

Algebra relazionale

L'algebra relazionale è un linguaggio formale utilizzato nel contesto dei database relazionali per manipolare e interrogare dati.

È stata introdotta da [Edgar F. Codd](#), il quale ha definito una serie di operazioni algebriche che possono essere eseguite su insiemi di dati organizzati in tabelle (relazioni).

L'obiettivo principale è fornire un mezzo dichiarativo per specificare le operazioni da eseguire sui dati, senza specificare come effettivamente devono essere eseguite.

L'algebra relazionale è un linguaggio formale utilizzato per manipolare i dati contenuti nelle tabelle (relazioni) di un database. È basata su operatori matematici che permettono di definire operazioni su set di dati. I simboli principali dell'algebra relazionale sono:

1. Proiezione (π)

- **Simbolo:** π
- **Descrizione:** L'operatore di proiezione seleziona una o più colonne (attributi) da una relazione, eliminando le colonne non selezionate. È simile all'operazione di *SELECT* in SQL.
- **Sintassi:** $\pi_{A1, A2, \dots, An}(R)$
 - Seleziona solo gli attributi $A1, A2, \dots, An$ dalla relazione R .

2. Selezione (σ)

- **Simbolo:** σ
- **Descrizione:** L'operatore di selezione estrae le righe (tuple) che soddisfano una determinata condizione da una relazione. È simile all'operazione di *WHERE* in SQL.
- **Sintassi:** $\sigma_{\text{condizione}}(R)$
 - Seleziona tutte le tuple di R che soddisfano la condizione condizione .

3. Unione (\cup)

- **Simbolo:** \cup
- **Descrizione:** L'operatore di unione combina due relazioni con lo stesso schema (stessi attributi), restituendo tutte le tuple uniche presenti in entrambe le relazioni.
- **Sintassi:** $R \cup S$
 - Restituisce una relazione che contiene tutte le tuple di R e S .

4. Intersezione (\cap)

- **Simbolo:** \cap
- **Descrizione:** L'operatore di intersezione restituisce le tuple comuni tra due relazioni con lo stesso schema.
- **Sintassi:** $R \cap S$
 - Restituisce le tuple che sono presenti sia in R che in S .

5. Differenza (–)

- **Simbolo:** –
- **Descrizione:** L'operatore di differenza restituisce le tuple che sono in una relazione ma non nell'altra. È simile all'operazione di *EXCEPT* in SQL.
- **Sintassi:** $R - S$
 - Restituisce le tuple che sono in R ma non in S .

6. Prodotto cartesiano (\times)

- **Simbolo:** \times
- **Descrizione:** L'operatore di prodotto cartesiano restituisce tutte le combinazioni possibili di tuple tra due relazioni. Ogni tupla della prima relazione viene combinata con ogni tupla della seconda relazione.
- **Sintassi:** $R \times S$
 - Restituisce una relazione che è il prodotto cartesiano delle relazioni R e S .

7. Join (\bowtie)

- **Simbolo:** \bowtie
- **Descrizione:** L'operatore di join unisce due relazioni basandosi su una condizione di corrispondenza tra gli attributi. Esistono diversi tipi di join, ma il più comune è l'**inner join**.
- **Sintassi:** $R \bowtie S$
 - Restituisce una relazione che unisce R e S su una condizione di uguaglianza fra attributi comuni (spesso una chiave primaria e una chiave esterna).

8. Ridenominazione (ρ)

- **Simbolo:** ρ
- **Descrizione:** L'operatore di ridenominazione cambia i nomi degli attributi o delle relazioni. È utile quando si vogliono evitare conflitti di nomi o semplificare i nomi degli attributi.
- **Sintassi:** $\rho_{\text{NomeNuovo}}(R)$
 - Ridenomina la relazione R con il nome NomeNuovo .

