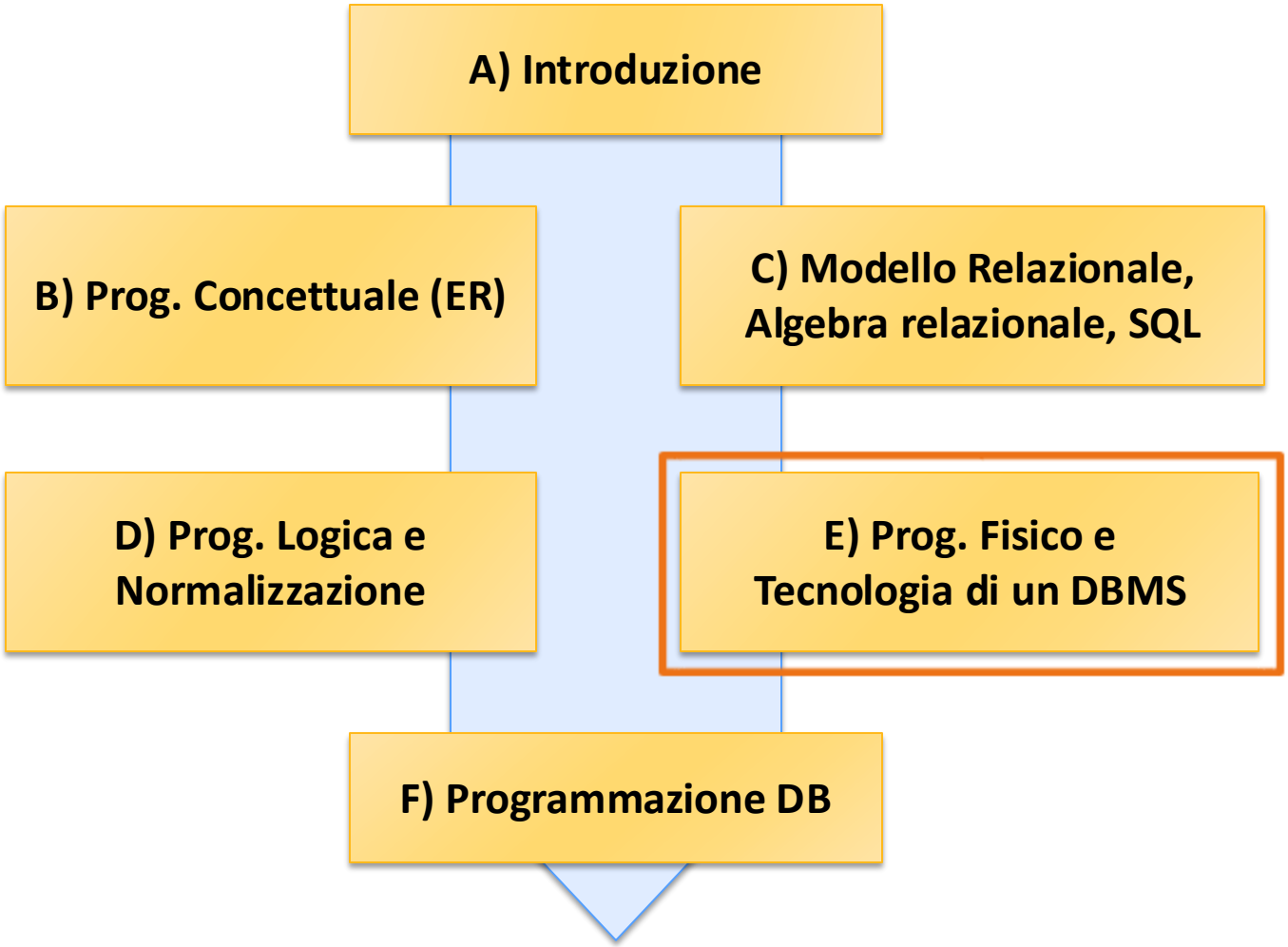
Basi di Dati

Calcolo del Costo di Accesso ai Dati (Parte 1)

