Basi di dati Appunti
-
secondo parziale

Dimitri

A.A. 2024/2025

Indice

1 Algebra Relazionale **2**

1.1 Introduzione . . . 2

1.2 Operatori . . . 2

1.2.1 Introduzione . . . 2

1.2.2 Selezione σ . . . 3

1.2.3 Proiezione Π . . . 3

1.2.4 Ridenominazione ρ . . . 3

1.2.5 Unione $\cup$ . . . 3

1.2.6 Differenza - . . . 4

1.2.7 Prodotto Cartesiano $\times$ . . . 4

1.2.8 Intersezione $\cap$ . . . 4

1.3 Join . . . 4

1.3.1 Introduzione . . . 4

1.3.2 Join Naturale $\bowtie$ . . . 5

1.3.3 Theta join ed equi-join . . . 6

1.3.4 Join esterni . . . 7

1.4 Ottimizzazione . . . 7

1.4.1 Valori nulli . . . 7

1.4.2 Ottimizzatore . . . 7

2 Calcolo Relazionale **11**

Chapter 1

Algebra Relazionale

1.1 Introduzione

L'algebra relazionale è un linguaggio procedurale formale di tipo algebrico i cui operandi sono relazioni. Questo linguaggio non è usato nelle implementazioni dei vari DBMS ma definisce in maniera semplice tutte le operazioni tipiche dei diversi linguaggi di interrogazione. Da un punto di vista didattico l'algebra relazionale è utile perché essendo svincolata dai "dettagli implementativi" dell'SQL (o di altri linguaggi), permette di comprendere rapidamente la tecnica d'uso dei linguaggi di interrogazione per basi di dati relazionali.

1.2 Operatori

1.2.1 Introduzione

In Algebra relazionale è possibile classificare i diversi operatori in base alla derivabilità oppure in modo funzionale. Questi operatori possono essere binari o unari, e tramite questi ultimi vengono descritte le procedure di interrogazione.
Nella classificazione in base alla derivabilità si distiguono 6 operatori di base e diversi derivati:
Operatori di Base:

- Selezione
- Proiezione
- Ridenominazione
- Unione
- Differenza
- Prodotto cartesiano

Operatori derivati:

- Intersezione
- Join

1.2.2 Selezione σ

La selezione è un'operazione unaria. Seleziona le tuple di una relazione che soddisfano un predicato producendo un sottoinsieme delle stesse, (ossia seleziona le righe della tabella che rispettano la condizione data, ricordiamo che con tuple si intendono le righe della tabella mentre con attributi le singole colonne).
Lo schema della relazione risultato è lo stesso di quella di origine ed il predicato è costituito dal nome di un attributo, da un operatore, e da un altro argomento che può essere un attributo o un valore costante.

$$\sigma_{\text{predicato}}(relazione)$$

La cardinalità di questa operazione è la seguente:

$$0 \leq |\sigma_{\text{F}}(R)| \leq |R|$$

1.2.3 Proiezione Π

La proiezione è un'operazione unaria. Data una relazione, la sua proiezione su un dato insieme di attributi è costituita dalla tabella generata dagli attributi specificati, contenente tutte le tuple della tabella di partenza, (ossia estrae tutte le colonne corrispondenti alla lista di attributi specificati).

$$\Pi_{\text{lista attributi}}(relazione)$$

La cardinalità di questa operazione è la seguente:

$$min(|R|, 1) \leq \Pi_{\text{y}}(R) \leq |R|$$

1.2.4 Ridenominazione ρ

A volte in preparazione all'esecuzione di una interrogazione o in seguito ad una sua esecuzione si ha bisogno di rinominare gli attributi di una relazione. A tal fine l'operatore di ridenominazione che permette di ottenere una nuova tabella con i nuovi nomi per gli attributi modificati e che ha le stesse tuple della tabella originale.

$$\rho_{\text{vecchio nome} \rightarrow \text{nuovo nome}}(relazione)$$

1.2.5 Unione $\cup$

L'unione fra due tabelle è rappresentata da una tabella costituita dall'unione "matematica" delle due tabelle, dove quindi sono presenti le tuple della prima tabella e quelle della seconda. Affinché l'unione abbia senso, è necessario che:

- le due tabelle abbiano lo stesso numero di attributi;
- i tipi degli attributi corrispondenti siano uguali;

