# Join, una difficoltà

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

| Reparto | Capo |
|---|---|
| B | Mori |
| C | Bruni |

| Impiegato | Reparto | Capo |
|---|---|---|
| Neri | B | Mori |
| Bianchi | B | Mori |

- alcune ennuple non contribuiscono al risultato: vengono "tagliate fuori"

# Join esterno

- Il join esterno estende, con valori nulli, le ennuple che verrebbero tagliate fuori da un join (interno)
- esiste in tre versioni:
  - sinistro, destro, completo

# Join, esterno

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

| Reparto | Capo |
|---|---|
| B | Mori |
| C | Bruni |

- Vedere anche su Relax (nella base di dati mostrata per gli esempi, usare R1 e R2)

## Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

## Reparti

| Reparto | Capo |
|---|---|
| B | Mori |
| C | Bruni |

## Impiegati $\text{JOIN}_{\text{LEFT}}$ Reparti

| Impiegato | Reparto | Capo |
|---|---|---|
| Neri | B | Mori |
| Bianchi | B | Mori |
| Rossi | A | *NULL* |

## Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

## Reparti

| Reparto | Capo |
|---|---|
| B | Mori |
| C | Bruni |

## Impiegati $JOIN_{RIGHT}$ Reparti

| Impiegato | Reparto | Capo |
|---|---|---|
| Neri | B | Mori |
| Bianchi | B | Mori |
| *NULL* | C | Bruni |

## Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

## Reparti

| Reparto | Capo |
|---|---|
| B | Mori |
| C | Bruni |

## Impiegati $JOIN_{FULL}$ Reparti

| Impiegato | Reparto | Capo |
|---|---|---|
| Neri | B | Mori |
| Bianchi | B | Mori |
| Rossi | A | *NULL* |
| *NULL* | C | Bruni |

- Su Relax il risultato ha una forma leggermente diversa, ma il concetto è lo stesso

# Join e proiezioni

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

| Reparto | Capo |
|---|---|
| B | Mori |
| C | Bruni |

| Impiegato | Reparto | Capo |
|---|---|---|
| Neri | B | Mori |
| Bianchi | B | Mori |

| Impiegato | Reparto |
|---|---|
| Neri | B |
| Bianchi | B |

| Reparto | Capo |
|---|---|
| B | Mori |

# Join e proiezioni

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

$$PROJ_{X_1}(R_1 \text{ JOIN } R_2) \subseteq R_1$$

# Proiezioni e join

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

| Impiegato | Reparto |
|---|---|
| Neri | B |
| Bianchi | B |
| Verdi | A |

| Reparto | Capo |
|---|---|
| B | Mori |
| B | Bruni |
| A | Bini |

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

# Join e proiezioni

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

$$PROJ_{X_1}(R_1 \text{ JOIN } R_2) \subseteq R_1$$

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

$$(PROJ_{X_1}(R)) \text{ JOIN } (PROJ_{X_2}(R)) \supseteq R$$

# 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)

## Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

## Reparti

| Codice | Capo |
|---|---|
| A | Mori |
| B | Bruni |

## Impiegati JOIN Reparti

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

- Il prodotto cartesiano, in pratica, ha senso (quasi) solo se seguito da selezione:

$$SEL_{Condizione}\ (R_1\ JOIN\ R_2)$$

- L'operazione viene chiamata theta-join e indicata con

$$R_1\ JOIN_{Condizione}\ R_2$$

# Equi-join

- Se l'operatore di confronto nel theta-join è sempre l'uguaglianza (=) allora si parla di equi-join

Nota: ci interessa davvero l'equi-join, non il theta-join più generale

- Equi-join: prodotto cartesiano seguito da selezione di uguaglianza

## Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | A |
| Neri | B |
| Bianchi | B |

## Reparti

| Codice | Capo |
|---|---|
| A | Mori |
| B | Bruni |

## Impiegati $JOIN_{Reparto=Codice}$ Reparti

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

# Join naturale ed equi-join

- In pratica, ciò che ci interessa è l'equi-join
- Il join naturale lo abbiamo usato solo a fini didattici, perché i concetti sono più semplici
- Nelle interrogazioni "pratiche" useremo l'equi-join

