

# Basi Di Dati e di conoscenza

Algebra Relazionale

# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

# Linguaggi per basi di dati

- operazioni sullo schema
  - **DDL: Data Definition Language**
- operazioni sui dati
  - **DML: Data Manipulation Language**
    - interrogazione ("query")
    - aggiornamento

# Linguaggi di interrogazione per basi di dati relazionali

- **Dichiarativi**: specificano le proprietà del risultato ("**che cosa**")
- **Procedurali**: specificano le modalità di generazione del risultato ("**come**")

# Linguaggi relazionali

- **Algebra relazionale: procedurale**
  - Insieme di operatori
    - su relazioni
    - che producono relazioni
    - e possono essere composti
- **Calcolo relazionale: dichiarativo (teorico)**
  - Basato sul calcolo dei predicati del primo ordine
  - Connettivi e clausole che consentono di descrivere la relazione risultato
- **SQL (Structured Query Language): intermedio (reale)**
- **QBE (Query by Example): dichiarativo (reale)**

# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

# Algebra Relazionale

- Linguaggio procedurale, in cui le operazioni vengono descritte descrivendo la procedura per ottenere la soluzione.

## Operatori di base:

- *Unione, differenza, intersezione* derivati dalla teoria degli insiemi
- *Ridenominazione, selezione, proiezione* specifici dell'algebra relazionale
- *join* che può assumere diverse forme (naturale, theta-join, prodotto cartesiano)

# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query



# Operatori insiemistici

- Le relazioni sono **insiemi** e quindi è naturale estendere ad esse le operazioni relative.
- Tuttavia le relazioni sono insiemi di tuple **omogenee** e quindi ha senso definire ed applicare tali operatori solo a tuple **definite sugli stessi attributi**.
- Es. l'unione fra due relazioni su tuple non omogenee **non** è una relazione.

# Operatori derivati dagli insiemi

- **Unione**

L'unione fra due relazioni  $r_1$  e  $r_2$  definite sullo stesso insieme di attributi  $X$  è indicata con  $r_1 \cup r_2$  ed è una relazione su  $X$  contenente le tuple che appartengono a  $r_1$  o  $r_2$  oppure ad entrambe.

- **Intersezione**

L'intersezione fra due relazioni  $r_1$  e  $r_2$  definite sullo stesso insieme di attributi  $X$  è indicata con  $r_1 \cap r_2$  ed è una relazione su  $X$  contenente le tuple che appartengono sia a  $r_1$  che a  $r_2$ .

- **Differenza**

La differenza fra due relazioni  $r_1$  e  $r_2$  definite sullo stesso insieme di attributi  $X$  è indicata con  $r_1 - r_2$  ed è una relazione su  $X$  contenente le tuple che appartengono a  $r_1$  e non a  $r_2$ .

# Algebra relazionale

**Laureati**

Matricola	Cognome	Età
1	Rossi	37
2	Neri	36
3	Bianchi	28

**Dirigenti**

Matricola	Cognome	Età
9	Verdi	51
2	Neri	36
3	Bianchi	28

**Laureati  $\cap$  Dirigenti**

Matricola	Cognome	Età
2	Neri	36
3	Bianchi	28

**Laureati - Dirigenti**

Matricola	Cognome	Età
1	Rossi	37

**Laureati  $\cup$  Dirigenti**

Matricola	Cognome	Età
1	Rossi	37
2	Neri	36
3	Bianchi	28
9	Verdi	51

# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

# Ridenominazione

- L'operatore di **ridenominazione** cambia il nome degli attributi allo scopo di facilitare operazioni insiemistiche.
- E' un operatore che consente di modificare il nome di un attributo per poterlo associare ad un altro attributo in una operazione algebrica.
  - operatore **monadico** (con un argomento)
  - "**modifica lo schema**" lasciando inalterata l'istanza dell'operando
- Si indica con  $\rho$  **nuovonome**  $\leftarrow$  **vecchionome** (**Relazione**)
- La ridenominazione  $\rho_{B_1, B_2, \dots, B_k \leftarrow A_1, A_2, \dots, A_k} (R)$  contiene tuple  $t'$  tali che  $t'$  è una tupla e  $t'[B_i] = t[A_i]$ , cioè cambiano i nomi degli attributi ma i valori non cambiano

