# 3 - Linguaggi di Interrogazioni per Basi di Dati Relazionali

I linguaggi per interrogare un database relazionale permettono di definire la relazione risultato a partire dalle relazioni che compongono la base di dati.

Il modello relazionale è stato il primo modello ad introdurre linguaggi **set oriented**, in grado di operare su insiemi di dati con **operatori insiemistici**, nei modelli precedenti invece questo era dato tramite operatori simili a quelli dei linguaggi imperativi per la manipolazione dei file

I linguaggi si dividono in tre famiglie:

- **Linguaggi algebrici**: un interrogazione (query) è definita da un'espressione con operatori su relazioni che producono come risultato altre relazioni
- **Linguaggi basati sul calcolo dei predicati**: un'interrogazione è definita da una formula del calcolo dei predicati
- **Linguaggi logici**: sono linguaggi ispirati dal linguaggio logico Prolog in cui un’interrogazione è definita da un insieme di clausole di Horn.

I linguaggi algebrici sono **procedurali**, ossia le loro espressioni descrivono passo passo la computazione del risultato a partire da un istanza di base di dati.
I linguaggi basati sul calcolo dei predicati ed i linguaggi logici sono **dichiarativi**, ossia le loro espressioni specificano le proprietà del risultato, piuttosto della procedura per ottenerlo

I linguaggi relazionali esistenti offrono anche altri operatori non previsti nell'algebra relazionale come le operazioni aritmetiche, la media, la somma, minimo o massimo, che consentono di agire su insiemi di valori per ottenere una singola quantità.
Un esempio di questo linguaggio logico è il **Datalog**, più espressivo del calcolo relazionale, riesce quindi ad esprimere interrogazioni ricorsive come la relazione transitiva

## Algebra relazionale

L'algebra relazionale è un linguaggio costituito da un insieme di operatori, definiti su relazioni e che producono ancora relazioni come risultati.
Prima si definisce un insieme minimo e completo di operatori (**operatori primitivi**) per poi definire altri operatori derivati dai precedenti, utili per esprimere operazioni complesse in modo sintetico, molto frequente nell'uso dei DB relazionali

Gli operatori sono distinti in 3 gruppi:

- **Operatori insiemistici tradizionali**: unione, intersezione, differenza
- **Operatori specifici**: ridenominazione, selezione, proiezione
- **Operatori di join**: prodotto cartesiano, join naturale, theta-join, equi-join, join-esterno

Gli operatori **primitivi** (da cui si possono derivare tutti gli altri) per categoria sono:

- Unione e differenza
- Ridenominazione, selezione, proiezione
- Prodotto Cartesiano

## Operatori insiemistici

Gli operatori insiemistici sono possibili solo con **relazioni binarie**

### Unione

L'unione di due relazioni $r_1(X)$ ed $r_2(X)$ definite sullo stesso insieme di attributi $X$, indicata con $r_1 \bigcup r_2$, è una relazione ancora su $X$ contenente le tuple che appartengono ad $r_1$ oppure ad $r_2$, oppure ad entrambe.

$$r_1 \bigcup r_2 = \{t | t \in r_1 \vee t \in r_2\}$$

Laureati

| Matricola | Cognome | Età |
|---|---|---|
| 7274 | Rossi | 37 |
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |

Dirigenti

| Matricola | Cognome | Età |
|---|---|---|
| 9297 | Neri | 56 |
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |

Laureati ∪ Dirigenti

| Matricola | Cognome | Età |
|---|---|---|
| 7274 | Rossi | 37 |
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |
| 9297 | Neri | 56 |

### Intersezione

L'intersezione di due relazioni definite sullo stesso insieme di attributi , indicata con $r_1 \bigcap r_2$, è una relazione su $X$ contenente le tuple che appartengono ad $r_1$ e $r_2$

$$r_1 \bigcap r_2 = \{t | t \in r_1 \wedge t \in r_2\}$$

Laureati

| Matricola | Cognome | Età |
|---|---|---|
| 7274 | Rossi | 37 |
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |

Dirigenti

| Matricola | Cognome | Età |
|---|---|---|
| 9297 | Neri | 56 |
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |

Laureati ∩ Dirigenti

| Matricola | Cognome | Età |
|---|---|---|
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |

C. d'Amato

## Differenza

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

$$r_1 - r_2 = \{t | t \in r_1 \wedge t \notin r_2\}$$

Laureati

| Matricola | Cognome | Età |
|---|---|---|
| 7274 | Rossi | 37 |
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |

Dirigenti

| Matricola | Cognome | Età |
|---|---|---|
| 9297 | Neri | 56 |
| 7432 | Neri | 39 |
| 9824 | Verdi | 38 |

Laureati – Dirigenti

| Matricola | Cognome | Età |
|---|---|---|
| 7274 | Rossi | 37 |

# Operatori specifici

## Ridenominazione

Consente di cambiare i nomi degli attributi, lasciando inalterato il contenuto delle relazioni e agendo soltanto sullo schema.

La possiamo definire con:
Sia $R(X)$ una relazione definita sull'insieme di attributi $X$ e sia $Y$ un altro insieme di attributi di stessa cardinalità.
Siano $A_1, A_2 \dots A_k$ e $B_1, B_2 \dots B_k$ rispettivamente un ordinamento per gli attributi in $X$ e un ordinamento per quelli in $Y$, allora la ridenominazione:

$$\rho\ B_1, B_2 \dots B_k \leftarrow A_1, A_2 \dots A_k(r)$$

contiene una tupla $t'$ su $Y$ per ciascuna tupla $t$ in $r$ (su $X$), definita come:

$$\rho\, B_1, B_2 \dots B_k \leftarrow A_1, A_2 \dots A_k(r) = \{t' | \exists t \in r \text{ t.c } \forall i = 1, \dots, k\; t'[B_i] = t[A_i]\}$$

Nelle liste $A_1, A_2 \dots A_k$ e $B_1, B_2 \dots B_k$ si indicheranno solo gli attributi che vengono rinominati, cioè quelli per cui $A_i \neq B_i$

Paternità

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

Maternità

| Madre | Figlio |
|---|---|
| Eva | Caino |
| Eva | Set |
| Sara | Isacco |
| Agar | Ismaele |

Paternità ∪ Maternità ??

