# Algebra relazionale

Daniele Riboni
Università degli Studi di Cagliari
Dipartimento di Matematica e Informatica

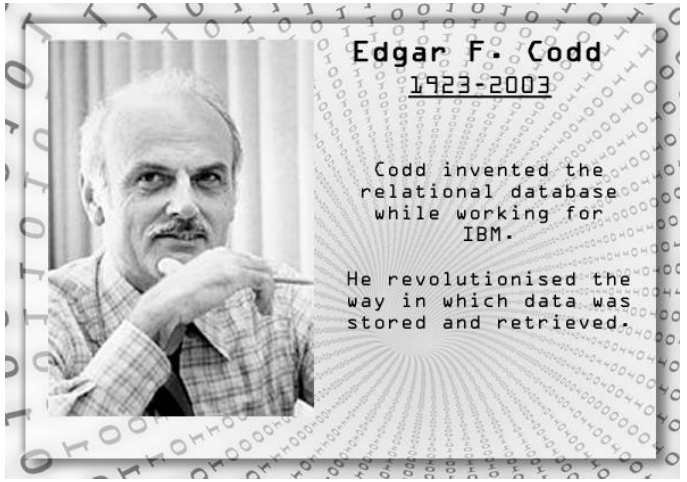
Slide tratte in parte dal materiale didattico del libro:
Atzeni, Ceri, Fraternali, Paraboschi, Torlone. Basi di dati. Quarta edizione. McGraw-Hill, 2014

# Algebra e Algebra relazionale

- Algebra (molto in generale):
  una branca della matematica che studia come rappresentare informazione numerica tramite simboli e le regole per manipolarli

- Algebra relazionale: un'algebra per modellare dati di strutture relazionali ed esprimere interrogazioni

# Perchè l'algebra relazionale?




- Definita nei primi anni '70
  da Edgar F. Codd ad IBM

- È il fondamento teorico
  di linguaggi di interrogazione
  per database relazionali (tra cui SQL)

- È la base per implementare e ottimizzare i DBMS
  reali

# Perchè l'algebra relazionale?

La studiamo di pari passo con SQL

- Ci aiuta a capire come scrivere le query
- Gli operatori "di base" sono gli stessi, ma in SQL ne abbiamo di aggiuntivi
  (in SQL riusciamo a fare query più complesse)

# 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 di interrogazione

- Algebra relazionale: procedurale
- SQL: (parzialmente) dichiarativo

# Algebra relazionale

- Insieme di operatori
  - su relazioni
  - che producono relazioni
  - e possono essere composti

# Operatori dell'algebra relazionale

- Unione, intersezione, differenza
- Ridenominazione
- Selezione e proiezione
- Join (join naturale, prodotto cartesiano, theta-join, equi-join)

# Operatori insiemistici

- Le relazioni sono insiemi
- I risultati devono essere relazioni
- È possibile applicare unione, intersezione, differenza solo a relazioni definite sugli stessi attributi

# Unione

Laureati

| Matricola | Nome | Età |
|---|---|---|
| 7274 | Rossi | 42 |
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |

Specialisti

| Matricola | Nome | Età |
|---|---|---|
| 9297 | Neri | 33 |
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |

Laureati ∪ Specialisti

| Matricola | Nome | Età |
|---|---|---|
| 7274 | Rossi | 42 |
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |
| 9297 | Neri | 33 |

# Differenza

Laureati

| Matricola | Nome | Età |
|---|---|---|
| 7274 | Rossi | 42 |
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |

Specialisti

| Matricola | Nome | Età |
|---|---|---|
| 9297 | Neri | 33 |
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |

Laureati – Specialisti

| Matricola | Nome | Età |
|---|---|---|
| 7274 | Rossi | 42 |

# Intersezione

Laureati

| Matricola | Nome | Età |
|---|---|---|
| 7274 | Rossi | 42 |
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |

Specialisti

| Matricola | Nome | Età |
|---|---|---|
| 9297 | Neri | 33 |
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |

Laureati ∩ Specialisti

| Matricola | Nome | Età |
|---|---|---|
| 7432 | Neri | 54 |
| 9824 | Verdi | 45 |

# Un’unione sensata ma impossibile

Paternità

| Padre | Figlio |
|---|---|
| Adamo | Abele |
| Adamo | Caino |
| Abramo | Isacco |

Maternità

| Madre | Figlio |
|---|---|
| Eva | Abele |
| Eva | Set |
| Sara | Isacco |

Paternità $\cup$ Maternità

??

# Ridenominazione

- Mi permette di cambiare nome agli attributi e alle relazioni
- “Modifica lo schema" lasciando inalterate le istanze

Paternità

| Padre | Figlio |
|---|---|
| Adamo | Abele |
| Adamo | Caino |
| Abramo | Isacco |

$REN_{Genitore \leftarrow Padre}$ (Paternità)

| Genitore | Figlio |
|---|---|
| Adamo | Abele |
| Adamo | Caino |
| Abramo | Isacco |

Paternità

| Padre | Figlio |
|---|---|
| Adamo | Abele |
| Adamo | Caino |
| Abramo | Isacco |

Maternità

| Madre | Figlio |
|---|---|
| Eva | Abele |
| Eva | Set |
| Sara | Isacco |

$REN_{Genitore \leftarrow Padre}$ (Paternità)

| Genitore | Figlio |
|---|---|
| Adamo | Abele |
| Adamo | Caino |
| Abramo | Isacco |

$REN_{Genitore \leftarrow Madre}$ (Maternità)

| Genitore | Figlio |
|---|---|
| Eva | Abele |
| Eva | Set |
| Sara | Isacco |

$REN_{Genitore \leftarrow Padre}$ (Paternità)

| Genitore | Figlio |
|---|---|
| Adamo | Abele |
| Adamo | Caino |
| Abramo | Isacco |

$REN_{Genitore \leftarrow Madre}$ (Maternità)

| Genitore | Figlio |
|---|---|
| Eva | Abele |
| Eva | Set |
| Sara | Isacco |

$REN_{Genitore \leftarrow Padre}$ (Paternità)

$\cup$

$REN_{Genitore \leftarrow Madre}$ (Maternità)

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

Impiegati

| Cognome | Ufficio | Stipendio |
|---|---|---|
| Rossi | Roma | 55 |
| Neri | Milano | 64 |

Operai

| Cognome | Fabbrica | Salario |
|---|---|---|
| Bruni | Monza | 45 |
| Verdi | Latina | 55 |

$$\text{REN}_{\text{Sede, Retribuzione} \leftarrow \text{Ufficio, Stipendio}}(\text{Impiegati}) \cup \text{REN}_{\text{Sede, Retribuzione} \leftarrow \text{Fabbrica, Salario}}(\text{Operai})$$

| Cognome | Sede | Retribuzione |
|---|---|---|
| Rossi | Roma | 55 |
| Neri | Milano | 64 |
| Bruni | Monza | 45 |
| Verdi | Latina | 55 |

# Ridenominazione

- Necessaria quando devo incrociare dati presi dalla stessa relazione
- Sintassi formale (lettera RHO):

$$\rho_{R(A_1,\ldots,A_n)}(E)$$

Data la relazione E (o una generica espressione) avente n attributi, rinomino
E in R e i suoi attributi in A1, A2, ...,An

# Ridenominazione

- Se voglio rinominare solo pochi attributi posso usare la notazione più compatta:

