



## Unità e abbreviazioni (1)


grandezza	nome	abbreviazione
$10^{18} \approx 2^{60}$	exa	e, E
$10^{15} \approx 2^{50}$	peta	p, P
$10^{12} \approx 2^{40}$	tera	t, T
$10^9 \approx 2^{30}$	giga, billion	g, b, G, B
$10^6 \approx 2^{20}$	mega	m, M
$10^3 \approx 2^{10}$	kilo	k, K
$10^0 = 2^0$		
$10^{-3}$	milli	m
$10^{-6}$	micro	$\mu$
$10^{-9}$	nano	n
$10^{-12}$	pico	p
$10^{-15}$	femto	f



Il livello fisico di un DB




7




## Unità e abbreviazioni (2)

unità	abbreviazione
bit	b
byte (8 bit)	B
bits per second	bps
bytes per second	Bps
instructions per second	ips
I/O operations per second	I/Ops
transactions per second	tps
bits per inch	bpi
rounds per minute	rpm



Il livello fisico di un DB



8

## Gerarchia di memorie

- ✦ 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.

- Internal - **Processor registers and cache**
- Main - system **RAM and controller cards**
- On-line mass storage - **Secondary storage**
- Off-line bulk storage - **Tertiary and Off-Line storage**

### ✦ 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).

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

Il livello fisico di un DB



9

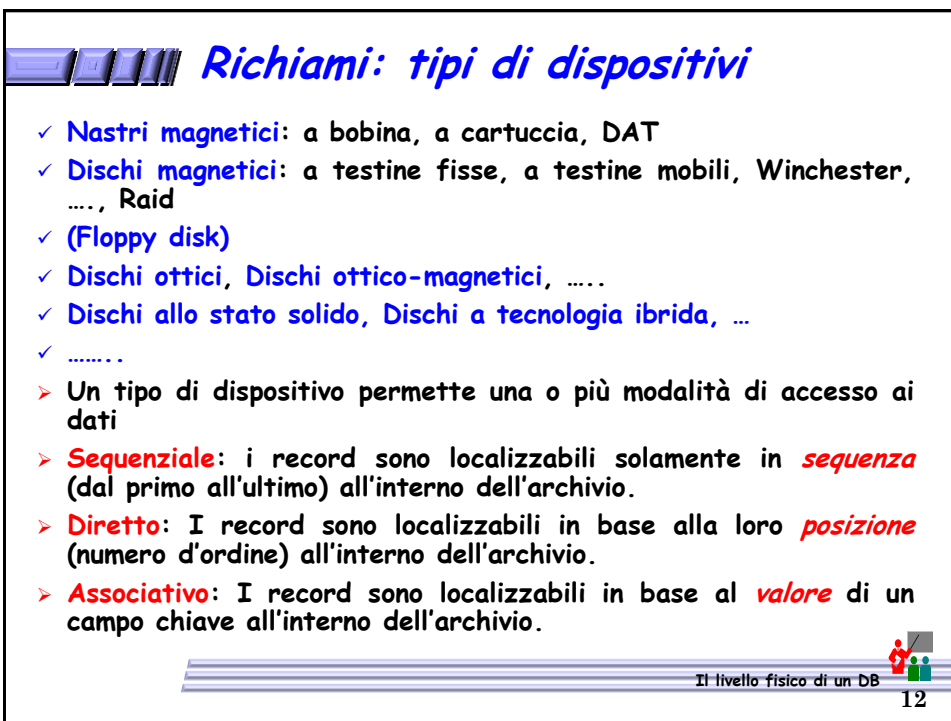
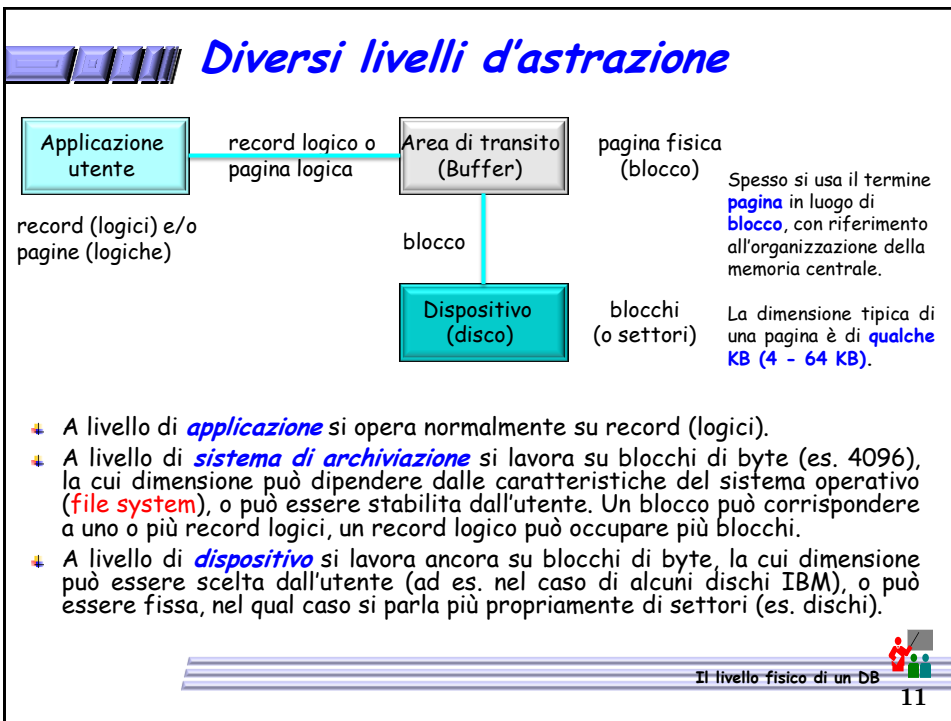
## Conseguenze per i DB

