



# Basi di Dati

## Modulo Tecnologie

### Ottimizzazione di interrogazioni (1/2)

Dr. Sara Migliorini

# Osservazione

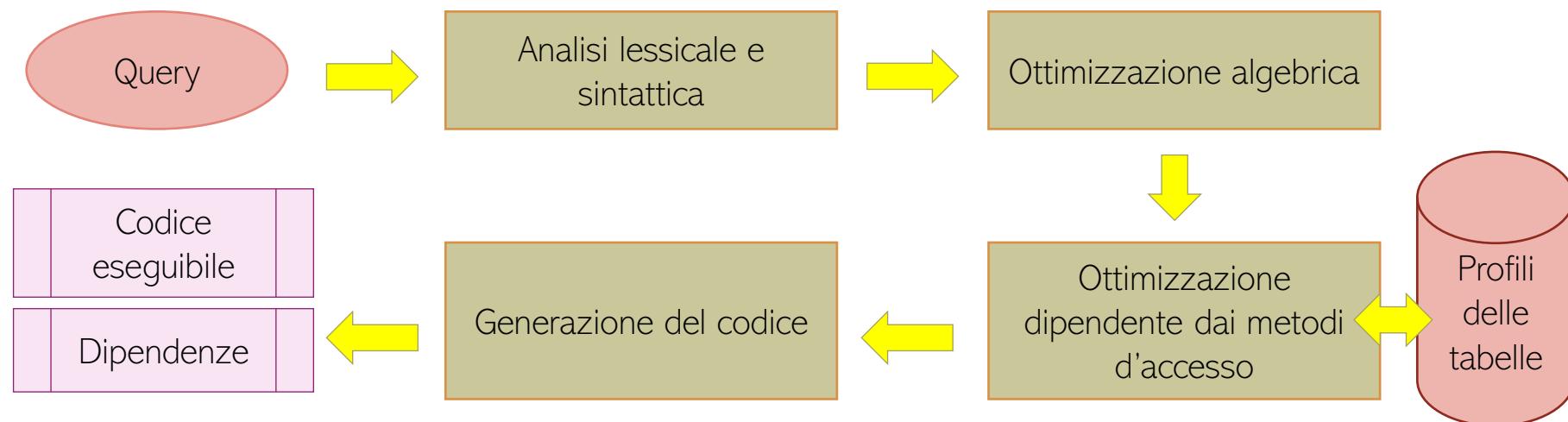
---

- Ogni interrogazione sottomessa al DBMS viene espressa in un **linguaggio dichiarativo** (ad esempio SQL).
- È quindi necessario trovare un equivalente espressione in **linguaggio procedurale** (ad esempio in algebra relazionale) per generare un piano di esecuzione.
- L'**espressione algebrica** va ottimizzata rispetto alle caratteristiche del DBMS a livello fisico (metodi d'accesso disponibili) e della base di dati corrente (statistiche del dizionario dei dati).

# Ottimizzazione

---

- “Compilazione” di un’interrogazione
- Analisi lessicale e sintattica
- Ottimizzazione algebrica (indipendente dal modello di costo)
- Ottimizzazione basata sui costi di esecuzione



# Ottimizzazione algebrica

---

L'ottimizzazione algebrica si basa fondamentalmente sulle regole di ottimizzazione già note dell'algebra relazionale:

- Anticipo delle selezioni (selection push)
- Anticipo delle proiezioni (projection push)

# Ottimizzazione dipendente dai metodi di accesso

---

Operazioni tipiche di accesso supportate dai DMBS:

- Scansione (scan) delle tuple di una relazione
- Ordinamento di un insieme di tuple
- Accesso diretto alle tuple attraverso indice
- Diverse implementazioni del join

# Scansione

---

- Una operazione di scan opera contestualmente altre operazioni. Varianti possibili:
  - scan + **proiezione** senza eliminazione di duplicati
  - scan + **selezione** in base ad un predicato semplice
  - scan + inserimento/cancellazione/modifica
- Costo di una scansione sulla relazione R: **NP(R)**
- **NP(R)** = Numero Pagine dati della relazione R.

# Ordinamento

---

- L'ordinamento viene utilizzato per:
  - ordinare il risultato di un'interrogazione (clausola order by),
  - eliminare duplicati (select distinct),
  - raggruppare tuple (group by).
- Ordinamento su memoria secondaria: **Z-way Sort-Merge**
  - **Sort interno**: si leggono una alla volta le pagine della tabella; le tuple di ogni pagina vengono ordinate facendo uso di un algoritmo di sort interno (es. QuickSort); ogni pagina così ordinata, detta anche “run”, viene scritta su memoria secondaria in un file temporaneo.
  - **Merge**: applicando uno o più passi di fusione, le run vengono unite, fino a produrre un'unica run.