---

Paternità

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

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

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

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

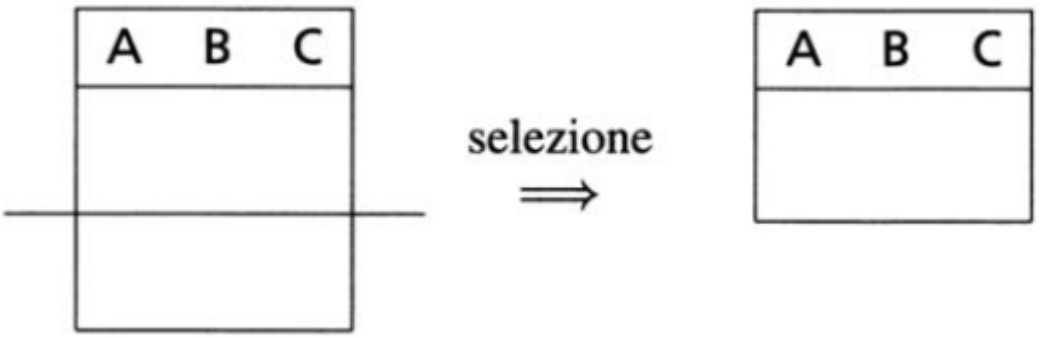
| Genitore | Figlio |
|---|---|
| Adamo | Caino |
| Adamo | Abele |
| Abramo | Isacco |
| Abramo | Ismaele |
| Eva | Caino |
| Eva | Set |
| Sara | Isacco |
| Agar | Ismaele |

La ridenominazione in questo caso è stata aggregata con l'operazione di unione delle due tabelle precedenti

## Selezione

Produce il sottoinsieme delle tuple di una relazione che soddisfano la “condizione di selezione” che possono prevedere:

- confronti fra attributi
- confronto fra attributi e costanti
- possono essere complesse (ossia ottenute combinando condizioni semplici con i connettivi logici $\wedge$, $\vee$, e $\neg$)
  e viene indicato con $\sigma$ a pedice




Formalmente viene definito:
Data una relazione $r(X)$, una formula proposizionale $F$ su $X$ è una formula ottenuta combinando, con i connettivi $\wedge$, $\vee$ e $\neg$, condizioni atomiche del tipo $A\theta B$ o $A\theta c$, dove:

- $\theta$ è un operatore di confronto ($=, \neq, >, <, \geq, \leq$)
- $A$ e $B$ sono attributi in $X$ dove il confronto $\theta$ abbia senso
- $c$ è una costante compatibile con il dominio di $A$

Date una formula $F$ e una tupla $t$, è definito un valore di verità (cioè vero o falso) per $F$ su $t$:

- $A\theta B$ è vera su $t$ se $t[A]$ è in relazione $\theta$ con $t[B]$, altrimenti è falsa;
- $A\theta c$ è vera su $t$ se $t[A]$ è in relazione $\theta$ con $c$, altrimenti è falsa;
- $F_1 \vee F_2, F_1 \wedge F_2$ e $\neg F_1$ hanno l’usuale significato

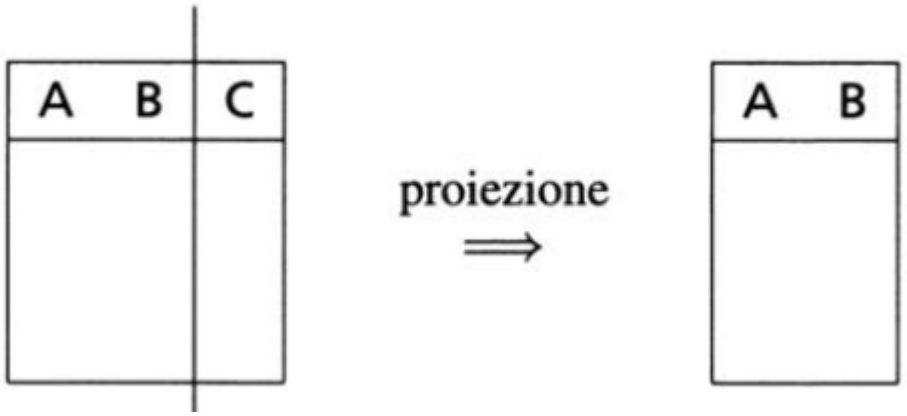
Date una relazione $r(X)$ ed una formula proposizionale $F$ su $X$, la selezione $\sigma_F(r)$ produce una relazione su $X$ che contiene le tuple $t$ di $r$ su cui $F$ è vera:

$$\sigma_F(r) = \{t | t \in r \wedge F(t)\}$$

Esempio:Data la relazione:
Studente(Matr, Nome, Genere, DataNascita, DataImmatr)
Esempioi di Formula:
(Genere = f ) ∧ (DataNascita > DataImmatricolazione) C

## Proiezione

Data una relazione $r(X)$ e un sottoinsieme $Y$ di $X$, la proiezione di $r$ su $Y$, $\pi_Y(r)$, è l'insieme di tuple su $Y$ ottenute dalle tuple di $r$ considerando solo i valori su $Y$:




Esempio:

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

Impiegati

| Cognome | Nome | Età | Stipendio |
|---|---|---|---|
| Rossi | Mario | 25 | 1000 |
| Neri | Luca | 40 | 1500 |
| Verdi | Nicola | 36 | 2200 |
| Rossi | Marco | 40 | 1900 |

$\pi_{Età}$(Impiegati)

| Età |
|---|
| 25 |
| 36 |
| 40 |

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

| Cognome | Nome |
|---|---|
| Rossi | Mario |
| Neri | Luca |
| Verdi | Nicola |
| Rossi | Marco |

Il risultato di una proiezione contiene al più tante tuple quante l'operando.
$\pi_Y(r)$ manterrà lo stesso numero di tuple di $r$ sse $Y$ è superchiave per $r$.

C. d'Amato

# Operatori di Join

## Prodotto cartesiano

Date due relazioni $r_1(X)$ ed $r_2(Y)$, con $X$ e $Y$ insiemi di attributi distinti, $X \bigcap Y = \varnothing$, il prodotto cartesiano è una relazione $r_1 \times r_2(XY)$ così definita:

$$r_1 \times r_2 = \{tt' | t \in r_1 \wedge t' \in r_2\}$$

dove $tt'$ è una $k$ + $m$ $n$-upla ottenuta dalla concatenazione di due tuple $t$ e $t'$ dove $k$ è il grado di $r_1$ ed $m$ è il grado di $r_2$.
La relazione risultante ha:

- **grado** uguale alla somma dei gradi dei due operandi
- **cardinalità** uguale al prodotto delle cardinalità degli operandi

Impiegati

| Cognome | Progetto |
|---|---|
| Rossi | A |
| Neri | A |
| Neri | B |

Progetti

| Codice | Nome |
|---|---|
| A | Venere |
| B | Marte |

Impiegati × Progetti

| Cognome | Progetto | Codice | Nome |
|---|---|---|---|
| Rossi | A | A | Venere |
| Neri | A | A | Venere |
| Neri | B | A | Venere |
| Rossi | A | B | Marte |
| Neri | A | B | Marte |
| Neri | B | B | Marte |

Esempio

## Join naturale

Il join naturale, denotato con $\bowtie$, è un operatore che correla dati in relazioni diverse, sulla base di valori uguali in attributi con lo stesso nome.
Viene formalmente definito con:
Date due relazioni $r_1(X_1)$ ed $r_2(X_2)$, con attributi comuni a $r_1$ ed $r_2$ definiti sugli stessi domini, il join naturale è una relazione definita sull'unione degli insiemi degli attributi degli operandi ( $X_1X_2$) e le cui tuple sono ottenute combinando le tuple degli operandi con valori uguali sugli attributi comuni:

$$r_1 \bowtie r_2 = \{t \text{ su } X_1X_2 | \exists\, t_1 \in r_1 \text{ e } t_2 \in r_2 \text{ con } t[X_1] = t_1 \text{ e } t[X_2] = t_2\}$$

Esempio:

Paternità

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

Maternità

| Madre | Figlio |
|---|---|
| Eva | Caino |
| Eva | Set |
| Sara | Isacco |
| Agar | Ismaele |

Paternità ▷◁ Maternità

| Padre | Figlio | Madre |
|---|---|---|
| Adamo | Caino | Eva |
| Abramo | Isacco | Sara |
| Abramo | Ismaele | Agar |

Si parla di **join completo** se ogni tupla di ciascun operando (ossia tabella) contribuisce ad almeno una tupla del risultato.
Se ciascuna tupla di un operando è compatibile con tutte le tuple dell'altro operando, il risultato conterrà un numero di tuple pari al prodotto delle cardinalità degli operandi.

Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | vendite |
| Neri | produzione |
| Bianchi | produzione |

CapiReparto

| Reparto | Capo |
|---|---|
| produzione | Mori |
| vendite | Bruni |

Impiegati▷◁CapiReparto

| Impiegato | Reparto | Capo |
|---|---|---|
| Rossi | vendite | Bruni |
| Neri | produzione | Mori |
| Bianchi | produzione | Mori |

Può accadere che alcune tuple degli operandi non contribuiscano al risultato, perché l'altra relazione non contiene tuple con gli stessi valori sull'attributo comune, tali tuple si definiscono dangling (dondolanti).
Come caso limite è possibile che nessuna delle tuple degli operandi sia combinabile e allora il risultato del join naturale è la relazione vuota.

Esempio:

Impiegati

| Impiegato | Reparto |
|---|---|
| Rossi | vendite |
| Neri | produzione |
| Bianchi | vendite |

CapiReparto

| Reparto | Capo |
|---|---|
| produzione | Mori |
| acquisti | Bruni |

Impiegati ▷◁CapiReparto

| Impiegato | Reparto | Capo |
|---|---|---|
| Neri | produzione | Mori |

Il join naturale presenta alcune proprietà:

- Il grado della relazione risultato di un join naturale è minore o uguale alla somma dei gradi degli operandi, poiché gli attributi omonimi degli operandi compaiono una sola volta nel risultato.
- **Commutatività**: $r_1 \bowtie r_2 = r_2 \bowtie r_1$
- **Associatività**: $r_1 \bowtie (r_2 \bowtie r_3) = (r_1 \bowtie r_2) \bowtie r_3$ (quindi è possibile scrivere sequenze di join senza parentesi)
- Se $r_1$ e $r_2$ non hanno attributi comuni allora $r_1 \bowtie r_2 = r_1 \times r_2$
- Se $r_1$ e $r_2$ hanno lo stesso schema ($X_1 = X_2$) allora $r_1 \bowtie r_2 = r_1 \bigcap r_2$ (il risultato sarà composto dagli elementi delle due istanze, quindi l'intersezione)

## Join esterno

Il join esterno è una variante del join naturale, il quale restituisce il join naturale di $r_1$ ed $r_2$ esteso con le tuple di $r_1$ ed $r_2$ che non appartengono al join naturale, completate con valori nulli per gli attributi mancanti.

## Esempio:

Paternità

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

Maternità

| Madre | Figlio |
|---|---|
| Eva | Caino |
| Eva | Set |
| Sara | Isacco |
| Agar | Ismaele |

Paternità $\rhd\lhd_{\text{Full}}$ Maternità

| Padre | Figlio | Madre |
|---|---|---|
| Adamo | Caino | Eva |
| Adamo | Abele | *null* |
| *null* | Set | Eva |
| Abramo | Isacco | Sara |
| Abramo | Ismaele | Agar |

Esistono altri due tipi di join esterni:

- Join esterno **sinistro**
- Join esterno **destro**
  Nel primo caso, solo le tuple dell'argomento sinistro $r_1$ che non appartengono al join naturale appaiono nel risultato, mentre nell'altro caso appaiono solo quelle dell'argomento destro $r_2$.

Paternità $\rhd\lhd_{\text{left}}$ Maternità

| Padre | Figlio | Madre |
|---|---|---|
| Adamo | Caino | Eva |
| Adamo | Abele | *null* |
| Abramo | Isacco | Sara |
| Abramo | Ismaele | Agar |

Paternità $\rhd\lhd_{\text{right}}$ Maternità

| Padre | Figlio | Madre |
|---|---|---|
| Adamo | Caino | Eva |
| *null* | Set | Eva |
| Abramo | Isacco | Sara |
| Abramo | Ismaele | Agar |

C. d'Ama

## Theta-Join

Il prodotto cartesiano ha poca utilità nella pratica, poiché concatena tuple non necessariamente correlate dal punto di vista semantico, infatti viene spesso seguito da una selezione, che centra l'attenzione sulle tuple correlate secondo le esigenze.
Per questo motivo si definisce l'operatore derivato theta-join come prodotto cartesiano seguito da una selezione.
Viene definito formalmente in:
Date due relazioni $R_1(X)$ e $R_2(Y)$, con $X \bigcup Y = \varnothing$, siano $A_i \in X$ e $B_j \in Y$ e $\theta$ un operatore di confronto ($=, \neq, >, <, \geq, \leq$) il theta-join è definito come:

$$r_1 \bowtie_{A_i \, \theta \, B_j} r_2 = \sigma_{A_i \, \theta \, B_j}(r_1 \times r_2)$$

Esempio:

Persone

| Cognome | Età |
|---|---|
| Rossi | 36 |
| Neri | 40 |
| Neri | 28 |

Offerte

| EtàMax | Impiego |
|---|---|
| 35 | Operaio |
| 38 | Analista |

Persone $\triangleright\!\triangleleft_{\text{Età} \leq \text{EtàMax}}$ Offerte

| Cognome | Età | EtàMax | Impiego |
|---|---|---|---|
| Rossi | 36 | 38 | Analista |
| Neri | 28 | 35 | Operaio |
| Neri | 28 | 38 | Analista |

## Equi-join

Un theta-join in cui la condizione di selezione sia una congiunzione di atomi di uguaglianza, con un attributo della prima relazione e uno della seconda, viene detto **equi-join**.

Esempio:

Impiegati

| Cognome | Progetto |
|---|---|
| Rossi | A |
| Neri | A |
| Neri | B |

Progetti

| Codice | Nome |
|---|---|
| A | Venere |
| B | Marte |

Impiegati $\triangleright\!\triangleleft_{\text{Progetto=Codice}}$ Progetti

| Cognome | Progetto | Codice | Nome |
|---|---|---|---|
| Rossi | A | A | Venere |
| Neri | A | A | Venere |
| Neri | B | B | Marte |

Da un punto di vista pratico il theta-join e l'equi-join hanno una grande importanza, in quanto la maggior parte dei BDMS relazionali esistenti non utilizzano i nomi di attributo per correlare relazioni e quindi non ha senso per essi il join naturale.
Peraltro il join naturale può essere simulato per mezzo della ridenominazione, dell'equi-join e della proiezione.