- ✦ 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).
  - 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.
- ✦ Poiché spesso le operazioni di I/O costituiscono 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**

Il livello fisico di un DB



10



## Legge di Moore

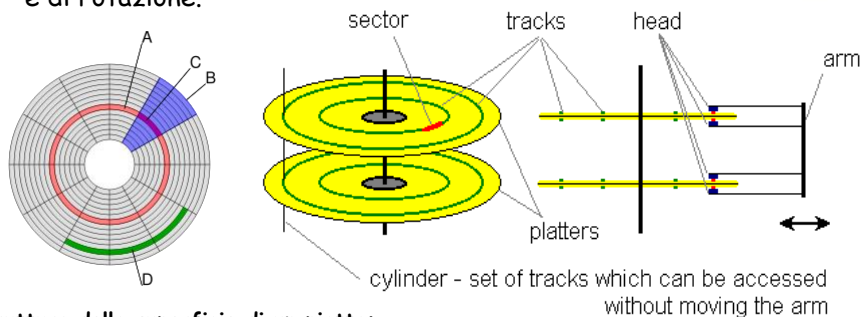
- **Gordon Moore:** "Integrated circuits are improving in many ways, following an exponential curve that doubles every 18 months"
  - velocità dei processori
  - numero di bit integrabili in un chip
  - numero di byte memorizzabili in un hard disk (HD)
- Parametri che **NON** seguono la legge di Moore:
  - tempo d'accesso alle memorie
  - velocità di rotazione degli hard disk
- **Conseguenza:** diventa relativamente sempre più oneroso muovere i dati lungo i livelli della gerarchia

Il livello fisico di un DB

13

## Hard disk drive

- Include organi di registrazione, di posizionamento e di rotazione.



Struttura della superficie di un piatto:

- A) Traccia
- B) Settore
- C) Settore di una traccia
- D) Cluster, insieme di settori contigui

➤Varianti principali:

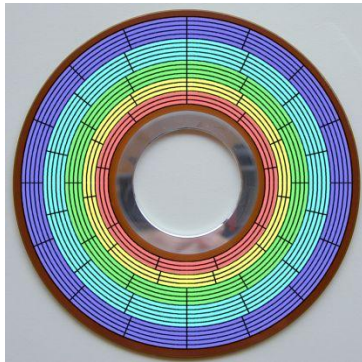
- testine mobili vs testine fisse
- dischi fissi vs asportabili (disk pack)

Il livello fisico di un DB

14

## **Caratteristiche costruttive**

- **Velocità angolare costante** : un numero fisso di settori per traccia implica che **la densità è minore per le tracce più esterne**.
- **Zoned Bit Recording (ZCAV)**: la maggior parte degli hard disk moderni adotta questo schema. **Velocità di rotazione tipiche: 3600, 5400, 7200, 10000, 15000 rpm.**



Le tracce sono raggruppate in zone in base alla loro distanza dal centro, e a ogni zona corrisponde un certo numero di settori per traccia. Muovendosi dal centro verso la periferia le zone contengono un maggior numero di settori per traccia. Ciò consente un'utilizzazione migliore dello spazio.