Se il numero degli attributi delle due relazioni (tabelle) non è uguale, si genera un errore.
La cardinalità di questa operazione è la seguente:

$$max(|R1|, |R2|) \leq |R1 \cup R2| \leq |R1| + |R2|$$

1.2.6 Differenza -

La differenza fra due tabelle A e B è una tabella che contiene le tuple che sono presenti in A ma non in B. Come nel caso dedll'unione, questa operazione può essere eseguita solo se le relazioni hanno lo stesso grado (numero di colonne) e gli attributi sono compatibili.
La cardinalità di questa operazione è la seguente:

$$0 \leq |R1 - R2| \leq |R1|$$

1.2.7 Prodotto Cartesiano $\times$

Il prodotto cartesiano fra due tabelle è una tabella con schema la somma degli schemi, se due attributi sono uguali questi sono ripetuti le tuple della tabella sono il risultato del prodotto cartesiano dei suoi elementi, ossia da tutte le coppie possibili composte dagli elementi appartenenti alle due relazioni.

$$R1 \times R2$$

La cardinalità di questa operazione è la seguente (ricordiamo che se il Join è completo il limite inferiore diventa $max(|R1|, |R2|)$):

$$|R1 \times R2| = |R1| \cdot |R2|$$

1.2.8 Intersezione $\cap$

Il risultato dell'operazione di intersezione tra due relazioni è una relazione contenente le tuple che appartengono ad entrambe le relazioni, anche in questo caso valgono le stesse condizioni di unione e differenza per la validità dell'operazione.

$$R1 \cap R2$$

La cardinalità di questa operazione è la seguente:

$$0 \leq |R1 \cap R2| \leq min(|R1|, |R2|)$$

1.3 Join

1.3.1 Introduzione

E' l'operatore più caratteristico dell'algebra relazionale, in quanto è quello che permette di correlare dati contenuti in relazioni diverse confrontando i valori comuni contenuti in esse.
Esistono diverse varianti di tale operatore comunque riconducibili l'una con l'altra:

- Join Naturale
- Join Esterni
- Theta Join ed Equi Join

1.3.2 Join Naturale $\bowtie$

Il Join naturale è un operatore binario che correla dati in relazioni diverse sulla base dei valori uguali in attributi con lo stesso nome. La relazione risultante è una tabella che ha come attributi l'unione degli attributi delle tabelle iniziali e contiene solamente le tuple che hanno valori uguali negli attributi in comune.
Quando non si hanno attributi comuni il join naturale diventa un prodotto cartesiano, perché genera tutte le possibili coppie.

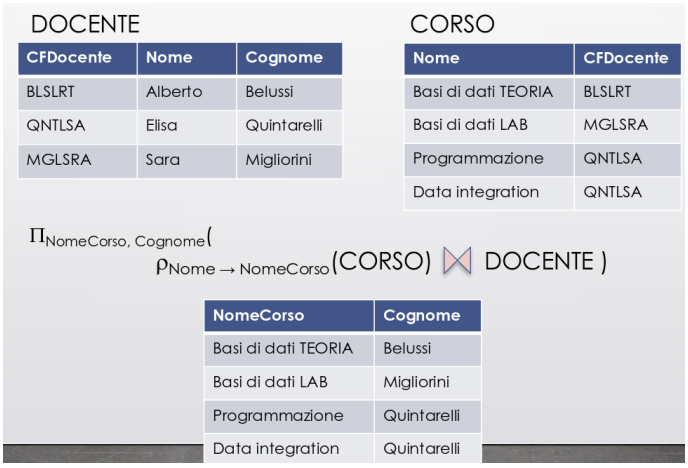
Esempio. Prendiamo lo schema:

```
DOCENTE(CFDocente, Nome, Cognome)
CORSO(Nome, CFDocente)
```

Si richiede di produrre l'insieme dei corsi riportando: nome del corso e cognome del docente.

$$\Pi_{\text{NomeCorso, Cognome}}(\rho_{\text{Nome} \to \text{NomeCorso}}(CORSO) \bowtie DOCENTE)$$

Graficamente si ottiene:



Il Join può essere ottenuto tramite un prodotto cartesiano e una selezione imponendo l'uguaglianza su tutti gli attributi in comune:

$$\sigma_{\text{NomeCorso} = \text{Nome}}(DOCENTE \times CORSO)$$

Il Join si dice completo se ogni tupla della relazione A contribuisce a generare almeno una tupla della relazione risultato, altrimenti si dice incompleto e le tuple che non contribuiscono al risultato si chiamano **dangling tuples**.

