Algebra relazionale

Index

Introduction

L'algebra relazione è un linguaggio **formale** per interrogare un database relazionale: consiste di un insieme di operatori che possono essere applicati a una (operatori unari) o due (operatori binari) istanze di relazione e forniscono come risultato una nuova istanza di relazione (che può essere "salvata" in una "variabile").
Ma è anche un linguaggio **procedurale** in quando l'interrogazione consiste in un'espressione in cui compaiono operatori dell'algebra e istanze di relazioni della base di dati, in una sequenza che prescrive in maniera puntuale l'ordine delle operazioni e i loro operandi

---

Proiezione

La **proiezione** consente di effettuare un taglio verticale su una relazione cioè di selezionare solo alcune colonne (attributi).

$$\pi_{\mathrm{A1},\,\mathrm{A2},\,\ldots,\,\mathrm{Ak}}(r)$$

Seleziona quindi le colonne di r che corrispondono agli attributi $\mathrm{A1},\,\mathrm{A2},\,\ldots,\mathrm{Ak}$

Esempio

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Query: Nomi dei clienti

$\pi_{\mathrm{Nome}}(\mathrm{Cliente})$

Rossi
Bianchi
Verdi

⚠ **Attenzione**

Si seguono le regole insiemistiche. Nella relazione risultato **non** ci sono **duplicati**. Se vogliamo conservare i clienti omonimi dobbiamo aggiungere un ulteriore attributo in questo caso la **chiave**

Cliente

Nome	C#	Città	.....
Rossi	C1	Roma	.....
Rossi	C2	Milano	.....
Rossi	C3	Roma	.....
Bianchi	C4	Roma	.....
Verdi	C5	Roma	.....

$\pi_{\text{Nome,C\#}}(\text{Cliente})$

Query: Nomi e codici dei clienti

Rossi	C1
Rossi	C2
Rossi	C3
Bianchi	C4
Verdi	C5

Selezione

La **selezione** consente di effettuare un "taglio orizzontale" su una relazione, cioè di selezionare solo le righe (tuple) che soddisfano una data condizione

$$\sigma_{\mathrm{C}}(r)$$

Seleziona le tuple di r che soddisfano la condizione C la quale è un'espressione booleana composta in cui i termini semplici sono del tipo:

- $\mathrm{A}\ \theta\ \mathrm{B}$
- $\mathrm{A}\ \theta$ 'nome'
  dove:
- $\theta \rightarrow$ un operatore di confronto ($\theta \in \{<, =, >, \leq, \geq\}$)
- A e B $\rightarrow$ due attributi con lo stesso dominio ($\mathrm{dom}(\mathrm{A}) = \mathrm{dom}(\mathrm{B})$)
- nome $\rightarrow$ un elemento di $\mathrm{dom}(\mathrm{A})$ (costante o espressione)

Esempio

Cliente

Nome	C#	Città	……..
Rossi	C1	Roma	……..
Rossi	C2	Milano	……..
Rossi	C3	Roma	……..
Bianchi	C4	Roma	……..
Verdi	C5	Roma	……..

Query: Dati dei clienti che risiedono a Roma

$\sigma_{\text{Città='Roma'}}(\text{Cliente})$

Rossi	C1	Roma	……..
Rossi	C3	Roma	……..
Bianchi	C4	Roma	……..
Verdi	C5	Roma	……..

Unione

L'**unione** serve a costruire una relazione contenente tutte le ennuple che appartengono ad almeno uno dei due operandi

$$r_1 \cup r_2$$

> ⚠ **Warning**
>
> L'unione può essere applicata a due istanze **union compatibili**, ovvero solo se:
>
> 1. hanno lo stesso numero di attributi
> 2. gli attributi ordinatamente (corrispondenti) sono definiti sullo stesso dominio
> 3. ordinatamente hanno lo stesso significato (es. matricola ≠ numero di telefono)

Esempi

1.