**La velocità di trasferimento dati è più elevata nelle zone più esterne. In generale, la densità presenta ancora valori più elevati al centro.**

necessario mapping da geometria logica BIOS a geometria fisica del disco.

Il livello fisico di un DB



15

## **Prestazioni HD**

- **Le prestazioni possono essere classificate in:**
- **Interne**
  - fattori che influenzano le prestazioni interne:
    - **Caratteristiche meccaniche**
    - **Tecniche di memorizzazione e codifica dei dati**
    - **Caratteristiche del controller e cache**
- **Esterne**
  - fattori che influenzano le prestazioni esterne:
    - **Tipo di interfaccia**
    - **Architettura del sottosistema di I/O**
    - **File system**

Il livello fisico di un DB



16



## Prestazioni interne (1)


- Definizione (non universalmente rispettate):
- **Access Time** = **Command Overhead Time** +  
**Seek Time** + **Settle Time** + **Latency**

indica il tempo impiegato per raggiungere le informazioni di interesse.

**Command Overhead Time**: si riferisce al tempo necessario a impartire comandi al drive, è dell'ordine di 0.5 ms e può essere trascurato.


## Seek Time & Settle Time


- Rappresentano rispettivamente il tempo necessario a spostare le testine sul cilindro desiderato e il tempo necessario a stabilizzare le testine: un'operazione di seek comporta le seguenti fasi:
    - **speedup** (accelerazione)
    - **coast** (velocità costante)
    - **slowdown** (rallentamento)
    - **settle** (stabilizzazione)
- Seek Time** (bracketed for speedup, coast, slowdown)
- Settle Time** (pointed to by settle)
- Il **Settle Time** è dell'ordine di 0.1 ms, trascurabile rispetto al **Seek Time**. Alcuni costruttori inglobano nel **Seek Time** anche il **Settle Time**.




## Seek Time

- Per **seek brevi** (meno di 200-400 cilindri) prevale la fase di accelerazione, il cui tempo è proporzionale alla radice del numero di cilindri attraversati.
- Per **seek lunghe** prevale la seconda fase a velocità costante, e il tempo risulta proporzionale al numero di cilindri attraversati più una costante.
- I costruttori forniscono in genere:
  - Average Seek** : 3 ms (high-end server drives)-12 ms (mobile drives) e di solito si riferisce a letture
  - Track-to-Track** : 1 ms
  - Full-stroke** : 15-20 ms (dalla traccia più interna alla traccia più esterna)
- Il seek time per scritture è superiore (di circa 1 ms) rispetto al seek time per letture



Il livello fisico di un DB
 


19




## Average Seek Time

- Normalmente calcolato considerando la richiesta di ogni traccia ugualmente probabile. Ciò equivale, adottando un modello lineare, a considerare il seek time corrispondente a una distanza pari a 1/3 dei cilindri (nel caso di ugual numero di settori per traccia).
- Sia  $n=N_{Cyl}$  il numero di cilindri, e siano  $X$  e  $Y$ , rispettivamente, il cilindro corrente e quello di destinazione. Per la distanza  $d = |X-Y|$  tra i due cilindri (**seek distance**) si hanno  $n^2$  possibilità, così suddivise:

d	0	1	2	...	d	...	n-1
n. casi	n	2 (n-1)	2 (n-2)	...	2 (n-d)	...	2



Il livello fisico di un DB
 

20



## Average seek distance

$$E[d] = \sum_{d=0}^{n-1} d \times \frac{2 \times (n-d)}{n^2} = \frac{2}{n} \sum_{d=1}^{n-1} d - \frac{2}{n^2} \sum_{d=1}^{n-1} d^2$$