# Interrogazioni in algebra relazionale

Un interrogazione può essere definita come una funzione che, applicata a istanze di basi di dati, produce relazioni.

Formalmente:
Dato uno schema $R$ di basi di dati, un'interrogazione è una funzione che, per ogni istanza di $r$ di $R$, produce una relazione su un dato insieme di attributi $X$
In algebra relazionale, le interrogazioni su uno schema di base di dati $R$ vengono formulati con espressioni i cui atomi (nomi di) sono relazioni in $R$ (le variabili)

## Esempi di interrogazioni in algebra relazionale

La prima interrogazione che consideriamo è molto semplice, in quanto coinvolge una sola relazione: *trovare matricola, nome ed età degli impiegati che guadagnano più di 40 mila euro*. In questo caso, con una selezione possiamo porre l'attenzione sulle sole tuple che soddisfano la condizione (stipendio maggiore di 40 mila euro) e con una proiezione eliminiamo gli attributi non richiesti:

$$\pi_{\text{Matr,Nome,Eta}}(\sigma_{\text{Stipendio}>40}(\text{Impiegati})) \tag{3.2}$$

| Matr | Nome | Età |
|---|---|---|
| 104 | Luigi Neri | 38 |
| 210 | Marco Celli | 49 |
| 231 | Siro Bisi | 50 |
| 252 | Nico Bini | 44 |
| 301 | Sergio Rossi | 34 |
| 375 | Mario Rossi | 50 |

La seconda interrogazione coinvolge entrambe le relazioni, in modo molto naturale: *trovare le matricole dei capi degli impiegati che guadagnano più di 40 mila euro*:

$$\pi_{\text{Capo}}(\text{Supervisione} \underset{\text{Impiegato=Matr}}{\bowtie} \sigma_{\text{Stipendio}>40}(\text{Impiegati})) \tag{3.3}$$

| Capo |
|---|
| 210 |
| 301 |
| 375 |

Passiamo a esempi un po' più complessi, in cui il coinvolgimento delle due relazioni è più articolato. Cominciamo aggiungendo solo un piccolo elemento all'interrogazione precedente: *trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 mila euro*. In questo caso, possiamo ovviamente far uso dell'espressione precedente, ma dobbiamo poi produrre, per ciascuna tupla del risultato, le informazioni richieste sul capo, che vanno estratte dalla relazione Impiegati. È evidente, quindi, che ogni tupla del risultato è costruita a partire da tre tuple: la prima di Impiegati, relativa a un impiegato che guadagna più di 40 mila euro, la seconda di Supervisione, che indica la matricola del capo dell'impiegato in questione, e la terza di nuovo di Impiegati, con le informazioni relative al capo. Intuitivamente, la soluzione prevede il join della relazione Impiegati con il risultato dell'espressione precedente, ma con un'avvertenza: in generale, il capo e l'impiegato differiscono, quindi le due tuple di Impiegati che contribuiscono a una tupla del join sono diverse. Il join deve quindi essere preceduto da una ridenominazione che "cambi" tutti i nomi degli attributi. Una possibile espressione allo scopo è la seguente (in cui alcuni nomi di attributo sono stati abbreviati per ragioni di spazio):

$$\pi_{\text{NomeC,StipC}}(\rho_{\text{MatrC,NomeC,StipC,EtaC}\leftarrow\text{Matr,Nome,Stip,Eta}}(\text{Impiegati}) \bowtie_{\text{MatrC=Capo}} \text{Supervisione} \bowtie_{\text{Imp=Matr}} \sigma_{\text{Stip>40}}(\text{Impiegati})) \quad (3.4)$$

| NomeC | StipC |
|---|---|
| Marco Celli | 60 |
| Sergio Rossi | 70 |
| Mario Rossi | 65 |

Il prossimo esempio è una variante del precedente, in quanto richiede il confronto di due valori dello stesso attributo, di tuple diverse: *trovare gli impiegati che guadagnano più del rispettivo capo, mostrando matricola, nome e stipendio di ciascuno di essi e del capo*. L'espressione è simile a quella precedente. e si nota ancora di più la necessità delle ridenominazioni

$$\pi_{\text{Matr,Nome,Stip,MatrC,NomeC,StipC}}(\sigma_{\text{Stip>StipC}}(\rho_{\text{MatrC,NomeC,StipC,EtaC}\leftarrow\text{Matr,Nome,Stip,Eta}}(\text{Impiegati}) \bowtie_{\text{MatrC=Capo}} \text{Supervisione} \bowtie_{\text{Imp=Matr}} \text{Impiegati})) \quad (3.5)$$

| Matr | Nome | Stip | MatrC | NomeC | StipC |
|---|---|---|---|---|---|
| 104 | Luigi Neri | 61 | 210 | Marco Celli | 60 |
| 252 | Nico Bini | 70 | 375 | Mario Rossi | 65 |

L'ultimo esempio richiede ancora più attenzione: *trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 mila euro.* L'interrogazione include una sorta di quantificazione universale e l'unico operatore dell'algebra riconducibile a tale quantificatore è quello di divisione, il cui uso però in questo caso sarebbe laborioso. Abbiamo visto che la divisione può essere simulata con due differenze, che corrispondono a due negazioni e procediamo quindi in questo modo, cercando i capi per i quali non vi sia alcun impiegato con stipendio non superiore a 40 mila euro. Questa interrogazione, pur contorta, può essere realizzata in algebra relazionale per mezzo dell'operatore di differenza: prendiamo tutti i capi meno quelli che hanno un impiegato che guadagna non più di 40 mila euro. L'espressione è la seguente:

$$\pi_{\text{Matr,Nome}}(\text{Impiegati} \bowtie_{\text{Matr=Capo}} (\pi_{\text{Capo}}(\text{Supervisione}) - \pi_{\text{Capo}}(\text{Supervisione} \bowtie_{\text{Imp=Matr}} \sigma_{\text{Stip}\leq 40}(\text{Impiegati}))) \quad (3.6)$$

$$\pi_{\text{Matr,Nome}}(\text{Impiegati} \bowtie_{\text{Matr=Capo}} (\pi_{\text{Capo}}(\text{Supervisione}) - \pi_{\text{Capo}}(\text{Supervisione} \bowtie_{\text{Imp=Matr}} \sigma_{\text{Stip}\leq 40}(\text{Impiegati}))) \quad (3.6)$$

| Matr | Nome |
|---|---|
| 301 | Sergio Rossi |
| 375 | Mario Rossi |

## Efficienza del join

Il join è l'operazione più dispendiosa dell'algebra relazionale, il metodo più semplice per calcolare un join consiste nel confrontare tutte le coppie di tuple (la complessità è $O(n^2)$ per relazioni di cardinalità $n$.)