**Esempio:** Date le relazioni

- **Paternità**(Padre, Figlio)
- **Maternità**(Madre, Figlio)

è possibile ottenere

$\rho$  **Genitore**  $\leftarrow$  **Padre** (**Paternità**)  $\cup$   $\rho$  **Genitore**  $\leftarrow$  **Madre** (**Maternità**)

# Ridenominazione: Esempio

## Paternità

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

## $\rho_{\text{Genitore}} \leftarrow \text{Padre}$ (Paternità)

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

## Maternità

Madre	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

## $\rho_{\text{Genitore}} \leftarrow \text{Madre}$ (Maternità)

Genitore	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

# Ridenominazione: Esempio

$\rho_{\text{Genitore} \leftarrow \text{Padre}}$  (Paternità)

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

$\rho_{\text{Genitore} \leftarrow \text{Madre}}$  (Maternità)

Genitore	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

$\rho_{\text{Genitore} \leftarrow \text{Padre}}$  (Paternità)

$\cup$   
 $\rho_{\text{Genitore} \leftarrow \text{Madre}}$  (Maternità)

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco
Eva	Abele
Eva	Set
Sara	Isacco

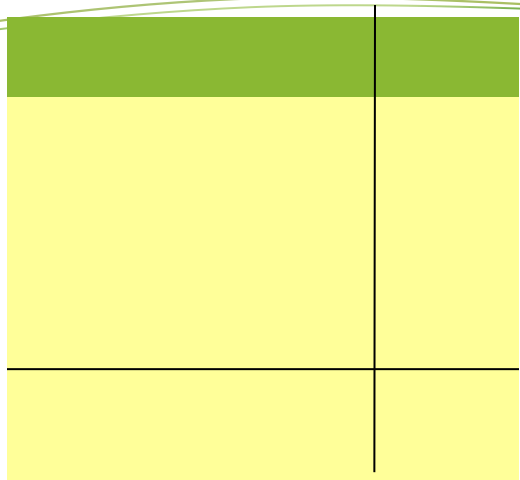
# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

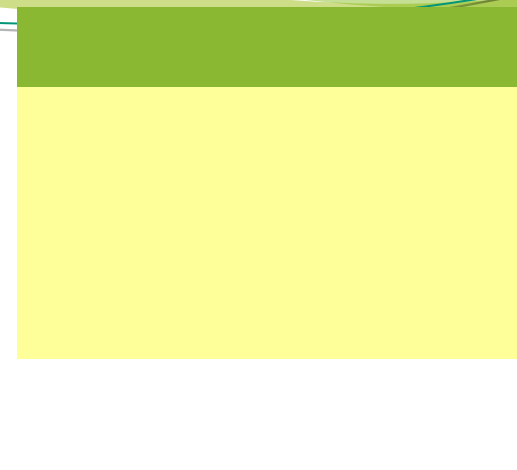


# Selezione e Proiezione

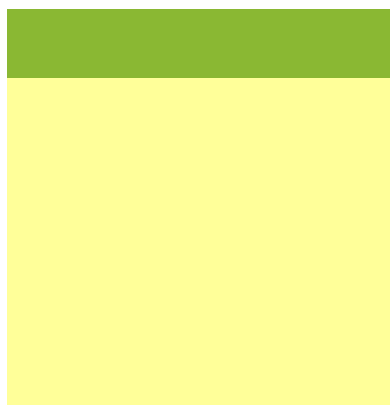
- Le operazioni di **selezione** e di **proiezione** si applicano ad una relazione e ne restituiscono una porzione.
- Possono essere considerate ortogonali o complementari, in quanto una opera sulle **righe** e l'altra sulle **colonne**.
- La **selezione** produce un insieme di tuple, su tutti gli attributi.
- La **proiezione** produce un risultato definito su un insieme di attributi, cui contribuiscono tutte le tuple.



**selezione**




**proiezione**



# Selezione

- operatore **monadico**
- produce un risultato che
  - ha lo stesso schema dell'operando
  - contiene un sottoinsieme delle ennuple dell'operando che soddisfano una **specificata condizione di selezione**.
- Si indica con  $\sigma_F(r)$  o  $SEL_F(r)$   
dove:
  - **F** è una condizione da verificare
  - **r** è la relazione a cui la selezione è applicata definita su un insieme di attributi X



Quindi,  $\sigma_F(r)$  produce una relazione sugli stessi attributi di r contenente le ennuple su cui F è vera (**semantica**).