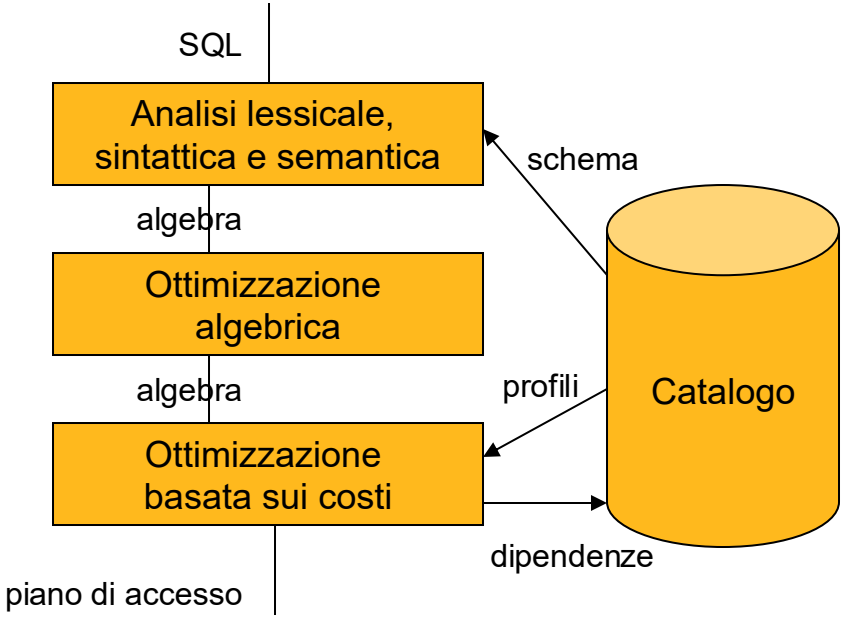
Basi di Dati – Dove ci troviamo?



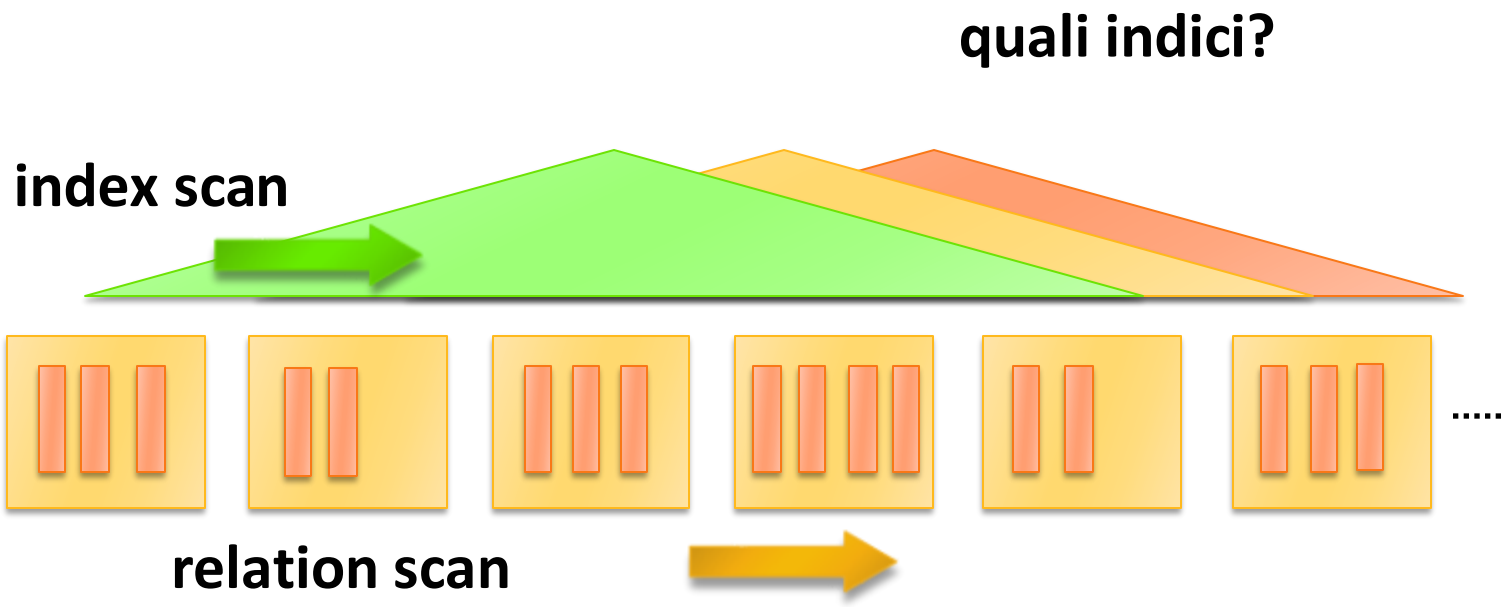
Ottimizzatori

- I criteri che vedremo sono in linea con i metodi e le scelte utilizzati dai query-optimizer dei DBMS relazionali
- lo scopo dei query-optimizer è infatti valutare quale sia la migliore strategia di accesso (access path) per le interrogazioni SQL degli utenti
- gli ottimizzatori non prendono decisioni sull'ordinamento delle relazioni e su quali indici costruire
- queste decisioni sono lasciate al DBA che deve valutare sulla base del carico di lavoro

Il processo di esecuzione delle interrogazioni



Decisioni



Relazione di NT tuple in NB blocchi (pagine)

Utilizzo degli indici

- **Un indice può essere utilizzato per eseguire una interrogazione SQL se l'attributo su cui è costruito:**
  - **compare nella clausola WHERE**
    - **è contenuto in un FATTORE BOOLEANO**
    - **il fattore booleano è ARGOMENTO DI RICERCA attraverso indice**
  - **compare in un ORDER BY o GROUP BY**