**Poiché:**  $\sum_{d=1}^{n-1} d^2 = \frac{(n-1) \times (n-1/2) \times n}{3}$

$$\begin{aligned} E[d] &= \frac{2}{n} \frac{(n-1) \times n}{2} - \frac{2}{n^2} \frac{(n-1) \times (n-1/2) \times n}{3} \\ &= (n-1) - \frac{(n-1) \times (2 \times n - 1)}{3 \times n} \\ &= (n-1) \left[ \frac{3 \times n - (2 \times n - 1)}{3 \times n} \right] \\ &= \frac{n^2 - 1}{3 \times n} \approx \frac{n}{3} \end{aligned}$$

Il livello fisico di un DB



21



## Un modello per il seek time

$$t_s(d) = a \times \sqrt{d-1} + b \times (d-1) + c \quad 0 < d < N\_Cyl$$

**Il primo termine modella le fasi di accelerazione/decelerazione;  
il secondo la fase a velocità costante;  
il terzo corrisponde al minimo tempo di seek.**

**Attenzione:**  
 $E(t_s(d)) \neq t_s(E(d))$

$$a = (-10 \times t_s(\min) + 15 \times t_s(\text{aver}) - 5 \times t_s(\max)) / (3 \times \sqrt{N\_Cyl})$$

$$b = (7 \times t_s(\min) - 15 \times t_s(\text{aver}) + 8 \times t_s(\max)) / (3 \times N\_Cyl)$$


$$c = t_s(\min)$$

**valide per  $N\_Cyl > 200$**

Il livello fisico di un DB



22




# Latency

$$(60/\text{Spindle Speed}) * 0.5 * 1000$$


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

**Rotational Latency**  
 rappresenta il tempo necessario affinché il settore interessato dall'operazione passi sotto la testina

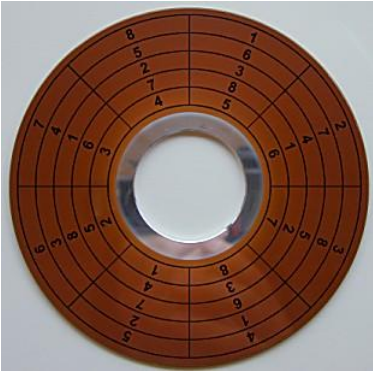
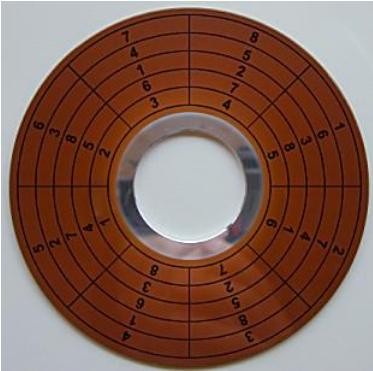


Il livello fisico di un DB


23



# Cylinder and head skew

Due piatti adiacenti (o due superfici dello stesso piatto). A sinistra il concetto di cylinder skew (offset di 3 settori tra tracce consecutive); a destra il concetto di head skew (offset di 1 settore tra piatti adiacenti). Rispettivamente per compensare i ritardi **cylinder switch time** e **head switch time**.



Il livello fisico di un DB

24

## Prestazioni interne (2)

### ➤ Internal Media Transfer Rate (media rate)

rappresenta la velocità massima alla quale il drive può leggere o scrivere bit, espressa in Mbit/sec or Mb/s. Tipicamente dell'ordine di qualche centinaio di Mb/s, si riferisce alla velocità con cui si trasferiscono bit **dai** (sui) piatti **sulla** (dalla) cache del controller.

### ➤ Si può stimare (in MB/sec) come:

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

### ➤ Sustained Transfer Rate

include overhead per **head switch time** e **cylinder switch time**, misurata in Mbytes/sec (MB/s).

 Il livello fisico di un DB



25

## Read-ahead, Command Queuing

### ➤ Read-ahead

Il controllore "anticipa" le richieste di lettura, caricando nella sua cache il contenuto di una o più tracce, e andando poi a verificare se una richiesta può essere soddisfatta dalla cache. Se la cache è "segmentata", è possibile mantenere nella cache dati da zone diverse del disco.

### ➤ Command Queuing

I drive SCSI permettono la gestione di più richieste in contemporanea da parte del controllore, che può quindi decidere autonomamente in quale ordine sia meglio servirle.

