Basi di Dati

Linguaggi per le basi di dati

Marco Maratea
Laurea in Informatica, DeMaCS, UNICAL

20 Novembre 2025

Equivalenza di espressioni

- Due espressioni sono equivalenti se producono lo stesso risultato qualunque sia l'istanza attuale della base di dati
- L'equivalenza è importante in pratica perché i DBMS cercano di eseguire espressioni equivalenti a quelle date, ma meno "costose"

Un'equivalenza importante

- Push selections (se A è attributo di R_2)

$$\text{SEL}_{A=10}(R_1 \text{ JOIN } R_2) = R_1 \text{ JOIN SEL}_{A=10}(R_2)$$

- Riduce in modo significativo la dimensione del risultato intermedio (e quindi il costo dell'operazione)

Nota

- In questo corso, ci preoccupiamo poco dell'efficienza:
  - l'obiettivo è di scrivere interrogazioni corrette e leggibili
- Motivazione:
  - I DBMS si preoccupano di scegliere le strategie realizzative efficienti

Selezione con valori nulli

Impiegati

Matricola	Cognome	Filiale	Età
7309	Rossi	Roma	32
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

$$SEL_{Età > 40}\ (Impiegati)$$

- **la condizione atomica è vera solo per valori non nulli**

Un risultato non desiderabile

$$\text{SEL}_{\text{Età}>30}(\text{Persone}) \cup \text{SEL}_{\text{Età}\leq 30}(\text{Persone}) \neq \text{Persone}$$

- Perché? Perché le selezioni vengono valutate separatamente!
- Ma anche

$$\text{SEL}_{\text{Età}>30 \vee \text{Età}\leq 30}(\text{Persone}) \neq \text{Persone}$$

- Perché? Perché anche le condizioni atomiche vengono valutate separatamente!

Selezione con valori nulli: soluzione

$$SEL_{Età > 40}(Impiegati)$$

- la condizione atomica è vera solo per valori non nulli
- per riferirsi ai valori nulli esistono forme apposite di condizioni:

  IS NULL

  IS NOT NULL
- si potrebbe usare (ma non serve) una "logica a tre valori" (vero, falso, sconosciuto)

- Quindi:

$$\text{SEL}_{\text{Età}>30}(\text{Persone}) \cup \text{SEL}_{\text{Età}\leq 30}(\text{Persone}) \cup \text{SEL}_{\text{Età IS NULL}}(\text{Persone}) = \text{SEL}_{\text{Età}>30 \vee \text{Età}\leq 30 \vee \text{Età IS NULL}}(\text{Persone}) = \text{Persone}$$

Impiegati

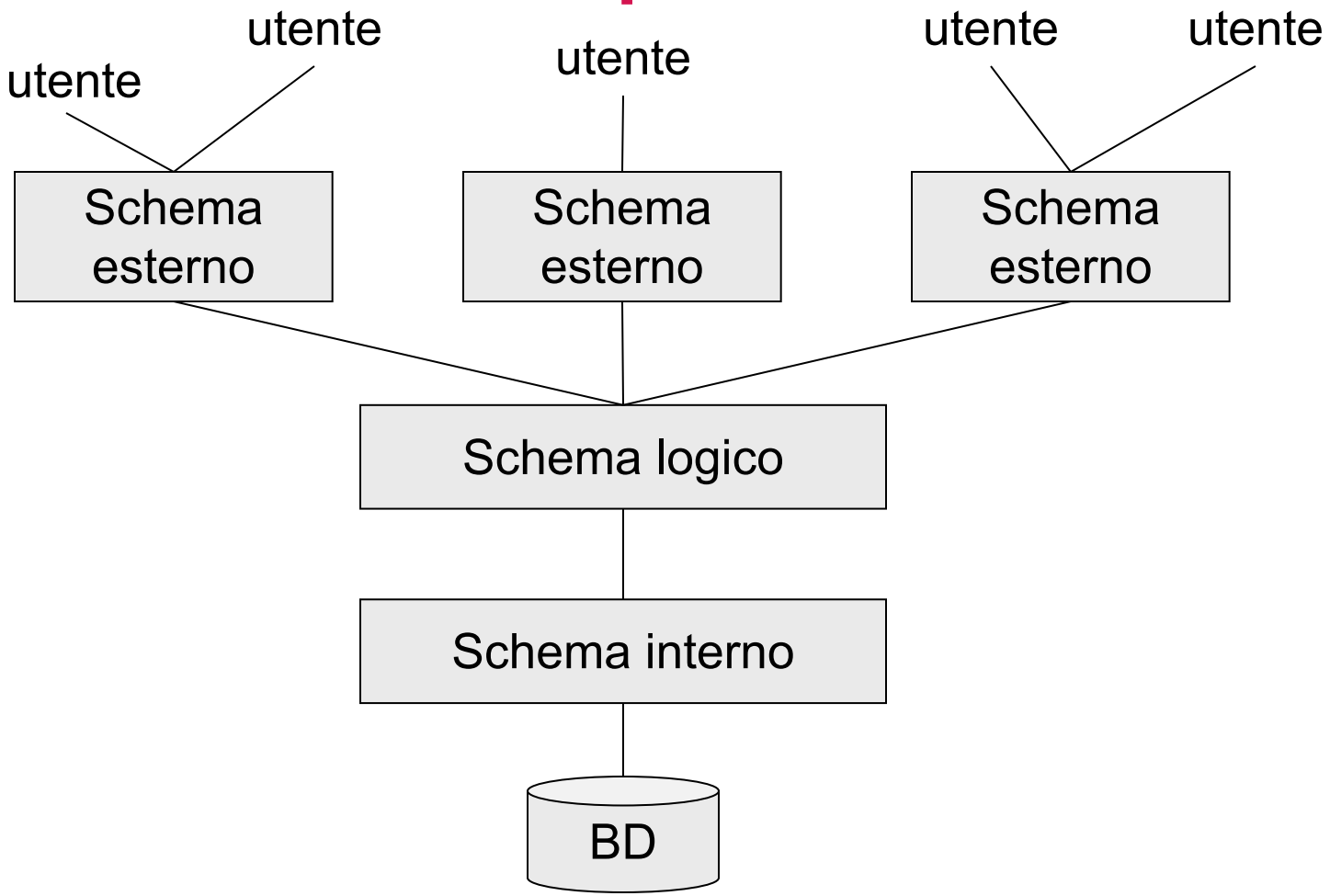
Matricola	Cognome	Filiale	Età
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