REN $_{\text{Sede, Retribuzione} \leftarrow \text{Fabbrica, Salario}}$ (Operai)

oppure:

$$\rho_{Sede,Retribuzione \leftarrow Fabbrica,Salario}(Operai)$$

# Selezione

- “Filtro” sulle tuple in base a condizioni sui valori

- Produce un risultato che:
  - ha lo stesso schema dell'operando
  - contiene un sottoinsieme delle tuple dell'operando
    (quelle che soddisfano la condizione)

## Impiegati

| Matricola | Cognome | Filiale | Stipendio |
|---|---|---|---|
| 7309 | Rossi | Roma | 55 |
| 5998 | Neri | Milano | 64 |
| 9553 | Milano | Milano | 44 |
| 5698 | Neri | Napoli | 64 |

- Impiegati che
  - guadagnano più di 50
  - guadagnano più di 50 e lavorano a Milano
  - hanno lo stesso nome della filiale presso cui lavorano

# Selezione, sintassi e semantica

- Sintassi

  SEL $_{Condizione}$ (*Espressione*)

  - *Condizione*: espressione booleana (come quelle dei vincoli di ennupla)

- Semantica
  - il risultato contiene le ennuple dell'operando che soddisfano la condizione

- impiegati che guadagnano più di 50

$SEL_{Stipendio > 50}$ (Impiegati)

| Matricola | Cognome | Filiale | Stipendio |
|---|---|---|---|
| 7309 | Rossi | Roma | 55 |
| 5998 | Neri | Milano | 64 |
| 5698 | Neri | Napoli | 64 |

$SEL_{Stipendio > 50}$ (Impiegati)

- impiegati che guadagnano più di 50 e lavorano a Milano

$$SEL_{Stipendio > 50 \text{ AND } Filiale = 'Milano'}(Impiegati)$$

| Matricola | Cognome | Filiale | Stipendio |
|---|---|---|---|
| 5998 | Neri | Milano | 64 |

$$SEL_{Stipendio > 50 \text{ AND } Filiale = 'Milano'}(Impiegati)$$

# Sintassi alternativa

$\text{SEL}_{\text{Stipendio > 50 AND Filiale = 'Milano'}}$ (Impiegati)

è equivalente a (lettera SIGMA):

$$\sigma_{Stipendio>50 \wedge Filiale='Milano'}(Impiegati)$$

- impiegati che hanno lo stesso nome della filiale presso cui lavorano

$$\text{SEL}_{\text{Cognome} = \text{Filiale}}(\text{Impiegati})$$

| Matricola | Cognome | Filiale | Stipendio |
|---|---|---|---|
| 9553 | Milano | Milano | 44 |

$$\text{SEL}_{\text{Cognome} = \text{Filiale}}(\text{Impiegati})$$

# Selezione e proiezione

Operatori "ortogonali"

- Selezione:
  - "filtraggio" orizzontale
- Proiezione:
  - "filtraggio" verticale

# Proiezione

- Produce un risultato che
  - ha parte degli attributi dell'operando
  - contiene ennuple cui contribuiscono tutte le ennuple dell'operando

## Impiegati

| Matricola | Cognome | Filiale | Stipendio |
|---|---|---|---|
| 7309 | Neri | Napoli | 55 |
| 5998 | Neri | Milano | 64 |
| 9553 | Rossi | Roma | 44 |
| 5698 | Rossi | Roma | 64 |

- per tutti gli impiegati:
  - matricola e cognome
  - cognome e filiale

# Proiezione, sintassi e semantica

- Sintassi

$$\text{PROJ}_{\text{ListaAttributi}}\ (\text{Operando})$$

- Semantica
  - il risultato contiene le ennuple ottenute da tutte le ennuple dell'operando ristrette agli attributi nella lista

- matricola e cognome  di tutti gli impiegati

| Matricola | Cognome |
|---|---|
| 7309 | Neri |
| 5998 | Neri |
| 9553 | Rossi |
| 5698 | Rossi |

**PROJ $_{\text{Matricola, Cognome}}$ (Impiegati)**

- cognome e filiale  di tutti gli impiegati

| Cognome | Filiale |
|---|---|
| Neri | Napoli |
| Neri | Milano |
| Rossi | Roma |

PROJ $_{\text{Cognome, Filiale}}$ (Impiegati)

# Sintassi alternativa

- PROJ $_{\text{Cognome, Filiale}}$ (Impiegati)

è equivalente a (lettera PI):

$$\pi_{Cognome,Filiale}(Impiegati)$$

# Cardinalità delle proiezioni

- Una proiezione
  - contiene al più tante ennuple quante l'operando
  - può contenerne di meno!!

# Cardinalità delle proiezioni

- Consideriamo $PROJ_K(R)$

- Se K è una superchiave di R, quante tuple contiene $PROJ_k(R)$ ?

A) può contenere meno tuple di R
B) può contenere più tuple di R
C) esattamente tante tuple quanto R

Risposta corretta: C

# Selezione e proiezione

- Combinando selezione e proiezione, possiamo estrarre interessanti informazioni da una relazione

- matricola e cognome degli impiegati che guadagnano più di 50

| Matricola | Cognome |
|---|---|
| 7309 | Rossi |
| 5998 | Neri |
| 5698 | Neri |

$$PROJ_{Matricola,Cognome}\ (SEL_{Stipendio > 50}\ (Impiegati))$$

# Esercizio

Data la seguente relazione:

- Persona(CF, Età, CittàNatale, CittàAttuale)

si scriva l'espressione in algebra relazionale per rispondere alla seguente domanda:

- 1) In quali città abitano dei 30enni?

$$PROJ_{Citta'Attuale}(Sel_{Eta'=30}(Persona))$$

# Esercizio

Data la seguente relazione:

- Persona(CF, Età, CittàNatale, CittàAttuale)

si scriva l'espressione in algebra relazionale per rispondere alla seguente domanda:

- 2) In quali città è nato qualcuno ma non abita nessuno?

$$\rho_{citta' \leftarrow Citta' Natale} \prod_{Citta' Natale} (Persona) - \rho_{citta' \leftarrow Citta' Attuale} \prod_{Citta' Attuale} (Persona)$$

# Esercizio

Data la seguente relazione:

- Persona(CF, Età, CittàNatale, CittàAttuale)

si scriva l'espressione in algebra relazionale per rispondere alla seguente domanda:

- 3) Quali sono le città in cui è nato qualcuno e abita qualcuno?

$$\rho_{citta' \leftarrow Citta' Natale} \prod_{Citta' Natale} (Persona) \cap \rho_{citta' \leftarrow Citta' Attuale} \prod_{Citta' Attuale} (Persona)$$

# Esercizio

Data la seguente relazione:

- Persona(CF, Età, CittàNatale, CittàAttuale)
- 4) In quali città risiede qualche nativo?

$$PROJ_{Citta'Natale}(Sel_{Citta'Natale=Citta'Attuale}(Persona))$$

equivalente a:

$$\pi_{Citta'Natale}(\sigma_{Citta'Natale=Citta'Attuale}(Persona))$$

# Esercizio

Data la seguente relazione:

- Persona(CF, Età, CittàNatale, CittàAttuale)

si scriva l'espressione in algebra relazionale per rispondere alla seguente domanda:

- 5) Quali sono tutte le città presenti nella base di dati?

$$\rho_{citta' \leftarrow Citta' Natale} \prod_{Citta' Natale} (Persona) \cup$$
$$\rho_{citta' \leftarrow Citta' Attuale} \prod_{Citta' Attuale} (Persona)$$

