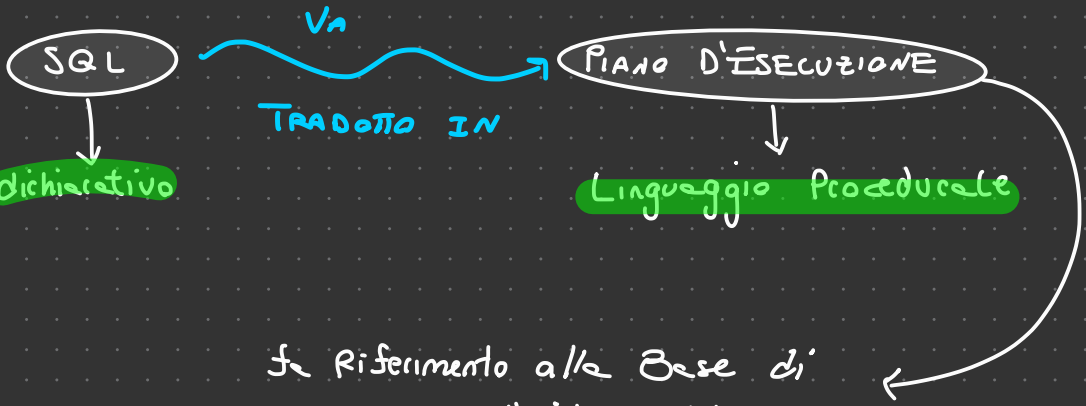
MODULO OTTIMIZZATORE:

Primo che si Attiva Quando Arriva un INTERROGAZIONE.



Si Parte da una QUERY, che verrà ANALIZZATA lessicalmente ed Sintatticamente e Solo dopo un OTTIMIZZAZIONE ALGEBRICA.
Dopo di che viene fatta un OTTIMIZZ. che dipende dai METODI D'ACESSO (se ci Sono Indici, Guardo Statistiche, ...) e Solo dopo passo Generare il CODICE/PIANO D'ESECUZIONE

OTTIMIZZAZIONE ALGEBRICA:

- Anticipo PROIEZIONI & SELEZIONI

OTTIMIZ. METODI D'ACCESSO:

- Quali Sono le operazioni tipiche di Accesso Supportate da DBMS:
  1. SCANSIONE SEQ
  2. ORDINAMENTO TUPLE

3. Accesso Diretto Tramite Indice
4. Join

1. Seq. Scan consiste nel leggere una riga alla volta la mia TABELLA.

Spesso fatta in maniera "parallela" con altre operazioni:

- Scan + Proiezione (senza eliminazione duplicati)
- Scan + Selezione (con predicato semplice)
- Scan + INSERT/DELETE/UPDATE

Costo? (#Accessi in memoria secondaria $\Leftrightarrow$ #Pagine leggo/scrivo)

Numero di Pagine ← NP(R) → Relazione/Tabella

2. Ordinamento, Quando ho:
   - ORDER BY (Ordinare)
   - SELECT DISTINCT (devo Elim. i duplicati)
   - GROUP BY (Raggruppare tuple)

Che Algoritmo Uso? Z-Way Sort Merge

Devo fare un ORDINAMENTO di DATI in Memoria Secondaria con l'IPOTESI che NON Possono Essere CONTENUTI in Memoria Centrale

Z-Way Sort Merge ha 2 fasi:

① ORDINAMENTO INTERNO ⇒ Legge 1 pagina Alla Volta, ordina le TUPLE presenti Nella pagina con un Algoritmo Classico (QUICK-SORT).

Ogni pagina Ordinata Viene Salvata in MEM. SECONDARIA in un file TEMPORANEO. → Chiamata RUN

② FASE DI MERGE ⇒ Si Applicano i Passi di fusione dove le RUN Vengono Unite fino a PRODURRE un UNICA Run.