La cardinalità del Join Naturale è, in generale:

$$0 \leq |R1 \bowtie R2| \leq |R1| \cdot |R2|$$

Se il join è completo la cardinalità diventa:

$$max(|R1|, |R2|) \leq |R1 \bowtie R2| \leq |R1| \cdot |R2|$$

Se $X_1 \cap X_2$ è una superchiave per $R2$:

$$0 \leq |R1 \bowtie R2| \leq |R1|$$

Il caso più classico che ricounduce a questa situazione è quando si fa l'uguaglianza tra una *chiave* e una *chiave esportata*, ovvero abbiamo esportato la chiave di $R2$ su $R1$. Se prendo una tupla di $R1$ che ha tra i suoi attributi una *superchiave* di $R2$, si avrà al massimo una tupla di $R2$ che va in combinazione con la tupla di partenza di $R1$, perciò, *al massimo* la cardinalità sarà quella di $R1$. Se $X_1 \cap X_2$ è una superchiave di $R2$ ed esiste un *vincolo di integrità referenziale* tra $X_1 \cap X_2$, o una parte di $R1$, e $R2$: $|R1 \bowtie R2| = |R1|$.

1.3.3 Theta join ed equi-join

Nel theta join il predicato di join viene esplicitato, è indipendente dallo schema. Date le relazioni $R1$ e $R2$, rispettivamente di schema X_1 e X_2, la **precondizione** al theta join è che gli schemi di $R1$ e $R2$ devono essere *disgiunti*: $X_1 \cap X_2 = \varnothing$.

$$R1 \bowtie_\theta R2 = \sigma_\theta(R1 \bowtie R2)$$

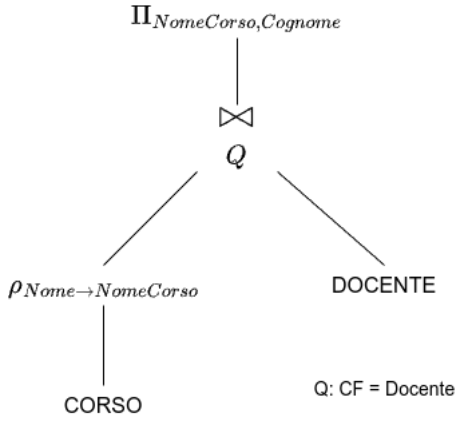
Dove il join naturale all'interno della selezione funge da *prodotto cartesiano* e θ è un predicato conforme alla sintassi prevista per l'operatore di selezione.

Esempio

`DOCENTE(CF, Nome, Cognome)`
`CORSO(Nome, Docente)`
Si richiede di produrre l'elenco di tutti i corsi riportando: nome del corso e cognome del docente.

$$\Pi_{\text{NomeCorso, Cognome}}(\rho_{\text{Nome} \to \text{NomeCorso}}(CORSO) \overset{\bowtie}{\underset{\text{CF = Docente}}{}} DOCENTE)$$

Graficamente:



Equi-join

Il theta join si dice *equi-join* se la condizione θ è una congiunzione di uguaglianze:

$$R1 \overset{\bowtie}{\underset{A_1 = B_1 \wedge \cdots \wedge A_n = B_n}{}} R2$$

Si noti che **non esiste** un operatore misto che applica la logica del join naturale per gli attributi comuni e aggiunge la condizione specificata esplicitamente attraverso il predicato θ.

Proprietà dell'equi-join

Date due relazioni $R1$ e $R2$ di schema X_1 e X_2, con $X_1 \cap X_2 = \{C_1, \ldots, C_m\}$ con $m > 0$, vale la seguente equivalenza:

$$R1 \bowtie R2 = \Pi_{X_1 \cup X_2}\left(R1 \overset{\bowtie}{\underset{C_1 = C'_1 \wedge \cdots \wedge C_m = C'_m}{}} \rho_{C_1,\ldots,C_m \to C'_1,\ldots,C'_m}(R2)\right)$$

