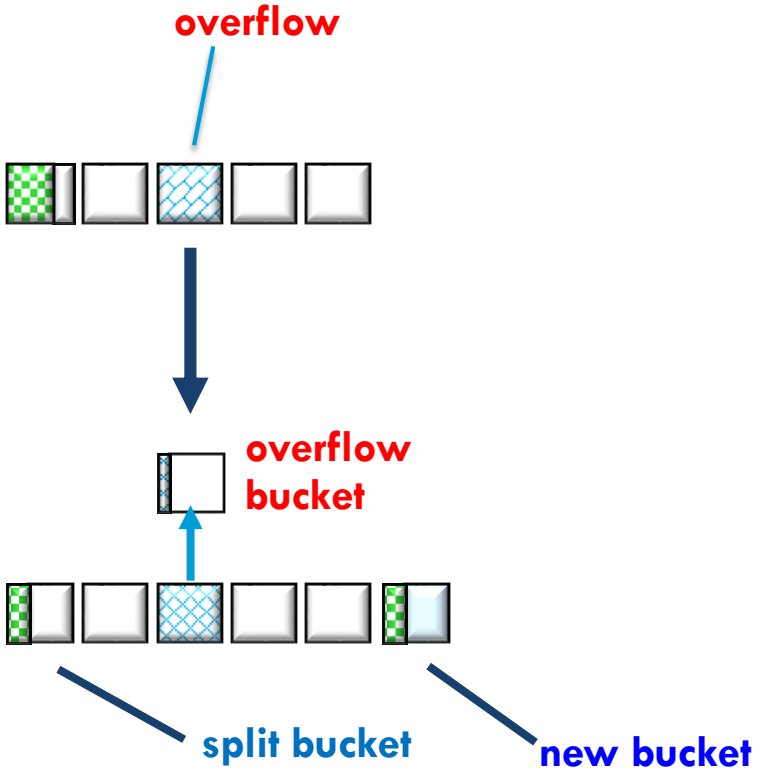
Poiché, a un certo istante, solo alcuni buddy sono effettivamente in uso, è necessario fare ricorso a una struttura ausiliaria che permetta di determinare se occorre utilizzare la vecchia funzione hash o la nuova.

Linear hashing

- L'idea di base del Linear hashing e delle altre organizzazioni hash dinamiche a "espansione lineare", è come segue:

  non si esegue lo split del bucket in cui si è verificato un overflow, ma si suddivide un altro bucket, scelto secondo un criterio prefissato.

- Le principali conseguenze sono:
  - non è necessaria una directory, in quanto si è a conoscenza dei bucket che sono stati suddivisi;
  - occorre gestire l'overflow in area primaria o in area separata;
  - l'area primaria cresce "linearmente"; non si effettuano raddoppi.
- Gestendo gli overflow in area separata, ed eseguendo gli split in sequenza ordinata, a partire dal bucket 0, si ha lo schema base del linear hashing.



Gestione dell'area primaria

- Inizialmente si allocano NP_0 bucket e si usa la funzione hash:

$$H_0(k) = k \bmod NP_0$$

- Si mantiene un puntatore (detto split pointer, SP) al prossimo bucket che deve essere suddiviso. Inizialmente $SP = 0$.
- Se si verifica un overflow si aggiunge in coda un bucket di indirizzo $NP_0 + SP$, si riallocano i record del bucket SP (inclusi quelli eventualmente presenti in area di overflow) facendo uso della nuova funzione hash:

$$H_1(k) = k \bmod (2 \times NP_0)$$ e si incrementa SP ($= SP + 1$).

- Dopo NP_0 overflow si è operata un'espansione completa dell'area primaria, in quanto il numero di bucket è ora pari a $2 \times NP_0$.
- Dopo un'espansione completa, ci si predispone per una nuova espansione…

Hash file: prestazioni

Modello semplificato: si considera che non ci siano record in overflow.

Operazione	**Descrizione**	**Costo num. op. I/O**
Ricerca di un valore di chiave	Si accede a una sola pagina	1
Ricerca all'interno di un intervallo	Si leggono tutte le pagine	NP
Inserimento di un record	Si suppone vi sia spazio per l'inserimento nella pagina	1 (lettura) + 1 (riscrittura pagina)
Eliminazione di un record	Ricerca del record e marcatura (si assume che esista)	1 (lettura) + 1 (riscrittura pagina)
Modifica di un record su attributi diversi dalla chiave.	Ricerca del record e aggiornamento (si assume che esista)	1 (lettura) + 1 (riscrittura pagina)

Osservazioni sulle organizzazioni primarie

- Ciascuna delle organizzazioni primarie esaminate sin ora presenta vantaggi e svantaggi e la scelta più consona dipende da molteplici fattori tra cui: occupazione di memoria, carico di lavoro, esigenze particolari di tempi di risposta, ecc.
- Nessuna di esse esibisce comportamenti soddisfacenti a fronte delle diverse tipologie di operazioni richieste. Certamente non sono in grado di rispondere efficientemente a ricerche su attributi diversi da quelli usati per l'allocazione dei record.
- Queste sono le motivazioni per far ricorso a strutture di indicizzazione, che possono essere di varia natura. Nel contesto dei DBMS per convenzionali applicazioni gestionali molto utilizzati sono indici B-tree, B^+-tree e loro varianti.
- Sono stati sviluppati nel tempo molti tipi di indici, alcuni specificatamente progettati per soddisfare particolari esigenze di gestione dei dati; ad esempio Kd-tree e R-tree a supporto di operazioni su dati multidimensionali in campi di impiego quali CAD (Computer Aided Design) e GIS (Geographical Information Systems).

Domande?