Utilizzo degli indici

- **Esempi: per la relazione**

IMPIEGATI (matr, cognome, nome, lavoro,
qualifica, salario, straordinario, dno)

1) la query: **SELECT cognome, salario**

FROM impiegati

WHERE dno = 51

AND salario > 2000

AND (lavoro = ‘fattorino’

OR lavoro = ’guardiano’)

oppure **….**

Utilizzo degli indici

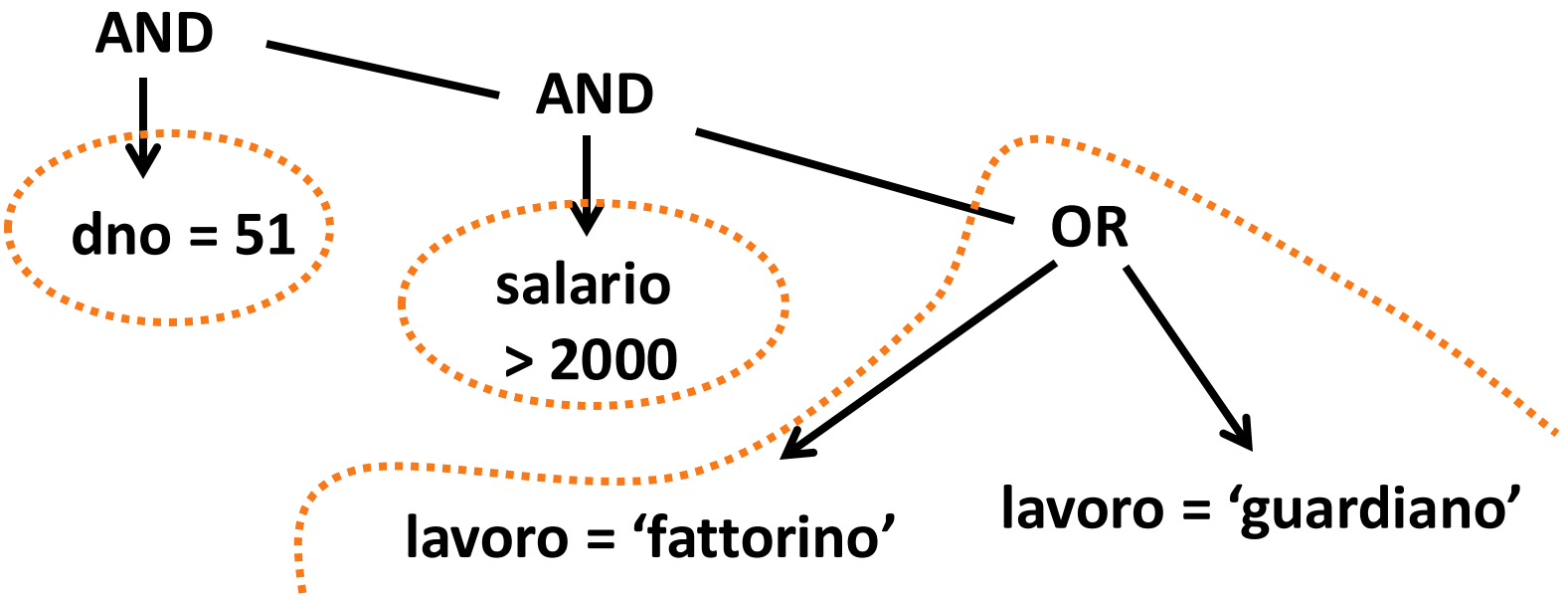
2) la query:

```
SELECT cognome, salario
FROM impiegati
WHERE dno = 51
AND salario + straordinario > 3000
AND (lavoro = 'fattorino'
        OR qualifica = 7)
```

Utilizzo degli indici

Separazione della condizione WHERE in fattori booleani: un predicato è un fattore booleano se è collegato alla radice del WHERE-tree da AND

