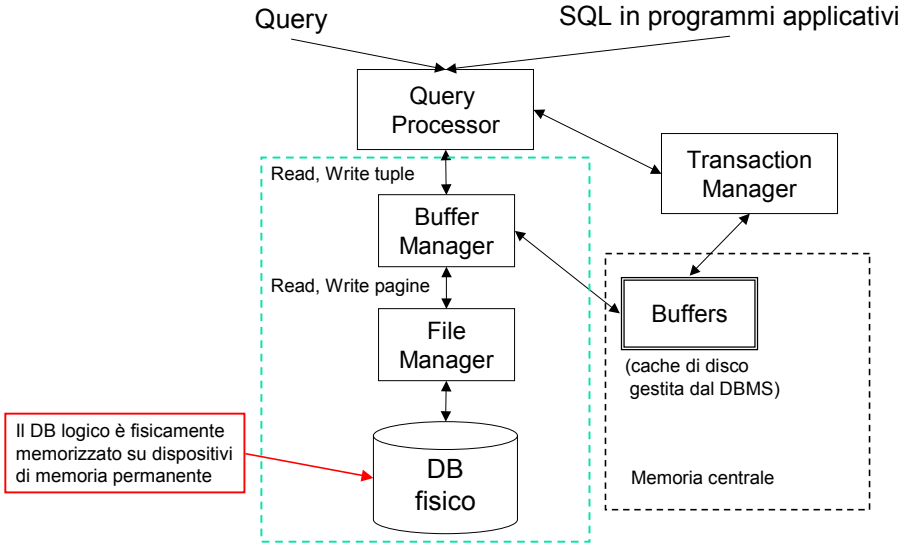
Il Data Base fisico

Sistemi Informativi L-B

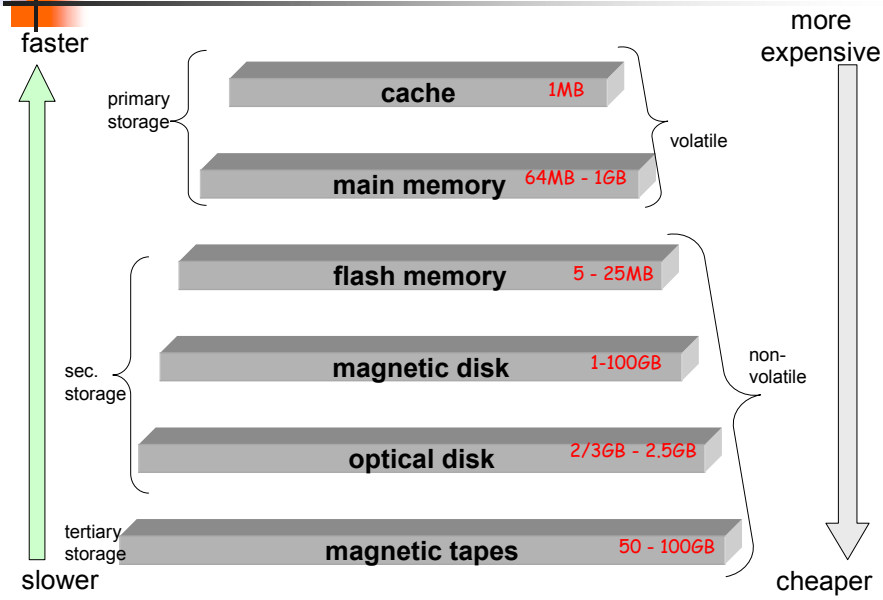
Home Page del corso:
http://www-db.deis.unibo.it/courses/SIL-B/

Versione elettronica: DBfisico.pdf

L'architettura (semplificata) di un DBMS



Gerarchia delle memorie (1)



Gerarchia delle memorie (2)

- La memoria di un sistema di calcolo è organizzata in una gerarchia. Al livello più alto memorie di piccola dimensione, molto veloci, costose; scendendo lungo la gerarchia la dimensione aumenta, diminuiscono la velocità e il costo
- Prestazioni di una memoria:
  - dato un indirizzo di memoria, le prestazioni si misurano in termini di tempo di accesso, determinato dalla somma della
    - latenza (tempo necessario per accedere al primo byte), e del
    - tempo di trasferimento (tempo necessario per muovere i dati)

$$\text{tempo di accesso} = \text{latenza} + \frac{\text{dimensione dati da trasferire}}{\text{velocità di trasferimento}}$$

Legge di Moore

- Gordon Moore: "*Integrated circuits are improving in many ways, following an exponential curve that doubles every 18 months*"
  - Velocità dei processori
  - Numero di bit integrabili in un chip
  - Numero di byte memorizzabili in un hard disk (HD)
- Parametri che NON seguono la legge di Moore:
  - Tempo di accesso alle memorie
  - Velocità a cui gli HD ruotano
- La conseguenza è che diventa relativamente sempre più oneroso muovere i dati lungo i livelli della gerarchia

Implicazioni per i DBMS

- Un DB, a causa della sua dimensione, risiede normalmente su dischi (e eventualmente anche su altri tipi di dispositivi)
- I dati devono essere trasferiti in memoria centrale per essere elaborati dal DBMS
- Il trasferimento non avviene in termini di singole tuple, bensì di blocchi (o **pagine**, termine comunemente usato quando i dati sono in memoria)
- Poiché spesso le operazioni di I/O cosituiscono il collo di bottiglia del sistema, si rende necessario ottimizzare l'implementazione fisica del DB, attraverso:
  - Organizzazione efficiente delle tuple su disco
  - Strutture di accesso efficienti
  - Gestione efficiente dei buffer in memoria
  - Strategie di esecuzione efficienti per le query