 Il livello fisico di un DB



26

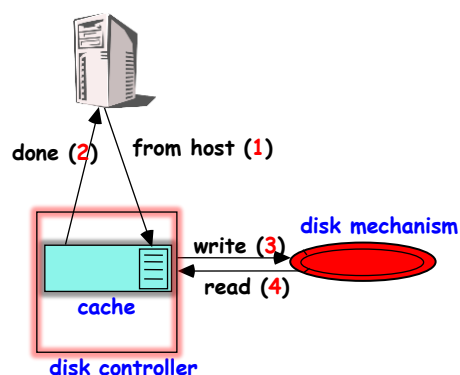
## **write-back vs write-through**

- Con la tecnica **write-through** la scrittura si considera eseguita solo dopo che i dati sono stati effettivamente scritti su disco.
- Con la tecnica **write-back** il controller segnala che la richiesta di scrittura è stata soddisfatta quando i dati sono stati scritti nella sua cache.
- Poiché la cache è normalmente volatile, la tecnica di write-back non è esente da rischi. I problemi possono nascere nel caso di errori (transitori) nella circuiteria del controllore, cadute di tensione, settori danneggiati, ecc.
- Esistono diversi accorgimenti per garantire che una scrittura avvenga realmente.



## **Read after write**

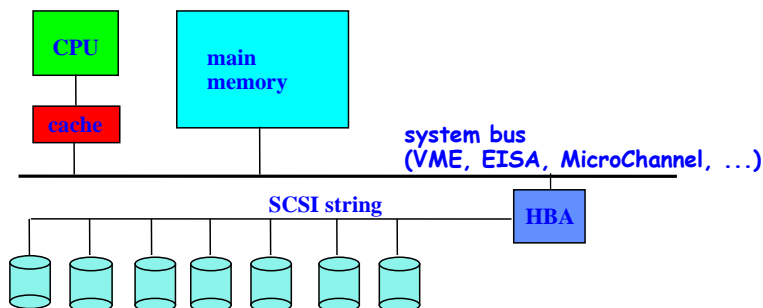
- Il controller, dopo aver scritto un blocco, aspetta una rotazione completa e rilegge il blocco. In caso di discordanza ripete il ciclo di scrittura, eventualmente su altro settore di disco.



## I/O Controller

Il percorso disco-memoria centrale include dispositivi e schemi di interconnessione variamente caratterizzati.

**I/O controller:** gestisce uno o più dispositivi e fornisce l'interfaccia con il bus di sistema (o con quello di I/O nel caso di calcolatori a bus multipli). È anche detto **Host-Bus Adapter (HBA)** e, nel caso di grossi calcolatori, I/O Processor o **I/O Channel Processor**.



Il livello fisico di un DB



29

## DMA - Direct Access Memory

Molti controllori sono di fatto processori specializzati che sollevano (parzialmente) la CPU dalla gestione diretta del trasferimento (byte a byte) dei dati dal buffer del controllore alla memoria centrale, o viceversa. Con la tecnica detta di **DMA**, il controllore (canale) accede direttamente alla memoria e gestisce autonomamente il trasferimento dei dati da/per la memoria centrale.

Il trasferimento è in generale avviato dalla CPU che comunica al canale:

- tipo di operazione sul dispositivo: lettura (**scrittura**)
- indirizzo iniziale di memoria dove scrivere (**leggere**)
- indirizzo dell'unità esterna dove leggere (**scrivere**)
- dimensione dei dati da trasferire

Al termine dell'operazione il canale invia un'interruzione alla CPU.

Il livello fisico di un DB



30

## Il DB fisico

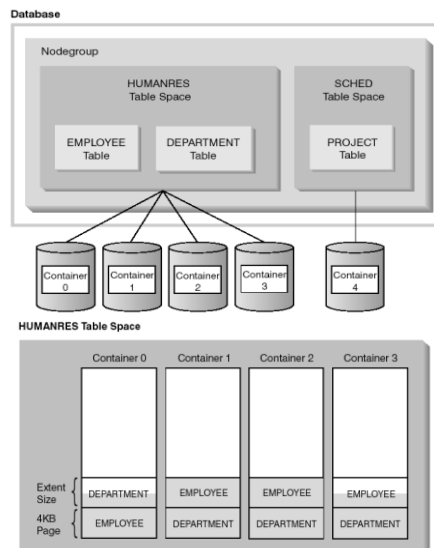
- A livello fisico un DB consiste di un insieme di file, ognuno dei quali viene visto come una collezione di pagine, di dimensione fissa (es: 4 KB)
- Ogni pagina memorizza più record (corrispondenti alle tuple logiche)
- A sua volta un record consiste di più campi, di lunghezza fissa e/o variabile, che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