Docenti

Nome	CodDoc	Dipartimento
Rossi	C1	Matematica
Rossi	C2	Lettere
Bianchi	C3	Matematica
Verdi	C4	Lingue

Amministrativi

Nome	CodAmm	Dipartimento
Esposito	C1	Lingue
Riccio	C2	Matematica
Pierro	C3	Lettere
Bianchi	C4	Lingue

sono union compatibili

$$\text{Personale} = \text{Docenti} \cup \text{Amministrativi}$$

Personale

Nome	Cod	Dipartimento
Rossi	C1	Matematica
Rossi	C2	Lettere
Bianchi	C3	Matematica
Verdi	C4	Lingue
Esposito	C1	Lingue
Riccio	C2	Matematica
Pierro	C3	Lettere
Bianchi	C4	Lingue

2

Docenti

Nome	CodDoc	Dipartimento
Rossi	D1	Matematica
Rossi	D2	Lettere
Bianchi	D3	Matematica
Verdi	D4	Lingue

Amministrativi

Nome	CodAmm	Dipartimento	Stip
Esposito	A1	Lingue	1250
Riccio	A2	Matematica	2000
Pierro	A3	Lettere	1000

In questo caso non posso fare l'unione (in Amministrativi ci sta un attributo in più). Per risolvere dunque devo prima fare una proiezione per poter poi fare l'unione. (non era necessario fare la proiezione sui docenti)

$$\text{Personale} = \text{Docenti} \cup \pi_{\text{Nome, CodDoc, Dipartimento}}(\text{Amministrativi})$$

3

Docenti

Nome	CodDoc	Dipartimento
Rossi	D1	Matematica
Rossi	D2	Lettere
Bianchi	D3	Matematica
Verdi	D4	Lingue

Amministrativi

Nome	CodAmm	AnniServizio
Esposito	A1	10
Riccio	A2	15
Pierro	A3	2
Bianchi	A4	12

In questo esempio non è possibile unire le due relazioni in quanto non sono union compatibili (attributi corrispondenti sono definiti su domini diversi $\mathrm{Dipartimento}$ e $\mathrm{AnniServizio}$). Devo per questo fare una proiezione su entrambe le relazioni

$$\mathrm{Personale} = \pi_{\mathrm{Nome},\ \mathrm{CodDoc}}(\mathit{Docente}) \cup \pi_{\mathrm{Nome},\ \mathrm{CodAmm}}(\mathrm{Amministrativi})$$

4

Docenti

Nome	CodDoc	Dipartimento
Rossi	D1	Matematica
Rossi	D2	Lettere
Bianchi	D3	Matematica
Verdi	D4	Lingue

Amministrativi

Nome	CodAmm	Mansioni
Esposito	A1	Contabilità
Riccio	A2	Didattica
Pierro	A3	Segreteria
Bianchi	A4	Didattica

In questo esempio le due relazioni sono union compatibili ma gli attributi anche se definiti sugli stessi domini hanno un significato diverso ($\mathrm{Dipartimento}$ e $\mathrm{Mansioni}$). Devo dunque fare una proiezione su entrambe le relazioni

$$\mathrm{Personale} = \pi_{\mathrm{Nome},\ \mathrm{CodDoc}}(\mathit{Docente}) \cup \pi_{\mathrm{Nome},\ \mathrm{CodAmm}}(\mathrm{Amministrativi})$$

Differenza

La **differenza** consente di costruire una relazione contentente tutte le tuple che appartengono al primo operando e non appartengono al secondo operando e si applica a operandi union compatibili

$$r_1 - r_2$$

> ⚠ **La differenza non è commutativa**

Esempio

Studenti

Nome	CodFiscale	Dipartimento
Rossi	C1	Matematica
Rossi	C2	Lettere
Bianchi	C3	Matematica
Verdi	C4	Lingue

Amministrativi

Nome	CodFiscale	Dipartimento
Esposito	C5	Lettere
Riccio	C6	Matematica
Pierro	C7	Lingue
Bianchi	C3	Matematica