# Esercizio

- Persona(CF, Età, CittàNatale, CittàAttuale)

si scriva l'espressione in algebra relazionale per rispondere alla seguente domanda:

- 6) In quali città abitano solo anziani (età >= 70)?

$$PROJ_{CittàAttuale}(Persona) - PROJ_{CittàAttuale}(Sel_{Età<70}(Persona))$$

- Combinando selezione e proiezione, possiamo estrarre informazioni da una relazione

- Non possiamo però correlare informazioni presenti in relazioni diverse, né informazioni in ennuple diverse di una stessa relazione

# Join

- Il join è l'operatore più interessante dell'algebra relazionale

- Permette di correlare dati in relazioni diverse

# Prove scritte in un concorso pubblico

- I compiti sono anonimi e ad ognuno è associata una busta chiusa con il nome del candidato

- Ciascun compito e la relativa busta vengono contrassegnati con uno stesso numero

| | |
|---|---|
| 1 | 25 |
| 2 | 13 |
| 3 | 27 |
| 4 | 28 |

| | |
|---|---|
| 1 | Mario Rossi |
| 2 | Nicola Russo |
| 3 | Mario Bianchi |
| 4 | Remo Neri |

| | |
|---|---|
| Mario Rossi | 25 |
| Nicola Russo | 13 |
| Mario Bianchi | 27 |
| Remo Neri | 28 |

| Numero | Voto |
|---|---|
| 1 | 25 |
| 2 | 13 |
| 3 | 27 |
| 4 | 28 |

| Numero | Candidato |
|---|---|
| 1 | Mario Rossi |
| 2 | Nicola Russo |
| 3 | Mario Bianchi |
| 4 | Remo Neri |

| Numero | Candidato | Voto |
|---|---|---|
| 1 | Mario Rossi | 25 |
| 2 | Nicola Russo | 13 |
| 3 | Mario Bianchi | 27 |
| 4 | Remo Neri | 28 |

# Join naturale

- Operatore binario (generalizzabile)

- Produce un risultato
  - sull'unione degli attributi degli operandi
  - con tuple costruite ciascuna a partire da una tupla di ognuno degli operandi

Impiegati

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

Reparti

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

Impiegati NAT_JOIN Reparti

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

- Ogni tupla contribuisce al risultato:
  - join completo

# Un join non completo

Impiegati

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

Reparti

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

Impiegati NAT_JOIN Reparti

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

# Un join vuoto

Impiegati

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

Reparti

| Reparto | Capo |
|---|---|
| D | Mori |
| C | Bruni |

Impiegati NAT_JOIN Reparti

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

# Un join completo, con n x m ennuple

Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | B |
| Neri | B |

Reparti

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

Impiegati NAT_JOIN Reparti

| Impiegato | Reparto | Capo |
|---|---|---|
| Rossi | B | Mori |
| Rossi | B | Bruni |
| Neri | B | Mori |
| Neri | B | Bruni |

# Sintassi alternativa

- Impiegati NAT_JOIN Reparti

è equivalente a (simbolo “bowtie”, papillon):

$$Impiegati \bowtie Reparti$$

# Cardinalità del join

- Il join naturale di $R_1$ e $R_2$ contiene un numero di ennuple …
  - compreso fra zero e il prodotto di $|R_1|$ e $|R_2|$
- se il join coinvolge una chiave di $R_2$, allora il numero di ennuple è …
  - compreso fra zero e $|R_1|$
- se il join coinvolge una chiave di $R_2$ e un vincolo di integrità referenziale (da $R_1$ a $R_2$), allora il numero di ennuple è ...
  - pari a $|R_1|$

# Cardinalità del join, 2

- Relazione $R_1(A,B)$ , relazione $R_2(B,C)$
- In generale

$$0 \leq |R_1 \text{ JOIN } R_2| \leq |R_1| \times |R_2|$$

- Se B è chiave in $R_2$

$$0 \leq |R_1 \text{ JOIN } R_2| \leq |R_1|$$

- Se B è chiave in $R_2$ ed esiste vincolo di integrità referenziale fra B (in $R_1$) e $R_2$:

$$|R_1 \text{ JOIN } R_2| = |R_1|$$

# 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 tuple 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

- Sinistro: mantiene tutte le ennuple del primo operando, estendendole con valori nulli, se necessario
- Destro: ... del secondo operando ...
- Completo: … di entrambi gli operandi ...

## Impiegati

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

## Reparti

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

## Impiegati NAT_JOIN $_{LEFT}$ 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 NAT_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 NAT_JOIN FULL Reparti

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

# Prodotto cartesiano

- Come nella teoria degli insiemi...
- Contiene sempre un numero di tuple pari al prodotto delle cardinalità degli operandi (le tuple 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 |

# Sintassi alternativa

- Impiegati JOIN Reparti

è equivalente a (simbolo “cross”, moltiplic.):

$$Impiegati \times Reparti$$

# Theta-join

- 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 :

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

# Perché "theta-join"?

- La condizione C è spesso una congiunzione (AND) di atomi di confronto $A_1 \vartheta A_2$ dove $\vartheta$ è uno degli operatori di confronto (=, >, <, …)

- È l'operatore di join che useremo più spesso

# Equi-join

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

Nota: di solito ci serve usare l'equi-join, non il theta-join più generale

## 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 |

# Sintassi alternativa

- Impiegati $\text{JOIN}_{\text{Reparto=Codice}}$ Reparti

è equivalente a:

$$Impiegati \bowtie_{Reparto=Codice} (Reparti)$$

Invece di = posso avere != , > , <= , ecc...
e comporre con AND (raramente OR)

# Possiamo esprimere il theta-join usando il prodotto cartesiano...

Impiegati

| Impiegato | Reparto |
|---|---|

Reparti

| Codice | Capo |
|---|---|

$$\text{Impiegati JOIN}_{\text{Reparto=Codice}}\ \text{Reparti}$$

$$\text{SEL}_{\text{Reparto=Codice}}(\ \text{Impiegati JOIN}\ \ \text{Reparti}\ )$$

# Possiamo esprimere il join naturale usando il prodotto cartesiano?

Impiegati

| Impiegato | Reparto |
|---|---|

Reparti

| Reparto | Capo |
|---|---|

Impiegati NAT_JOIN Reparti

$$PROJ_{Impiegato,Reparto,Capo}\ (\ SEL_{Reparto=Codice}\ (Impiegati\ JOIN\ \ REN_{Codice \leftarrow Reparto}\ (Reparti)\ )\ )$$

# 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

- Prima cosa che devo chiedermi: in quante relazioni devo cercare per trovare i dati che servono?
- Devo fare la query sul numero minimo di relazioni!
  - La query è più leggibile
  - La query è più “leggera” per il DBMS

$$SEL_{Stipendio>40}(Impiegati)$$

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

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

- Trovare i capi degli impiegati che guadagnano più di 40

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

- Ho capi duplicati?
- Equivalente a:

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

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

- 1) è equivalente a 2) ?