31

## 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.**



Il livello fisico di un DB

32





## Tipi di tablespace in DB2

- ✦ 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** sono **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 migliori prestazioni nel caso di DB di grandi dimensioni

Il livello fisico di un DB



33



## Attributi dei tablespace di DB2

- ✦ 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 sono usati dall'ottimizzatore

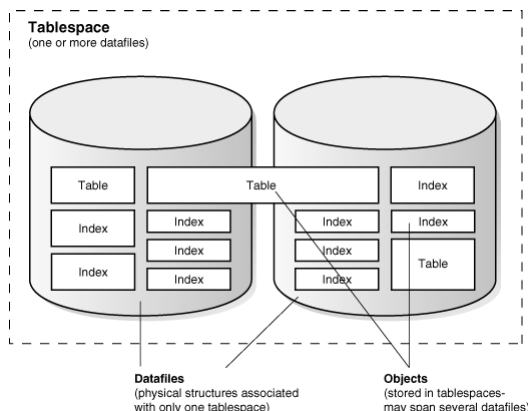
Il livello fisico di un DB



34

## Modello di memorizzazione di Oracle

- Oracle memorizza i dati dal punto di vista logico in **tablespace** e fisicamente in **datafile** associati con il corrispondente tablespace.



Un database Oracle consiste in una o più unità logiche di memorizzazione dette **tablespace**.

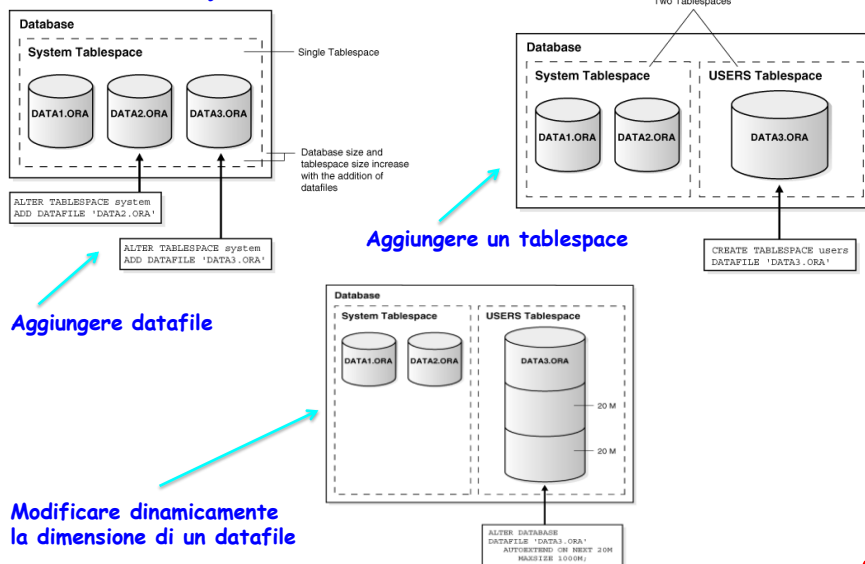
Ogni tablespace consiste di uno o più **datafile** che sono strutture fisiche conformi al sistema operativo su cui Oracle è eseguito.