# Equivalenza di espressioni

- Due espressioni sono equivalenti se producono risultati uguali fra loro qualunque su ogni istanza 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 $R_1$ )

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

# 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

# Viste (relazioni derivate)

- Relazioni di base: contenuto autonomo, le relazioni nella base di dati
- Relazioni derivate:
  - relazioni il cui contenuto è funzione del contenuto di altre relazioni (definito per mezzo di interrogazioni)
- Le relazioni derivate possono essere definite su altre derivate, ma …

# 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)

# 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

Nota bene:

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

Vantaggi:

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

# Viste come strumento di programmazione

- Trovare gli impiegati che hanno lo stesso capo di Rossi (vedremo meglio il concetto più avanti)
- 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:

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

# Un servizio online per esercitazioni in algebra relazionale

(Io abbiamo già visto ma ripetiamo i dettagli per comodità – ora è necessario usare lo strumento)

- RelaX
  - http://dbis-uibk.github.io/relax/calc
- Verrà proposto un “homework” il cui svolgimento sarà necessario per partecipare alla prova parziale

# RelaX

- Utilizza una sintassi molto simile a quella vista a lezione e sul libro
- L'editor aiuta nella scrittura degli operatori e dei nomi di relazione e di attributo (basta cliccare sul simbolo desiderato)
- Talvolta è utile scrivere direttamente – allora attenzione a maiuscole e minuscole (è "case-sensitive")
- Le espressioni sono talvolta di lettura non semplice, perché tutto su una linea, senza "pedici":
  - scriviamo σ Stipendio>40 (Impiegati) invece di

    $\sigma_{\text{Stipendio>40}}$ (Impiegati)
- Attenzione agli spazi (talvolta lo strumento si confonde) e spesso è utile qualche parentesi in più
- Una differenza nella "assegnazione"; serve una "ridenominazione" esplicita della relazione; invece di

  Capi := Impiegati

  dobbiamo scrivere

  Capi = ρ Capi (Impiegati)

# Rappresentazione grafica

- RelaX fornisce anche una rappresentazione grafica delle espressioni sotto forma di albero, molto espressiva
- Ogni operatore è un nodo, con uno o due nodi discendenti (a seconda che abbia uno o due operandi) e le foglie sono relazioni nella base di dati
- Nei lucidi seguenti sono mostrate le interrogazioni discusse in aula e per ciascuna è mostrata la formulazione mostrata in aula, quelle in RelaX (molto simile) e l'albero generato da RelaX

# Dati

- Accedendo al servizio si possono specificare interrogazioni su una base di dati
  - fra quelle disponibili sul servizio, oppure
  - su una “caricata” dall’utente
- Per i primi esempi (in questa presentazione), le basi di dati sono state predisposte e possono essere caricate selezionando il link “Select DB ..” (in alto a sinistra) e inserendo nel campo “Load dataset stored in a gist” il relativo link

  1a9dc6cd0f3478388fc177dfc9b5a314 (prima bd)
  b7a8eac38317e0d6a7f0b904a9a10bd3 (seconda bd)
- oppure, più semplicemente richiamando RelaX con l’url:

  http://dbis-uibk.github.io/relax/calc/gist/1a9dc6cd0f3478388fc177dfc9b5a314
  http://dbis-uibk.github.io/relax/calc/gist/b7a8eac38317e0d6a7f0b904a9a10bd3

- Ulteriori basi di dati (data-set nella terminologia di RelaX) possono essere predisposti con una sintassi molto semplice e caricati su github (vedere l’help)

```
Impiegati = {
  Matricola, Nome, Eta:number, Stipendio:number
              7309, Rossi, 34, 45
              5998, Bianchi, 37, 38
              9553, Neri, 42, 35
              5698, Bruni, 43, 42
              4076, Mori, 45, 50
              8123, Lupi, 46, 60
}

Supervisione = {
  Impiegato, Capo
              7309, 5698
              5998, 5698
              9553, 4076
              5698, 4076
              4076, 8123
}
```