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

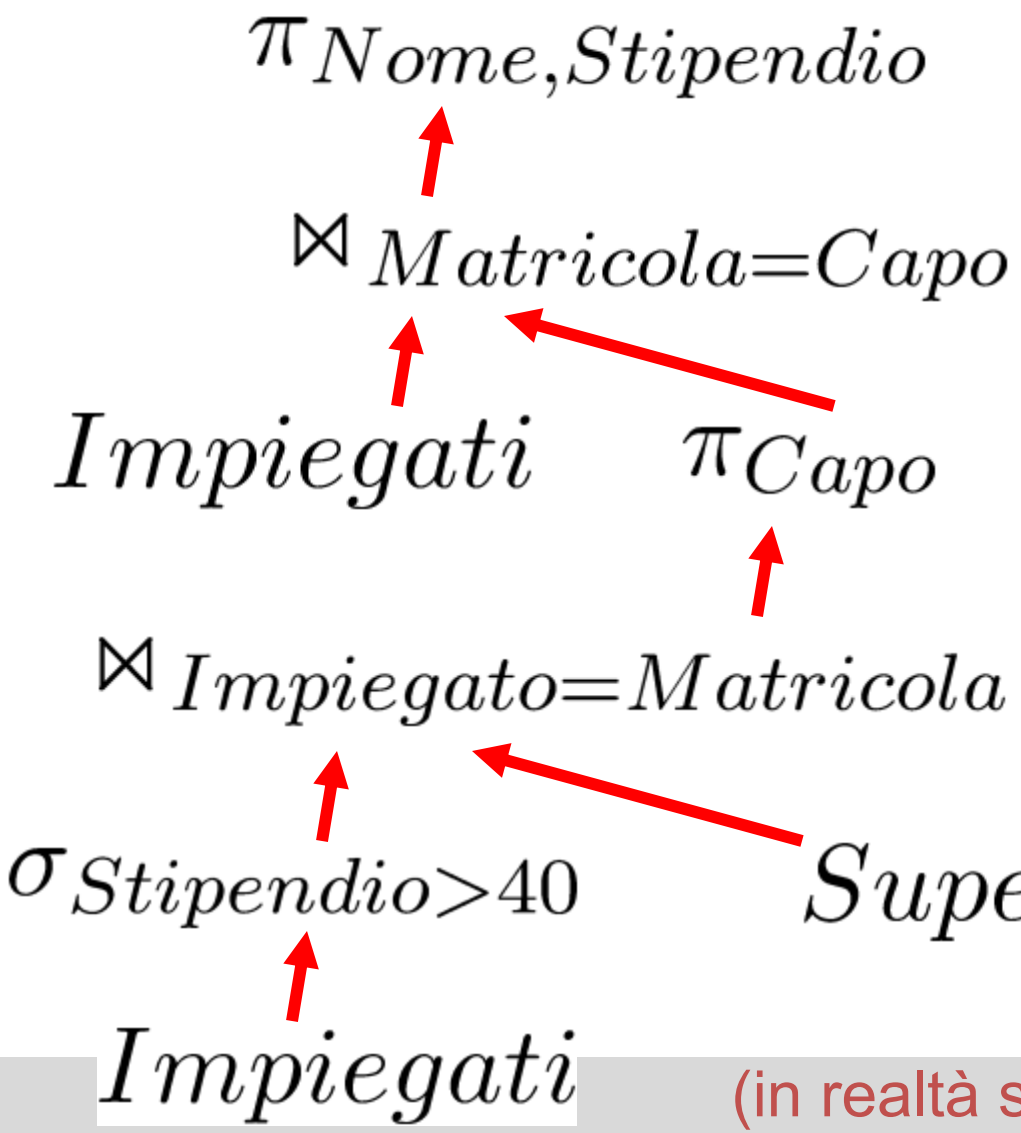
- No! In 2) sto applicando una selezione su Stipendo ad una relazione che ha un solo attributo Capo (è un errore sintattico)

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

$$\text{PROJ}_{\text{Nome,Stipendio}} ( \text{Impiegati JOIN}_{\text{Matricola=Capo}} \text{PROJ}_{\text{Capo}}(\text{Supervisione JOIN}_{\text{Impiegato=Matricola}} (\text{SEL}_{\text{Stipendio>40}}(\text{Impiegati}))))$$

- Mal di testa? Esiste una notazione alternativa: “expression tree”

# Expression tree




PROJ$_{Nome,Stipendio}$ (
Impiegati JOIN

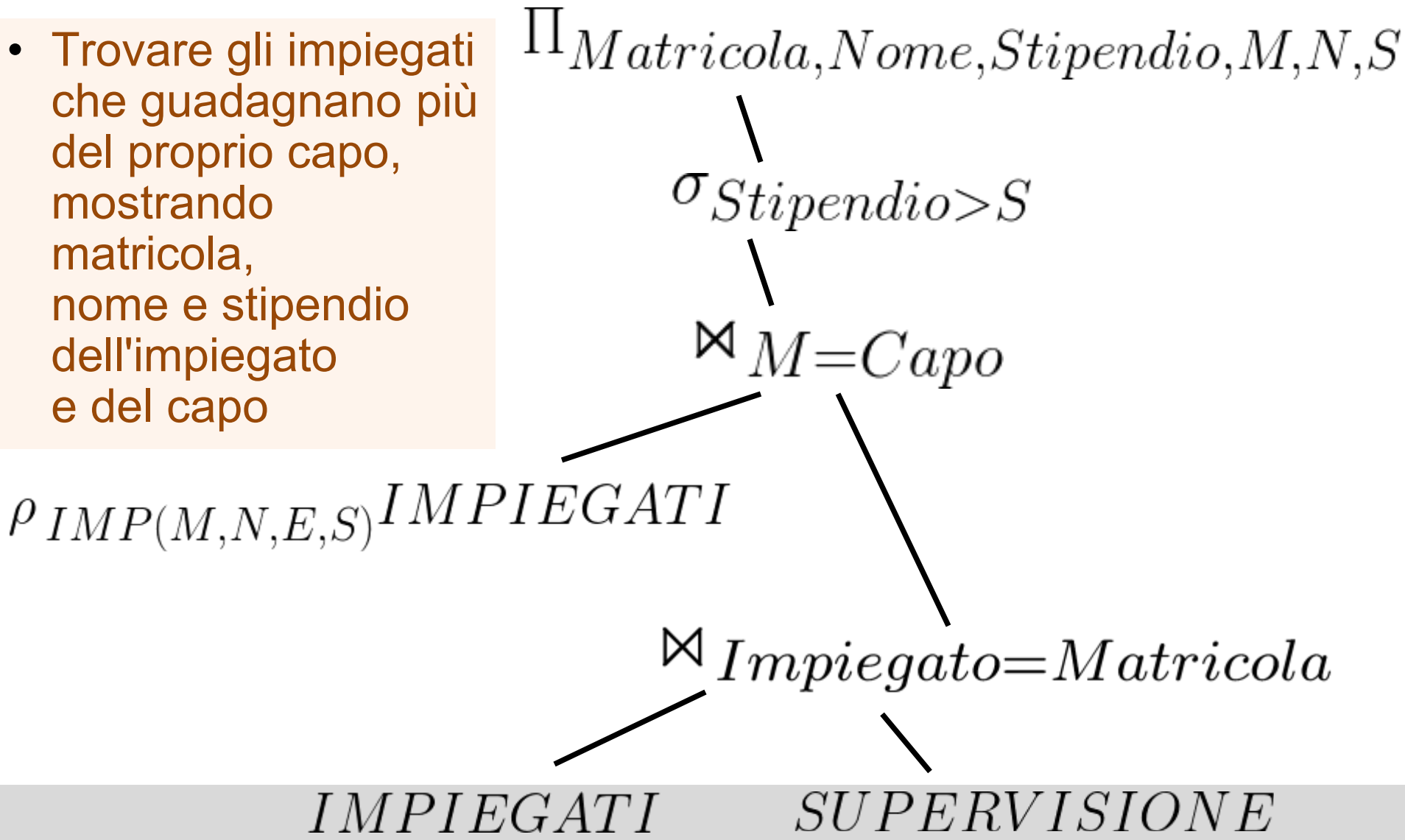
Matricola=Capo
PROJ$_{Capo}$(
Supervisione