$\text{Studenti} - \text{Amministrativi} = \text{studenti che non sono anche amministrativi}$

$\text{Amministrativi} - \text{Studenti} = \text{amministrativi che non sono anche studenti}$

Studenti – Amministrativi

Studenti

Nome	CodFiscale	Dipartimento
Rossi	C1	Matematica
Rossi	C2	Lettere
Verdi	C4	Lingue

Amministrativi - Studenti

Amministrativi

Nome	CodFiscale	Dipartimento
Esposito	C5	Lettere
Riccio	C6	Matematica
Pierro	C7	Lingue

Nascerebbe però un problema se avessi degli studenti che sono amministrativi in dipartimenti diversi da quelli in cui studiano (e viceversa). In questo caso infatti dovremmo fare una proiezione su Nome e CodFiscale per poter avere gli stessi risultati

Intersezione

L'intersezione consente di costruire una relazione contenente tutte le tuple che appartengono ad entrambi gli operandi e si applica a operandi union compatibili.

$$r_1 \cap r_2 = (r_1 - (r_1 - r_2))$$

Esempio

Studenti

Nome	CodFiscale	Dipartimento
Rossi	C1	Matematica
Rossi	C2	Lettere
Bianchi	C3	Matematica
Verdi	C4	Lingue

Amministrativi

Nome	CodFiscale	Dipartimento
Esposito	C5	Lettere
Riccio	C6	Matematica
Pierro	C7	Lingue
Bianchi	C3	Matematica

$$\text{Studenti} \cap \text{Amministrativi} = \text{studenti che sono anche amministrativi}$$

Studenti Amm

Nome	CodFiscale	Dipartimento
Bianchi	C3	Matematica

Informazioni in più relazioni

Vedremo che per garantire determinate "buone" qualità di una relazione occorre rappresentare **separatamente** (in relazioni diverse) **concetti diversi**
Capita che molto spesso che le informazioni che interessano per rispondere ad una interrogazione sono **distribuite** in più relazioni, in quanto coinvolgono **più oggetti** in qualche modo associati. Occorre quindi individuare le relazioni in cui si trovano le informazioni che ci interessano, e combinare queste informazioni in maniera opportuna

Prodotto cartesiano

Il **prodotto cartesiano** permette di costruire una relazione che contiene tutte le ennuple ottenute unendo tutte le ennuple di una relazione e tutte le ennuple di una seconda relazione

$$r_1 \times r_2$$

Si usa quando le informazioni che occorrono a rispondere a una query si trovano in **relazioni diverse**

> ⚠ **Non sempre il prodotto cartesiano ha un significato**

Esempio

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Ordine

O#	C#	A#	N-pezzi
O1	C1	A1	100
O2	C2	A2	200
O3	C3	A2	150
O4	C4	A3	200
O1	C1	A2	200
O1	C1	A3	100

In questo caso però non posso fare direttamente $\text{Cliente} \times \text{Ordine}$ in quanto ho un attributo identico nelle due relazioni. Per questo motivo abbiamo necessità di utilizzare la **ridenominazione** (ρ)

$$\text{OrdineR} = \rho_{\text{CC\#} \leftarrow \text{C\#}}(\text{Ordine})$$

Dunque posso fare:

$$\text{Dati dei clienti e degli ordini} = (\text{Cliente} \times \text{OrdineR})$$

Nome	C#	Città	O#	CC#	A#	N-pezzi
Rossi	C1	Roma	O1	C1	A1	100
Rossi	C1	Roma	O2	C2	A2	200
Rossi	C1	Roma	O3	C3	A2	150
Rossi	C1	Roma	O4	C4	A3	200
Rossi	C1	Roma	O1	C1	A2	200
Rossi	C2	Milano	O1	C1	A1	100
...	...	...		...	...	...
Bianchi	C3	Roma	O3	C1	A1	100
...	...	...		...	...	...
Verdi	C4	Roma	O4		A3	200
...	...	...		...	...	...