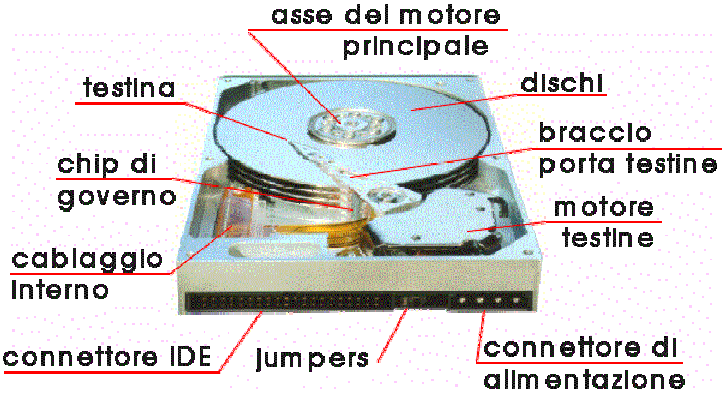
Tipi di dispositivi

- Nastri magnetici: a bobina, a cartuccia, DAT
- Dischi magnetici: a testine fisse, a testine mobili, Winchester, ….,Raid
- Floppy disk
- Dischi ottici, Dischi ottico-magnetici, …..

- Per il loro interesse e la loro diffusione, vediamo qualcosa sulla tecnologia dei dischi magnetici (hard disk)
  - …anche per capire quanto può richiedere l'esecuzione di una query!

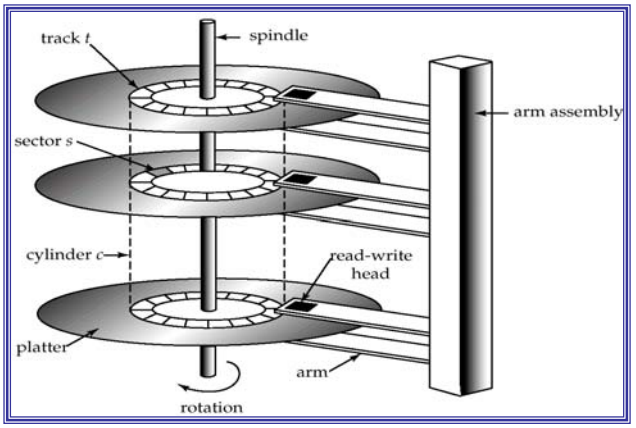
Gli hard disk

- Un hard disk (HD) è un dispositivo elettro-meccanico per la conservazione di informazioni sotto forma magnetica, su supporto rotante a forma di piatto su cui agiscono delle testine di lettura/scrittura



Il meccanismo del disk drive

Include organi di registrazione, di posizionamento e di rotazione



NB: Il diagramma è estremamente semplificato

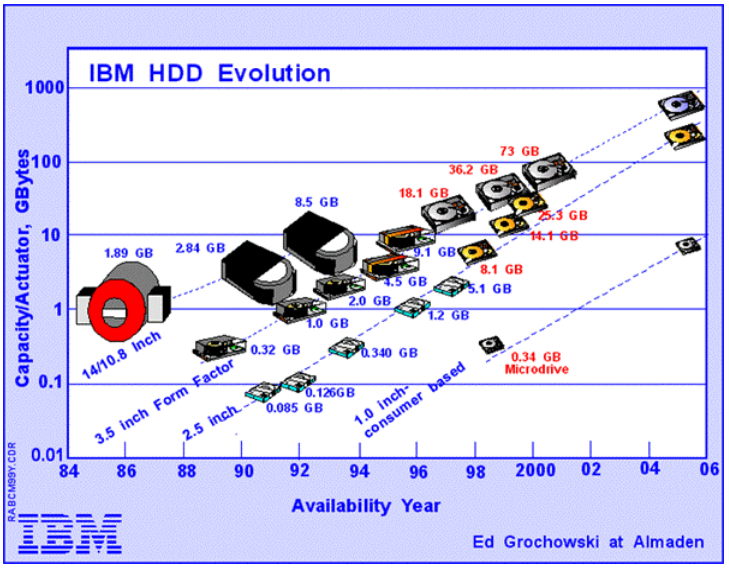
HD: elementi di base

- Testine di lettura e scrittura
- Superficie di un piatto divisa in tracce concentriche
  - Più di 16000 tracce per superficie su tipici HD
- Ogni traccia è divisa in settori
  - Un settore è la più piccola unità di dati che può essere letta o scritta
  - Sector size: tipicamente 512 byte
  - Numero settori per traccia: dell'ordine dei 100-1000
- Per leggere/scrivere un settore:
  - Il braccio si posiziona sulla traccia
  - I dati vengono letti/scritti quando il settore passa sotto la testina
- Dischi a piatti multipli
  - una testina per superficie, tutte montate su un braccio comune
- L'i-mo cilindro consiste delle i-me tracce di tutti i piatti

Caratteristiche costruttive

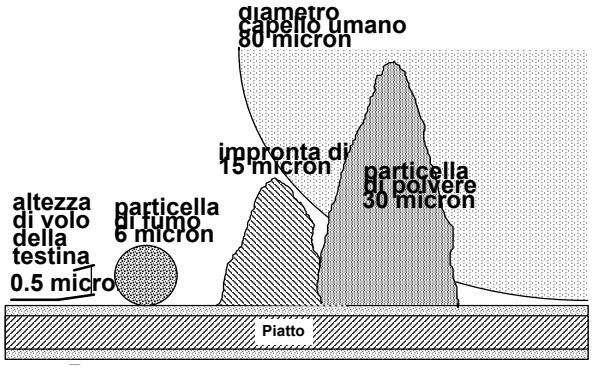
- Piatti di alluminio ricoperti da materiale ferromagnetico. Recentemente introdotti piatti in materiale vetroso, più resistenti agli urti (accidentali) con le testine
- Diametri: 2.5, 3.5, 5.25 inch. A parità di densità, diminuiscono la superficie, la capacità di registrazione, la potenza assorbita e la distanza tra tracce; aumenta la velocità di rotazione
- La densità lineare è limitata dalla difficoltà delle operazioni di registrazione/lettura delle variazioni del campo magnetico sulla superficie del disco
- Sono state ottenute densità di area superiori a 10 Gbits/in^2, e nel 1999 i laboratori IBM hanno raggiunto 35.3 Gbits/in^2 (524000 bpi (bits/inch) di densità lineare e 67300 tpi (tracks/inch) di densità di traccia)
- In contrasto, il primo HD per PC aveva una densità d'area di circa 0.004 Gbits/in^2!
- Poiché a velocità angolare costante un numero fisso di settori per traccia implica una densità minore per le tracce più esterne, si è passati a HD in cui la velocità angolare varia a zone, permettendo quindi di avere più settopri nelle tracce più esterne
- Velocità di rotazione tipiche: 3600, 5400, 7200, 10000 rpm (rounds/minute)