Un alternativa per calcolare l'equi-join o il join naturale è l'ordinare entrambe le relazioni rispetto agli attributi coinvolti per poi fondere le liste, producendo le tuple del join.
Questo tipo di realizzazione ha una complessità $O(m + nlogn)$ per le relazioni di cardinalità $n$ ed $m$ numero di tuple risultanti dalla join.
**In generale si deve evitare join di grandi relazioni.**

## Proprietà algebriche

L'algebra relazionale permette di formulare espressioni fra loro equivalenti sfruttando alcune proprietà degli operatori;
Queste trasformazioni sono utili perché possono ridurre di ordini di grandezza il costo di esecuzione delle espressioni (ottimizzazione algebrica).
Una buona tecnica per ottimizzare un'espressione algebrica consiste nell'anticipare l'applicazione degli operatori di proiezione e di selezione rispetto al prodotto, in modo da ridurre la dimensione dei risultati intermedi, infatti l'operatore di selezione produce una

relazione con un numero inferiore di tuple rispetto alla relazione a cui viene applicato e la proiezione riduce la dimensione delle tuple dell’operando ed elimina eventuali tuple dal risultato.

Definiamo le formule:
Siano $E$ un'espressione dell’algebra relazionale e $C_x$ una condizione sull'insieme di attributi $X$

1. *Atomizzazione di selezioni*: una selezione congiuntiva può essere sostituita da una cascata di selezioni atomiche:

$$\sigma_{Cx \wedge Cy}(E) \equiv \sigma_{Cx}(\sigma_{Cy}(E))$$

   Questa trasformazione permette di applicare successive trasformazioni che operano su selezioni con condizioni atomiche.

2. *Commutatività della selezione e della proiezione*:

$$\pi_Y(\sigma_{Cx}(E)) \equiv \sigma_{Cx}(\pi_Y(E)) \text{ se } X \subseteq Y$$
$$\pi_Y(\sigma_{Cx}(E)) \equiv \pi_Y(\sigma_{Cx}(\pi_{ZY}(E))) \text{ se } X \nsubseteq Y$$

3. *Anticipazione della selezione rispetto al prodotto*:

   - $\sigma_{Cx}(E_1 \times E_2) = \sigma_{Cx}(E_1) \times E_2$ se Cx usa solo attributi di $E_1$
   - $\sigma_{Cx \wedge Cy}(E_1 \times E_2) = \sigma_{Cx}(E_1) \times \sigma_{Cy}(E_2)$ se X sono attributi di $E_1$ e Y attributi di $E_2$
   - $\sigma_{Cx \wedge Cy \wedge Cz}(E_1 \times E_2) = \sigma_{Cz}(\sigma_{Cx}(E_1) \times \sigma_{Cy}(E_2))$ se X sono attributi di $E_1$, Y attributi di $E_2$ e $Cz$ condizione che usa sia attributi di $E_1$ che attributi di $E_2$

4. *Idempotenza* (Raggruppamento) di proiezioni, una proiezione può essere trasformata in una cascata di proiezioni che eliminano i vari attributi in fasi successive:

$$\pi_Z(\pi_Y(E)) = \pi_Z(E) \text{ dove } \pi_z \text{ è l'operatore di proiezione sull'insieme di attributi Z, con } Z \in Y$$

5. *Eliminazione di proiezioni superflue*:

$$\pi_Z(E) = E, \text{ se Z sono gli attributi di E}$$

6. *Anticipazione della proiezione rispetto al prodotto*:

$$\pi_{XY}(E_1 \times E_2) = \pi_x(E_1) \times (E_2), \text{ dove X sono gli attributi di } E_1 \text{ e Y gli attributi di } E_2$$

7. *Distributività della selezione rispetto all'unione*:

$$\sigma_F(E_1 \cup E_2) = \sigma_F(E_1) \cup \sigma_F(E_2)$$

8. Distributività della selezione rispetto alla differenza:

$$\sigma_F(E_1 - E_2) = \sigma_F(E_1) - \sigma_F(E_2)$$

9. Distributività della proiezione rispetto all'unione:

$$\pi_X(E_1 \cup E_2) = \pi_X(E_1) \cup \pi_X(E_2)$$

10. Distributività della giunzione rispetto all'unione:

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

11. $\sigma_{F_1 \vee F_2}(R) = \sigma_{F_1}(R) \cup \sigma_{F_2}(R)$
12. $\sigma_{F_1 \wedge F_2}(R) = \sigma_{F_1}(R) \cup \sigma_{F_2}(R) = \sigma_{F_1}(R) \bowtie \sigma_{F_2}(R)$
13. $\sigma_{F_1 \wedge \neg F_2}(R) = \sigma_{F_1}(R) - \sigma_{F_2}(R)$

## Algebra e calcolo con valori nulli

Trattiamo un caso dove in una relazione si ha un (o più) valore nullo:

PERSONE

| Nome | Età | Reddito |
|---|---|---|
| Aldo | 35 | 15000 |
| Andrea | 27 | 21000 |
| Maria | NULL | 42000 |

Se volessimo fare un interrogazione del tipo:

$$\sigma_{\text{Età}>30}(Persone)$$

non si può dire se la terza tupla faccia parte o meno del risultato.
Vi è stato quindi proposto di utilizzare una logica a 3 valori, dove un predicato può essere vero, falso oppure sconosciuto, rendendo il risultato della relazione precedente:

- Prima tupla appartenente certamente al risultato (vero)
- Seconda tupla non appartenente certamente al risultato (falso)
- Terza tupla forse appartenente al risultato (sconosciuto)

In caso di operazioni complesse come:

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

si conduce ad un comportamento non chiaro per la relazione Persone, nella logica a tre valori invece restituirebbe la terza tupla con appartenenza sconosciuta.
Si ottiene una valida alternativa introducendo due condizioni atomiche di selezione (nell'algebra) e due predicati atomici aggiuntivi (nel calcolo) allo scopo di verificare che un valore sia nullo oppure no:

- $A$ is null assume valore vero (falso) su una tupla $t$ se il valore di $t$ su $A$ è (non) nullo
- $A$ is not null assume valore vero (falso) su una tupla $t$ se il valore di $t$ su $A$ è non (è) nullo

Esempio:

$$\sigma_{Eta>30 \vee Eta\ IS\ NULL}(Persone)$$
$$\{p.* \mid p(Persone) \mid p.Eta > 30 \vee p.Eta\ IS\ NULL\}$$

Questa logica è utilizzabile in SQL, visto che prevede una gestione a tre valori

## Viste

Abbiamo visto che può risultare utile mettere a disposizione degli utenti rappresentazioni diverse per gli stessi dati, in una base di dati relazionale questo si ottiene distinguendo relazioni di base il cui contenuto è autonomo e relazioni derivate il cui contenuto è funzione del contenuto di altre relazioni, ed inoltre è possibile che una relazione derivata sia funzione di un’altra relazione derivata a patto di stabilire un ordinamento fra le relazioni derivate stesse

In linea di principio possono esistere due tipi di relazioni derivate:

- **Viste materializzate**: relazioni derivate effettivamente memorizzate nella base di dati
- **Relazioni virtuali (o viste):** relazioni definite per mezzo di espressioni del linguaggio di interrogazione non memorizzate nella base di dati, ma utilizzate nelle interrogazioni come se lo fossero.