Questa relazione però risulta essere sbagliata; ho infatti la seconda tupla ha due codici

diversi per $\mathrm{Cliente}$ e Ordine. Il che vuol dire che devo effettuare una selezione sul codice cliente

$$\sigma_{\mathrm{C\#}=\mathrm{CC\#}}(\mathrm{Cliente} \times \mathrm{OrdineR})$$

Nome	C#	Città	O#	CC#	A#	N-pezzi
Rossi	C1	Roma	O1	C1	A1	100
Rossi	C1	Roma	O1	C1	A2	200
Rossi	C1	Roma	O1	C1	A3	100
Rossi	C2	Milano	O2	C2	A2	200
Bianchi	C3	Roma	O3	C3	A2	150
Verdi	C4	Roma	O4	C4	A3	200

A questo punto mi ritrovo sostanzialmente con un attributo duplicato. Lo rimuovo quindi con una proiezione

$$\pi_{\mathrm{Nome},\ \mathrm{C\#},\ \mathrm{Città},\ \mathrm{O\#},\ \mathrm{A\#},\ \mathrm{N\text{-}pezzi}}(\sigma_{\mathrm{C\#}=\mathrm{CC\#}}(\mathrm{Cliente} \times \mathrm{OrdineR}))$$

Nome	C#	Città	O#	A#	N-pezzi
Rossi	C1	Roma	O1	A1	100
Rossi	C1	Roma	O1	A2	200
Rossi	C1	Roma	O1	A3	100
Rossi	C2	Milano	O2	A2	200
Bianchi	C3	Roma	O3	A2	150
Verdi	C4	Roma	O4	A3	200

Adesso voglio rimuovere tutti gli elementi in cui il $\mathrm{N\text{-}pezzi}$ è ≤ a 100

$$\pi_{\mathrm{Nome\ C\#\ Città\ O\#\ A\#\ N\text{-}pezzi}}(\sigma_{\mathrm{C\#}=\mathrm{CC\#}\ \wedge\ \mathrm{N\text{-}pezzi}>100}(\mathrm{Cliente} \times \mathrm{OrdineR}))$$

Nome	C#	Città	O#	A#	N-pezzi
Rossi	C1	Roma	O1	A2	200
Rossi	C2	Milano	O2	A2	200
Bianchi	C3	Roma	O3	A2	150
Verdi	C4	Roma	O4	A3	200

Join naturale

Il join naturale consente di selezionare automatiche le tuple del prodotto cartesiano dei due operandi che soddisfano la condizione

$$(R_1.\,A_1 = R_2.\,A_1) \wedge (R_1.\,A_2 = R_2.\,A_2) \wedge \cdots \wedge (R_1.\,A_k = R_2.\,A_k)$$

(dove R_1 ed R_2 sono i nomi delle relazioni operando e $A_1, A_2, \ldots, A_k$ sono gli attributi comuni, cioè **con lo stesso nome**, delle relazioni operando) eliminando le ripetizioni degli attributi

$$r_1 \bowtie r_2 = \pi_{\mathrm{XY}}(\sigma_{\mathrm{C}}(r_1 \times r_2))$$

dove:

- $\mathrm{C} \rightarrow (R_1.\,A_1 = R_2.\,A_1) \wedge \cdots \wedge (R_1.\,A_k = R_2.\,A_k)$
- $\mathrm{X} \rightarrow$ è l'insieme di attributi di r_1
- $\mathrm{Y} \rightarrow$ insieme di attributi di r_2 che non sono attributi di r_1

> **Info**
>
> Nel join naturale gli attributi della condizione che consente di unire solo le ennuple giuste che hanno lo **stesso nome**
> Vengono unite le ennuple in cui questi attributi hanno lo **stesso valore**