L'evoluzione della capacità

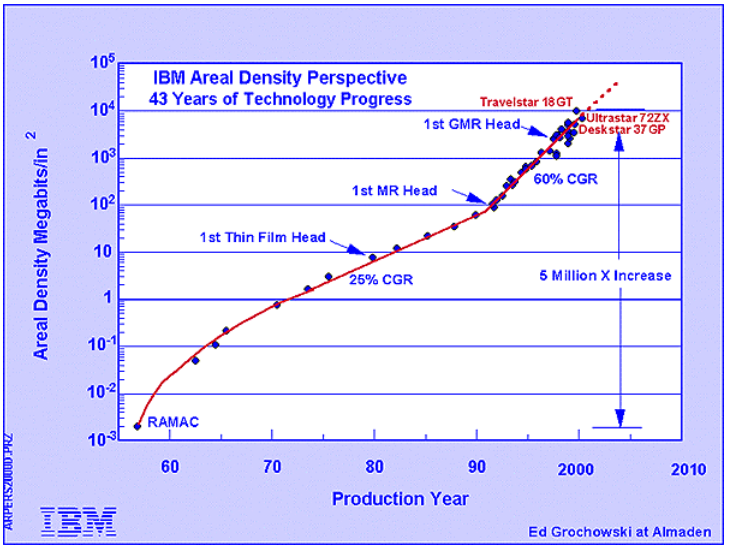


Le testine

- In scrittura, le testine svolgono il lavoro di conversione dei bit in impulsi magnetici che vengono quindi registrati sulle superfici dei dischi; in lettura attuano la conversione inversa
- Sono componenti molto critici per gli effetti che hanno sulle prestazioni e costituiscono le parti più costose di un HD



L'evoluzione della densità

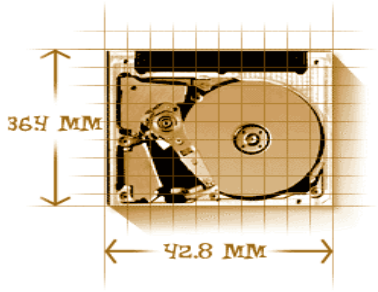


Settori, tracce, cilindri: un esempio

Modello Fujitsu MPF3XXXAT 5400 rpm, 3,5”

Storage Capacity (formatted)	**MPF3102AT**	**10.2 GB**
	MPF3153AT	**15.3 GB**
	MPF3204AT	**20.4 GB**
Disks	**MPF3102AT**	**1**
	MPF3153AT	**2**
	MPF3204AT	**2**
Heads (read / write)	**MPF3102AT**	**2**
	MPF3153AT	**3**
	MPF3204AT	**4**
Tracks capacity (formatted)		**188,416 to 344,046 bytes**
Bytes / sectors		**512**
Cylinders		**20,020**
Sectors/Track		**368 to 672**

Il più piccolo..



IBM Microdrive

ha un singolo piatto, diametro di 1”,
progettato per essere inserito
in telecamere e in personal organizer,
ha una capacità di 340 MB

Prestazioni

- Le prestazioni possono essere classificate in:
- Interne: dipendono da
  - Caratteristiche meccaniche
  - Tecniche di memorizzazione e codifica dei dati
  - Disk controller (interfaccia tra l'HW del disco e il sistema di calcolo)
    - Accetta comandi di alto livello per leggere/scrivere settori e controlla il meccanismo
    - Aggiunge nei settori informazioni per il controllo degli errori (checksum)
    - Verifica la correttezza delle scritture rileggendo i settori scritti
    - Esegue il mapping tra indirizzi logici dei blocchi e settori su disco
- Esterne: dipendono da
  - Tipo di interfaccia
  - Architettura del sottosistema di I/O
  - File system

Prestazioni interne

- La figura che più incide è la latenza (ovvero il tempo impiegato per raggiungere le informazioni di interesse), composta da:
  - Command Overhead Time: tempo necessario a impartire comandi al drive
    - È dell'ordine di 0.5 ms e può essere trascurato
  - Seek Time (**Ts**): tempo impiegato dal braccio a posizionarsi sulla traccia desiderata
    - Il tempo medio di seek, dell'ordine di 2-10 ms, è, nel caso di tracce con lo stesso numero di settori, 1/3 del tempo massimo di seek
    - Il seek time per scritture è superiore di circa 1 ms rispetto al seek time per letture
  - Settle Time: tempo richiesto per la stabilizzazione del braccio
  - Rotational Latency (**Tr**): tempo di attesa del primo settore da leggere
    - La latenza rotazionale media è 1/2 rispetto al caso peggiore
    - Da 2 a 11 ms

Esempio: IBM 34GXP drive

Component	Best-Case Figure (ms)	Worst-Case Figure (ms)
Command Overhead	0.5	0.5
Seek Time (Ts)	2.2	15.5
Settle Time	<0.1	<0.1
Rotational Latency (Tr)	0.0	8.3
Total	2.8	28.4