9. Divisione (\div)

- **Simbolo:** \div
- **Descrizione:** L'operatore di divisione è utilizzato per ottenere le tuple di una relazione che sono associate a tutte le tuple di una seconda relazione. È particolarmente utile quando si desidera trovare "cose che appartengono a tutto".
- **Sintassi:** $R \div S$
 - Restituisce le tuple di R che sono associate a tutte le tuple di S .

10. Aggregazione

- **Simbolo:** (Nessun simbolo universale per l'aggregazione, ma si utilizza un insieme di funzioni come `SUM` , `AVG` , `COUNT` , `MIN` , `MAX` , etc.)
- **Descrizione:** L'aggregazione non è un operatore primario dell'algebra relazionale classica, ma viene utilizzata nelle implementazioni più moderne per calcolare valori aggregati come somma, media, numero di occorrenze, ecc.
- **Sintassi:** `R[agg_func(A)]`
 - Esegue una funzione di aggregazione `agg_func` sull'attributo `A` nella relazione `R` .

Esempio di combinazione degli operatori:

Immagina di voler trovare tutte le persone che abitano in città con un affitto mensile maggiore di 500 euro. Potremmo utilizzare:

```
 $\pi_{\text{Nome}, \text{Città}}(\sigma_{\text{CostoAffittoMensile} > 500}(\text{ALLOGGIO}))$ 
```

In questo caso:

- $\sigma_{\text{CostoAffittoMensile} > 500}(\text{ALLOGGIO})$ seleziona solo gli affitti maggiori di 500.
- $\pi_{\text{Nome}, \text{Città}}$ proietta solo i nomi e le città delle persone che soddisfano il criterio.

Conclusione:

Gli operatori dell'algebra relazionale sono fondamentali per esprimere query sui dati e sono la base per comprendere linguaggi più complessi come SQL. Utilizzando questi operatori, possiamo effettuare una vasta gamma di operazioni sui dati, come la selezione, la proiezione, l'unione, l'intersezione, la differenza, e molto altro.

Alcuni simboli dell'algebra relazionale sono rappresentati con lettere greche o caratteri speciali. Ecco l'elenco con i simboli e le loro corrispondenze:

1. Proiezione - π (Pi greco)

- Rappresenta l'operatore di proiezione.

2. Selezione - σ (Sigma maiuscola)

- Rappresenta l'operatore di selezione.

3. Unione - \cup (Unione)

- Rappresenta l'operatore di unione tra due relazioni.

4. Intersezione - \cap (Intersezione)

- Rappresenta l'operatore di intersezione tra due relazioni.

5. Differenza - $-$ (Segno meno)

- Rappresenta l'operatore di differenza tra due relazioni.

6. Prodotto cartesiano - \times (Perse cartesian)

- Rappresenta l'operatore di prodotto cartesiano tra due relazioni.

7. Join - \bowtie (Join)

- Rappresenta l'operatore di join (spesso inner join) tra due relazioni.

8. Ridenominazione - ρ (Rho)

- Rappresenta l'operatore di ridenominazione di una relazione o di un attributo.

9. Divisione - \div (Divisione)

- Rappresenta l'operatore di divisione tra due relazioni.

Esempi

Ecco degli esempi per ciascun operatore dell'algebra relazionale, con un contesto semplice per aiutarti a capire come utilizzarli.

1. Proiezione (π)

Se abbiamo una relazione **STUDENTI** con gli attributi **ID** , **Nome** , **Città** , e vogliamo selezionare solo i nomi e le città degli studenti, l'espressione di proiezione sarà:

```
 $\pi_{\text{Nome}, \text{Città}}(\text{STUDENTI})$ 
```

Questo restituirà una nuova relazione contenente solo i dati degli studenti relativi a **Nome** e **Città**.

2. Selezione (σ)

Supponiamo di avere una relazione **STUDENTI** con gli attributi **ID** , **Nome** , **Età** , e vogliamo selezionare gli studenti che hanno un'età maggiore di 20 anni. L'espressione di selezione sarà:

```
 $\sigma_{\text{Età} > 20}(\text{STUDENTI})$ 
```

Questo restituirà una relazione contenente solo gli studenti che soddisfano la condizione **Età > 20**.