JOIN $_{Impiegato=Matricola}$
(SEL$_{Stipendio>40}$(
Impiegati))))

(in realtà sono archi, non hanno frecce)

# Expression tree

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



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

$$\text{PROJ}_{\text{Matr,Nome,Stip,MatrC,NomeC,StipC}}(\text{SEL}_{\text{Stipendio>StipC}}(\text{REN}_{\text{MatrC,NomeC,StipC,EtàC} \leftarrow \text{Matr,Nome,Stip,Età}}(\text{Impiegati}) \text{ JOIN}_{\text{MatrC=Capo}} (\text{Supervisione JOIN}_{\text{Impiegato=Matricola}} \text{Impiegati}))))$$

# Expression tree

- Nell'expression tree si possono usare sia le lettere greche (es. π) che le label (es. PROJ)
- Nel compito scriverete sia l'expression tree che la formula algebrica

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

$$\text{PROJ}_{\text{Capo}}(\text{Supervisione}) - \text{PROJ}_{\text{Capo}}(\text{Supervisione JOIN}_{\text{Impiegato=Matricola}}(\text{SEL}_{\text{Stipendio} \leq 40}(\text{Impiegati})))$$

# 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

- 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

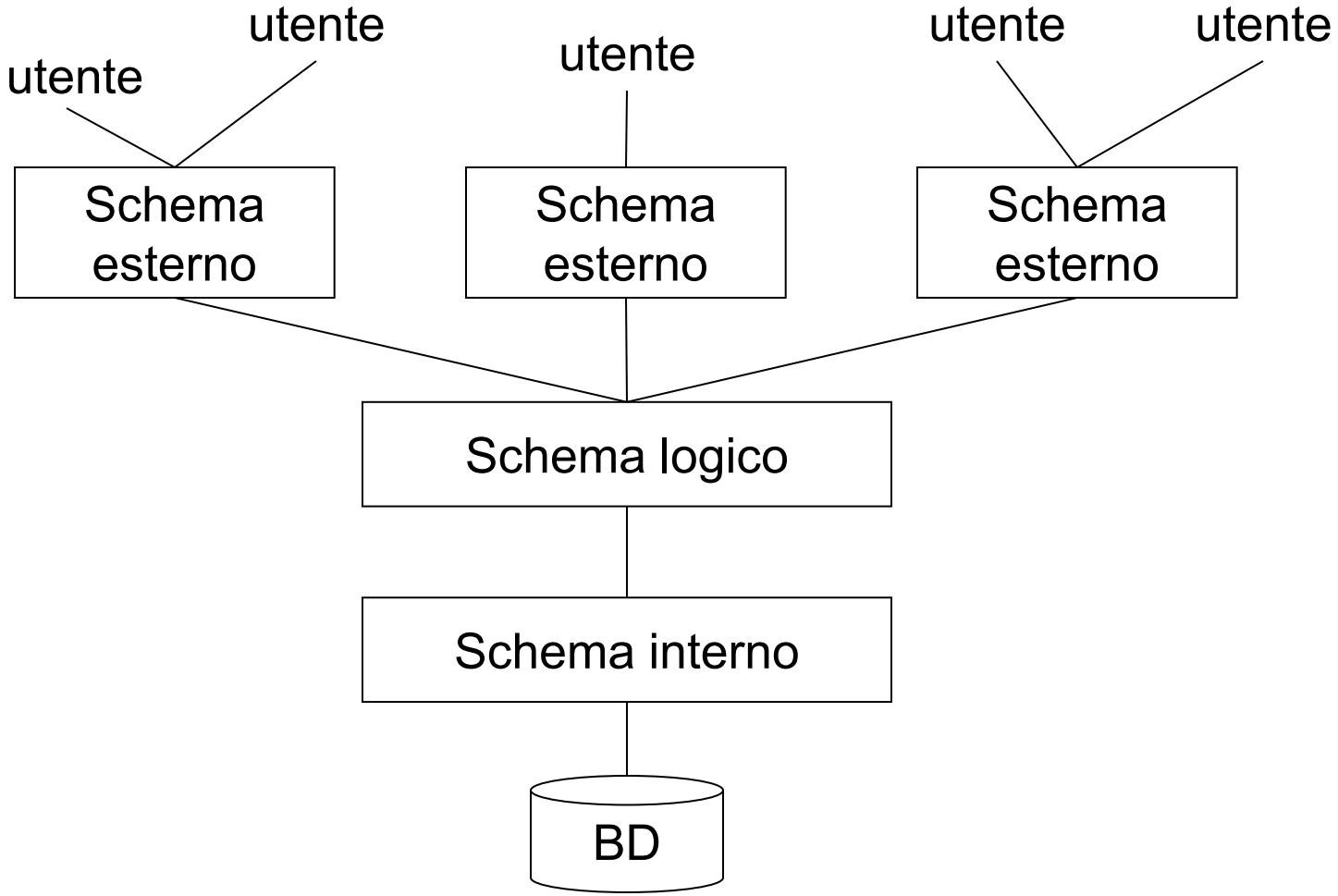
| Matricola | Cognome | Filiale | Età |
|---|---|---|---|
| 5998 | Neri | Milano | 45 |
| 9553 | Bruni | Milano | NULL |

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

# Viste (relazioni derivate)

- Rappresentazioni diverse per gli stessi dati (schema esterno)
- Relazioni derivate (viste):
  - 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 NAT_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

- Schema esterno: ogni utente vede solo
  - ciò che gli interessa
  - ciò che e' autorizzato a vedere (autorizzazioni)

- Strumento di programmazione:
  - si può semplificare la scrittura di interrogazioni

- Utilizzo di programmi esistenti su schemi ristrutturati

# Algebra relazionale: limiti

- Ci sono interrogazioni interessanti non esprimibili:
  - Calcolo di valori derivati: possiamo solo estrarre valori, non calcolarne di nuovi
    - A livello di tupla o di singolo valore (conversioni, somme, etc.)
    - Su insiemi di tuple (somme, medie, etc.)
  - Interrogazioni inerentemente ricorsive, come la chiusura transitiva

# Chiusura transitiva

Supervisione(Impiegato, Capo)

- Per ogni impiegato, trovare tutti i superiori (cioè il capo, il capo del capo, e cosi' via)

| Impiegato | Capo |
|---|---|
| Rossi | Lupi |
| Neri | Bruni |
| Lupi | Falchi |

| Impiegato | Superiore |
|---|---|
| Rossi | Lupi |
| Neri | Bruni |
| Lupi | Falchi |
| Rossi | Falchi |

- Non esiste in algebra relazionale la possibilità di esprimere questa interrogazione

# Ripasso online

Relational algebra 1 e 2 (Stanford Lagunita):

- https://www.youtube.com/watch?v=tii7xcFilOA
- https://www.youtube.com/watch?v=GkBf2dZAES0

# Esercizio

**MAGAZZINO** (CodiceM, Città, Via, NumeroCivico)

**AZIENDA** (CodiceA, NomeAzienda, Città, CAP, Telefono)

**PRODOTTO** (CodiceP, NomeProdotto, Azienda, Prezzo)
**FK**: Azienda = AZIENDA (CodiceA)

**STOCCAGGIO** (Prodotto, Magazzino, Quantità)
**FK**: Prodotto= PRODOTTO (CodiceP)
Magazzino= MAGAZZINO(CodiceM)

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare il nome di tutti i prodotti che costano meno di 10 Euro