Il livello fisico di un DB



35

## Espandere i dati in Oracle



Il livello fisico di un DB



36

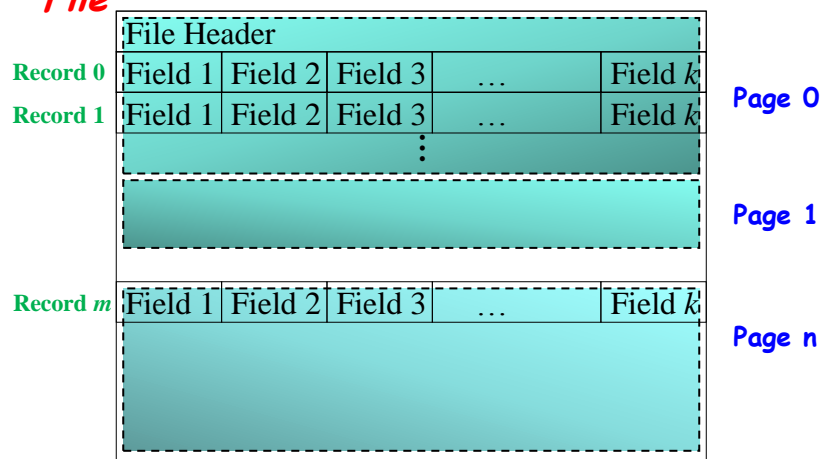
## Perché non usare sempre il file system?

- Le prestazioni di un DBMS dipendono fortemente dall'organizzazione fisica dei dati su disco.
- Intuitivamente, l'allocazione dei dati dovrebbe mirare a ridurre i tempi di accesso ai dati e, a tale scopo, bisogna conoscere come (*logicamente*) i dati dovranno essere elaborati e quali sono le *correlazioni logiche* tra i dati.
- 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

**File**

**Schema di riferimento (semplificato)**



## Rappresentazione dei valori

- Per ogni tipo di dati di SQL è definito un formato di rappresentazione, ad esempio:
- **Stringhe a lunghezza fissa: CHAR(n)**
  - si allocano n byte, eventualmente usando un carattere speciale per valori lunghi meno di n;
  - Esempio: se A è CHAR(5), 'cat' è memorizzato come cat␣␣
- **Stringhe a lunghezza variabile: VARCHAR(n)**
  - si allocano m+p byte, con m ( $\leq n$ ) byte usati per gli m caratteri effettivamente presenti e p byte per memorizzare il valore di m (per  $n \leq 254$  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, ...