3. Unione (\cup)

Immagina di avere due relazioni **STUDENTI_1** e **STUDENTI_2**, entrambe con gli stessi attributi **ID** e **Nome**. Se vogliamo unire gli studenti di entrambe le relazioni, l'espressione di unione sarà:

```
STUDENTI_1  $\cup$  STUDENTI_2
```

Questo restituirà una relazione contenente tutte le tuple (studente) di **STUDENTI_1** e **STUDENTI_2**, senza duplicati.

4. Intersezione (\cap)

Supponiamo di avere due relazioni **STUDENTI_1** e **STUDENTI_2**, entrambe con gli stessi attributi **ID** e **Nome**, e vogliamo trovare gli studenti che sono presenti in entrambe le relazioni. L'espressione di intersezione sarà:

```
STUDENTI_1  $\cap$  STUDENTI_2
```

Questo restituirà una relazione contenente solo gli studenti che sono presenti sia in **STUDENTI_1** che in **STUDENTI_2**.

5. Differenza (–)

Immagina di avere due relazioni **STUDENTI_1** e **STUDENTI_2** e vogliamo trovare gli studenti che sono presenti in **STUDENTI_1** ma non in **STUDENTI_2**. L'espressione di differenza sarà:

```
STUDENTI_1 – STUDENTI_2
```

Questo restituirà una relazione contenente gli studenti che sono in **STUDENTI_1**, ma non in **STUDENTI_2**.

6. Prodotto cartesiano (\times)

Se abbiamo due relazioni **CORSI** con gli attributi `CodCorso` e `NomeCorso`, e **STUDENTI** con gli attributi `ID` e `Nome`, e vogliamo fare il prodotto cartesiano tra queste due relazioni, l'espressione sarà:

```
CORSI  $\times$  STUDENTI
```

Questo restituirà una relazione che combina ogni tupla di **CORSI** con ogni tupla di **STUDENTI**, creando tutte le possibili combinazioni.

7. Join (⋈)

Supponiamo di avere due relazioni **STUDENTI** con gli attributi **ID** e **Nome** e **ISCRIZIONI** con gli attributi **ID_Studente** e **CodCorso**, e vogliamo trovare tutti gli studenti iscritti a un corso specifico. L'espressione di join sarà:

```
STUDENTI ⋈ ISCRIZIONI
```

Questo restituirà una relazione contenente tutte le combinazioni di **STUDENTI** e **ISCRIZIONI** dove l'attributo **ID** di **STUDENTI** corrisponde all'attributo **ID_Studente** di **ISCRIZIONI**.

8. Ridenominazione (ρ)

Immagina di avere una relazione **STUDENTI** con gli attributi **ID** e **Nome**, e vogliamo ridenominare la relazione **STUDENTI** in **ALUNNI** e cambiare l'attributo **ID** in **ID_Studente**. L'espressione di ridenominazione sarà:

```
 $\rho_{\text{ALUNNI}}(\text{ID\_Studente}, \text{Nome})(\text{STUDENTI})$ 
```

Questo restituirà la relazione **STUDENTI**, ma con il nome della relazione cambiato in **ALUNNI** e l'attributo **ID** ridenominato in **ID_Studente**.

9. Divisione (\div)

Supponiamo di avere due relazioni **STUDENTI_ISCRITTI** con gli attributi **ID_Studente** , **CodCorso** , e **CORSI** con gli attributi **CodCorso** , e vogliamo trovare gli studenti che sono iscritti a **tutti i corsi**. L'espressione di divisione sarà:

```
STUDENTI_ISCRITTI  $\div$  CORSI
```

Questo restituirà una relazione contenente gli **ID_Studente** degli studenti che sono iscritti a **tutti i corsi** presenti in **CORSI**.

Questi esempi mostrano come applicare gli operatori dell'algebra relazionale in scenari tipici di un database. Ogni operatore è utilizzato per estrarre, combinare o manipolare i dati in base a specifici criteri.