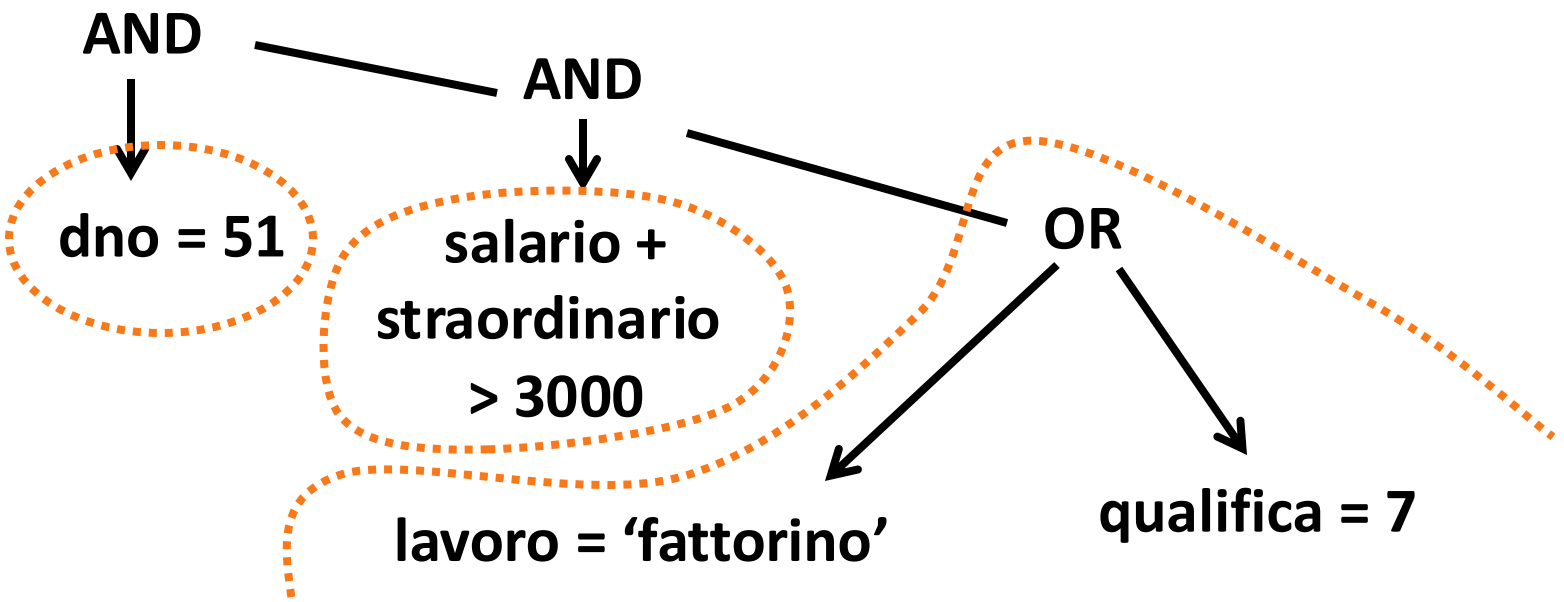
(query 1)



Utilizzo degli indici

un predicato è un fattore boleano se con risultato falso determina il risultato falso per la query

(query 2)

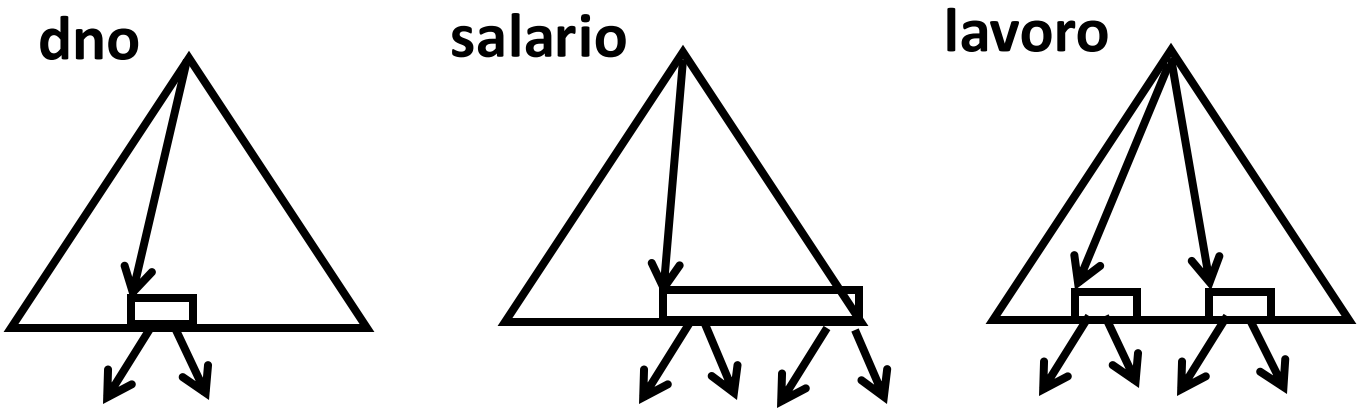


Utilizzo degli indici

1) per la query 1 sono fatt. bool. argomenti di ricerca:

   dno = 51 , salario > 2000

   (lavoro = ‘fattorino’ OR lavoro = ’guardiano’)