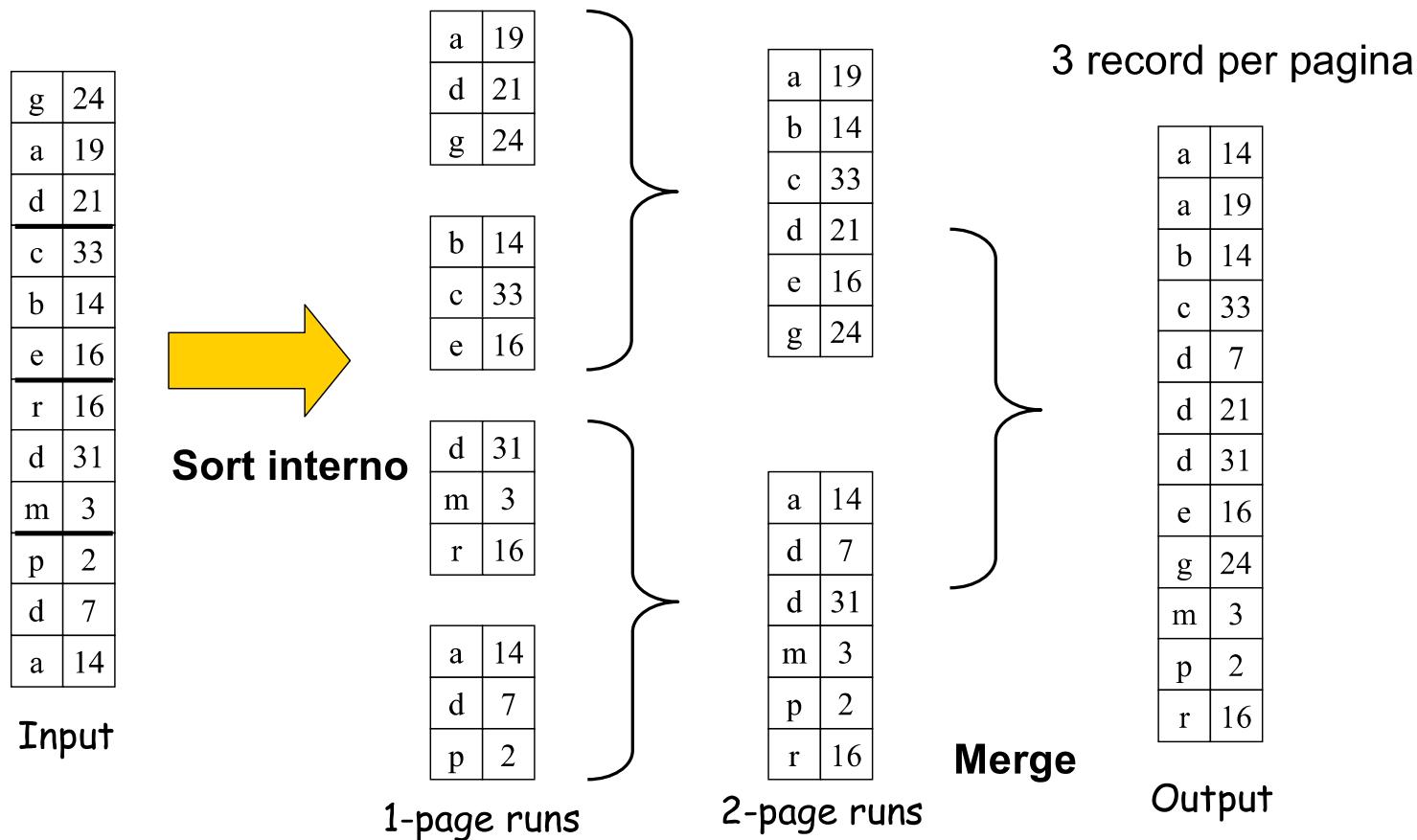
# Z-way Sort-Merge

---

## Esempio

- Supponiamo di dover ordinare un input che consiste di una tabella di  $NP$  pagine e di avere a disposizione solo  $NB$  buffer in memoria centrale, con  $NB < NP$ .
- Per semplicità consideriamo il caso base a due vie ( $Z = 2$ ), e supponiamo di avere a disposizione solo 3 buffer in memoria centrale ( $NB = 3$ ).