Rotational Latency

(60/Spindle Speed)* 0.5 * 1000

Spindle Speed (RPM)	Worst-Case Latency (Full Rotation) (ms)	Average Latency (Half Rotation) (ms)
3,600	16.7	8.3
4,200	14.2	7.1
4,500	13.3	6.7
4,900	12.2	6.1
5,200	11.5	5.8
5,400	11.1	5.6
7,200	8.3	4.2
10,000	6.0	3.0
12,000	5.0	2.5
15,000	4.0	2.0

Transfer Rate

- È la velocità massima alla quale il drive può leggere o scrivere dati
  - Tipicamente dell'ordine di qualche decina di MB/s, si riferisce alla velocità con cui si trasferiscono bit dai (sui) piatti sulla (dalla) cache del controller
- Si può stimare come:

$$\frac{\text{(bytes/sector) x (sectors/track)}}{\text{rotation time}}$$

Esempio: Con 512 bytes/sector, 368 sectors/track, 7200 rpm il transfer rate è pari a (512 x 368) / (60 / 7200) = 21.56 MB/sec

- In pratica le velocità di trasferimento da/per HD sono più basse di quelle nominali (4-10 MB/sec)

Pagine

- Un blocco (o **pagina**) è una sequenza contigua di settori su una traccia, e costituisce l'unità di I/O per il trasferimento di dati da/per la memoria principale
- La dimensione tipica di una pagina è di qualche KB (4 - 64 KB)
  - Pagine piccole comportano un maggior numero di operazioni di I/O
  - Pagine grandi tendono ad aumentare la frammentazione interna (pagine parzialmente riempite) e richiedono più spazio in memoria per essere caricate
- Il tempo di trasferimento di una pagina (**Tt**) da disco a memoria centrale dipende dalla dimensione della pagina (**P**) e dal transfer rate (**Tr**)

Esempio:

con un transfer rate di 21.56 MB/sec e P = 4 KB si ha Tt = 0.18 ms,
con P = 64 KB si ha Tt = 2.9 ms

Il DB fisico

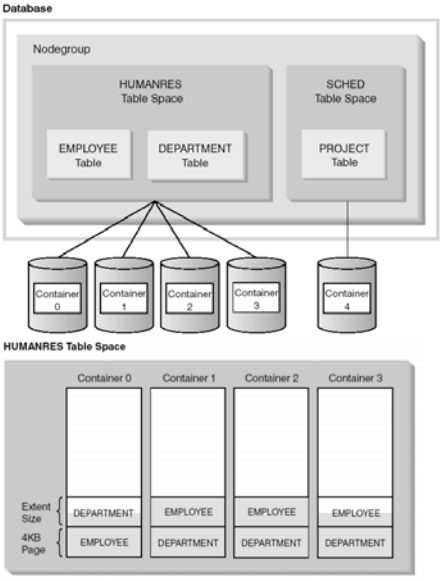
- A livello fisico un DB consiste di un insieme di file, ognuno dei quali viene visto come una collezione di pagine, di dimensione fissa (es: 4 KB)
- Ogni pagina memorizza più record (corrispondenti alle tuple logiche)
- A sua volta un record consiste di più campi, di lunghezza fissa e/o variabile, che rappresentano gli attributi

- I "file" del DBMS che qui consideriamo non corrispondono necessariamente a quelli del file system del sistema operativo
- Casi limite:
  - Ogni relazione del DB è memorizzata in un proprio file
  - Tutto il DB è memorizzato in un singolo file

- In pratica ogni DBMS a livello fisico adotta soluzioni specifiche più articolate e flessibili

Il modello di memorizzazione di DB2

- DB2 organizza lo spazio fisico in tablespace, ognuno dei quali è una collezione di container
- Tipicamente diversi container usano dischi differenti
- Ogni container è a sua volta diviso in extent, che rappresentano l'unità minima di allocazione su disco e sono costituiti da insiemi contigui di pagine di 4 KB (valore di default di P)
- La dimensione di un extent dipende dallo specifico tablespace, e viene scelta all'atto della creazione del tablespace
- Ogni relazione è memorizzata in un singolo tablespace, ma un tablespace può contenere più relazioni; viceversa un extent contiene dati di una singola relazione



Tipi di tablespace

- In un tablespace di tipo SMS (System Managed Space):
  - La gestione dello spazio su disco è demandata al sistema operativo
  - Un container corrisponde a una directory del file system
  - I file vengono estesi (dal file system) una pagina alla volta
    - Quest'ultimo fatto può portare a una **frammentazione** del file che può rallentare le operazioni di I/O
- In un tablespace di tipo DMS (Database Managed Space):
  - La gestione è a carico del DBMS
  - Un container è o un file di dimensione prefissata (statica) o un dispositivo (HD)
  - Il tablespace può essere esteso solo aggiungendo container
- L'uso dei tablespace DMS permette di raggiungere le migliori prestazioni nel caso di DB di grandi dimensioni

Attributi dei tablespace

- All'atto della creazione di un tablespace è possibile specificare una serie di parametri, tra cui:
  - EXTENTSIZE: numero di blocchi dell'extent
  - BUFFERPOOL: nome del pool di buffer associato al tablespace
  - PREFETCHSIZE: numero di pagine da trasferire in memoria prima che vengano effettivamente richieste
  - OVERHEAD: stima del tempo medio di latenza per un'operazione di I/O
  - TRANSFERRATE: stima del tempo medio per il trasferimento di una pagina
- Gli ultimi due parametri vengono usati dall'ottimizzatore