# Sintassi: Condizione di selezione

$$\sigma_F ( r(X) )$$

- $F$  è una *formula proposizionale* su  $X$ , cioè una formula ottenuta combinando con i simboli  $\wedge$  (*and*)  $\vee$  (*or*)  $\neg$  (*not*) espressioni del tipo  $A \theta B$  o  $A \theta c$ : dove :
  - $\theta$  è un operatore di confronto ( $\leq, <, =, >, \geq$ )
  - $A$  e  $B$  sono attributi di  $X$  su cui il confronto abbia senso
  - $c$  è una costante tale che il confronto con  $A$  sia definito
- $E'$  definito un valore di verità di  $F$  su una ennupla  $t \in r$  :
  - $A \theta B$  è vera se e solo se  $t[A] \theta t[B]$  è vero
  - $A \theta c$  è vera se  $t[A] \theta c$  è vera
  - $F_1 \wedge F_2, F_1 \vee F_2, \neg F$  hanno l'usuale significato

# Selezione: esempio

## Dirigenti

Matricola	Cognome	Età	Stipendio
9	Verdi	51	2700
7	Blu	35	3000
10	Viola	29	2000

$\sigma_{\text{Età} > 50 \wedge \text{Stipendio} > 2500}(\text{Dirigenti})$

Matricola	Cognome	Età	Stipendio
9	Verdi	51	2700

# Selezione: esempio

## Cittadini

Cognome	Nome	Nascita	Residenza
Rossi	Mario	Roma	Milano
Neri	Luca	Roma	Roma
Verdi	Nico	Firenze	Firenze
Rossi	Marco	Napoli	Firenze

$$\sigma_{\text{Nascita} = \text{Residenza}}(\text{Cittadini})$$

Cognome	Nome	Nascita	Residenza
Neri	Luca	Roma	Roma
Verdi	Nico	Firenze	Firenze

# Selezione con valori nulli

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

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

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

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

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

# Selezione con valori nulli

- Per riferirsi ai valori nulli esistono forme apposite di condizioni:

**IS NULL**

**IS NOT NULL**

$\sigma_{Età>30}(\text{Persone}) \cup \sigma_{Età\leq 30}(\text{Persone}) \cup \sigma_{Età \text{ IS NULL}}(\text{Persone})$

=

$\sigma_{Età>30 \vee Età\leq 30 \vee Età \text{ IS NULL}}(\text{Persone})$

=

**Persone**



# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

# Proiezione

Dati una relazione  $r(X)$  e un sottoinsieme  $Y$  di  $X$ , la proiezione di  $r$  su  $Y$  si indica con

$$\Pi_Y(r) \text{ o } \mathbf{PROJ}_Y(r)$$

l'insieme delle ennuple su  $Y$  ottenute dalle ennuple di  $r$  considerando solo i valori su  $Y$ .

La proiezione  $\Pi_Y(r)$  è l'insieme di tuple su un sottoinsieme  $Y$  di attributi  $X$  di  $R$ , ottenuta dalle tuple di  $R$  considerando solo i valori su  $Y$  cioè:

$$\Pi_Y(r) = \{ t[Y] \mid t \in r \}$$

- Una proiezione ha un numero di tuple *minore o uguale* rispetto alla relazione  $r$  cui è applicata. Il numero di tuple è uguale se e solo se  $Y$  è **superchiave per  $r$**

# Proiezione: Esempio

- visualizzare Cognome e Nome di tutti i cittadini

**Cittadini**

Cognome	Nome	Nascita	Residenza
Rossi	Mario	Roma	Milano
Neri	Luca	Roma	Roma
Verdi	Nico	Firenze	Firenze
Rossi	Marco	Napoli	Firenze

$\pi_{\text{Cognome, Nome}}(\text{Cittadini})$

Cognome	Nome
Rossi	Mario
Neri	Luca
Verdi	Nico
Rossi	Marco

# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

# Algebra relazionale: Join

- Combinando selezione e proiezione, si possono estrarre informazioni da **una** relazione
- Non si possono però correlare informazioni presenti in relazioni diverse
- Il **join** è l'operatore più interessante (potente) dell'algebra relazionale in quanto permette di correlare dati in relazioni diverse
- Evidenzia la proprietà del modello relazionale di essere **basato su valori**.
- Due tipi di join:
  - **Join naturale**
  - **Theta join**