Esempio 1

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Ordine

O#	C#	A#	N-pezzi
O1	C1	A1	100
O2	C2	A2	200
O3	C3	A2	150
O4	C4	A3	200
O1	C1	A2	200
O1	C1	A3	100

$$\text{Dati dei clienti e dei loro ordini} = \text{Cliente} \bowtie \text{Ordine}$$

Nome	C#	Città	O#	A#	N-pezzi
Rossi	C1	Roma	O1	A1	100
Rossi	C1	Roma	O1	A2	200
Rossi	C1	Roma	O1	A3	100
Rossi	C2	Milano	O2	A2	200
Bianchi	C3	Roma	O3	A2	150
Verdi	C4	Roma	O4	A3	200

Adesso voglio rimuovere tutti gli elementi in cui il $\mathrm{N\text{-}pezzi}$ è ≤ a 100

$$\sigma_{\mathrm{N\text{-}pezzi}>100}(\mathrm{Cliente} \bowtie \mathrm{Ordine})$$

Nome	C#	Città	O#	A#	N-pezzi
Rossi	C1	Roma	O1	A2	200
Rossi	C2	Milano	O2	A2	200
Bianchi	C3	Roma	O3	A2	150
Verdi	C4	Roma	O4	A3	200

Come prima ma stavolta voglio solo i nomi dei clienti

$$\pi_{\mathrm{Nome}}(\sigma_{\mathrm{N\text{-}pezzi}>100}(\mathrm{Cliente} \bowtie \mathrm{Ordine}))$$

Nome
Rossi
Bianchi
Verdi

> ⚠ **Warning**
>
> Dato che Nome non identifica il cliente, i duplicati vengono cancellati (in questo caso ho solo un Rossi). Sarebbe stato meglio utilizzare anche una **chiave** ($\mathrm{C\#}$)

Adesso oltre che il nome voglio anche vedere la città

$$\pi_{\mathrm{Nome},\ \mathrm{Città}}(\sigma_{\mathrm{N\text{-}pezzi}>100}(\mathrm{Cliente} \bowtie \mathrm{Ordine}))$$

Nome	Città
Rossi	Roma
Rossi	Milano
Bianchi	Roma
Verdi	Roma

Esempio 2

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Ordine

O#	C#	A#	N-pezzi
O1	C1	A1	100
O2	C2	A2	200
O3	C3	A2	150
O4	C4	A3	200
O1	C1	A2	200
O1	C1	A3	100

Articolo

A#	Denom.	Prezzo
A1	Piatto	3
A2	Bicchiere	2
A3	Tazza	4

Nomi e città dei clienti che hanno ordinato più di 100 pezzi per almeno un articolo con prezzo superiore a 2

$$\pi_{\text{Nome, Città}}(\sigma_{\text{N-pezzi}>100 \wedge \text{Prezzo}>2}((\text{Cliente} \bowtie \text{Ordine}) \bowtie \text{Articolo}))$$

Nome	Città
Verdi	Roma

Alternativa

In alternativa prima potremmo fare selezione sugli ordini con numero di pezzi maggiore di 100

$$\sigma_{\text{N-pezzi}>100}(\text{Ordine})$$

Quindi fare il join con Cliente

$$\text{Cliente} \bowtie \sigma_{\text{N-pezzi}>100}(\text{Ordine})$$

Per quanto riguarda il prezzo invece mi conviene fare una proiezione sul numero dell'articolo e sul prezzo per poi selezionare quelli con un prezzo maggiore di due

$$\sigma_{\text{Prezzo}>2}(\pi_{\text{A\#, Prezzo}}(\text{Articolo}))$$

E solo ora fare il join naturale con il resto:

$$(\text{Cliente} \bowtie \sigma_{\text{N-pezzi}>100}(\text{Ordine})) \bowtie \sigma_{\text{Prezzo}>2}(\pi_{\text{A\#, Prezzo}}(\text{Articolo}))$$

Infine fare la proiezione su Nome e Città

$$\pi_{\text{Nome, Città}}((\text{Cliente} \bowtie \sigma_{\text{N-pezzi}>100}(\text{Ordine})) \bowtie \sigma_{\text{Prezzo}>2}(\pi_{\text{A\#, Prezzo}}(\text{Articolo})))$$