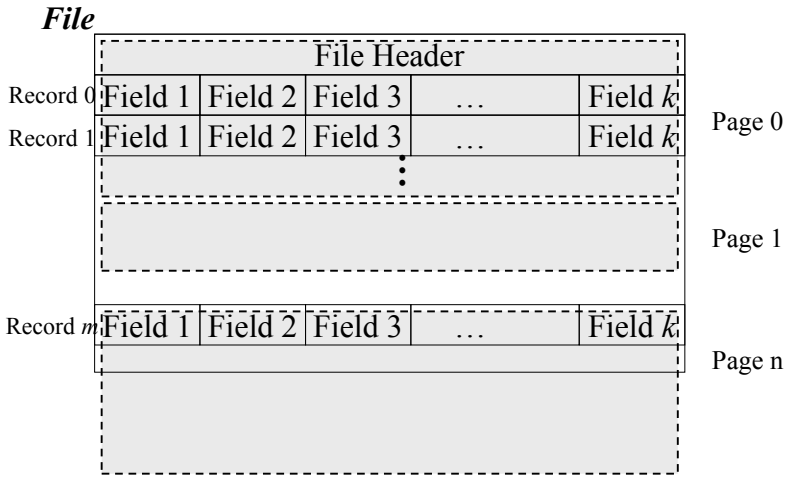
Perché non usare sempre il file system?

- Le prestazioni di un DBMS dipendono fortemente da come i dati sono organizzati su disco
- Intuitivamente, l'allocazione dei dati dovrebbe mirare a ridurre i tempi di accesso ai dati, e per far questo bisogna sapere come (logicamente) i dati dovranno essere elaborati e quali sono le relazioni (logiche) tra i dati
- Tutte queste informazioni non possono essere note al file system

  Esempi:
  - Se due relazioni contengono dati tra loro correlati (mediante join) può essere una buona idea memorizzarle in cilindri vicini, in modo da ridurre i tempi di seek
  - Se una relazione contiene attributi BLOB, può essere una buona idea memorizzarli separatamente dagli altri attributi

Organizzazione dei dati nei file

Schema di riferimento (semplificato)



Rappresentazione dei valori

- Per ogni tipo di dati di SQL viene definito un formato di rappresentazione, ad es.:
- Stringhe a lunghezza fissa: CHAR(n)
  - Si usano n byte, eventualmente usando un carattere speciale per valori lunghi meno di n

  Esempio: se A è CHAR(5), 'cat' viene memorizzato come cat⊥⊥
- Stringhe a lunghezza variabile: VARCHAR(n)
  - Si allocano m+p byte, con m (≤ n) byte usati per gli m caratteri effettivamente presenti e p byte per memorizzare il valore di m (per n ≤ 254 p = 1)

  Esempio: se A è VARCHAR(10), 'cat' viene memorizzato in 4 byte come 3cat
- DATE e TIME sono normalmente rappresentati con stringhe di lunghezza fissa
  - DATE: 10 caratteri YYYY-MM-DD; TIME: 8 caratteri HH:MM:SS
- Tipi enumerati: si usa una codifica intera

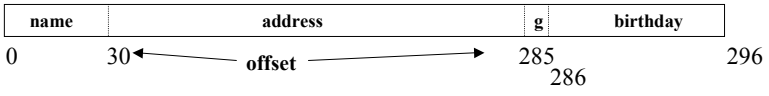
  Esempio: week = {SUN, MON, TUE, …, SAT} richiede un byte per valore

  SUN: 00000001, MON: 00000010, TUE: 00000011, …

Record a lunghezza fissa

- Per ogni tipo di record nel DB deve essere definito uno schema (fisico) che permetta di interpretare correttamente il significato dei byte che costituiscono il record
- La situazione più semplice si ha evidentemente quando tutti i record hanno lunghezza fissa, in quanto, oltre alle informazioni logiche, è sufficiente specificare l'ordine in cui gli attributi sono memorizzati nel record (se differente da quello di default)

```
CREATE TABLE MovieStar (
name       CHAR(30) PRIMARY KEY,
address    CHAR(255),
gender     CHAR(1),
birthdate  DATE )
```

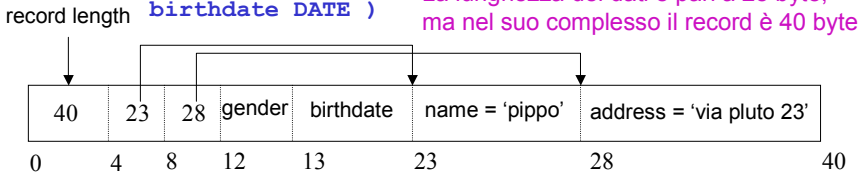


Record a lunghezza variabile

- Nel caso di record a lunghezza variabile si hanno diverse alternative, che devono considerare anche i problemi legati agli aggiornamenti che modificano la lunghezza dei campi (e quindi dei record)
- Una soluzione consolidata consiste nel memorizzare prima tutti i campi a lunghezza fissa, e quindi tutti quelli a lunghezza variabile; per ogni campo a lunghezza variabile si ha un "prefix pointer" che riporta l'indirizzo del primo byte del campo

```
CREATE TABLE MovieStar (
name      VARCHAR(30) PRIMARY KEY,
address   VARCHAR(255),
gender    CHAR(1),
birthdate DATE )
```

La lunghezza dei dati è pari a 28 byte, ma nel suo complesso il record è 40 byte

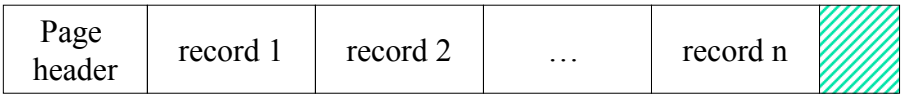


Record Header

- In generale ogni record include un header che, oltre alla lunghezza del record, può contenere:
  - L'identificatore della relazione cui il record appartiene
  - L'identificatore univoco del record nel DB
  - Un timestamp, che indica quando il record è stato inserito o modificato l'ultima volta
- Il formato specifico dell'header ovviamente varia da un DBMS all'altro