# Z-way Sort-Merge: esempio

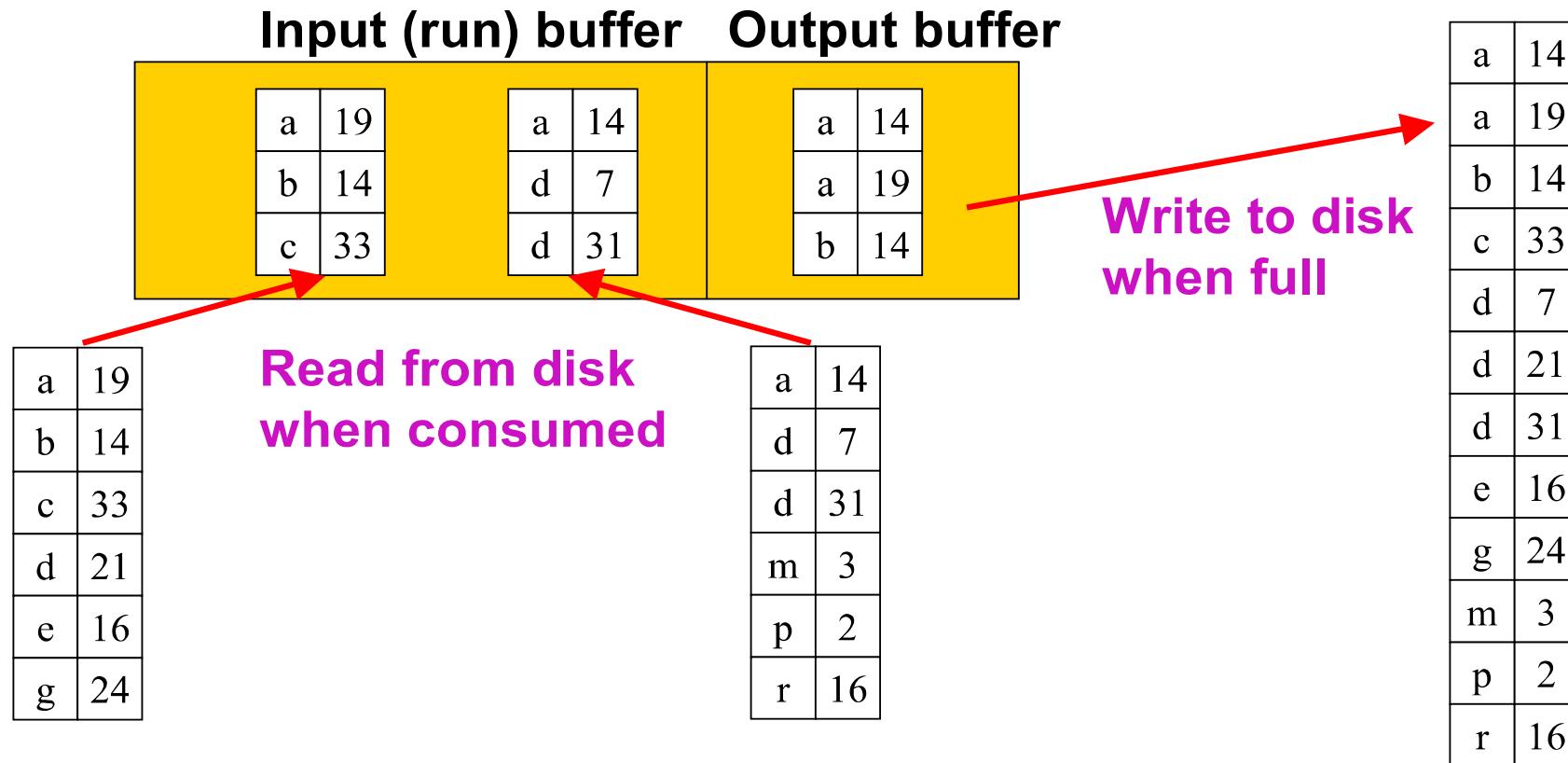


# Z-way Sort-Merge

---

- Dopo la fase di “sort interno”, nel caso base  $Z = 2$  si fondono 2 run alla volta.
- Con  $NB = 3$ , si associa un buffer a ognuna delle run, il terzo buffer serve per produrre l’output, 1 pagina alla volta.
- Si legge la prima pagina da ciascuna run e si genera la prima pagina dell’output; quando tutti i record di una pagina di run sono stati consumati, si legge un’altra pagina della run.

# Z-way Sort-Merge



# Z-way Sort-Merge: costo

---

- Consideriamo come costo solo il numero accessi a memoria secondaria.
- Nel caso base  $Z = 2$  e con  $NB = 3$  si può osservare che:
  - Nella fase di sort interno si leggono e si riscrivono  $NP$  pagine
  - Ad ogni passo di merge si leggono e si riscrivono  $NP$  pagine
  - Il numero di passi di merge («fusione») è pari a:

$$\lceil \log_2(NP) \rceil$$

in quanto ad ogni passo il numero di run si dimezza ( $Z = 2$ ).

- Il costo complessivo è pertanto pari a:


$$2 \times NP \times (\lceil \log_2 NP \rceil + 1)$$

lettura+scrittura

# Accesso diretto via indice

---

- Interrogazioni che fanno uso dell'indice:
- Selezioni con condizione atomica di uguaglianza ( $A = v$ ):
  - richiede indice hash o B+-tree.
- Selezioni con condizione di range ( $A \geq v1 \text{ AND } A \leq v2$ ):
  - richiede indice B+-tree.
- Selezioni con condizione costituita da una **congiunzione** di condizioni di uguaglianza ( $A = v1 \text{ AND } B = v2$ ):
  - in questo caso si sceglie per quale delle condizioni di uguaglianza utilizzare l'indice; la scelta ricade sulla condizione più selettiva. L'altra si verifica direttamente sulle pagine dati.
- Selezioni con condizione costituita da una **disgiunzione** di condizioni di uguaglianza ( $A = v1 \text{ OR } B = v2$ ):
  - in questo caso è possibile utilizzare più indici in parallelo, facendo un merge dei risultati eliminando i duplicati oppure, se manca anche solo uno degli indici, è necessario eseguire una scansione sequenziale.