# Esempi

Impiegati

| Matricola | Nome | Età | Stipendio |
|---|---|---|---|
| 7309 | Rossi | 34 | 45 |
| 5998 | Bianchi | 37 | 38 |
| 9553 | Neri | 42 | 35 |
| 5698 | Bruni | 43 | 42 |
| 4076 | Mori | 45 | 50 |
| 8123 | Lupi | 46 | 60 |

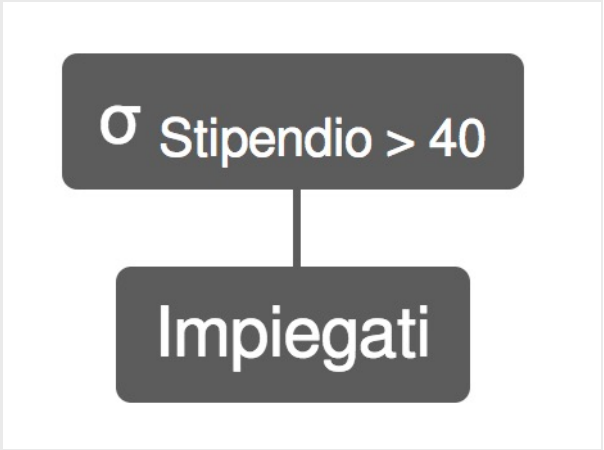
Supervisione

| Impiegato | Capo |
|---|---|
| 7309 | 5698 |
| 5998 | 5698 |
| 9553 | 4076 |
| 5698 | 4076 |
| 4076 | 8123 |

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

$SEL_{Stipendio>40}(Impiegati)$

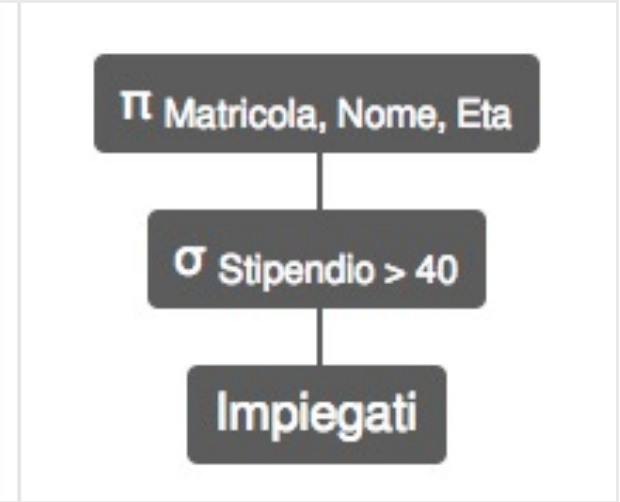
$\sigma$ Stipendio>40 (Impiegati))



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

$PROJ_{Matricola,\ Nome,\ Età}\ (SEL_{Stipendio>40}(Impiegati))$

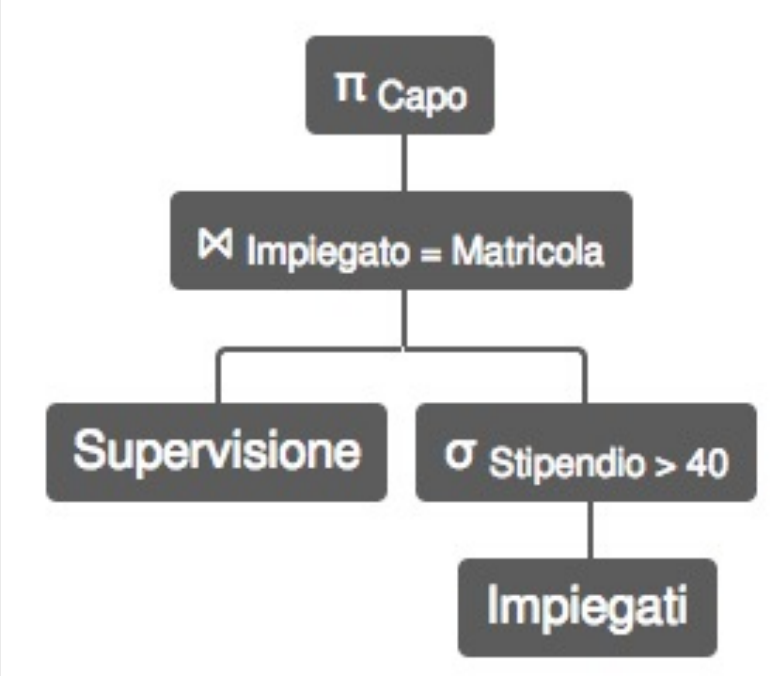
$\pi$ Matricola, Nome, Eta ($\sigma$ Stipendio>40 (Impiegati))