Casi limite

Per quanto riguarda il join naturale sono presenti due casi limite:

Caso limite 1

Quando le relazioni contengono attributi con lo stesso nome ma non esistono ennuple con lo stesso valore per tali attributi in entrambe le relazioni, il risultato del join naturale è **vuoto**

Per esempio:

Nome	C#	Città	O#	A#	N-pezzi
Rossi	C1	Roma	O1	A2	200
Rossi	C2	Milano	O2	A2	200
Bianchi	C3	Roma	O3	A2	150
Verdi	C4	Roma	O4	A3	200

Cliente $\triangleright\triangleleft$ $\sigma_{\text{N-pezzi}>100}$(Ordine)

$\triangleright\triangleleft$

A#	Prezzo
A4	1

$\sigma_{\text{Prezzo}<2}$ ($\pi_{\text{A\#,Prezzo}}$(Articolo))

il risultato è vuoto

Caso limite 2

Quando le relazioni non contengono contengono attributi con lo stesso nome, il join naturale degenera nel **prodotto cartesiano**

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Ordine

O#	CC#	A#	N-pezzi
O1	C1	A1	100
O2	C2	A2	200
O3	C3	A2	150
O4	C4	A3	200
O1	C1	A2	200
O1	C1	A3	100

$$\text{Cliente} \bowtie \text{Ordine}$$

Nome	C#	Città	O#	CC#	A#	N-pezzi
Rossi	C1	Roma	O1	C1	A1	100
Rossi	C1	Roma	O2	C2	A2	200
Rossi	C1	Roma	O3	C3	A2	150
Rossi	C1	Roma	O4	C4	A3	200
Rossi	C1	Roma	O1	C1	A2	200
Rossi	C2	Milano	O1	C1	A1	100
…	…	…		…	…	…
Bianchi	C3	Roma	O3	C1	A1	100
…	…	…		…	…	…
Verdi	C4	Roma	O4		A3	200
…	…	…		…	…	…

Possibili errori

Ovviamente perché il join naturale abbia senso gli attributi devono avere lo stesso significato

Artista

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Quadro

Titolo	C#	Artista
Tit1	C1	C1
Tit2	C2	C3
Tit3	C3	C1
Tit4	C4	C2
Tit5	C5	C4
Tit6	C6	C2

Perché il join tra queste due cose abbia senso, il join va effettuato tra $\mathrm{Artista.C\#}$ e $\mathrm{Quadro.Artista}$. Quindi posso utilizzare o il θ-join oppure la ridenominazione
Procedendo con la ridenominazione:

$$\rho_{\mathrm{CA\#}\leftarrow\mathrm{C\#}}(\mathrm{Artista}) \bowtie \rho_{\mathrm{CA\#}\leftarrow\mathrm{Artista}}(\mathrm{Quadro})$$

θ-join

Il **θ-join** consente di selezionare le tuple del prodotto cartesiano dei due operandi che soddisfano una condizione del tipo $\mathrm{A}\theta\mathrm{B}$
dove:

- θ è un operatore di confronto ($\theta \in \{<, =, >, \leq, \geq\}$)
- A è un attributo dello schema del primo operando
- B è un attributo dello schema del secondo operando
- $\mathrm{dom(A)} = \mathrm{dom(B)}$

$$r_1 \underset{A\theta B}{\bowtie} r_2 = \sigma_{A\theta B}(r_1 \times r_2)$$

Condizioni negative

Alle query si possono anche applicare delle condizioni negative

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Rossi	C2	Milano

$$\sigma_{\neg(\text{Città}=\text{"Roma"})\wedge\text{Nome}=\text{"Rossi"}}(\text{Cliente})$$