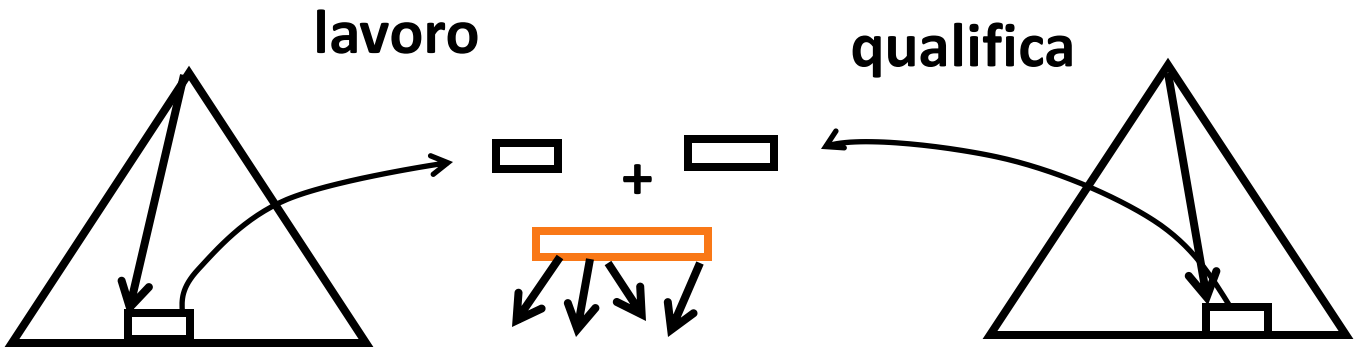


Utilizzo degli indici

2) per la query 2 è fatt. bool. argomento di ricerca

solo: dno = 51

per (lavoro = 'fattorino' OR qualifica = 7) se il DBMS può usare più indici per uno stesso fattore booleano:



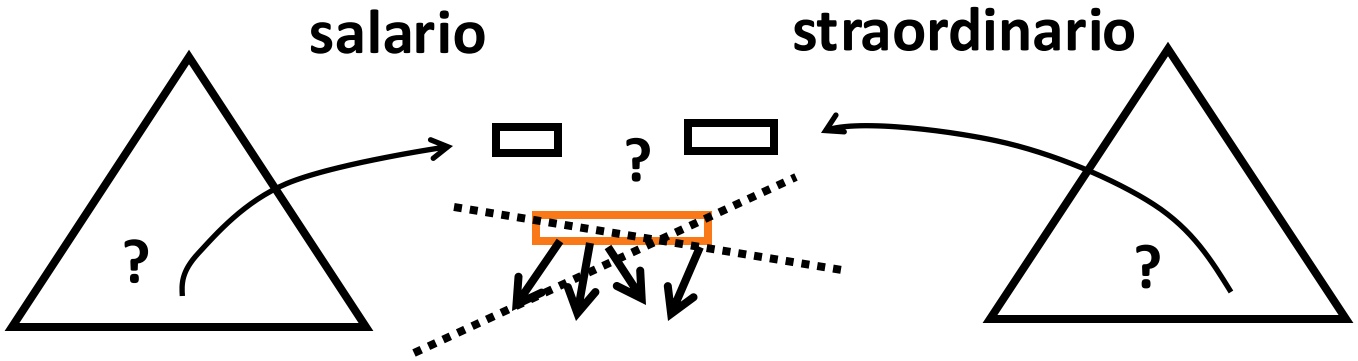
si può effettuare l'unione delle liste di TID

Utilizzo degli indici

2) per la query 2 non è fatt. bool. argomento di ricerca:

salario + straordinario > 3000

sia che il DBMS possa usare più indici per un fattore booleano che uno solo :



non si può effettuare l'unione delle liste di TID

Utilizzo degli indici

per una query sono argomento di ricerca i predicati del tipo: attributo . comparatore. valore,

ad es.:

dno = 47 (<, < = ,> = , > , between)	SI
dno = $D (variabile di programma)	SI
dno = 47 OR dno = 32	SI
(IN corrisponde ad OR ma non sempre è SI)	
(dno = 47) OR (qualifica = 3)	SI/NO
salario + straordinario > 3000	NO
salario = straordinario (stessa relazione)	NO

Modello di costo

- Un indice è utile per una query solo se il costo di accesso con l'indice è < costo dell'accesso sequenziale cioè < NB (NB/2 se attributo unique)
- il modello comunemente utilizzato (ce ne sono di molto più sofisticati e precisi) serve per previsioni di massima e si basa su:
  - per la relazione : NT, NB
  - per ogni indice : NF (numero di foglie)
  - per ogni attributo : NK (cardinalità), max, min
- tutti valori desumibili dai cataloghi dei DBMS

  Es: http://wiki.postgresql.org/wiki/Disk_Usage
  http://wiki.postgresql.org/wiki/Index_Maintenance
- le grandezze sono uniformemente distribuite