In questo caso si hanno due relazioni che hanno attributi comuni ($\{C_1, \ldots, C_m\}$) e quindi si devono ridenominare gli attributi di una delle due relaizoni, nell'esempio quelli di $R2$. A questo punto gli schemi sono disgiunti e la condizione del theta-join sarà una congiunzione di uguaglianze: $C_1 = C'_1 \wedge \cdots \wedge C_m = C'_m$. Gli attributi che si ottengono nel risultato del theta-join sono la somma degli attributi di $R1$ e di $R2$, cosa che non avviene nel join naturale. Per questo motivo si proietta su $\Pi_{X_1 \cup X_2}$, in questo modo l'unione considera gli attributi di tutte e due le relazioni ma esclude i $C'_1, \ldots, C'_m$ generati nell'espressione.

1.3.4 Join esterni

Consentono di ottenere nel risultato del join tutte le tuple, anche le **dangling tuples**, di una o di entrambe le relazioni coinvolte nel join, eventualmente estese con valori nulli:

- **left join**: $r_1 \bowtie_{LEFT} r_2$;
- **right join**: $r_1 \bowtie_{RIGHT} r_2$;
- **full join**: $r_1 \bowtie_{FULL} r_2$;

1.4 Ottimizzazione

1.4.1 Valori nulli

E' opportuno estendere l'algebra relazionale affinché possa manipolare anche relazioni che contengono *valori nulli* (NULL). Le operazioni che devono essere raffinate per gestire relazioni che contengono valori nulli sono in particolare **selezione** e **join naturale**.
Le altre operazioni riportano semplicemente nelle tuple del risultato il valore nullo presente sulle tuple di input.

Selezione

Nel caso della **selezione** abbiamo che la selezione applicata a:

- $A\theta B$ risulta essere falsa (quindi non seleziona quella determinata tupla) se uno dei due attributi A o B è NULL.
- $A\theta const$ risulta essere falsa se l'attributo A è NULL.

Inoltre, si aggiungono le seguenti condizioni atomiche:

- `A IS NULL`: è una condizione atomica che viene valutata sulla tupla t. È **vero** se $t[A]$ *contiene* il valore NULL, altrimenti è **falso**.
- `A IS NOT NULL`: è una condizione atomica che viene valutata sulla tupla t. È **vero** se $t[A]$ <u>non</u> *contiene* il valore NULL, altrimenti è **falso**.

Join naturale

La condizione di uguaglianza sugli attributi comuni alle due relazioni è **falsa** sulle tuple t_1 e t_2 se almeno uno degli attributi comuni di t_1 o di t_2 è NULL. Inoltre, il confronto particolare $NULL = NULL$ è **sempre valutato FALSO**.

1.4.2 Ottimizzatore

Ogni espressione DML[1] ricevuta dal DBMS viene sottoposta ad un processo di elaborazione, tra cui, anche uno di ottimizzazione. L'ottimizzatore genera un'espressione equivalente all'interrogazione di input e di costo inferiore. quest'ultimo viene valutato in termini di **dimensione dei risultati intermedi**. L'ottimizzatore esegue **trasformazioni di equivalenza** allo scopo di riddurre la dimensione dei risultati intermedi.

Equivalenza tra espressioni algebriche

Equivalenza dipendente dallo schema. Dato uno schema R: $E_1 \equiv E_2$ se $E_1(r) = E_2(r)$ per ogni istanza di r di schema R.
Esempio:

$$\Pi_{AB}(R_1) \bowtie \Pi_{AC}(R_2) \equiv_R \Pi_{ABC}(R_1 \bowtie R_2)$$

Con $R = \{R_1(A, B, D), R_2(A, C, E)\}$.

La condizione generale che lo schema deve soddisfare è che l'**unico attributo comune tra R_1 e R_2 sia A**.

[1] Solitamente specificata in linguaggio dichiarativo

Equivalenza assoluta. È indipendente dallo schema.

$E_1 \equiv E_2$ se $E_1 \equiv_R E_2$ per ogni schema R compatibile con E_1 ed E_2.