Il livello fisico di un DB

39

## 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 )
```



Il livello fisico di un DB

40



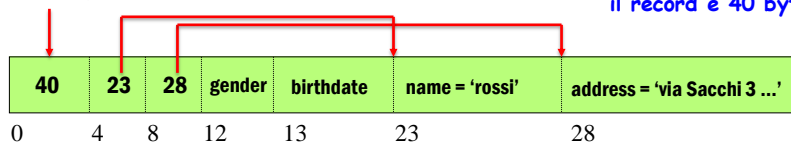
## Record a lunghezza variabile

- 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 length



Il livello fisico di un DB

41



## Record Header

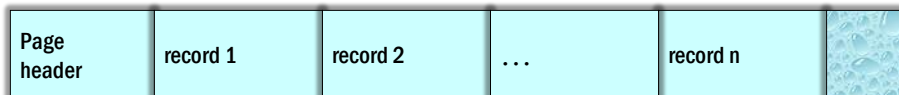
- ✦ In generale ogni record include un **header** che, oltre alla lunghezza del record, può contenere:
  - ✦ l'identificatore della relazione cui il record appartiene;
  - ✦ l'identificatore univoco del record nel DB;
  - ✦ un **timestamp** che indica quando il record è stato inserito o modificato l'ultima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

Il livello fisico di un DB

42

## Organizzare i record in pagine

- Normalmente la dimensione di un record è (molto) minore di quella di una pagina.
  - Esistono tecniche particolari 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 a cui le tuple nella pagina appartengono, ecc.
- Normalmente un record è contenuto interamente in una pagina, quindi si può avere uno spreco di spazio.

Il livello fisico di un DB



43

## Un semplice esempio

- Nel caso visto prima, con record di lunghezza fissa pari a **296 byte**, si supponga di usare pagine di dimensione  $P = 4 \text{ KB} = 4096 \text{ byte}$ .
- 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**, nel caso peggiore nell'ipotesi di allocazione non contigua delle pagine su disco.

Il livello fisico di un DB



44

## Organizzazione a slot delle pagine

✚ **Formato tipico di una pagina in un DBMS**

- ✚ La **directory** contiene un **puntatore per ogni record nella pagina**.
- ✚ Con questa soluzione l'identificatore di un record (**RID o TID**) 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 nella pagina senza modificare il RID

Il livello fisico di un DB 45

## 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 indirizione.
- Avere molti record in overflow ovviamente porta a un degrado delle prestazioni, per cui si può periodicamente rendere necessario **riorganizzare il file**.

Il livello fisico di un DB 46

## *Lettura e scrittura di pagine*

- ✦ La lettura di una tupla richiede che la pagina corrispondente sia prima trasferita 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)**.
- ✦ BM è chiamato in causa anche nel caso di scritture, ovvero quando bisogna riscrivere su disco una pagina modificata.
- ✦ BM ha un ruolo fondamentale nella gestione delle transazioni, per garantire l'integrità del DB a fronte di guasti.
- ✦ Esempio: in DB2 si possono definire più buffer pool, ma ogni tablespace deve essere associato a un singolo buffer pool.

Il livello fisico di un DB



47

## *Il Buffer Manager*

- ✦ A fronte di una **richiesta di una pagina**, il Buffer Manager opera come segue (ipotesi dimensione buffer = dimensione pagina):
  - ✦ Se la pagina è già in un buffer, si restituisce al programma chiamante l'indirizzo del buffer.
  - ✦ Se la pagina non è in memoria:
    - ✦ BM seleziona un buffer per la pagina richiesta. Se tale buffer è già occupato da un'altra pagina (**rimpiazzamento**), 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 BM può leggere la pagina e copiarla nel buffer prescelto, rimpiazzando così quella prima presente

Il livello fisico di un DB



48



## Interfaccia del Buffer Manager

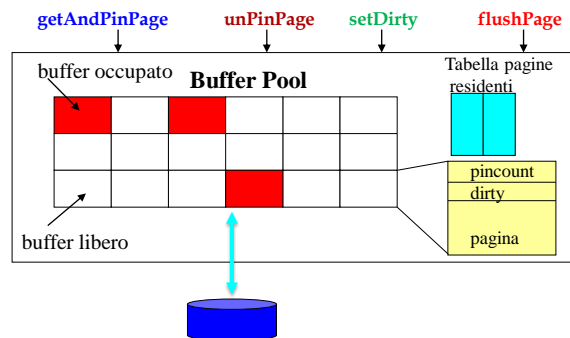
- L'interfaccia che BM offre agli altri moduli del DBMS ha quattro metodi di base:

**getAndPinPage:** richiede la pagina al BM e vi pone un **pin** ("spillo"), a 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"



49

## Politiche di rimpiazzamento

- Nei sistemi operativi una comune politica adottata per decidere quale pagina rimpiazzare è **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 molto le prestazioni.
- Il valore **hit ratio**, ovvero la frazione di richieste che non provocano una operazione di I/O, indica sinteticamente quanto buona è una politica di rimpiazzamento.

Esempio: 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 sistemi operativi...



Il livello fisico di un DB

50

## Organizzazione dei file

- ✦ Il modo con cui i record sono organizzati nei file incide sull'efficienza delle operazioni e sull'occupazione di memoria.
- ✦ Nel seguito vedremo alcune organizzazioni di base e le valuteremo relativamente ad alcune tipiche operazioni.
- ✦ Per semplicità:
  - ✦ considereremo **record a lunghezza fissa**
  - ✦ valuteremo i "costi" solo in termini di **numero di operazioni di I/O**, assumendo che ogni richiesta di una pagina comporti un'operazione di I/O.
- ✦ Per valutare i costi è necessario comunque disporre di alcune informazioni...

Il livello fisico di un DB

51

## Le statistiche dei cataloghi SQL

- Ogni DBMS mantiene **cataloghi**, ovvero **relazioni che descrivono il DB sia a livello logico che fisico**. I cataloghi riportano **informazioni statistiche sulle relazioni**:

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)

- e statistiche per ogni indice costruito, in particolare:

SQL Catalog	SQL attribute	Descrizione	Simbolo
SYSSTAT.INDEXES	NLEAF	Numero di pagine foglia (ossia del livello più basso) dell'indice	NL o NL(index)
SYSSTAT.INDEXES	NLEVELS	Numero di livelli dell'indice	h o h(index)
SYSSTAT.INDEXES	FULLKEYCARD	Numero di valori distinti di chiave nell'indice	NK o NK(index)

Il livello fisico di un DB

52



## Sommario:

- ✦ 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 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.