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

$$PROJ_{Capo}\,(Supervisione\ JOIN_{Impiegato=Matricola}\ (SEL_{Stipendio>40}(Impiegati)))$$

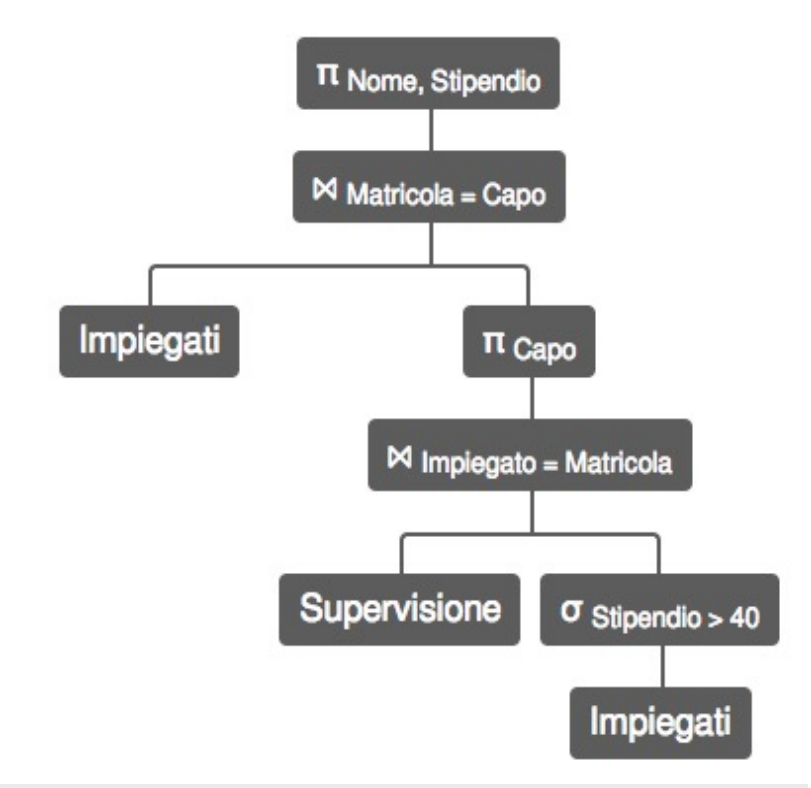
$$\pi_{Capo}\ ((Supervisione) \bowtie_{Impiegato=Matricola} (\sigma_{Stipendio>40}\ (Impiegati)))$$



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

$PROJ_{Nome,Stipendio}$ (
Impiegati JOIN $_{Matricola=Capo}$
$PROJ_{Capo}$(Supervisione
JOIN $_{Impiegato=Matricola}$
($SEL_{Stipendio>40}$(Impiegati))))

π Nome,Stipendio (
Impiegati ⋈ Matricola = Capo
(π Capo ((Supervisione)
⋈ Impiegato=Matricola
(σ Stipendio>40 (Impiegati)))))



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

- un po' complessa, vediamo prima un'altra interrogazione con caratteristiche simili, ma più semplice

- Trovare matricola, nome e stipendio dei capi degli impiegati che guadagnano più di 40; per ciascuno, mostrare, matricola, nome e stipendio anche dell'impiegato

- Il problema:
  - cii interessano, insieme, valori di uno stesso attributo, ma di ennuple diverse