SEL (Età > 40) OR (Età IS NULL) (Impiegati)

Viste (relazioni derivate)

- Rappresentazioni diverse per gli stessi dati (schema esterno)
- Relazioni derivate:
  - relazioni il cui contenuto è funzione del contenuto di altre relazioni (definito per mezzo di interrogazioni)
- Relazioni di base: contenuto autonomo
- Le relazioni derivate possono essere definite su altre derivate

Architettura standard (ANSI/SPARC) a tre livelli per DBMS



Viste, esempio

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Direzione

Reparto	Capo
A	Mori
B	Bruni

- una vista:

Supervisione =

PROJ $_{\text{Impiegato, Capo}}$ (Afferenza JOIN Direzione)

Viste virtuali e materializzate

- Due tipi di relazioni derivate:
  - viste materializzate
  - relazioni virtuali (o viste)

Viste materializzate

- relazioni derivate memorizzate nella base di dati
  - vantaggi:
    - immediatamente disponibili per le interrogazioni
  - svantaggi:
    - ridondanti
    - appesantiscono gli aggiornamenti
    - sono raramente supportate dai DBMS

Viste virtuali

- relazioni virtuali (o viste):
  - sono supportate dai DBMS (tutti)
  - una interrogazione su una vista viene eseguita "ricalcolando" la vista (o quasi)

Interrogazioni sulle viste

- Sono eseguite sostituendo alla vista la sua definizione:

$$SEL_{Capo='Leoni'}\ (Supervisione)$$

viene eseguita come

$$SEL_{Capo='Leoni'}(\ PROJ_{\ Impiegato,\ Capo}\ (Afferenza\ JOIN\ Direzione))$$

Viste, motivazioni

- Schema esterno: ogni utente vede solo
  - ciò che gli interessa e nel modo in cui gli interessa, senza essere distratto dal resto
  - ciò che e' autorizzato a vedere (autorizzazioni)
- Strumento di programmazione:
  - si può semplificare la scrittura di interrogazioni: espressioni complesse e sottoespressioni ripetute
- Utilizzo di programmi esistenti su schemi ristrutturati

Invece:

- L'utilizzo di viste non influisce sull'efficienza delle interrogazioni

Viste come strumento di programmazione

- Trovare gli impiegati che hanno lo stesso capo di Rossi
- Senza vista:

$$\text{PROJ}_{\text{Impiegato}}((\text{Afferenza JOIN Direzione}) \text{ JOIN } \text{REN}_{\text{ImpR,RepR} \leftarrow \text{Imp,Reparto}}(\text{SEL}_{\text{Impiegato='Rossi'}}(\text{Afferenza JOIN Direzione})))$$

- Con la vista:

$$\text{PROJ}_{\text{Impiegato}}(\text{Supervisione JOIN } \text{REN}_{\text{ImpR} \leftarrow \text{Imp}}(\text{SEL}_{\text{Impiegato='Rossi'}}(\text{Supervisione})))$$

Viste e aggiornamenti, attenzione

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Verdi	A

Direzione

Reparto	Capo
A	Mori
B	Bruni
C	Bruni

Supervisione

Impiegato	Capo
Rossi	Mori
Neri	Bruni
Verdi	Mori

- Vogliamo inserire, nella vista, il fatto che Lupi ha come capo Bruni; oppure che Belli ha come capo Falchi; come facciamo?

Viste e aggiornamenti

- "Aggiornare una vista":
  - modificare le relazioni di base in modo che la vista, "ricalcolata" rispecchi l'aggiornamento
- L'aggiornamento sulle relazioni di base corrispondente a quello specificato sulla vista deve essere univoco
- In generale però non è univoco!
- Ben pochi aggionamenti sono ammissibili sulle viste