# Join naturale

- L'operatore di **join naturale**  $r_1 \bowtie r_2$  (o  $r_1 \mid X \mid r_2$ ) correla dati in relazioni diverse sulla base di valori uguali in attributi con lo stesso nome.
- Il **join naturale**  $r_1 \bowtie r_2$  di  $r_1(X_1)$  e  $r_2(X_2)$  è una relazione definita su  $X_1 \cup X_2$  (che si può scrivere  $X_1 X_2$ ):

$$r_1 \bowtie r_2 = \{ t \text{ su } X_1 X_2 \mid t[X_1] \in r_1 \text{ e } t[X_2] \in r_2 \}$$

- Il grado della relazione ottenuta è minore o uguale al grado della somma dei gradi delle due relazioni in quanto gli attributi omonimi compaiono una sola volta.
- Se  $X_1 \cap X_2$  è vuoto il join naturale equivale al *prodotto cartesiano* fra le relazioni.
- Se  $X_1 = X_2$  il join naturale equivale all'intersezione fra le relazioni

# Join naturale

- Se ciascuna ennuple di ciascuno degli operandi contribuisce ad almeno una ennuple del risultato il join si dice *completo*.
- Se per alcune ennuple non è verificata la corrispondenza e non contribuiscono al risultato, le ennuple si dicono *dangling*.
- Ai due estremi si pongono il join vuoto in cui nessuna ennuple degli operandi è combinabile, e quello in cui ciascuna delle ennuple di un operando è combinabile con tutte le ennuple dell'altro. In questo caso la cardinalità della relazione risultante è pari al prodotto della cardinalità degli operandi

# Join naturale completo

- ogni ennupla contribuisce al risultato:
  - join **completo**

**R<sub>1</sub>**

Impiegato	Reparto
Rossi	vendite
Neri	produzione
Bianchi	produzione

**R<sub>2</sub>**

Reparto	Capo
produzione	Mori
vendite	Bruni

**R<sub>1</sub> ⋈ R<sub>2</sub>**

Impiegato	Reparto	Capo
Rossi	vendite	Bruno
Neri	produzione	Mori
Bianchi	produzione	Mori



# Un join non completo: Esempio

**R<sub>1</sub>**

Impiegato	Reparto
Rossi	vendite
Neri	produzione
Bianchi	produzione

**R<sub>2</sub>**

Reparto	Capo
produzione	Mori
marketing	Bruni

**R<sub>1</sub> ⋈ R<sub>2</sub>**

Impiegato	Reparto	Capo
Neri	produzione	Mori
Bianchi	produzione	Mori

# Join vuoto: Esempio

**R<sub>1</sub>**

Impiegato	Reparto
Rossi	vendite
Neri	produzione
Bianchi	produzione

**R<sub>2</sub>**

Reparto	Capo
amministrazione	Mori
marketing	Bruni

**R<sub>1</sub> ⋈ R<sub>2</sub>**

Impiegato	Reparto	Capo
-----------	---------	------

# Join naturale: proprietà

1. Il **join di  $r_1$  e  $r_2$**  contiene un numero di ennuple compreso fra zero e il prodotto di  $|r_1|$  e  $|r_2|$
2. se il join di  $r_1$  e  $r_2$  è **completo** allora contiene un numero di tuple pari almeno al massimo fra  $|r_1|$  e  $|r_2|$
3. se  $X_1 \cap X_2$  contiene una chiave per  $r_2$ , allora il join di  $r_1(X_1)$  e  $r_2(X_2)$  contiene almeno  $|r_2|$  tuple.
4. se il join coinvolge una chiave di  $R_2$  e **un vincolo di integrità referenziale**, allora il numero di tuple è pari a  $|R_1|$
5.  $r_1 \bowtie r_2 = r_2 \bowtie r_1$  il join è **commutativo**
6.  $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$  il join è **associativo**  
Quindi sequenze di join possono essere scritte senza parentesi