Z è il #RUN che Si riescono ad UNIRE in un UNICO Passo di MERGE

ESEMPIO:

Z = 2, NB = 3 e NP(R) > NB, NP(R) = 4

G	24	
A	19	
D	21	1
C	33	
B	14	
E	16	2
R	16	
D	31	
M	3	3
P	2	
D	7	
A	14	4

① SORT INTERNO ⇒

A	19
D	21
G	24

B	14
C	33
E	16

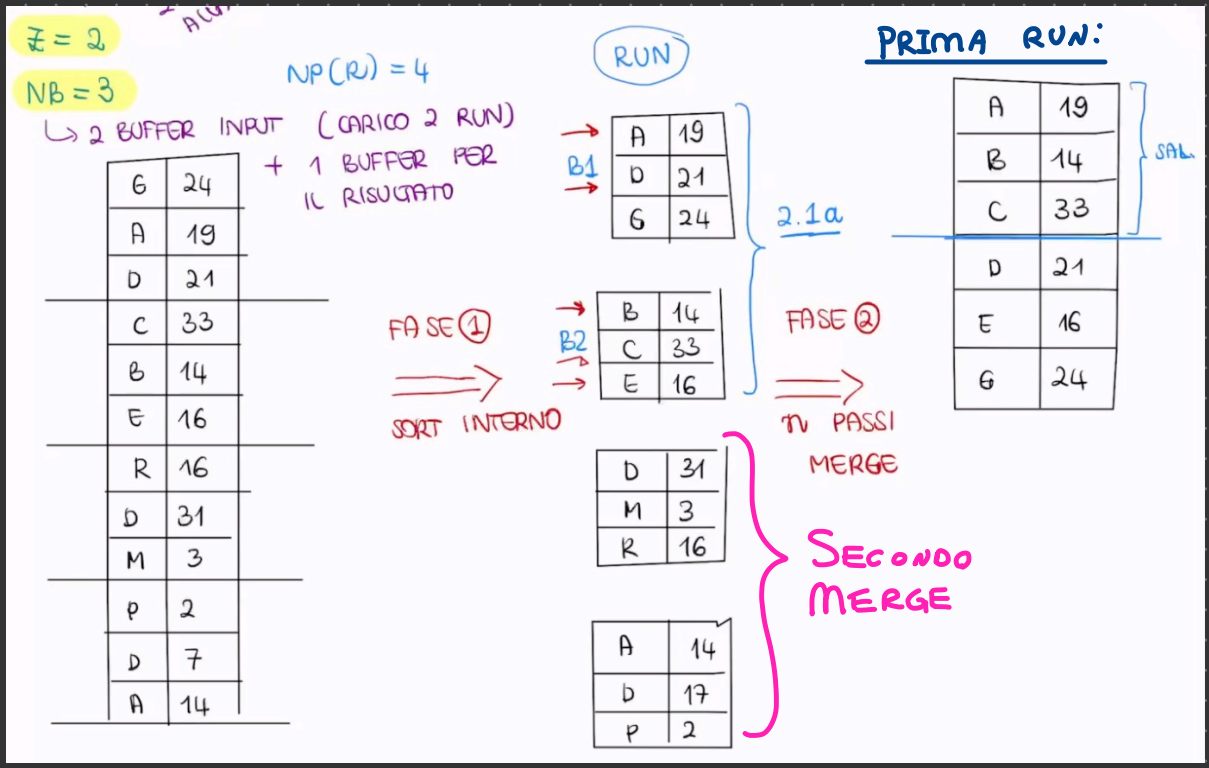
D	31
M	3
R	16

A	7
D	2
P	14

② FASE DI MERGE ⇒ N passi di Merge, unisco 2 RUN alla Volta. (Visto che NB = 3 e 1 mi Serve per OUTPUT)