Le espressioni dell'algebra relazionale rappresentano operazioni su relazioni (tabelle) in un database e sono lette in un modo specifico. Ecco una guida su come leggere ciascuna delle espressioni.

1. Proiezione (π)

Espressione:

```
 $\pi_{\text{Nome}, \text{Città}}(\text{STUDENTI})$ 
```

Lettura:

"Proiezione degli attributi **Nome** e **Città** dalla relazione **STUDENTI**."

La proiezione serve a selezionare solo alcune colonne da una relazione, escludendo le altre.

2. Selezione (σ)

Espressione:

```
 $\sigma_{\text{Età} > 20}(\text{STUDENTI})$ 
```

Lettura:

"Selezione degli studenti dalla relazione **STUDENTI** per cui l'attributo **Età** è maggiore di 20."

La selezione è un filtro che estrae solo le righe (tuple) che soddisfano una certa condizione.

3. Unione (\cup)

Espressione:

```
STUDENTI_1  $\cup$  STUDENTI_2
```

Lettura:

"Unione delle relazioni **STUDENTI_1** e **STUDENTI_2**."

L'unione combina tutte le righe delle due relazioni, eliminando i duplicati.

4. Intersezione (\cap)

Espressione:

```
STUDENTI_1  $\cap$  STUDENTI_2
```

Lettura:

"Intersezione delle relazioni **STUDENTI_1** e **STUDENTI_2**."

L'intersezione restituisce le righe che sono comuni a entrambe le relazioni.

5. Differenza (–)

Espressione:

```
STUDENTI_1 – STUDENTI_2
```

Lettura:

"Differenza tra le relazioni **STUDENTI_1** e **STUDENTI_2**."

La differenza restituisce tutte le righe che sono presenti nella prima relazione ma non nella seconda.

6. Prodotto cartesiano (\times)

Espressione:

```
CORSI  $\times$  STUDENTI
```

Lettura:

"Prodotto cartesiano delle relazioni **CORSI** e **STUDENTI**."

Il prodotto cartesiano combina ogni riga della prima relazione con ogni riga della seconda, creando tutte le possibili combinazioni.

7. Join (⋈)

Espressione:

```
STUDENTI ⋈ ISCRIZIONI
```

Lettura:

"Join tra le relazioni **STUDENTI** e **ISCRIZIONI**."

Il join combina le righe delle due relazioni dove esiste una corrispondenza fra gli attributi specificati. Spesso si fa il join basandosi su un campo chiave (come **ID_Studente**).

8. Ridenominazione (ρ)

Espressione:

```
 $\rho_{\text{ALUNNI}}(\text{ID\_Studente}, \text{Nome})(\text{STUDENTI})$ 
```

Lettura:

"Ridenominazione della relazione **STUDENTI** in **ALUNNI** e dell'attributo **ID** in **ID_Studente**."

La ridenominazione cambia il nome di una relazione o di uno degli attributi di una relazione.

9. Divisione (\div)

Espressione:

```
STUDENTI_ISCRITTI  $\div$  CORSI
```

Lettura:

"Divisione della relazione **STUDENTI_ISCRITTI** per la relazione **CORSI**."

La divisione è un'operazione complessa che restituisce le righe della prima relazione (ad esempio, gli studenti) che sono "collegate" a tutte le righe della seconda relazione (ad esempio, a tutti i corsi). Si usa spesso per rispondere a domande come "quali studenti sono iscritti a tutti i corsi?"

Sintesi generale per leggere le espressioni:

- Gli operatori (π , σ , \cup , \cap , $-$, \times , \bowtie , ρ , \div) sono letti come "**operazione su**".
- Le relazioni (ad esempio **STUDENTI**, **CORSI**, **ISCRIZIONI**) sono le tabelle sui quali l'operatore agisce.
- Gli attributi sono specificati nelle espressioni quando è necessario, come nei casi di π (proiezione) o σ (selezione) dove specifichiamo i campi da considerare o le condizioni da applicare.