# Prodotto cartesiano

- un join naturale su relazioni senza attributi in comune
- contiene sempre un numero di ennuple pari al prodotto delle cardinalità degli operandi (le ennuple sono tutte combinabili )
- Il **prodotto cartesiano**  $r_1 \times r_2$  di  $r_1(X_1)$  e  $r_2(X_2)$  è una relazione definita su  $X_1 \cup X_2$   
:

$$r_1 \times r_2 = \{ t \text{ su } X_1 X_2 \mid t[X_1] \in r_1 \text{ e } t[X_2] \in r_2 \}$$

# Prodotto cartesiano: esempio

## Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

## Reparti

Codice	Capo
A	Mori
B	Bruni

## Impiegati x Reparti

Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	A	Mori
Bianchi	B	A	Mori
Rossi	A	B	Bruni
Neri	B	B	Bruni
Bianchi	B	B	Bruni

# Join esterni: Outer Join

- Il **join naturale** traslascia le ennuple in cui non vi è corrispondenza fra gli attributi legati dal join
- L'operatore di **join esterno (outer join)** prevede che tutte le tuple diano sempre un contributo al risultato, eventualmente estese con valori nulli ove non vi siano controparti opportune.

## Tre tipi di outer join:

- **left join:** Contribuiscono tutte le ennuple del primo operando eventualmente estese con valori nulli
- **right join:** Contribuiscono tutte le ennuple del secondo operando eventualmente estese con valori nulli
- **full join:** Contribuiscono tutte le ennuple del primo e del secondo operando eventualmente estese con valori nulli

# Left Join

- **Left Join** ritorna tutte le tuple dalla relazione di sinistra a prescindere dal fatto che siano combinabili con quelle della relazione di destra.
- Assegna valori nulli per i record che non matchano.

# Left Join: Esempio

## Impiegati

Impiegato	Reparto
Rossi	vendite
Neri	produzione
Bianchi	produzione

## Reparti

Reparto	Capo
produzione	Mori
acquisti	Bruni

## Impiegati $\triangleright \triangleleft_{\text{LEFT}}$ Reparti

Impiegato	Reparto	Capo
Rossi	vendite	NULL
Neri	produzione	Mori
Bianchi	produzione	Mori



# Right Join Join

- **Right Join** ritorna tutte le tuple dalla relazione di destra a prescindere dal fatto che siano combinabili con quelle della relazione di sinistra.
- Assegna valori nulli per i record che non matchano.

# Right Join: Esempio

## Impiegati

Impiegato	Reparto
Rossi	vendite
Neri	produzione
Bianchi	produzione

## Reparti

Reparto	Capo
produzione	Mori
acquisti	Bruni

## Impiegati $\triangleright \triangleleft$ RIGHT Reparti

Impiegato	Reparto	Capo
Neri	produzione	Mori
Bianchi	produzione	Mori
NULL	acquisti	Bruni

# Full Join

- **Full Join** combina i risultati di due relazioni tenendo conto di tutte le tuple delle relazioni, anche di quelle che non hanno corrispondenza tra di loro.
- Il risultato contiene sempre tutte le tuple della relazione di sinistra ("left"), estraendo dalla relazione di destra ("right") solamente le tuple che trovano corrispondenza nella regola di confronto; inoltre verranno estratte tutte le tuple della relazione di sinistra ("left") che non trovano corrispondenza nella relazione di destra ("right") impostando a nulli i valori di tutti gli attributi della relazione di destra, e viceversa

# Full Join: Esempio

## Impiegati

Impiegato	Reparto
Rossi	vendite
Neri	produzione
Bianchi	produzione

## Reparti

Reparto	Capo
produzione	Mori
acquisti	Bruni

## Impiegati $\bowtie$ FULL Reparti

Impiegato	Reparto	Capo
NULL	acquisti	Bruni
Neri	produzione	Mori
Bianchi	produzione	Mori
Rossi	Vendite	NULL

# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

# Theta-Join

- Se si devono correlare attributi con nome diverso è possibile fare il *theta-join*, definito come un prodotto cartesiano seguito da una selezione

$$\mathbf{r}_1 \bowtie_F \mathbf{r}_2 = \sigma_F (\mathbf{r}_1 \times \mathbf{r}_2)$$

- dove F è una formula e  $\mathbf{r}_1$  e  $\mathbf{r}_2$  non hanno attributi di nome comune
- Se F è una relazione di uguaglianza, con un attributo della prima relazione e uno della seconda, allora siamo in presenza di un *equi-join*.
- **Sono importanti formalmente:**
  - il join naturale è basato sui *nomi* degli attributi
  - equi-join e theta-join sono basati sui *valori*