$\mathbf{PROJ}_{\text{Nome}}(\mathbf{SEL}_{\text{Prezzo} < 10}(\text{PRODOTTO}))$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare i numeri civici dei magazzini di via Ripamonti a Milano

$$\mathbf{PROJ}_{\text{NumeroCivico}}(\mathbf{SEL}_{\text{Città=“Milano” AND Via=“Ripamonti”}}(\text{MAGAZZINO}))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO (CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO (Prodotto, Magazzino, Quantità)

Elencare il codice dei prodotti stoccati in un magazzino avente lo stesso codice del prodotto

$$\mathbf{PROJ}_{\text{Prodotto}}(\mathbf{SEL}_{\text{Prodotto=Magazzino}}(\text{STOCCAGGIO}))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare il nome dei prodotti delle aziende di Milano

$\mathbf{PROJ}_{\text{NomeProdotto}}$ (PRODOTTO
$\mathbf{JOIN}_{\text{CodiceA = Azienda}}$ ($\mathbf{SEL}_{\text{Città = 'Milano'}}$(AZIENDA)))

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare il codice di tutti i magazzini in cui è stoccato un prodotto di nome “iPhone”.

$$\mathbf{PROJ}_{\text{Magazzino}}\ (\text{STOCCAGGIO}$$

$$\mathbf{JOIN}_{\text{Prodotto = CodiceP}}\ ($$

$$\mathbf{SEL}_{\text{NomeProdotto = ‘iPhone’}}(\text{PRODOTTO})))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare la città di tutti i magazzini in cui è stoccato un prodotto di nome “iPhone”.

$$\mathbf{PROJ}_{\text{Città}}\,(\text{MAGAZZINO}\ \mathbf{JOIN}_{\text{Magazzino=CodiceM}}\,(\text{STOCCAGGIO}\ \mathbf{JOIN}_{\text{Prodotto = CodiceP}}\,(\mathbf{SEL}_{\text{NomeProdotto = ‘iPhone’}}(\text{PRODOTTO}))))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)

AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)

PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)

STOCCAGGIO(Prodotto, Magazzino, Quantità)

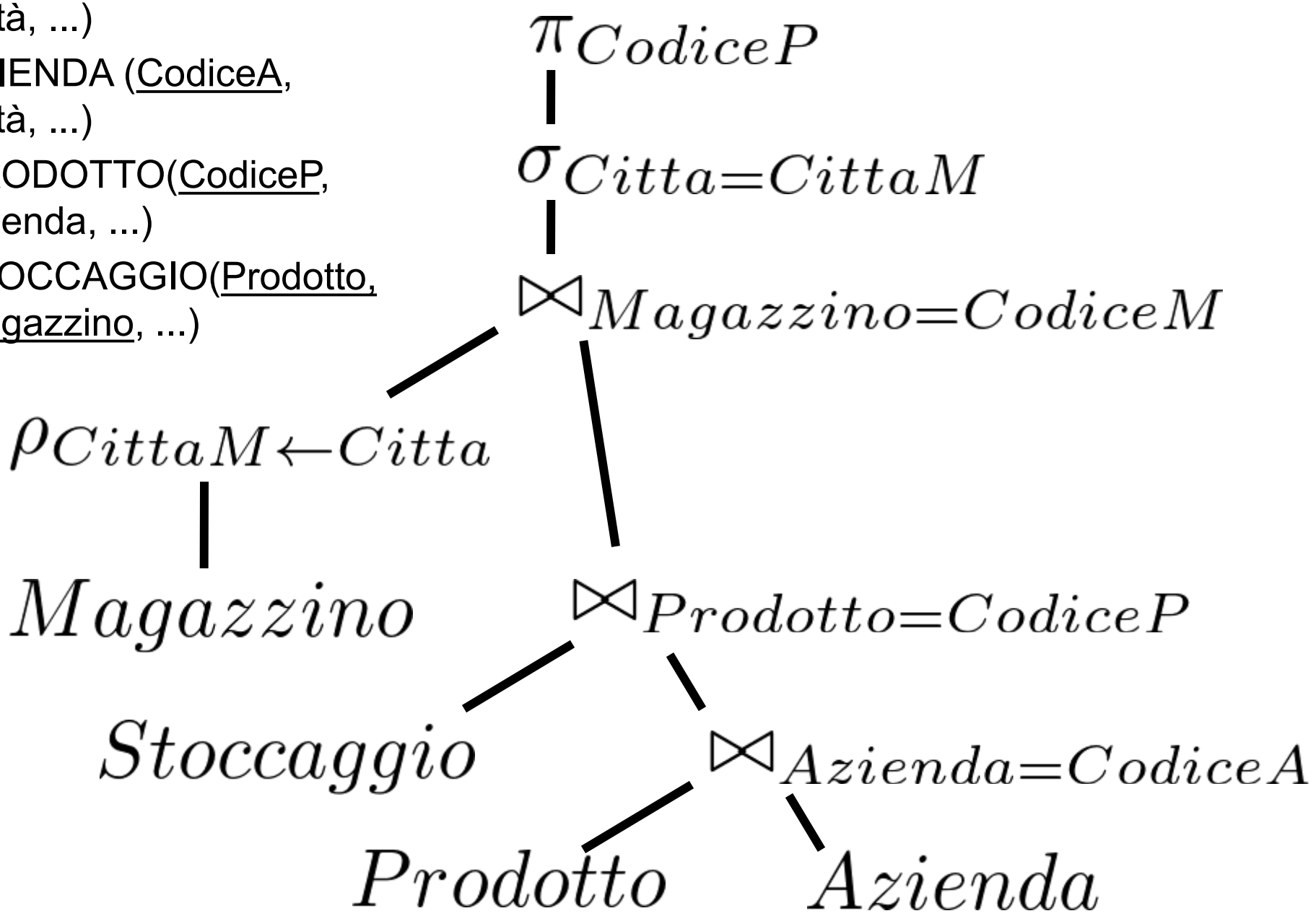
Elencare il codice di quei prodotti che sono stoccati nella città dell'azienda che li produce

MAGAZZINO (CodiceM, Città, ...)

AZIENDA (CodiceA, Città, ...)

PRODOTTO(CodiceP, Azienda, ...)

STOCCAGGIO(Prodotto, Magazzino, ...)