Condizioni che implicano il quantificatore universale

Fino ad ora abbiamo visto query che implicavano condizioni equivalenti al quantificatore esistenziale ∃ (esiste almeno un)
Il meccanismo di valutazione consente di rispondere facilmente a questo tipo di query infatt **in qualunque posizione** appaiono nell'espressione di algebra relazionale, la valutazione delle **condizioni** avviene in **sequenza**, tupla per tupla, e quando si incontra una tupla che soddisfa le condizioni, questa viene inserita nel risultato (eventualmente parziale)

La condizione potrebbe richiedere la valutazione di gruppi interi di tuple prima di decidere se inserirle tutte, qualcuna o nessuna nella risposta e le tuple non sono ordinate e la valutazione avviene in sequenza tupla per tupla, e una volta inserita una tupla nel risultato non possiamo più eliminarla. In questo caso la condizione equivale a valutare il quantificatore universale ∀ (per ogni) oppure !∃ (non esiste nessun)

Esempio 1

Nomi e città dei clienti che hanno **SEMPRE** ordinato più di 100 pezzi per articolo

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Ordine

C#	A#	N-pezzi
C1	A1	100
C2	A2	200
C3	A2	150
C4	A3	200
C1	A2	200
C1	A3	100

Visto che risulterebbe complesso fare una selezione in cui ho solo i clienti che hanno ordinato sempre più di 100 pezzi, mi conviene piuttosto trovare quelli che hanno fatto ordini con numero di pezzi minori di 100 ed escluderli

$$\sigma_{\text{N-pezzi} \leq 100}(\text{Cliente} \bowtie \text{Ordine})$$

Facciamo quindi la proiezione sul nome e città

$$\pi_{\text{Nome, Città}}(\sigma_{\text{N-pezzi} \leq 100}(\text{Cliente} \bowtie \text{Ordine}))$$

Solo a questo punto posso fare la differenza con il totale

$$\pi_{\text{Nome, Città}}(\text{Cliente} \bowtie \text{Ordine}) - \pi_{\text{Nome, Città}}(\sigma_{\text{N-pezzi} \leq 100}(\text{Cliente} \bowtie \text{Ordine}))$$

> ⚠ **Attenzione**
>
> Nel primo membro della sottrazione faccio il join naturale tra Cliente e Ordine in modo tale da poter togliere tutti i casi in cui sono presenti clienti che non hanno mai fatto ordini. Il join infatti assicura di effettuare la sottrazione non a partire da tutti i clienti, ma solo da quelli che hanno effettuato almeno un ordine

Esempio 2

Nomi e città dei clienti che non hanno **MAI** ordinato più di 100 pezzi per un articolo

Cliente

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Ordine

C#	A#	N-pezzi
C1	A1	100
C2	A2	200
C3	A2	150
C4	A3	200
C1	A2	200
C1	A3	100

Applichiamo il ragionamento di prima e selezioniamo prima i nomi e città di clienti che NON ci interessano

$$\sigma_{\text{N-pezzi} > 100}(\text{Cliente} \bowtie \text{Ordine})$$

Facciamo la proiezione sul nome e città

$$\pi_{\text{Nome, Città}}(\sigma_{\text{N-pezzi}>100}(\text{Cliente} \bowtie \text{Ordine}))$$

Posso quindi fare la differenza

$$\pi_{\text{Nome, Città}}(\text{Cliente} \bowtie \text{Ordine}) - \pi_{\text{Nome, Città}}(\sigma_{\text{N-pezzi}>100}(\text{Cliente} \bowtie \text{Ordine}))$$

Condizioni che richiedono il prodotto di una relazione con sé stessa

Come negli esempi precedenti abbiamo visto casi in cui oggetti di relazioni diverse vengono associati, ci sono anche casi in cui sono in qualche modo associati oggetti della stessa relazione.