Le viste materializzate hanno il vantaggio di essere immediatamente disponibili per le interrogazioni, ma è spesso oneroso mantenere il loro contenuto allineato con quello delle relazioni da cui derivano mentre al contrario le relazioni virtuali devono essere ricalcolate per ogni interrogazione ma non presentano problemi di allineamento.
Per inciso, per mantenere costantemente allineate le viste materializzate occorre disporre di meccanismi di trigger per l'aggiornamento automatico (basi di dati attive), i DBMS attuali forniscono meccanismi per la loro gestione (poiché il loro allineamento è difficile manualmente).

Un'interrogazione su una relazione virtuale viene trasformata sostituendo ad ogni occorrenza della relazione virtuale l'espressione che la definisce.
Esempio:

Dati i seguenti schemi di relazioni di base: $R_1(ABC), R_2(DEF), R_3(GH)$ e la vista $R = \sigma_{A>D}(R_1 \times R_2)$, l'interrogazione $\sigma_{B=G}(R \times R_3)$ viene eseguita sostituendo ad R la sua definizione $\sigma_{B=G}(\sigma_{A>D}(R_1 \times R_2) \times R_3)$ .

Mentre per quanto riguarda le interrogazioni, le viste possono essere trattate come relazioni di base, lo stesso non si può dire per le operazioni di aggiornamento, in molti casi non è possibile stabilire facilmente una semantica degli aggiornamenti sulle viste:
Ad esempio l'inserimento di una tupla nella vista non corrisponde univocamente ad un insieme di aggiornamenti sulle relazioni di base, per questo motivo i DBMS limitano aggiornamenti sulle viste.

# Calcolo relazionale

Con il termine **calcolo relazionale** si fa riferimento ad una famiglia di linguaggi di interrogazione, basati sul calcolo dei predicati del primo ordine, che hanno la caratteristica di essere dichiarativi, cioè di specificare le proprietà del risultato delle interrogazioni anziché la procedura seguita per generarlo.
Nel calcolo relazionale, l'istanza di un DB relazionale è vista come una interpretazione di una logica del primo ordine, mentre un'interrogazione è espressa come una formula ben formata.
Il risultato si ottiene, dunque, interpretando l'espressione rispetto all'istanza del DB disponibile.

Esistono diverse versioni del calcolo relazionale:

- **il calcolo relazionale su domini**
- **il calcolo relazionale su tuple**

## Calcolo relazionale su domini

Le espressioni del calcolo relazionale su domini hanno la forma:

$$\{A_1 : x_1 \ldots, A_k : x_k | f\}$$

dove:

- $A_1 : x_1 \ldots, A_k$ sono attributi distinti (che possono anche non comparire nello schema del DB rispetto a cui viene formulata l'interrogazione)
- $x_1, \ldots, x_k$ sono variabili distinte quantificate universalmente
- $f$ è una formula logica dove le variabili locali sono da intendersi quantificate esistenzialmente, a meno di quantificazioni universali esplicite

I simboli che compaiono in una formula $f$ sono:

- **Costanti**: elementi di un dominio di interesse (Per semplicità si assume che tutti gli attributi siano definiti sullo stesso dominio $D$)
- **Variabili**: elementi di un insieme numerabile $V$ disgiunto dal dominio $D$
- **Nomi di relazione e attributi**: presi dallo schema di DB di interesse
- **Operatori di confronto**: $=, \neq, >, <, \geq, \leq$
- **Connettori logici:** $\wedge$, $\vee$, e $\neg$
- **Quantificatori**: $\forall, \exists$
- **Simboli di punteggiatura e parentesi**

Questi simboli sono composti secondo le regole:

- **Formule atomiche**, di due tipi:
  - $R(A_1 : x_1 \ldots, A_p : x_p)$ dove $R(A_1 \ldots, A_p)$ è uno schema di relazione e $x_1 \ldots x_p$ sono variabili distinte
  - $x \theta y$ o $x \theta c$, con $x$ e $y$ variabili di $V$, $c$ costante e $\theta$ operatore di confronto
- Se $f_1$ e $f_2$ sono formule allora $f_1 \vee f_2, f_1 \wedge f_2, \neg f_1$ sono formule
- Se $f$ è una formula e $x$ una variabile allora $\exists x(f)$ e $\forall x(f)$ sono formule

La lista di coppie $A_1 : x_1 \ldots, A_p : x_p$ nelle espressioni del calcolo relazionale su domini è detta **target list** in quanto definisce la struttura del risultato, ovvero una relazione con schema $\{A_1, \ldots, A_k\}$ e tuple i cui valori sostituiti a $x_1, \ldots, x_k$ rendono vera la formula $f$ rispetto ad una istanza di DB cui $f$ è applicata

Per definire la semantica di un'espressione bisogna definire la nozione di valore di verità di una formula rispetto ad una sostituzione, si deve inoltre fare riferimento ad uno schema $R = \{R_1(X_1), \ldots, R_m(X_m)\}$ ed una sua istanza $r = \{r_1, \ldots, r_m\}$

- Interpretazione delle formule atomiche:
  - $R_j(A_1 : x_1, \ldots, A_p : x_p)$ è vera sui valori sui valori $a_1$ per $x_1, \ldots, a_p$ per $x_p$ se la relazione $r_j$ in $R$ contiene una tupla con valore $a_1$ per $A_1, \ldots, a_p$ per $A_p$.
    In altre parole la formula è vera se nella base di dati esiste almeno una tupla tale per cui è possibile sostituire ad ogni variabile della formula un valore di attributo della tupla
  - $x \theta y$ è vera sui valori $a_1$ per $x$, $a_2$ per $y$ se il confronto $a_1 \theta a_2$ è soddisfatto
  - $x \theta c$ è vera sul valore $a$ per $x$ se il confronto $a \theta c$ è soddisfatto
- Interpretazione di congiunzioni, disgiunzioni e negazioni:

  - $f_1 \vee f_2$ è vera se almeno una delle sotto-formule è vera
  - $f_1 \wedge f_2$ è vera se entrambe le sotto-formule sono vere
  - $\neg f_1$ è vera (falsa) su una sostituzione se $f - 1$ è falsa (vera) sulla stessa sostituzione
- Interpretazione di formule quantificate:
  - $\exists x(f)$ ,con variabili libere $y_1 \ldots y_q$, è vera sui valori $a_1, \ldots a_q$ se esiste ameno un valore $a$ tale che $f$ è vera sui valori $a$ per $x$, $a_1$ per $y_1 \ldots a_q$ per $y_q$
  - $\forall x(f)$,con variabili libere $y_1 \ldots y_q$ è vera sui valori $a_1, \ldots a_q$ se per ogni elemento $a$ del dominio $D$, la formula $f$ risulta veri sui valori $a$ per $x$, $a_1$ per $y_1, \ldots, a_q$ per $y_q$

## Esempi di calcolo relazionale su domini

Cominciamo in effetti con un'interrogazione ancora più semplice di quelle già viste: *trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 mila euro*, che formuleremmo in algebra con una selezione:

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

Nel calcolo relazionale su domini abbiamo una formulazione altrettanto semplice, con l'espressione:

$$\{\text{Matr} : m,\ \text{Nome} : n,\ \text{Età} : e,\ \text{Stipendio} : s \mid \text{Impiegati}(\text{Matr} : m,\ \text{Nome} : n,\ \text{Età} : e,\ \text{Stipendio} : s) \wedge s > 40\} \qquad (3.7)$$

L'interrogazione appena più complessa che richiede solo alcuni degli attributi: *trovare matricola, nome ed età degli impiegati che guadagnano più di 40 mila euro*, e che quindi in algebra abbiamo formulato con una proiezione (Espressione 3.2):

$$\pi_{\text{Matr,Nome,Eta}}(\sigma_{\text{Stipendio}>40}(\text{mpiegati}))$$

$$\{\text{Matr} : m,\ \text{Nome} : n,\ \text{Età} : e \mid \text{Impiegati}(\text{Matr} : m,\ \text{Nome} : n,\ \text{Età} : e,\ \text{Stipendio} : s) \wedge s > 40\} \qquad (3.9)$$

La stessa struttura si estende a interrogazioni più complesse, che in algebra relazionale abbiamo formulato per mezzo dell'operatore di join: avremo bisogno di più condizioni atomiche, una per ciascuna relazione coinvolta, e possiamo utilizzare variabili ripetute per indicare le condizioni di join. Per esempio, l'interrogazione che vuole *trovare le matricole dei capi degli impiegati che guadagnano più di 40 mila euro*, formulata in algebra con l'Espressione 3.3:

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

può essere formulata nel calcolo con:

$$\{\text{Capo} : c \mid \text{Impiegati}(\text{Matr} : m,\ \text{Nome} : n,\ \text{Età} : e,\ \text{Stipendio} : s) \wedge \text{Supervisione}(\text{Impiegato} : m,\ \text{Capo} : c) \wedge s > 40\} \qquad (3.10)$$

Se in un'espressione è richiesto il coinvolgimento di tuple diverse di una stessa relazione (in algebra, il join di una relazione con se stessa), è sufficiente includere nella formula più condizioni sullo stesso predicato, con variabili diverse. L'interrogazione che vuole *trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 mila euro*, realizzata in algebra con l'Espressione 3.4:

$$\pi_{\text{NomeC,StipC}}(\rho_{\text{MatrC,NomeC,StipC,EtaC}\leftarrow\text{Matr,Nome,Stip,Eta}}(\text{Impiegati}) \bowtie_{\text{MatrC=Capo}} \text{Supervisione} \bowtie_{\text{Impiegato=Matr}} \sigma_{\text{stipendio}>40}(\text{Impiegati}))$$

viene formulata nel calcolo richiedendo, per ciascuna tupla del risultato, l'esistenza di tre tuple: una relativa a un impiegato che guadagna più di 40 mila euro, una seconda che indica chi è il suo capo e l'ultima (di nuovo nella relazione Impiegati) che fornisce le informazioni di dettaglio sul capo:

$$\{\text{NomeC} : nc,\ \text{StipC} : sc \mid \text{Impiegati}(\text{Matr} : m,\ \text{Nome} : n,\ \text{Età} : e,\ \text{Stipendio} : s) \wedge s > 40 \wedge \text{Supervisione}(\text{Impiegato} : m,\ \text{Capo} : c) \wedge \text{Impiegati}(\text{Matr} : c,\ \text{Nome} : nc,\ \text{Età} : ec,\ \text{Stipendio} : sc)\} \quad (3.11)$$

L'ultimo esempio richiede una soluzione più complessa. Dobbiamo *trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 mila euro*. In algebra abbiamo utilizzato una differenza (Espressione 3.6):

$$\pi_{\text{Matr,Nome}}(\text{Impiegati} \bowtie_{\text{Matr=Capo}} (\pi_{\text{Capo}}(\text{Supervisione}) - \pi_{\text{Capo}}(\text{Supervisione} \bowtie_{\text{Imp=Matr}} \sigma_{\text{Stip}\leq 40}(\text{Impiegati}))))$$

Nel calcolo dobbiamo utilizzare un quantificatore. Seguendo la stessa strada seguita nell'algebra (che genera l'insieme richiesto considerando tutti i capi esclusi quelli che hanno almeno un impiegato che guadagna meno di 40 mila euro), possiamo utilizzare un quantificatore esistenziale negato (in effetti ne usiamo diversi, uno per ciascuna variabile coinvolta), trovando i capi per i quali non esiste un impiegato che guadagna non più di 40 mila euro:

$$\{\text{Matr} : c,\ \text{Nome} : n \mid \text{Impiegati}(\text{Matr} : c,\ \text{Nome} : n,\ \text{Età} : e,\ \text{Stip} : s) \wedge \text{Supervisione}(\text{Impiegato} : m,\ \text{Capo} : c) \wedge \neg\exists m'(\exists n'(\exists e'(\exists s'(\text{Impiegati}(\text{Matr} : m',\ \text{Nome} : n',\ \text{Età} : e',\ \text{Stip} : s') \wedge \text{Supervisione}(\text{Impiegato} : m',\ \text{Capo} : c) \wedge s' \leq 40))))\} \quad (3.13)$$

In alternativa, possiamo utilizzare quantificatori universali:

$$\{\text{Matr} : c,\ \text{Nome} : n \mid \text{Impiegati}(\text{Matr} : c,\ \text{Nome} : n,\ \text{Età} : e,\ \text{Stip} : s) \wedge \text{Supervisione}(\text{Impiegato} : m,\ \text{Capo} : c) \wedge \forall m'(\forall n'(\forall e'(\forall s'(\neg(\text{Impiegati}(\text{Matr} : m',\ \text{Nome} : n',\ \text{Età} : e',\ \text{Stip} : s') \wedge \text{Supervisione}(\text{Impiegato} : m',\ \text{Capo} : c)) \vee s' > 40))))\} \quad (3.14)$$

# Calcolo relazionale su tuple con dichiarazione di range

Le espressioni del calcolo su tuple con dichiarazione di range hanno la forma:

$$\{T|L|f\}$$

dove:

- $T$ è la target list, con elementi del tipo $Y : x.Z$, con $x$ variabile e $Y$ e $Z$ sequenze di attributi di pari lunghezza; ($x.* \text{ abbreviazione di } X : x.X$)
- $L$ è la range list: elenca le variabili libere di $f$ con i relativi campi di variabilità, la scrittura $x(R) \in L$ indica che la variabile $x$ può assumere come valore solo tuple nella relazione $r$ di schema $R$.
- $f$ è una formula con
  - Atomi di tipo $x.A\theta c$ o $x_1.A_1\theta x_2.A_2$ che confrontano rispettivamente il valore di $x$ sull'attributo $A$ con la costante $c$ e il valore di $x_1$ su $A_1$ con quello di $x_2$ su $A_2$
  - Connettivi come nel calcolo sui domini
  - Quantificatori che associano i range alle rispettive variabili, del tipo:
    - $\exists x(R)(f)$, (esiste una tupla $x$ nella relazione $r$ sullo schema $R$ che soddisfa la formula $f$)
      - $\forall x(R)(f)$ (ogni tupla $x$ nella relazione $r$ sullo schema $R$ soddisfa la formula $f$)

Il calcolo su tuple non permette di esprimere le interrogazioni i cui risultati possono provenire indifferentemente da due o più relazioni (che in algebra relazionale si realizzano con l'operatore di unione).