Esempio:

$$\Pi_{AB}(\sigma_{A>0}(R_1)) \equiv \sigma_{A>0}(\Pi_{AB}(R1))$$

Trasformazioni di equivalenza

Le principali trasformazioni sono quattro. Consideriamo, per gli esempi, il seguente schema relazionale:
`TRENO(NumTreno, OrarioPart, Cat, Dest, OrarioArr)`
`FERMATA(NumTreno, Stazione, Orario)`

Sia E un'espressione di schema X, si definiscono le seguenti trasformazioni di equivalenza:

- **Atomizzazione delle selezioni:** una congiunzione di selezioni può essere sostituita da una sequenza di selezioni atomiche.

  $$\sigma_{\mathrm{F1} \wedge \mathrm{F2}}(E) \equiv \sigma_{\mathrm{F1}}(\sigma_{\mathrm{F2}}(E))$$

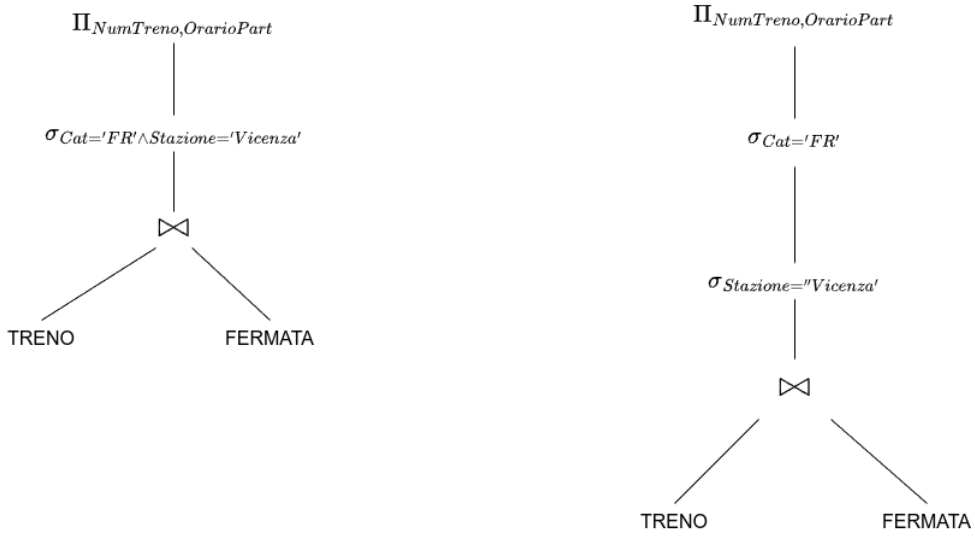
  È propedeutica ad altre trasformazioni. Non ottimizza se non è seguita da altre trasformazioni.
  Esempio:

  $$\Pi_{\mathrm{NumTreno,\ OrarioPart}}(\sigma_{\mathrm{Cat='FR'} \wedge \mathrm{Stazione='Vicenza'}}(TRENO \bowtie FERMATA)$$

  $$\Rightarrow$$

  $$\Pi_{\mathrm{NumTreno,\ OrarioPart}}(\sigma_{\mathrm{Cat='FR'}}(\sigma_{\mathrm{Stazione='Vicenza'}}(TRENO \bowtie FERMATA)))$$

  Graficamente:



- **Idempotenza delle proiezioni:** una proiezione può essere trasformata in una sequenza di proiezioni che eliminano i vari attributi in varie fasi.

  $$\Pi_{\mathrm{A}}(E) \equiv \Pi_{\mathrm{A}}(\Pi_{\mathrm{A,B}}(E)) \text{ dove } B \subseteq X$$

  È propedeutica ad altre trasformazioni. Non ottimizza se non è seguita da altre trasformazioni.

Siano E_1 ed E_2 espressioni di schema X_1 e X_2, si definiscono le seguenti trasformazioni di equivalenza:

- **Anticipazione della selezione rispetto al join:** questa espressione vale solo se F conivolge **solo** gli attributi di E_2.