# Theta-Join: Esempio

## Impiegati

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B

## Progetti

Codice	Nome
A	Venere
B	Marte

## Impiegati $\triangleright \triangleleft$ Progetto=Codice Progetti

Impiegato	Progetto	Codice	Nome
Rossi	A	A	Venere
Neri	A	A	Venere
Neri	B	B	Marte

# Join naturale ed equijoin

**Impiegati**

Impiegato

Reparto

**Reparti**

Codice

Capo

$\pi$  Impiegato, reparto, capo (Impiegati  $\bowtie$ <sub>reparto=codice</sub> Reparti)

=

$\pi$  Impiegato, reparto, capo ( $\sigma$ <sub>reparto=codice</sub>(Impiegati x Reparti))

=

Impiegati  $\bowtie$  ( $\rho$  Codice  $\leftarrow$  Reparto (Reparti) )



# Join e proiezioni: perdita di informazioni

- $R_1(X_1), R_2(X_2)$

$$\Pi_{X_1}(R_1 \bowtie R_2) \subseteq R_1$$

- $R(X), X = X_1 \cup X_2$

$$(\Pi_{X_1}(R)) \bowtie (\Pi_{X_2}(R)) \supseteq R$$

# Join e proiezioni: problemi

$R_1$

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

$R_2$

Reparto	Capo
B	Mori
C	Bruni

$R_1 \bowtie R_2$

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

$\Pi_{x_1}(R_1 \bowtie R_2) \subseteq R_1$

Impiegato	Reparto
Neri	B
Bianchi	B

$\Pi_{x_2}(R_1 \bowtie R_2) \subseteq R_2$

Reparto	Capo
B	Mori

# Proiezioni e join: problemi

**R**

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Bruni
Verdi	A	Bini

**$\Pi_{x_2}(R)$**

Reparto	Capo
B	Mori
B	Bruni
A	Bini

**$\Pi_{x_1}(R)$**

Impiegato	Reparto
Neri	B
Bianchi	B
Verdi	A

**$(\Pi_{x_1}(R)) \bowtie (\Pi_{x_2}(R)) \supseteq R$**

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Bruni
Neri	B	Bruni
Bianchi	B	Mori
Verdi	A	Bini

# Contenuti della lezione

- linguaggi di interrogazione
- algebra relazionale
- operatori insiemistici
- Ridenominazione
- Selezione
- Proiezione
- Join
- Theta Join
- Query

# Interrogazioni- query

- Un'interrogazione è una funzione  $E(R)$  che applicata ad istanze di una base di dati  $R$  produce una relazione su un dato insieme di attributi  $X$ .
- Le interrogazioni su uno schema di base di dati  $R$  in algebra relazionale sono espressioni i cui atomi (le variabili) sono relazioni in  $R$  o costanti.

**Le interrogazioni sono in pratica espressioni di relazioni che producono relazioni**

# Esempi: Schema relazionale

## Impiegati

Matricola	Cognome	Età	Stipendio
101	Rossi	34	40
103	Bianchi	23	35
104	Neri	38	61
210	Celli	49	60
231	Bisi	50	60
252	Bini	44	70
301	S. Rossi	34	70
375	M. Rossi	50	65

## Supervisione

Capo	Impiegato
210	101
210	103
210	104
301	210
301	231
375	252

# Esempio

- Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 milioni

$\sigma_{\text{Stipendio} > 40}(\text{Impiegati})$

Matricola	Cognome	Età	Stipendio
104	Neri	38	61
210	Celli	49	60
231	Bisi	50	60
252	Bini	44	70
301	S. Rossi	34	70
375	M. Rossi	50	65

# Esempio

- Trovare matricola, nome ed età degli impiegati che guadagnano più di 40 milioni

$\pi$  Matricola, Nome, Età ( $\sigma$  Stipendio $>40$ (Impiegati))

Matricola	Cognome	Età
104	Neri	38
210	Celli	49
231	Bisi	50
252	Bini	44
301	S. Rossi	34
375	M. Rossi	50