# Esercizio

Elencare il codice di quei prodotti che sono stoccati nella città dell'azienda che li produce

```
PROJ CodiceP(
  SEL Città=CittaM(
    REN CittaM ← Città(MAGAZZINO)
     JOIN Magazzino=CodiceM (
       STOCCAGGIO JOIN Prodotto=CodiceP (
         PRODOTTO JOIN Azienda=CodiceA (AZIENDA)
        )
      )
    )
)
```

# Schema “Stoccaggi”

**MAGAZZINO** (CodiceM, Città, Via, NumeroCivico)

**AZIENDA** (CodiceA, NomeAzienda, Città, CAP, Telefono)

**PRODOTTO (**CodiceP, NomeProdotto, Azienda, Prezzo)
**FK**: Azienda = AZIENDA (CodiceA)

**STOCCAGGIO (**Prodotto, Magazzino, Quantità)
**FK**: Prodotto= PRODOTTO (CodiceP)
Magazzino= MAGAZZINO(CodiceM)

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare il codice, il nome e il prezzo dei prodotti che costano più di 100

$$\mathbf{PROJ}_{\text{CodiceP, NomeProdotto, Prezzo}}(\mathbf{SEL}_{\text{Prezzo} > 100}(\text{PRODOTTO}))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare il nome e il prezzo dei prodotti dell'azienda di nome "Apple"

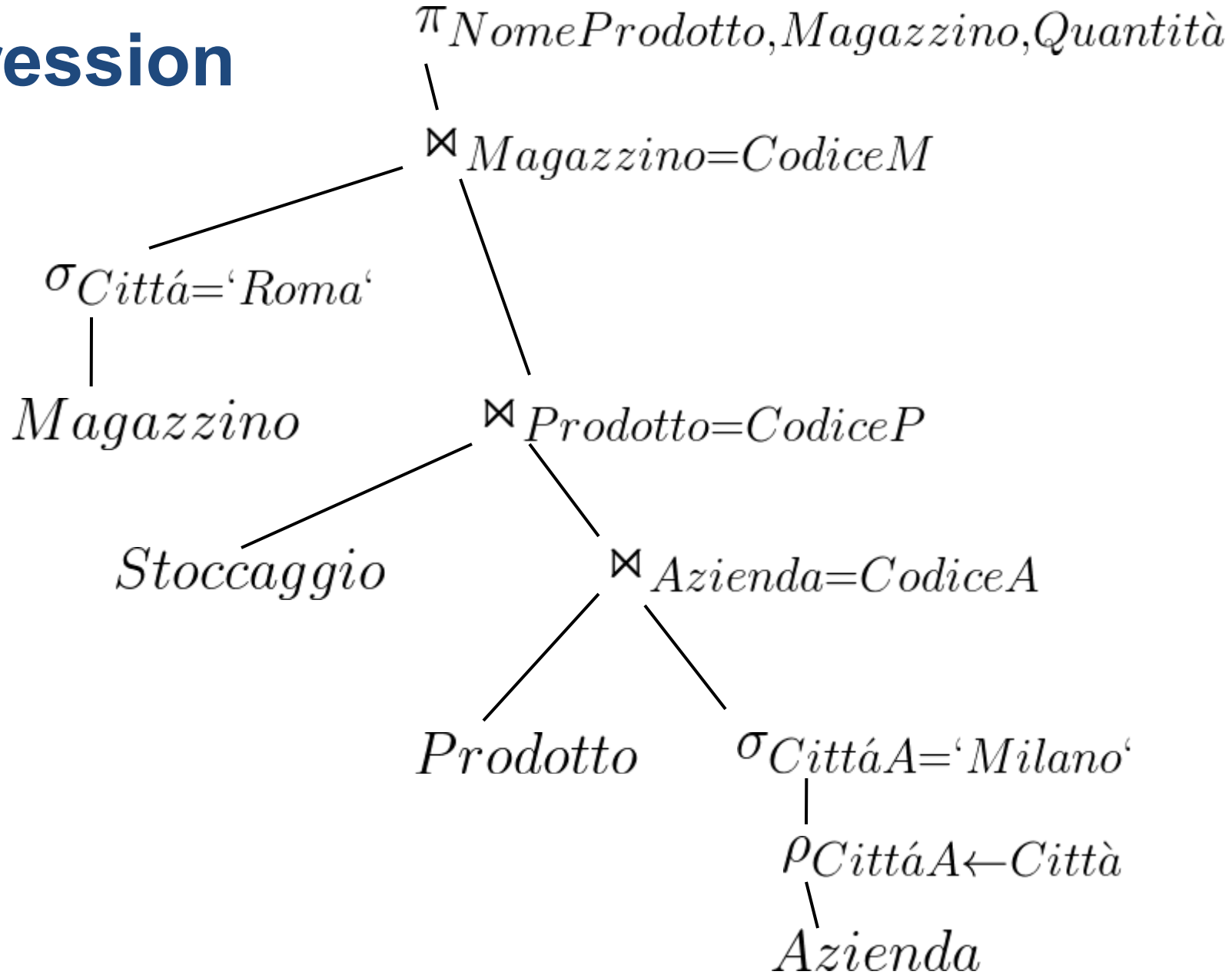
$$\mathbf{PROJ}_{\text{NomeProdotto, Prezzo}}(\text{PRODOTTO } \mathbf{JOIN}_{\text{Azienda = CodiceA}} (\mathbf{SEL}_{\text{NomeAzienda = "Apple"}}(\text{AZIENDA})))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Selezionare i prodotti di aziende milanesi stoccati in magazzini di Roma, indicando per ogni stoccaggio il nome del prodotto, il codice del magazzino, e la quantità stoccata

# Espression tree



# Espressione algebrica

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

$$\mathbf{PROJ}_{\text{NomeProdotto, Magazzino, Quantità}}($$

$$(\mathbf{SEL}_{\text{Città = “Roma”}}(\text{MAGAZZINO}))\mathbf{JOIN}_{\text{Magazzino = CodiceM}}$$

$$\text{STOCCAGGIO } \mathbf{JOIN}_{\text{Prodotto = CodiceP}} (\text{PRODOTTO}$$

$$\mathbf{JOIN}_{\text{Azienda = CodiceA}} (\mathbf{SEL}_{\text{CittàA = “Milano”}}($$

$$\mathbf{REN}_{\text{CittàA} \leftarrow \text{Città}} \text{AZIENDA})))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare il codice di tutti i magazzini in cui non è stoccato nessun prodotto

$$\mathbf{REN}_{\text{Magazzino} \leftarrow \text{CodiceM}}(\mathbf{PROJ}_{\text{CodiceM}}(\text{MAGAZZINO})) - \mathbf{PROJ}_{\text{Magazzino}}(\mathbf{SEL}_{\text{Quantità} > 0}(\text{STOCCAGGIO}))$$

# Esercizio

MAGAZZINO (CodiceM, Città, Via, NumeroCivico)
AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)
STOCCAGGIO(Prodotto, Magazzino, Quantità)

Elencare le città che sono sede di una azienda o di un magazzino (o di entrambi)

$\mathbf{PROJ}_{\text{Città}}(\text{MAGAZZINO}) \cup \mathbf{PROJ}_{\text{Città}}(\text{AZIENDA})$

# Esercizio

PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)

Elencare le coppie di prodotti diversi che hanno lo stesso nome (indicando il codice di ognuno)

PROD1 = $\mathbf{REN}_{C1,\ N1,\ A1,\ P1}$ (PRODOTTO)

PROD2 = $\mathbf{REN}_{C2,\ N2,\ A2,\ P2}$ (PRODOTTO)

$\mathbf{PROJ}_{C1,C2}$ ($\mathbf{SEL}_{(N1=N2)\ \wedge\ (C1\neq C2)}$ (PROD1 **x** PROD2))

(in questo caso l'uso di viste aiuta ma non è necessario)

# Esercizio

$\mathbf{PROJ}_{C1,C2}\ \mathbf{SEL}_{(N1=N2)\ \Lambda\ (C1\neq C2)}$ (PROD1 x PROD2 ))

Con (C1≠ C2) ottengo anche coppie ripetute:
Ad es. [33, 66] e [66, 33]

Per tenere una sola coppia, invece di ≠ uso >

$\mathbf{PROJ}_{C1,C2}\ \mathbf{SEL}_{(N1=N2)\ \Lambda\ (\mathbf{C1>C2})}$ (PROD1 **x** PROD2 ))

e ottengo solo [66, 33]