Organizzare i record in pagine

- Normalmente la dimensione di un record è (molto) minore di quella di una pagina
  - Esistono tecniche particolari (che qui non vediamo) per gestire il caso di “long tuples”, la cui dimensione eccede quella di una pagina
- Nel caso di record a lunghezza fissa l’organizzazione in una pagina si potrebbe presentare così:



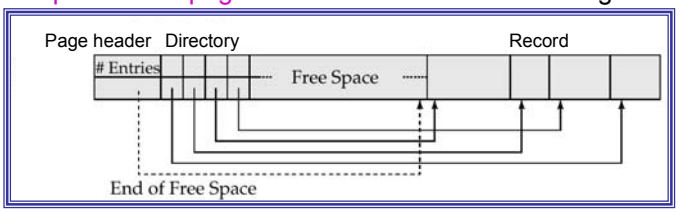
- Il page header mantiene informazioni quali:
  - ID della pagina nel DB, timestamp che indica quando la pagina è stata modificata l’ultima volta, relazione cui le tuple nella pagina appartengono, ecc.
- Normalmente un record è contenuto interamente in una pagina
  - Quindi si può avere dello spazio sprecato

Un semplice esempio

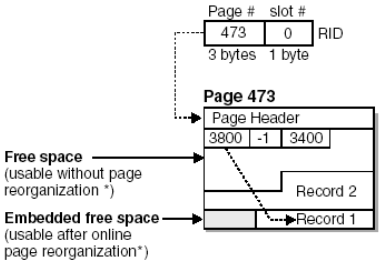
- Nel caso visto prima, con record di lunghezza fissa pari a 296 byte, si supponga di usare pagine di dimensione P = 4 KB = 4096 byte
- Supponendo che l’header della pagina richieda 12 byte ne restano 4084 per i dati
- Pertanto è possibile memorizzare in una pagina fino a 13 record ($13 = \lfloor 4084/296 \rfloor$)
  - In ogni pagina resteranno quindi sempre inutilizzati almeno 236 byte
- … se la relazione MovieStar contiene 10000 tuple serviranno quindi almeno 770 pagine per memorizzarla ($770 = \lceil 10000/13 \rceil$)
- … e se per leggere una pagina da disco ci vogliono 10 ms, la lettura di tutte le tuple richiederà circa 7.7 secondi

Organizzazione a slot delle pagine

- Il formato tipico di una pagina in un DBMS è descritto in figura

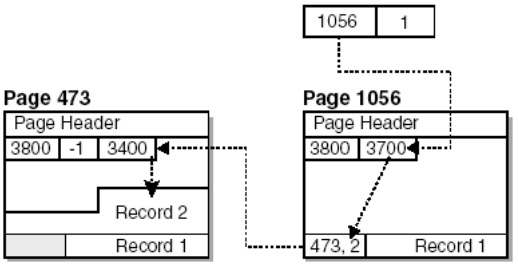


- La directory contiene un puntatore per ogni record nella pagina
- Con questa soluzione l'identificatore di un record (**RID**) nel DB è formato da una coppia:
  - **PID**: identificatore della pagina
  - **Slot**: posizione all'interno della directory
- È possibile sia individuare velocemente un record, sia permettere la sua riallocazione nell pagina senza modificare il RID



Record in overflow

- Se un update fa aumentare la dimensione di un record e non c'è più spazio per continuare a memorizzarlo nella stessa pagina, il record viene spostato in un'altra pagina (va in "overflow")
- Il RID del record tuttavia non cambia, ma si introduce un livello di indirezione
- Avere molti record in overflow ovviamente porta a un degrado delle prestazioni, per cui si può periodicamente rendere necessario riorganizzare il file



Lettura e scrittura di pagine

- La lettura di una tupla richiede che la pagina corrispondente sia prima portata in memoria, in un'area gestita dal DBMS detta buffer pool
- Ogni buffer nel pool può ospitare una copia di una pagina su disco
- La gestione del buffer pool, che è fondamentale dal punto di vista prestazionale, è demandata a un modulo del DBMS, detto Buffer Manager (BM)
- Il BM è chiamato in causa anche nel caso di scritture, ovvero quando bisogna riscrivere su disco una pagina modificata
- Il BM ha un ruolo fondamentale, come vedremo, nella gestione delle transazioni, per garantire l'integrità del DB a fronte di guasti

In DB2 si possono definire più buffer pool, ma ogni tablespace deve essere associato a un singolo buffer pool

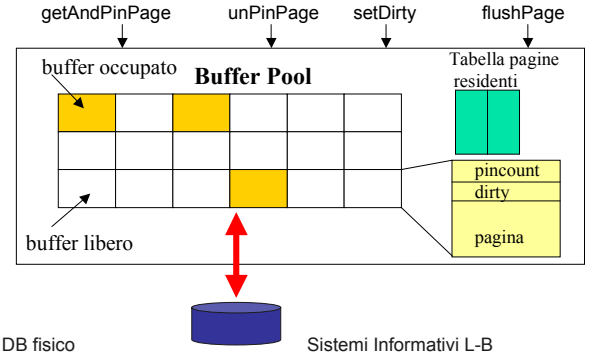
Il Buffer Manager

- A fronte di una richiesta di una pagina, il Buffer Manager (BM) opera come segue:
  - Se la pagina è già in un buffer, viene fornito al programma chiamante l'indirizzo del buffer
  - Se la pagina non è in memoria:
    - Il BM seleziona un buffer per la pagina richiesta. Se tale buffer è occupato da un'altra pagina, questa viene riscritta su disco solo se è stata modificata e non ancora salvata su disco e se nessuno la sta usando
    - A questo punto il BM può leggere la pagina e copiarla nel buffer prescelto, rimpiazzando così quella prima presente