# Esempio:

- Trovare le matricole dei capi degli impiegati che guadagnano più di 40 milioni

$\pi_{\text{Capo}}(\text{Supervisione} \bowtie_{\text{Impiegato=Matricola}} (\sigma_{\text{Stipendio}>40}(\text{Impiegati})))$

- nome e stipendio dei capi degli impiegati che guadagnano più di 40mila euro

$\text{Supervisione} \bowtie_{\text{Impiegato=Matricola}} (\sigma_{\text{Stipendio}>40}(\text{Impiegati})) = A$

$\pi_{\text{Cognome, Stimeptio}}(\text{Impiegati} \bowtie_{\text{capo=Matricola}} A)$

# Esercizi

- Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 milioni
- Trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo
- Trovare le matricole dei capi i cui impiegati guadagnano **tutti** più di 40 milioni

# Soluzione Esercizi

- Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 milioni

$\pi_{\text{Nome,Stipendio}} (\text{Impiegati} \bowtie_{\text{Matricola=Capo}} \pi_{\text{Capo}} (\text{Supervisione} \bowtie_{\text{Impiegato=Matricola}} (\sigma_{\text{Stipendio}>40} (\text{Impiegati}))))$

# Soluzione Esercizi

- Trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo

$\pi_{\text{Nome,Stip,MatrC,NomeC,StipC}} (\sigma_{\text{Stipendio} > \text{StipC}} ($

$\rho_{\text{MatrC,NomeC,StipC,EtàC}} \leftarrow \text{Matr,Nome,Stip,Età} (\text{Impiegati}) \bowtie$

$\text{MatrC=Capo} (\text{Supervisione} \bowtie_{\text{Impiegato=Matricola}} \text{Impiegati}))$

# Soluzione Esercizi

- Trovare le matricole dei capi i cui impiegati guadagnano **tutti** più di 40 milioni

$$\pi_{\text{Capo}}(\text{Supervisione}) -$$
$$\pi_{\text{Capo}}(\text{Supervisione}$$
$$\bowtie_{\text{Impiegato}=\text{Matricola}} (\sigma_{\text{Stipendio} <= 40}(\text{Impiegati})))$$

# Equivalenza di espressioni

Due espressioni sono **equivalenti** se:

- $E_1 \equiv_R E_2$  se  $E_1(r) = E_2(r)$  per ogni istanza  $r$  di  $R$   
(equivalenza dipendente dallo schema)
- $E_1 \equiv E_2$  se  $E_1 \equiv_R E_2$  per ogni schema  $R$   
(equivalenza assoluta)

L'equivalenza è importante in quanto consente di scegliere, a parità di risultato, l'operazione meno costosa.

# Equivalenze

- **Atomizzazione delle selezioni**

$$\sigma_{F_1 \wedge F_2}(E) \equiv \sigma_{F_1}(\sigma_{F_2}(E))$$

- **Idempotenza delle proiezioni**

$$\Pi_X(E) \equiv \Pi_X(\Pi_{XY}(E))$$

- **Anticipazione della selezione rispetto al join**

$$\sigma_F(E_1 \bowtie E_2) \equiv (\sigma_F(E_1) \bowtie \sigma_F(E_2))$$

# Equivalenze

- **Anticipazione della proiezione rispetto al join:**

$$\Pi_{X_1 Y_2} (E_1 \bowtie E_2) \equiv E_1 \bowtie \Pi_{Y_2} (E_2)$$

(se gli attributi in  $X_2 - Y_2$  non sono coinvolti nel join)

Allora (combinando con idempotenza delle proiezioni):

$$\Pi_Y (E_1 \bowtie_F E_2) \equiv \Pi_Y (\Pi_{Y_1} E_1 \bowtie_F \Pi_{Y_2} E_2)$$

dove  $Y_1$  e  $Y_2$  sono gli attributi di  $X_1$  e  $X_2$  compresi in  $Y$  o coinvolti nel join.