Consideriamo due tabelle di esempio:

Tabella "Studenti":

ID	Nome	Corso
1	Mario	Mat
2	Anna	Fis
3	Carlo	Mat
4	Elena	Fis

Tabella "Corsi":

Corso	Docente
Mat	Rossi
Fis	Bianchi
Storia	Verdi

Ora vediamo esempi di operazioni dell'algebra relazionale:

1. Selezione (σ):

- Trova tutti gli studenti che seguono il corso di Matematica:

[$\sigma_{\text{Corso}='Mat'}(\text{Studenti})$]

Risultato:

ID	Nome	Corso
1	Mario	Mat
3	Carlo	Mat

2. Proiezione (π):

- Seleziona solo il nome degli studenti:

[π_{Nome} (Studenti)]

Risultato:

Nome
Mario
Anna
Carlo
Elena

3. Unione (\cup):

- Combina gli studenti di Matematica e Fisica:
[$\text{Studenti} \cup \text{Studenti}_{\{\text{Fis}\}}$]

Risultato:

ID	Nome	Corso
1	Mario	Mat
2	Anna	Fis
3	Carlo	Mat
4	Elena	Fis

4. Intersezione (\cap):

- Trova gli studenti che seguono entrambi i corsi di Matematica e Fisica:
[$\text{Studenti}_{\{\text{Mat}\}} \cap \text{Studenti}_{\{\text{Fis}\}}$]

Risultato:

ID	Nome	Corso
3	Carlo	Mat

5. Differenza (-):

- Trova gli studenti che seguono Matematica ma non Fisica:
[Studenti_{Mat} - Studenti_{Fis}]

Risultato:

ID	Nome	Corso
1	Mario	Mat
3	Carlo	Mat

6. Prodotto Cartesiano (\times):

- Crea tutte le possibili coppie tra studenti e corsi:

[Studenti \times Corsi]

Risultato:

ID	Nome	Corso	Docente
1	Mario	Mat	Rossi
1	Mario	Fis	Bianchi
1	Mario	Storia	Verdi
2	Anna	Mat	Rossi
2	Anna	Fis	Bianchi
2	Anna	Storia	Verdi
...

7. Join (⋈):

- Combina gli studenti e i corsi in base al corso:

[Studenti ⋈_{Studenti.Corso=Corsi.Corso} Corsi]

Risultato:

ID	Nome	Corso	Docente
1	Mario	Mat	Rossi
2	Anna	Fis	Bianchi
3	Carlo	Mat	Rossi
4	Elena	Fis	Bianchi

8. Division (\div):

- Trova gli studenti che seguono tutti i corsi:

[Studenti \div Corsi]

Risultato:

ID	Nome
1	Mario
2	Anna

Recap

Le operazioni principali dell'algebra relazionale includono:

1. **Selezione (σ):** Estrae righe dalla tabella che soddisfano una determinata condizione.

1. **Proiezione (π)**: Seleziona colonne specifiche dalla tabella.

2. **Unione (\cup):** Combina due tabelle mantenendo tutte le righe, eliminando duplicati.

3. Intersezione (\cap): Restituisce le righe presenti in entrambe le tabelle.

4. Differenza (-): Restituisce le righe presenti in una tabella ma non nell'altra.

5. Prodotto cartesiano (\times): Combina tutte le righe di una tabella con tutte le righe di un'altra tabella.

6. **Join (⋈)**: Combina le righe di due tabelle in base a una condizione specificata.

7. **Division (\div):** Trova tutte le tuple nella prima tabella che sono correlate a tutte le tuple nella seconda tabella.

Queste operazioni formano il nucleo dell'algebra relazionale e forniscono un modo rigoroso per manipolare e combinare dati all'interno di database relazionali.

L'utilizzo di un linguaggio dichiarativo come l'algebra relazionale consente agli utenti di concentrarsi sulla specifica di cosa desiderano ottenere, senza dover preoccuparsi dell'implementazione dettagliata delle operazioni.