Selettività

- **Un predicato è selettivo se ci si aspetta che non tutte le tuple lo soddisfino**
- **fattore di selettività (filtro) F di un predicato : frazione di tuple che soddisfano il predicato**

nt / NT = valori selezionati / NK = SK / NK

SK: valori selezionati

- **A = valore : F = 1 / NK_A , default = 1/10**

Esempio dno = 24 : $F_{dno} = 1/ NK_{dno}$

- **A IN valSet : F = card(valSet) / NK_A , default = 1/2**

Esempio dno IN (24, 36) : $F_{dno} = 2/ NK_{dno}$

analogamente per dno = 24 OR dno = 36

Selettività

- A > valore: $F = (max_A - valore)/ (max_A - min_A)$, default f=1/3

Esempio voto > 27 $F_{voto} = 4/ 15$

se i voti vanno da 17 a 31 e supponendo che tutti siano stati assegnati almeno una volta

per salario > 2000 (range la cui cardinalità non è controllabile) si può prendere:

$F_{salario}$ = valori selezionati / valori ammissibili =

= (max(sal) - 2000) / (max(sal) - min(sal))

(ipotesi: sia per numeratore che per denominatore si considera 1 solo estremo incluso, ad esempio valori ammissibili compresi tra max(sal) escluso e min(sal) incluso)

- Analogamente per BETWEEN 2000 AND 3500

$F_{salario}$ = (3500 - 2000) / (max(sal) - min(sal)) default f=1/4

Selettività

- **Predicati su attributi diversi**
  - **pred1 OR pred2 :**

$$F = F_{pred1} + F_{pred2} - F_{pred1} F_{pred2}$$

  - **attributo A = attributo B**

$$F = 1 / \max(NK_A, NK_B)$$

se i due domini sono sovrapposti, altrimenti F=0

- **negazione : A = not valore**

$$F = 1 - 1 / NK_A$$

per ottenere maggiori precisioni molti sistemi memorizzano istogrammi semplificati

Selettività

- **Numero di tuple del risultato:**
  - $E = NT \times F_{pred}$ e
  - **nell'ipotesi di assenza di correlazione tra i valori degli attributi, per più predicati:**

    $E = NT \times \prod_i F_{predi}$

- **L'ipotesi non è sempre verificata, bisognerebbe rilevare un fattore di correlazione o di clustering relativo tra valori di attributi differenti:**
  - **tipo di lavoro, data di nascita: scorrelati**
  - **qualifica, dipartimento: correlati**

Costo di accesso

I casi esaminati si differenziano a seconda che:

- **indice clustered / unclustered**
- **attributo unique / con ripetizione dei valori**
- **predicato di uguaglianza / di range / OR**
- **uso di un solo indice / più indici**

Ipotesi di buffer

Si considera, per ogni relazione o indice, che ogni cambio di riferimento a pagina referenzi una pagina fuori dal buffer di memoria centrale.

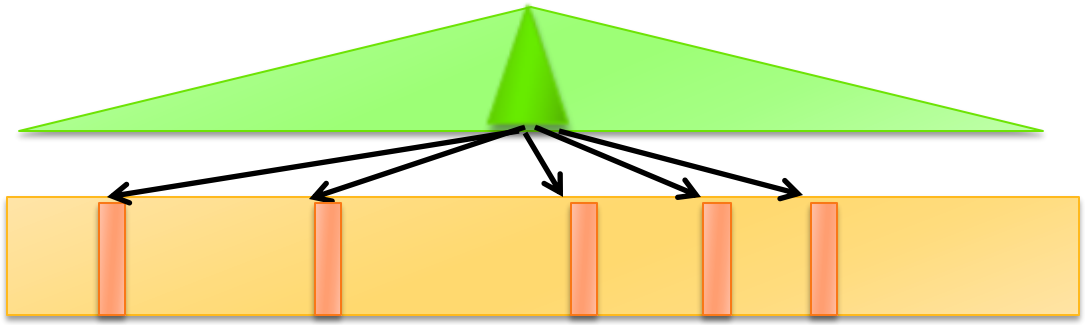
E' come se ogni relazione (o indice) avesse una sola pagina di buffer in memoria centrale.

E' un'ipotesi pessimistica che tende a calcolare upper bound.

Costo di accesso

Il costo C è dato dalla somma :

$$C_{indice} + C_{relazione}$$

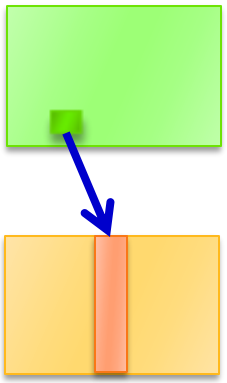


Costo di accesso

- **Indice clustered / unclustered su attributo unique con predicato di uguaglianza ovvero E = 1:**

  esempio, matr = 236 (sulla relazione impiegati)