NB = 3 Quindi 2 BUFFER DI INPUT e 1 OUTPUT

↳ Carico 2 RUN



G	24
A	19
D	21
C	33
B	14
E	16
R	16
D	31
M	3
P	2
D	7
A	14

A	19
D	21
G	24

B	14
C	33
E	16

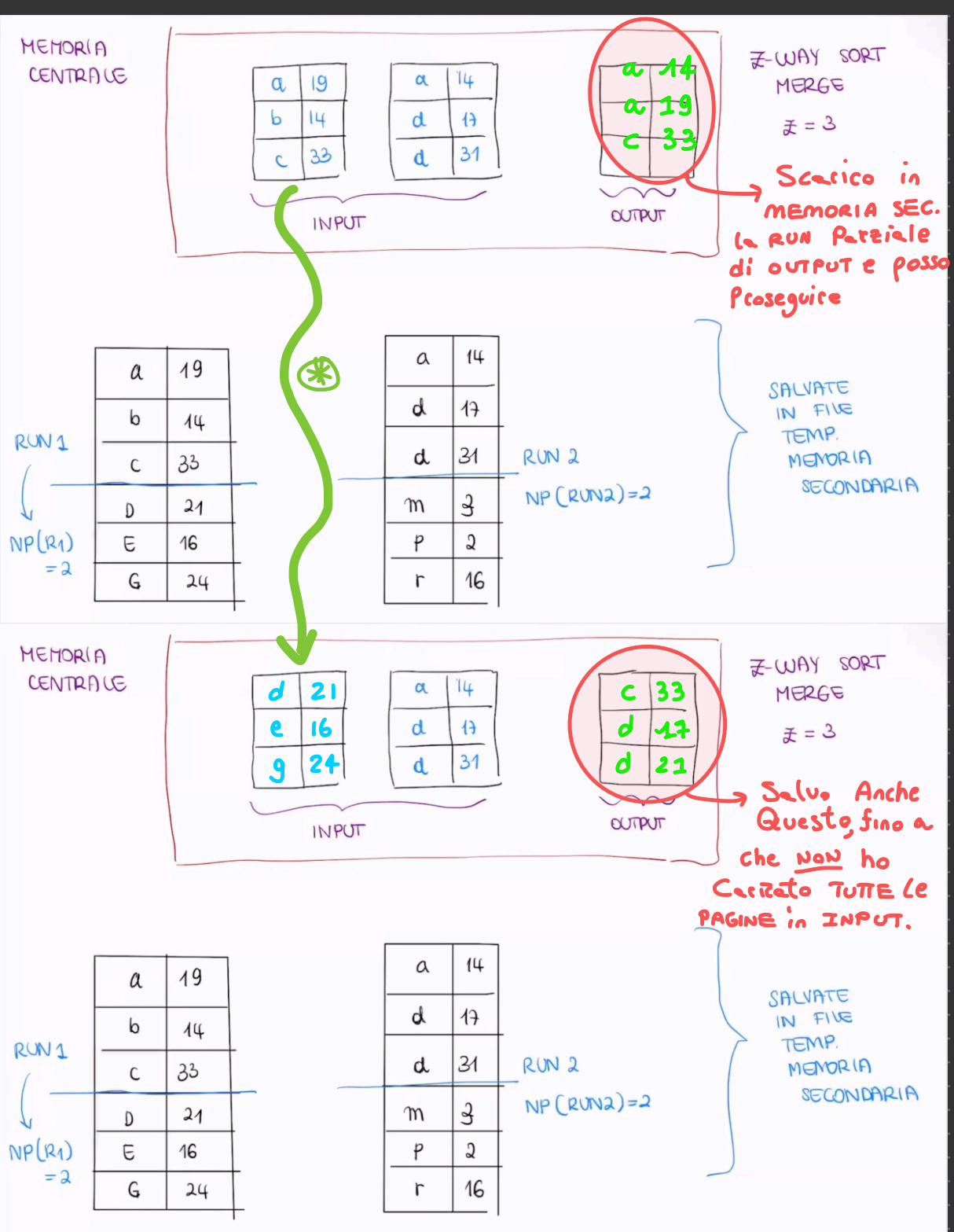
A	19
B	14
C	33
D	21
E	16
G	24

D	31
M	3
R	16

A	14
D	17
P	2

Faccio il Secondo MERGE e Salvo l'OUTPUT in Un BUFFER di OUTPUT, fino ad Arrivare ad:

R	P	M	G	E	D	D	D	C	B	A	A
16	2	3	24	16	31	21	7	33	14	19	14


MEMORIA CENTRALE
a 19
b 14
c 33
a 14
d 17
d 31
INPUT
a 14
a 19
c 33
OUTPUT
2-WAY SORT MERGE
2 = 3
Scarico in MEMORIA SEC. la RUN Parziale di OUTPUT e posso Proseguire
a 19
b 14
c 33
D 21
E 16
G 24
RUN 1
NP(R1) = 2
a 14
d 17
d 31
m 3
p 2
r 16
RUN 2
NP(RUN2) = 2
SALVATE IN FILE TEMP. MEMORIA SECONDARIA
MEMORIA CENTRALE
d 21
e 16
g 24
a 14
d 17
d 31
INPUT
c 33
d 17
d 21
OUTPUT
2-WAY SORT MERGE
2 = 3
Salvo Anche Questo, fino a che NON ho Caricato TUTTE Le PAGINE in INPUT.
a 19
b 14
c 33
D 21
E 16
G 24
RUN 1
NP(R1) = 2
a 14
d 17
d 31
m 3
p 2
r 16
RUN 2
NP(RUN2) = 2
SALVATE IN FILE TEMP. MEMORIA SECONDARIA

⑧ È Stato UTILIZZATO Tutto, Quindi Posso SOVRASCRIVERLO

QUANTO COSTA? È dipendente dai PARAMETRI, ma:

- SORT_INTERNO ⇒ NP(R) Letture + NP(R) Scritture
- FASE MERGE ⇒ Ad Ogni Passo di MERGE posso leggere e Scrivere NP(R) Pagine, ma Quanti Sono i Passi?
  - $z = 2$, $\lceil \log_2(NP(R)) \rceil$ ⇒ Ad Ogni Passo #Run si dimezza

Costo Complessivo è: $2 \cdot NP(R) \cdot (\lceil \log_2 NP(R) \rceil + 1)$

- 2 → Lettura + Scrittura
- $\lceil \log_2 NP(R) \rceil$ → #Passi Merge
- 1 → Sort Interno

③ ACCESSO TRAMITE INDICE

Dipende dal TIPO di INDICE (hash/B+Tree) il COSTO DELLA RICERCA

- hash → Potenzialmente lineare
  - Si USA solo con '='
- B+Tree → Logaritmico
  - Si USA con '=', '<', '>', '<=', '>=', LIKE, ILIKE, BETWEEN

AND ⇒ Posso Usare MULTI ATTRIBUTO e INDICE DEI SINGOLI

OR ⇒ No MULTI-ATTRIBUTO, Posso fare UNIONE dopo che ho Usato

L'INDICE SUI SINGOLI ATTRIBUTI ⇒ MERGE RISULTATI

④ OPERATORE DI JOIN

→ Operazione più Gravosa per il DBMS

Ci Sono ≠ Implementazioni:

- NESTED-LOOP JOIN ⇒ più Generale
- MERGE-SCAN JOIN
- HASH-BASED JOIN

Sebbene JOIN sia COMMUTATIVO: $A \times B \equiv B \times A$ per il RISULTATO ma NON per il COSTO, l'ORDINE impatta le PRESTAZIONI.

NESTED-LOOP JOIN ⇒ Si eseguono 2 cicli ANNIDATI

```
∀ TUPLA Tr in R {
    ∀ TUPLA Ts in S {
        SE COPIA (tr, ts) SODDISFA Predicato JOIN {
            Aggiungi (Tr, Ts) al RISULTATO
        }
    }
}
```

R è Relazione ESTERNA ed S RELAZIONE INTERNA

R:

A	B
22	a
87	s
45	h
32	b

S:

A	C	D
22	z	8
45	x	4
22	s	7
87	s	9
32	c	3
45	h	5
32	9	6

P. J. = R.A = S.A

$\forall t_r \in R \{$
  $\forall t_s \in S \{$
    IF $(t_r.A = t_s.A) \{$
      (t_r, t_s) Nel Risultato
    $\}$
  $\}$
$\}$

$R \bowtie S$

A	C	D	B
22	z	8	a
22	s	7	a
.			
.			
.			

Per il Costo, Mi Serve Sapere: ① NB?
② NP(R), NP(S)
③ NR(R), NR(S)

① dipende dallo Spazio a disposizione in MEM. Centrale

Caso SEMPLICE: 1 Buffer per R
1 Buffer per S

E Leggo 1 Volta tutte le pagine di R e ∀ riga di R devo leggere TUTTA S, Quindi CARICARE tutte le PAGINE di S (NR(R)·NP(S))

COSTO COMPLESSIVO ⇒ NP(R) + NR(R)·NP(S)

Data una TUPLA della TABELLA esterna R, la Scansione Completa di S può ESSERE Sostituita da SCANSIONE SU Indice Costruito Sugli Attributi di JOIN in S.
Questo mi potrebbe far Risparmiare NP(S) da Caricare

COSTO NESTED LOOP JOIN CON INDICE:

$$NP(R) + NR(R) \cdot \left(\text{Profondità Indice} + \frac{NR(S)}{VAL(A,S)}\right)$$

Leggere l'indice fino a trovare le Pagine di S che mi Interessano

Selettività dell'Attributo A.

VAL(A,S) è il #Possibili Valori di A in S.

Mi PRODUCE una STIMA di #Righe ∀. Valore di A.

S

A

1	22	Z	8	
2	45	K	4	1
1	22	S	7	
4	87	S	9	2
3	32	C	3	
2	45	h	5	3
3	32	g	6	4

SENZA INDICE
∀ RIGA R CARICO
4 PAGINE DI S

CON INDICE PROF. 2
∀ RIGA R CARICO

$$2 + \left\lceil \frac{7}{4} \right\rceil \downarrow 2$$

Da Ricordare che è UNA STIMA