# Esercizio

STOCCAGGIO(<u>Prodotto, Magazzino</u>, Quantità)

Elencare il codice dei prodotti stoccati in almeno 2 magazzini diversi

STOC1 = $\mathbf{REN}_{P1,\ M1,\ Q1}$ (STOCCAGGIO)

STOC2 = $\mathbf{REN}_{P2,\ M2,\ Q2}$ (STOCCAGGIO)

$\mathbf{PROJ}_{P1}$ ($\mathbf{SEL}_{(P1=P2)\ \wedge\ (M1\neq M2)}$ (STOC1 **x** STOC2))

(in questo caso l'uso di viste aiuta ma non è necessario)

# Esercizio

PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)

Elencare il prezzo del prodotto più caro

$$PROD1 = \mathbf{REN}_{P1 \leftarrow Prezzo}(\mathbf{PROJ}_{Prezzo}(PRODOTTO))$$

$$PROD2 = \mathbf{REN}_{P2 \leftarrow Prezzo}(\mathbf{PROJ}_{Prezzo}(PRODOTTO))$$

$$\mathbf{PROJ}_{Prezzo}(PRODOTTO) - \mathbf{REN}_{Prezzo \leftarrow P2}(\mathbf{PROJ}_{P2}(\mathbf{SEL}_{P2<P1}(PROD1 \times PROD2)))$$

# Esercizio

AZIENDA (CodiceA, NomeAzienda, Città, CAP, Telefono)
PRODOTTO(CodiceP, NomeProdotto, Azienda, Prezzo)

Elencare il nome delle aziende che producono almeno 2 prodotti diversi aventi lo stesso nome

PROD1 = $\mathbf{REN}_{C1, N1, A1, P1}$ (PRODOTTO)
PROD2 = $\mathbf{REN}_{C2, N2, A2, P2}$ (PRODOTTO)

$\mathbf{PROJ}_{NomeAzienda}$ (AZIENDA $\mathbf{JOIN}_{CodiceA = A1}$ (
$\mathbf{SEL}_{(A1=A2) \wedge (N1=N2) \wedge (P1<P2)}$ (PROD1 **x** PROD2)))

# Esercizio

Elencare il nome delle aziende che producono almeno 3 prodotti diversi aventi lo stesso nome

PROD1 = $\mathbf{REN}_{C1, N1, A1, P1}$ (PRODOTTO)
PROD2 = $\mathbf{REN}_{C2, N2, A2, P2}$ (PRODOTTO)
PROD3 = $\mathbf{REN}_{C3, N3, A3, P3}$ (PRODOTTO)

$\mathbf{PROJ}_{NomeAzienda}$ (AZIENDA $\mathbf{JOIN}_{CodiceA = A1}$ (

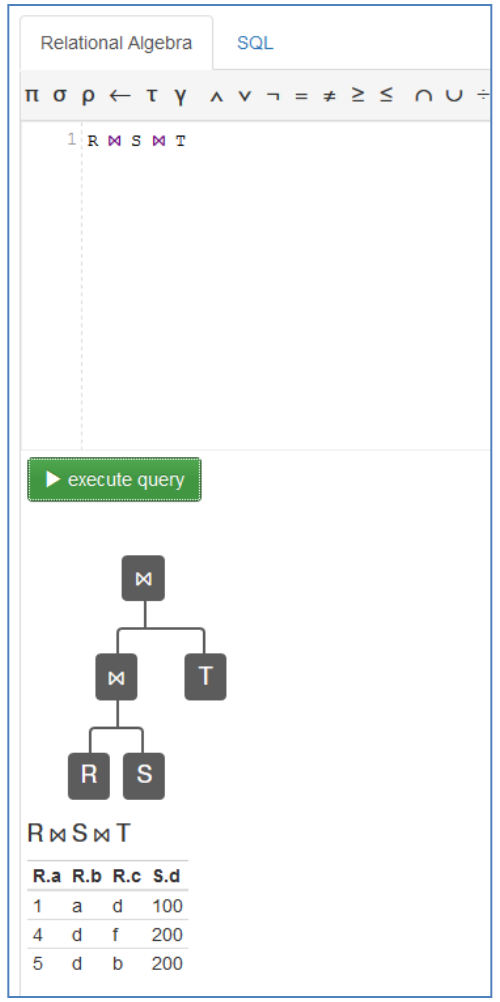
$\mathbf{SEL}_{(A1=A2) \wedge (A2=A3) \wedge (N1=N2) \wedge (N2=N3) \wedge (P1<P2) \wedge (P2<P3)}$ (PROD1 **x** PROD2 **x** PROD3)))

(in questo caso l'uso di viste aiuta ma non è necessario)

# RelaX
# relational algebra calculator

- Useremo questo tool online per allenarci con l'algebra relazionale

- https://dbis-uibk.github.io/relax/



# Esercizi con RelaX

- https://dbis-uibk.github.io/relax/calc.htm
- Dal menu a tendina in alto a sinistra scegliere: Kemper Datenbanksysteme (en)

# Esercizi con RelaX (1)

- Per ogni assistente (assistant), selezionare il nome dell'assistente e quello del boss (professor)

π a_name,p_name (
(ρ a_name <- assistant.name assistant) ⋈ boss =
professor.employee_id (ρ p_name <- professor.name professor))

# Esercizi con RelaX (2)

- Per ogni professore che insegna un corso (lecture) da 4 crediti, selezionare il nome del professore e quello degli assistenti

π professor.name, assistant.name ( ((σ credits=4 (lecture)) ⋈ lectured_by = employee_id (professor)) ⋈ boss = professor.employee_id (assistant))

# Esercizi con RelaX (2 cont.)

- Se vogliamo includere anche i nomi dei professori senza assistente: left join ⟕

π professor.name, assistant.name ( ((σ credits=4 (lecture)) ⋈ lectured_by = employee_id (professor)) ⟕ boss = professor.employee_id (assistant))

# Esercizi con RelaX (3)

- Quali sono i codici (lecture_id) dei corsi seguiti (attends) dallo studente di nome Jonas che egli deve ancora sostenere?

π lecture_id ((σ name='Jonas' student) ⋈ student.student_id=attends.student_id (attends)) -
π lecture_id ((σ name='Jonas' student) ⋈ student.student_id=grade.student_id (grade))

# Esercizi con RelaX (4)

- Quali sono le coppie di assistenti che hanno lo stesso boss (eliminando le coppie ripetute)?

π a1.name, a2.name (σ a1.boss=a2.boss ∧ a1.employee_id<a2.employee_id (ρ a1 assistant × RHO a2 assistant))

# Esercizi con RelaX (5)

- Quali sono le coppie di assistenti che hanno lo stesso boss (eliminando le coppie ripetute)? Scrivere anche il nome del relativo boss.

π a1.name, a2.name, professor.name ((σ a1.boss=a2.boss ∧ a1.employee_id<a2.employee_id (ρ a1 assistant × ρ a2 assistant)) ⋈ a1.boss = professor.employee_id professor)

# Esercizi con RelaX (6)

- Scrivere il nome degli studenti che hanno preso il voto più alto.

$$\pi_{\text{student.name}}((\pi_{\text{grade}}(\text{grade}) - \pi_{\text{g1.grade}}(\sigma_{\text{g1.grade}<\text{g2.grade}}(\rho_{\text{g1 grade}} \times \rho_{\text{g2 grade}}))) \bowtie \text{grade} \bowtie \text{student})$$