Interfaccia del Buffer Manager

- L'interfaccia che il BM offre agli altri moduli del DBMS ha quattro metodi di base:

  getAndPinPage: richiede la pagina al BM e vi pone un pin ("spillo"), ad indicarne l'uso
  unPinPage: rilascia la pagina e elimina un pin
  setDirty: indica che la pagina è stata modificata, ovvero è dirty ("sporca")
  flushPage: forza la scrittura della pagina su disco, rendendola così "pulita"



Politiche di rimpiazzamento

- Nei sistemi operativi una comune politica adottata per decidere quale pagina rimpiazzare è la LRU (Least Recently Used), ovvero si rimpiazza la pagina che da più tempo non è in uso
- Nei DBMS LRU non è sempre una buona scelta, in quanto per alcune query il "pattern di accesso" ai dati è noto, e può quindi essere utilizzato per operare scelte più accurate, in grado di migliorare anche di molto le prestazioni
- L'hit ratio, ovvero la frazione di richieste che non provocano una operazione di I/O, indica sinteticamente quanto buona è una poltica di rimpiazzamento

  Esempio: come vedremo, esistono algoritmi di join che scandiscono N volte le tuple di una relazione. In questo caso la politica migliore sarebbe la MRU (Most Recently Used), ovvero rimpiazzare la pagina usata più di recente!

- … altro motivo per cui i DBMS non usano (tutti) i servizi offerti dai sitemi operativi…

Organizzazione dei file

- Il modo con cui i record vengono organizzati nei file incide sull'efficienza delle operazioni e sull'occupazione di memoria
- Nel seguito vediamo alcune organizzazioni di base, ovvero:

  Heap file, Sequential file, Hash file

  … e le valutiamo relativamente ad alcune tipiche operazioni
- Per semplicità:
  - Consideriamo record a lunghezza fissa
  - Valutiamo i "costi" come numero di operazioni di I/O, assumendo che ogni richiesta di una pagina comporti un'operazione di I/O
- Per valutare i costi abbiamo comunque bisogno di alcune informazioni…

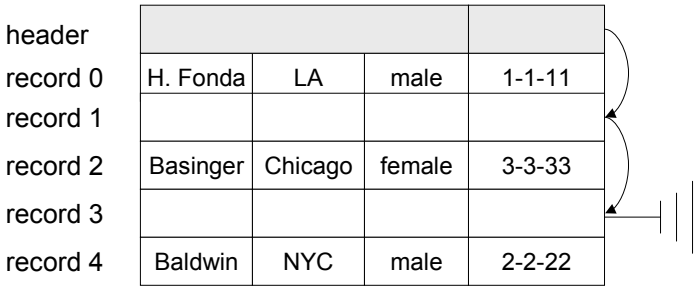
Le statistiche dei cataloghi SQL

- Ogni DBMS mantiene dei **cataloghi**, ovvero delle relazioni che descrivono il DB sia a livello logico che fisico
- I cataloghi che ci interessano in questa fase sono quelli che riportano informazioni statistiche sulle relazioni, in particolare:

SQL Catalog	SQL attribute	Descrizione	Simbolo
SYSSTAT.TABLES	CARD	Numero di tuple nella relazione	NR o NR(table)
SYSSTAT.TABLES	NPAGES	Numero di pagine occupate dalla relazione	NP o NP(table)
SYSSTAT.COLUMNS	COLCARD	Numero di valori distinti dell'attributo	NK o NK(attribute)
SYSSTAT.COLUMNS	LOW2KEY	Secondo valore minore	LK o LK(attribute)
SYSSTAT.COLUMNS	HIGH2KEY	Secondo valore maggiore	HK o HK(attribute)

Heap file

- Detta anche organizzazione seriale, è la più semplice in quanto si caratterizza per l'inserimento di nuovi record alla fine del file
- Se qualche record viene cancellato, per poter riutilizzare lo spazio senza dover scandire tutto il file, è necessario implementare un meccanismo per localizzare velocemente gli spazi liberi (ad es. lista)



Heap file: operazioni e prestazioni

- La tabella riassume i costi (numero di operazioni di I/O) per le operazioni di base (ricerca e modifica)

Operazione	Descrizione	Costo
Ricerca per chiave	La ricerca avviene leggendo sequenzialmente le pagine	NP/2 medio NP massimo NP se non presente
Ricerca per intervallo	Devono comunque essere lette tutte le pagine	NP
Inserimento	Si assume di inserire in fondo al file	2
Cancellazione	Si assume di cancellare un record	C(ricerca) + 1
Aggiornamento	Si assume di aggiornare un record	C(ricerca) + 1

Sequential file

- In un file sequenziale i record vengono mantenuti ordinati secondo i valori di un attributo (o di una combinazione di attributi)
- È evidente che gli inserimenti devono ora avvenire ordinatamente, e quindi normalmente viene lasciato dello spazio libero in ogni pagina (oppure si tollerano record in overflow e poi si riorganizza)

Brighton	A-127	750
Downtown	A-101	500
Downtown	A-101	600
Mianus	A-215	700
Perryridge	A-102	400
Perryridge	A-201	900

Sequential file: operazioni e prestazioni

- Per semplicità si considera che il file sia ordinato sui valori di chiave primaria (o di un'altra chiave)

Operazione	Descrizione	Costo
Ricerca per chiave	Si applica un algoritmo di ricerca binaria per localizzare la pagina che contiene il record	$\lceil \log_2(NP) \rceil$
Ricerca per intervallo	Si leggono le sole pagine con valori di chiave nell'intervallo [L,H]	$C(\text{ricerca}) - 1 + \frac{(H - L) * NP}{HK - LK}$
Inserimento	Si suppone che vi sia spazio per l'inserimento	C(ricerca) + 1
Cancellazione	Si assume di cancellare un record	C(ricerca) + 1
Aggiornamento	Si assume di aggiornare un record	C(ricerca) + 1