foglia indice



blocco dati

Costo = 1 foglia + 1 blocco = 2

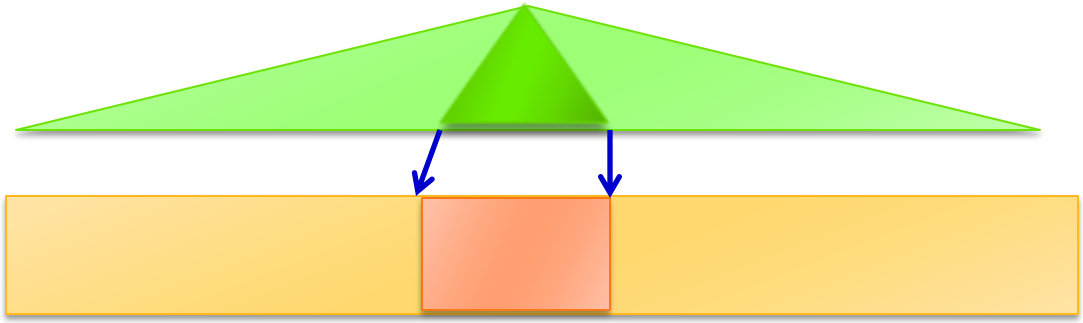
se l'indice è clustered o unclustered è lo stesso

Costo di accesso

- **Indice clustered / unclustered con E > 1:**

  es. matr between 236 and 312, lavoro = ‘guardiano’

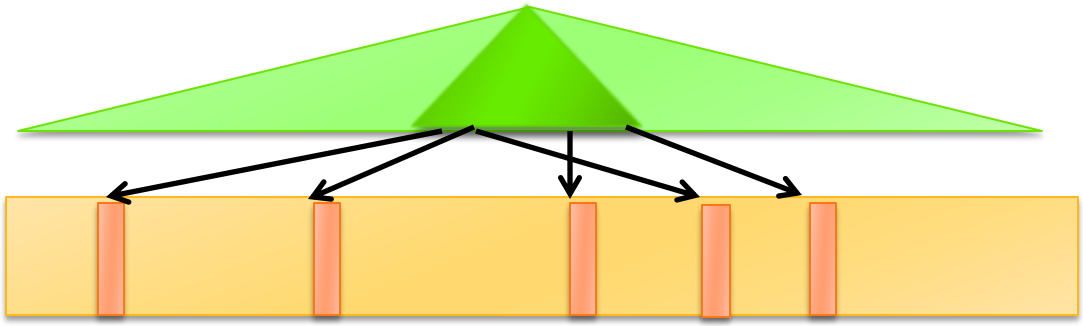
Caso clustered:



Costo = ⌈F x NF⌉ + ⌈F x NB⌉

Costo di accesso

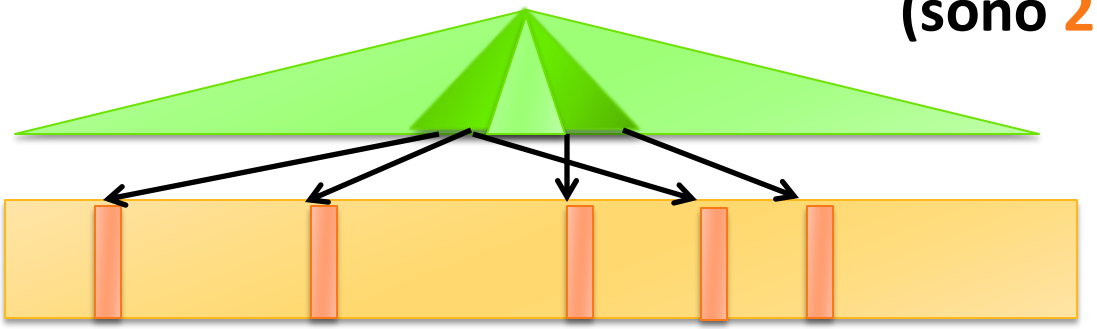
Caso unclustered:



Costo = (⌈F x NF⌉ + ⌈F x NT⌉) **con F = F $_{valore}$**

Costo di accesso

- **Indice unclustered / clustered (predicato OR):**
  lavoro = («guardiano» OR «portiere»)
  (sono 2 accessi distinti)



caso clustered:

Costo = 2 x (⌈F x NF⌉ + ⌈ F x NB ⌉)