Esempio 1

Nomi e codici degli impiegati che guadagnano quanto o più del loro capo

Impiegati

Nome	C#	Dipart	Stip	Capo#
Rossi	C1	B	100	C3
Pirlo	C2	A	200	C3
Bianchi	C3	A	500	NULL
Verdi	C4	B	200	C2
Neri	C5	B	150	C1
Tosi	C6	B	100	C1

Per poter confrontare le informazioni sullo stipendio di un impiegato e su quello del suo capo che si trovano in tuple diverse questi devono trovarsi nella stessa tupla. Per farlo creiamo una copia della relazione ed effettuiamo un prodotto in maniera da combinare le informazioni su di un impiegato con quelle del suo capo, che a questo punto possono essere confrontate. ImpiegatiC sarà collegata in join ad impiegati combinando le tuple col valore di $\text{C}\#$ uguale a $\text{Capo}\#$. In questo modo accodiamo i dati del capo a quelli dell'impiegato.
Utilizziamo la ridenominazione e facciamo in modo che i nuovi nomi aiutino a distinguere il ruolo delle due parti nel join finale

$$\text{ImpiegatiC} = \rho_{\text{Nome, C\#, Dipart, Stip, Capo\#} \leftarrow \text{CNome, CC\#, Cdipart, Cstip, Ccapo\#}}(\text{Impiegati})$$

$$\sigma_{\text{Capo\#}=\text{CC\#}}(\text{Impiegati} \times \text{ImpiegatiC})$$

A questo punto basta confrontare lo stipendio dell'impiegato con quello del capo per selezionare gli impiegati che ci interessano e infine proiettare

$$\mathrm{r} = \sigma_{\mathrm{Stip} \geq \mathrm{CStip}}(\sigma_{\mathrm{Capo\#}=\mathrm{CC\#}}(\mathrm{Impiegati} \times \mathrm{ImpiegatiC}))$$

$$\pi_{\mathrm{Nome},\ \mathrm{C\#}}(\mathrm{r})$$

Esempio 2

Query: Nomi e codici dei capi che guadagnano più di tutti i loro impiegati
Ripriendiamo la query dell'esempio precedente che trova gli impiegati che guadagnano quanto o più del loro capo. I capi che compaiono anche una sola volta in questo risultato sono quelli che non ci interessano

$$\mathrm{r} = \sigma_{\mathrm{Stip} \geq \mathrm{CStip}}(\sigma_{\mathrm{Capo\#}=\mathrm{CC\#}}(\mathrm{Impiegati} \times \mathrm{ImpiegatiC}))$$

$$\pi_{\mathrm{CNome},\ \mathrm{CC\#}}(\sigma_{\mathrm{Capo\#}=\mathrm{CC\#}}(\mathrm{Impiegati} \times \mathrm{ImpiegatiC})) - \pi_{\mathrm{CNome},\mathrm{CC\#}}(r)$$

Errori possibili

Lo stesso esercizio può essere svolto in altri modi altrettanto corretti, ma attenzione invece a quelli non corretti

$$\pi_{\mathrm{Nome},\ \mathrm{C\#}}(\mathrm{Impiegati}) - \pi_{\mathrm{CNome},\ \mathrm{CC\#}}(\mathrm{r})$$

è sbagliato perché ci sono impiegati che non sono capi e non verrebbero sottratti alla prima proiezione

$$\pi_{\mathrm{Nome},\ \mathrm{Capo\#}}(\mathrm{Impiegati}) - \pi_{\mathrm{CNome},\ \mathrm{CC\#}}(\mathrm{r})$$

è sbagliato perché nella prima proiezione il nome è dell'impiegato e il codice è del capo

Un'alternativa corretta è:

$$\pi_{\mathrm{Nome},\ \mathrm{C\#}}((\pi_{\mathrm{Capo\#}}(\mathrm{Impiegati}) - \pi_{\mathrm{CC\#}}(\mathrm{r})) \bowtie \mathrm{Impiegati})$$

che estrae prima i codici, effettua un join per ottenere le altre informazioni (un capo è anche un impiegato) e poi effettua la proiezione. Vogliamo che i codici da cui sottraiamo siano sicuramente di capi