In pratica è possibile ignorare in ciascuna relazione gli attributi non compresi in  $Y$  e non coinvolti nel join

- **Inglobamento di una selezione in un prodotto cartesiano a formare un join:**

$$s_F (E_1 \bowtie E_2) \equiv E_1 \bowtie_F E_2$$



# Equivalenze

- **Distributività** della selezione rispetto all'unione:

$$s_F (E_1 \cup E_2) \equiv s_F (E_1) \cup s_F (E_2)$$

- **Distributività della selezione rispetto alla differenza:**

$$s_F (E_1 - E_2) \equiv s_F (E_1) - s_F (E_2)$$

- Distributività della proiezione rispetto **all'unione**:

$$P_X (E_1 \cup E_2) \equiv P_X (E_1) \cup P_X (E_2)$$

**NB** La proiezione **NON** è distributiva rispetto alla differenza

- Tutti gli operatori binari eccetto la differenza godono delle proprietà associativa e commutativa.

# Equivalenze

- Corrispondenze fra operatori insiemistici e selezioni complesse

$$\begin{aligned}\sigma_{F_1 \vee F_2}(R) &\equiv \sigma_{F_1}(R) \cup \sigma_{F_2}(R) \\ \sigma_{F_1 \wedge F_2}(R) &\equiv \sigma_{F_1}(R) \cap \sigma_{F_2}(R) \\ \sigma_{F_1 \wedge \neg F_2}(R) &\equiv \sigma_{F_1}(R) - \sigma_{F_2}(R)\end{aligned}$$

- Proprietà distributiva del join rispetto all'unione:

$$E \bowtie (E_1 \cup E_2) \equiv (E \cup E_1) \bowtie (E \cup E_2)$$

# Algebra con valori nulli

- Estensione degli operatori logici ad una logica a 3 valori (VERO, FALSO, SCONOSCIUTO (U))

not		and	V	U	F	or	V	U	F
F	V	V	V	U	F	V	V	V	V
U	U	U	U	U	F	U	V	U	U
V	F	F	F	F	F	F	V	U	F

# Algebra con valori nulli

- A IS NULL è vero su una ennupla  $t$  se il valore di  $t$  su  $A$  è nullo; falso se è specificato.
- A IS NOT NULL è vero su una tupla  $t$  se il valore di  $t$  su  $A$  è specificato; falso se è nullo.

$\sigma_{\text{Età}>30}$  (Persone) restituisce le persone la cui età è nota e  $> 30$  anni

$\sigma_{\text{Età}>30 \vee \text{Età IS NULL}}$  (Persone) restituisce le persone che potrebbero avere più di 30 anni

# Viste (relazioni derivate)

Rappresentazioni diverse per gli stessi dati (**schema esterno**)

- **Relazioni di base:** contenuto autonomo
- **Relazioni derivate:** relazioni il cui contenuto è funzione del contenuto di altre relazioni (definito per mezzo di interrogazioni)

# Viste

- **Relazioni Virtuali (Viste)**

Relazioni definite mediante funzioni o espressioni del linguaggio di interrogazione, non memorizzate ma utilizzabili come se lo fossero. Devono essere ricalcolate tutte le volte.

- **Viste materializzate**

Relazioni virtuali effettivamente memorizzate nella base di dati.

Immediatamente disponibili ma critiche per il mantenimento dell'allineamento con le relazioni da cui derivano. Non sono supportate dai DBMS.

# Viste: Vantaggi

- Permettono di mostrare a un utente le sole componenti della base di dati che interessano
- Espressioni molto complesse possono essere definite come viste
- **Sicurezza:** è possibile definire dei diritti di accesso relativi ad una vista (e quindi ad una particolare porzione della base di dati)
- In caso di ristrutturazione della base di dati, le “vecchie” relazioni possono essere di nuovo ricavate mediante viste, consentendo l’uso di applicazioni che fanno riferimento al vecchio schema

# Viste, esempio

- una vista:

Supervisione =

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

## Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

## Direzione

Reparto	Capo
A	Mori
B	Bruni



# Interrogazioni sulle viste

- Sono eseguite sostituendo alla vista la sua definizione:

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

viene eseguita come

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

# 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

# Viste come strumento di programmazione

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

```
PROJ Impiegato (Afferenza JOIN Direzione) JOIN
      REN ImpR,RepR ← Imp,Reparto (
SEL Impiegato='Rossi' (Afferenza JOIN Direzione))
```

- Con la vista:

```
PROJ Impiegato (Supervisione) JOIN
      REN ImpR,RepR ← Imp,Reparto (
SEL Impiegato='Rossi' (Supervisione))
```

# Viste e aggiornamenti, attenzione

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

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

# 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 aggiornamenti sono ammissibili sulle viste