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

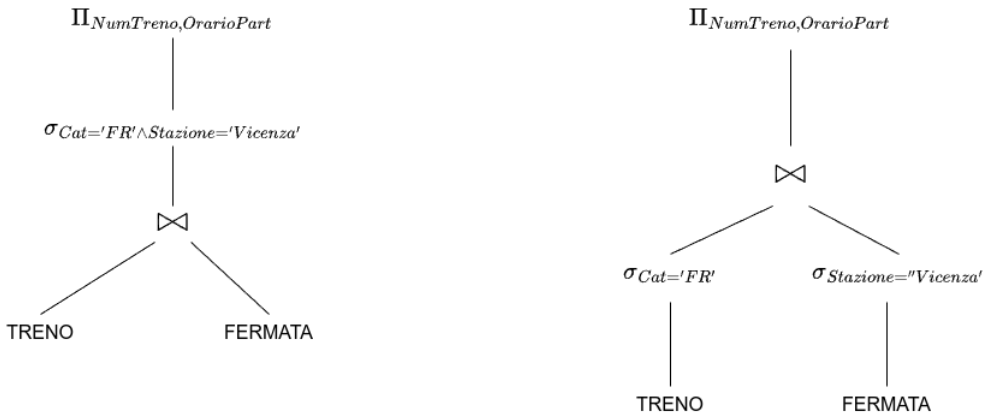
Esempio:

$$\Pi_{\text{NumTreno, OrarioPart}}(\sigma_{\text{Cat='FR'} \wedge \text{Stazione='Vicenza'}}(TRENO \bowtie FERMATA)$$

$$\Rightarrow$$

$$\Pi_{\text{NumTreno, OrarioPart}}(\sigma_{\text{Cat='FR'}}(TRENO)) \bowtie \sigma_{\text{Stazione='Vicenza'}}(FERMATA))$$

Graficamente:



- **Anticipazione della proiezione rispetto al Join:** vale solo se Y sono attributi di B e I suoi attributi sono coinvolti nel join.

$$\Pi_{\text{Y}}(A \bowtie B) \equiv A \bowtie (\Pi_{\text{Y}}(B))$$

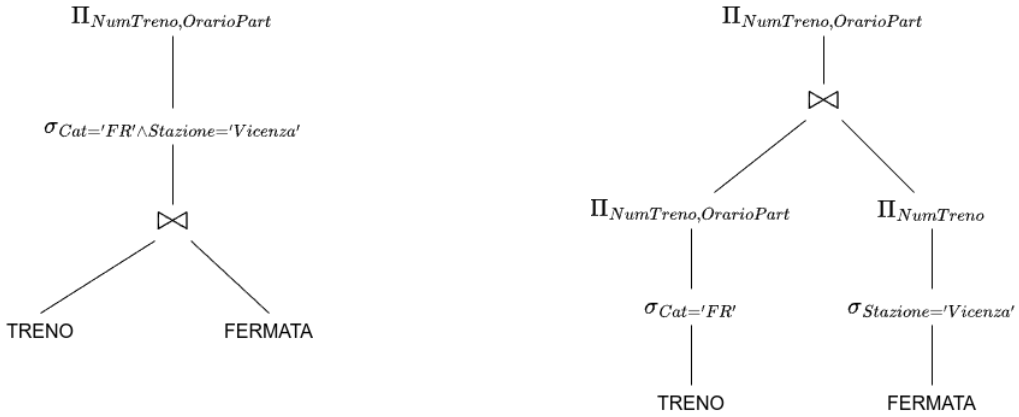
Esempio:

$$\Pi_{\text{NumTreno, OrarioPart}}(\sigma_{\text{Cat='FR'} \wedge \text{Stazione='Vicenza'}}(TRENO \bowtie FERMATA)$$

$$\Rightarrow$$

$$\Pi_{\text{NumTreno, OrarioPart}}(\Pi_{\text{NumTreno, OrarioPart}}(\sigma_{\text{Cat='FR'}}(TRENO)) \bowtie \Pi_{\text{NumTreno}}(\sigma_{\text{Stazione='Vicenza'}}(FERMATA))$$

Graficamente:



- Esistono poi altre otimizzazioni minori come l'inglobamento di una selezione in un join (da eseguire solo se non è possibile anticipare la selezione), in questo caso la condizione di selezione sarà assorbita nel **theta-join**.

$$\sigma_{\mathrm{F}}(A \bowtie B) \equiv A \bowtie_{\mathrm{F}} B$$

Infine è bene ricordare anche le trasformazioni con gli operatori insiemistici:

$$\sigma_{\mathrm{F\ or\ G}}(A) \equiv \sigma_{\mathrm{F}}(A) \cup \sigma_{\mathrm{G}}(A)$$
$$\sigma_{\mathrm{F\ and\ G}}(A) \equiv \sigma_{\mathrm{F}}(A) \cap \sigma_{\mathrm{G}}(A)$$

Chapter 2

Calcolo Relazionale