**Esempio**: date due relazioni definite sugli stessi attributi:

$$R_1(AB), R_2(AB)$$

si vuole formulare una interrogazione che restituisca l'unione delle due relazioni.
In algebra relazionale tale interrogazione verrebbe espressa mediante l'operatore di unione, mentre nel calcolo relazionale su domini si avrebbe:

$$\{A : a, B : b|R_1(A : a, B : b) \vee R_2A : a, B : b\}$$

Quindi non si riesce a formulare questa interrogazione nel calcolo su tuple, infatti se la query formulata avesse una sola variabile libera nella range list, questa dovrebbe far riferimento ad una sola delle relazioni senza acquisire tuple dall'altra per il risultato;
Se invece l'espressione avesse due variabili libere nella range list$\{t.*, t'.*|t(R1), t'(R2)|true\}$, ogni tupla del risultato dovrebbe corrispondere ad una tupla di ciascuna delle relazioni, che non è necessario perché l'unione richiede alle tuple del risultato di comparire in almeno uno degli operandi e non necessariamente in entrambi.
Si potrebbe pensare di esprimere il problema consentendo di associare ad una variabile un

range costituito da più relazioni.

$$\left\{t.^* \,|t(R_1) \bigcup t'(R_2)|true\right\}$$

Questo risolverebbe il problema dell'unione di due relazioni, ma non si riuscirebbe comunque a formulare unioni complesse i cui operandi siano sottoespressioni non direttamente corrispondenti a schemi di relazioni.
SQL, che si ispira al calcolo su tuple con dichiarazioni di range, prevede un costrutto esplicito di unione per esprimere interrogazioni che non potrebbero essere espresse altrimenti.

## Esempi di calcolo relazionale su tuple con dichiarazione di range

La prima interrogazione, che richiede *matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 mila euro*, diventa molto compatta e chiara (si veda l'Espressione 3.7):

$$\{i.* \mid i(\text{Impiegati}) \mid i.\text{Stipendio} > 40\} \qquad (3.15)$$

Per produrre solo alcuni degli attributi, *matricola, nome ed età degli impiegati che guadagnano più di 40 mila euro* (Espressione 3.2 in algebra e 3.9 in calcolo su domini), è sufficiente modificare la target list:

$$\{i.(\text{Matr}, \text{Nome}, \text{Età}) \mid i(\text{Impiegati}) \mid i.\text{Stipendio} > 40\} \qquad (3.16)$$

Per interrogazioni che coinvolgono più relazioni sono necessarie più variabili, con specifica delle condizioni di correlazione sugli attributi. L'interrogazione che vuole *trovare le matricole dei capi degli impiegati che guadagnano più di 40 mila euro* (Espressione 3.3 in algebra e 3.10 in calcolo su domini), può essere formulata con:

$$\{s.\text{Capo} \mid i(\text{Impiegati}),\ s(\text{Supervisione}) \mid i.\text{Matr} = s.\text{Impiegato} \wedge i.\text{Stipendio} > 40\} \qquad (3.17)$$

Nel caso delle espressioni corrispondenti al join di una relazione con se stessa, abbiamo più variabili aventi la stessa relazione come range. L'interrogazione: *trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 mila euro* (Espressioni 3.4 e 3.11), può essere realizzata con l'espressione:

$$\{\text{NomeC}, \text{StipC} : i'.(\text{Nome}, \text{Stip}) \mid i'(\text{Impiegati}),\ s(\text{Supervisione}),\ i(\text{Impiegati}) \mid i'.\text{Matr} = s.\text{Capo} \wedge s.\text{Impiegato} = i.\text{Matr} \wedge i.\text{Stipendio} > 40\} \qquad (3.18)$$

In modo analogo troviamo anche *gli impiegati che guadagnano più del rispettivo capo, mostrando matricola, nome e stipendio di ciascuno di essi e del capo*, (Espressione 3.5 in algebra e 3.12 in calcolo su domini):

$$\{i.(\mathsf{Nome}, \mathsf{Matr}, \mathsf{Stip}),\ \mathsf{NomeC}, \mathsf{MatrC}, \mathsf{StipC} : i'.(\mathsf{Nome}, \mathsf{Matr}, \mathsf{Stip}) \mid i(\mathsf{Impiegati}),\ s(\mathsf{Supervisione}),\ i'(\mathsf{Impiegati}) \mid i.\mathsf{Matr} = s.\mathsf{Impiegato} \wedge s.\mathsf{Capo} = i'.\mathsf{Matr} \wedge i.\mathsf{Stipendio} > i'.\mathsf{Stipendio}\} \quad (3.19)$$

Le interrogazioni con i quantificatori mostrano appieno la maggiore sinteticità e praticità del calcolo su tuple con dichiarazioni di range. L'interrogazione che richiede di *trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 mila euro* (Espressione 3.6 in algebra ed Espressione 3.13 o 3.14 in calcolo su domini) si esprime con un numero assai inferiore di quantificatori e variabili. Abbiamo di nuovo

varie alternative, sulla base dell'uso dei due quantificatori e della negazione. Con quantificatori universali:

$$\{i.(\mathsf{Matr}, \mathsf{Nome}) \mid i(\mathsf{Impiegati}),\ s(\mathsf{Supervisione}) \mid i.\mathsf{Matr} = s.\mathsf{Capo} \wedge \forall i'(\mathsf{Impiegati})(\forall s'(\mathsf{Supervisione}) (\neg(s.\mathsf{Capo} = s'.\mathsf{Capo} \wedge s'.\mathsf{Impiegato} = i'.\mathsf{Matr}) \vee i'.\mathsf{Stipendio} > 40))\} \quad (3.20)$$

Con quantificatori esistenziali negati:

$$\{i.(\mathsf{Matr}, \mathsf{Nome}) \mid i(\mathsf{Impiegati}),\ s(\mathsf{Supervisione}) \mid i.\mathsf{Matr} = s.\mathsf{Capo} \wedge \neg(\exists i'(\mathsf{Impiegati})(\exists s'(\mathsf{Supervisione}) (s.\mathsf{Capo} = s'.\mathsf{Capo} \wedge s'.\mathsf{Impiegato} = i'.\mathsf{Matr} \wedge i'.\mathsf{Stipendio} \leq 40)))\} \quad (3.21)$$

# Esercizi Linguaggi di interrogazione

Presenti nel PDF 3.1 - Esercizi Linguaggi di Interrogazione.pdf e 3.2 - Esercizi Calcolo relazionale.pdf