Hash file

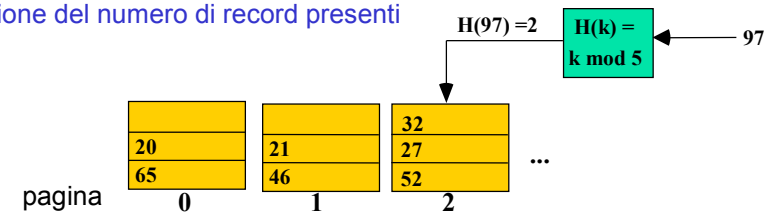
- In un file hash i record vengono allocati in una pagina il cui indirizzo dipende dal valore di chiave del record:

  key → H(key) → page address

- Una comune funzione hash è il resto della divisione intera:

  H(k) = k mod NP

- Si può applicare anche a chiavi alfanumeriche dopo averle convertite
- Anche in questo caso è necessario prevedere la possibilità di record in overflow
- Esistono anche file hash dinamici, in cui lo spazio degli indirizzi varia in funzione del numero di record presenti



Hash file: operazioni e prestazioni

- Si considera che non ci siano record in overflow. A tal fine è importante osservare che il valore di NP è tipicamente superiore di circa il 30-40% a quello di un heap file che memorizza gli stessi dati

Operazione	Descrizione	Costo
Ricerca per chiave	Si accede direttamente alla pagina	1
Ricerca per intervallo	Si leggono tutte le pagine	NP
Inserimento	Si suppone che vi sia spazio per l'inserimento	2
Cancellazione	Si assume di cancellare un record	2
Aggiornamento	Si assume di aggiornare un record	2

Esercizio riassuntivo (1)

- Consideriamo la relazione

```
CREATE TABLE Studenti (
matricola char(10) PRIMARY KEY,
nome char(50),
CF char(16),
indirizzo char(60),
telefono char(12),
datanascita DATE,
sesso char(1) )
```

che consiste di NT = 200000 tuple, memorizzate in pagine di dimensione P = 4 KB su un disco con le seguenti caratteristiche:

- Rotational latency: Tr = 8.5 ms
- Seek time: Ts = 12 ms
- Page transfer time: Tt = 1 ms

Esercizio riassuntivo (2)

A) Dimensione dei record: (10 + 50 + 16 + 60 + 12 + 10 + 1) + 1 = 160 byte
l'ultimo byte viene usato per indicare se il record è stato cancellato

B) Dimensione del file:
nel caso di heap file si riempiono tutte le pagine, nel caso di file sequenziale ogni pagina viene riempita solo all'85%, nel caso di hash file il riempimento è del 70%

In ogni caso si assume la presenza di un page header e di una directory che lasciano a disposizione uno spazio pari a 4000 byte

Organizzazione	Numero pagine (NP)	Dimensione file
Heap	$\lceil 200000 / \lfloor 4000 / 160 \rfloor \rceil = 8000$	31.25 MB
Sequenziale	$\lceil 200000 / \lfloor 0.85*4000 / 160 \rfloor \rceil = 9524$	37.20 MB
Hash	$\lceil 200000 / \lfloor 0.7*4000 / 160 \rfloor \rceil = 11765$	45.96 MB

Esercizio riassuntivo (3)

C) Ricerca per chiave:

nel caso di heap file si assume un'allocazione contigua delle pagine su disco, il che permette di evitare seek (tranne che per la prima lettura)

nel caso di file sequenziale le letture sono "random" in quanto si fa ricerca binaria

Organizzazione	Costo medio di ricerca per chiave
Heap	Ts + Tr + NP/2 * Tt = 4020.5 ms
Sequenziale	$\lceil \log_2(NP) \rceil$ * (Ts + Tr + Tt) = 301 ms
Hash	Ts + Tr + Tt = 21.5 ms

Esercizio riassuntivo (4)

D) Ricerca per intervallo di valori di chiave:

nel caso di file sequenziale si suppone che le chiavi che soddisfano il predicato occupino 900 pagine (≈ 0.1 * NP)

Organizzazione	Costo di ricerca per intervallo
Heap	Ts + Tr + NP * Tt = 8020.5 ms
Sequenziale	$\lceil \log_2(NP) \rceil$ * (Ts + Tr + Tt) + (900 - 1) * Tt = 1200 ms
Hash	Ts + Tr + NP * Tt = 11785.5 ms

Osservazioni conclusive

- Ognuna delle organizzazioni viste presenta pro e contro, e la scelta più adeguata dipende da diversi fattori (occupazione di memoria, tipi di operazioni, frequenza di inserimenti e cancellazioni, ecc.)
- In ogni caso, nessuna organizzazione è in grado di fornire, da sola, prestazioni soddisfacenti a fronte di interrogazioni che non coinvolgono l'attributo (o gli attributi) usato per determinare l'allocazione dei record
  - Per risolvere questi problemi si fa pertanto uso di "**indici**"

Riassumiamo:

- Un DB è fisicamente rappresentato mediante un insieme di file, ognuno dei quali può essere visto a sua volta come una collezione di pagine di una certa dimensione
- Il trasferimento dei dati in memoria centrale è gestito dal Buffer Manager, il quale gestisce un insieme di buffer in cui le pagine possono essere copiate
- I record in un file possono essere organizzati in vario modo (heap sequenziale, hash), e ogni organizzazione ha caratteristiche peculiari per quanto riguarda i costi di esecuzione delle operazioni (tipicamente valutati contando il numero di operazioni di I/O), l'occupazione di memoria e la possibilità di gestire nuovi inserimenti di record