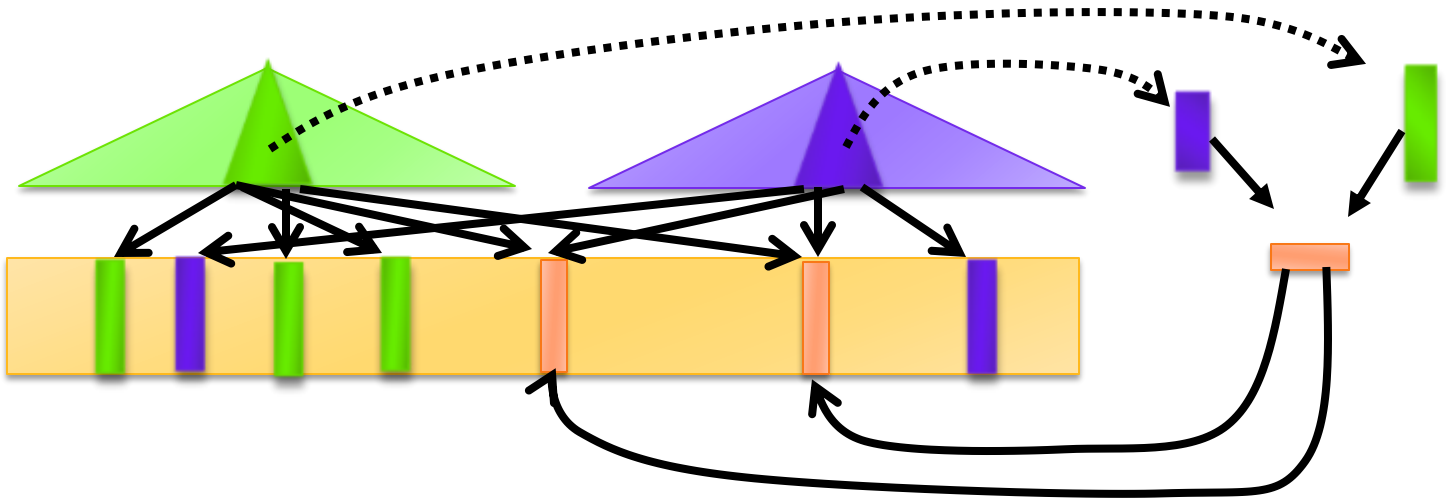
caso unclustered:

Costo = 2 x (⌈F x NF⌉ + ⌈ F x NT ⌉) **con $F = F_{valore}$**

Costo di accesso

- **Uso di più indici unclustered sulla stessa relazione con E > 1: intersezione dei TID estratti dagli indici**

  matr between 236 and 312 AND lavoro = «guardiano»



$$\text{Costo} = \sum_k \lceil F_k \times NF_k \rceil + \lceil \prod_k F_k \times NT \rceil$$

Costo di accesso

$$\text{Costo} = \sum_k \lceil F_k \times NF_k \rceil + \lceil \prod_k F_k \times NT \rceil$$

Questa formula è molto imprecisa perché presuppone la mancanza di correlazione tra attributi

Il costo può essere migliorato aggiungendo indici k solo se portano un vantaggio di selettività al secondo termine superiore allo svantaggio introdotto nella sommatoria del primo

Esercizio su Costo di accesso

SELECT * FROM IMPIEGATI
WHERE lavoro = 'fattorino' AND salario < 1500

con **NT = 10000, NB = 1000,**
NK_{sal} = 100 NK_{lav} = 50

indici: **clustered su salario con NF = 160**
unclustered su lavoro con NF = 100

Si suppone che salario abbia valori tra min=500 e max=10500

- **Stimare il costo di accesso (sequenziale / con indici) dell'operazione**
- **Stimare il numero di tuple del risultato**
- **Discutere la variazione di costo di accesso nel caso in cui l'indice fosse clustered su lavoro**

Costo di accesso

$F_{lav} = 1 / 50 = 0.02$

$F_{sal} = (1500 - min) / (max- min) = 0.1$

costo delle scansione sequenziale:

$C_{seq} = 1000$

costo dell'indice su lavoro (unclust.):

$C_{lav} = \lceil F_{lav} \times NF_{lav} \rceil + \lceil F_{lav} \times NT \rceil =$

$0.02 \times 100 + 0.02 \times 10000 = 2 + 200 = 202$

costo dell'indice su salario (clust):

$C_{sal} = \lceil F_{sal} \times NF_{sal} \rceil + \lceil F_{sal} \times NB \rceil =$

$0.1 \times 160 + 0.1 \times 1000 = 16 + 100 = 116$

$C_{seq} > C_{lav} > C_{sal}$

Costo di accesso

scambiamo adesso l'ordinamento per gli indici:

costo dell'indice su lavoro (clust.):

$C_{lav} = \lceil F_{lav} \times NF_{lav} \rceil + \lceil F_{lav} \times NB \rceil =$

$0.02 \times 100 + 0.02 \times 1000 = 2 + 20 = 22$

costo dell'indice su salario (unclust):

$C_{sal} = \lceil F_{sal} \times NF_{sal} \rceil + \lceil F_{sal} \times NT \rceil =$

$0.1 \times 160 + 0.1 \times 10000 = 16 + 1000 = 1016$

$C_{sal} > C_{seq} > C_{lav}$

Costo di accesso

tuple del risultato:

$$E = F_{lav} \times F_{sal} \times NT = 20$$

- **la scelta migliore per la query è avere un ordinamento su lavoro e un indice su lavoro, mentre l'indice su salario non deve essere costruito**
- **il miglioramento che si ottiene rispetto all'assenza di indici e ordinamenti è di 1 a 45**
- **nel caso in cui l'ordinamento su salario sia di utilità per altre query l'indice unclustered su lavoro porta ad un miglioramento di 1 a 5**