

# Modello Relazionale: Algebra Relazionale

Prof. Fedeli Massimo - Tutti i diritti riservati

ITS 4.0

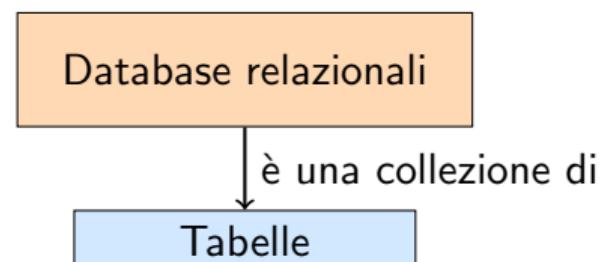
16 dicembre 2025

# Indice

- 1 Introduzione al Modello Relazionale
- 2 Linguaggi di Interrogazione
- 3 Algebra Relazionale: Operatori
- 4 Esempio di Database
- 5 Operatore di Selezione
- 6 Operatore di Proiezione
- 7 Operatore di Ridenominazione
- 8 Operatori Binari
- 9 Prodotto Cartesiano e Join

## Definizione

Un **database relazionale** è una collezione di **tabelle**



## Operazioni di base:

- Selezione
- Proiezione

(operazioni unarie)

## Operazioni complesse:

- Prodotto cartesiano
- Unione
- Differenza

(operazioni binarie)

## Operazioni derivate

Intersezione, Congiunzione (Join)

## E.F. Codd (1970)

Il modello relazionale è stato introdotto formalmente da Edgar F. Codd nel 1970

### Caratteristiche principali:

- Descrizione semplice ma rigorosa dei dati
- Basato sul concetto di **relazione matematica**
- Fondamento teorico: teoria degli insiemi e logica dei predicati

# Rappresentazione dei Dati

- I dati sono rappresentati come **tabelle bidimensionali**
- Ogni tabella rappresenta un'entità del mondo reale
- Le righe sono **tuple** (record)
- Le colonne sono **attributi** (campi)

## Esempio

Tabella STUDENTI: ogni riga rappresenta uno studente

Due famiglie principali:

## ① Algebra Relazionale

- Linguaggio procedurale
- Notazione algebrica
- Usa 5 operatori fondamentali

## ② Calcolo Relazionale

- Linguaggio dichiarativo
- Notazione logica
- Base per SQL

## Definizione

Linguaggio procedurale con notazione algebrica che utilizza **5 operatori fondamentali**

## Caratteristiche:

- Usa simboli matematici
- Permette di formulare interrogazioni complesse
- Fornisce il pieno potere espressivo

Usa simboli matematici!  $\sigma, \pi, \times, \cup, -$

## Calcolo Relazionale

Linguaggio di alto livello con caratteristiche dichiarative

### SQL (Structured Query Language):

- Parzialmente dichiarativo
- L'utente specifica **cosa** vuole ottenere
- Non specifica **come** ottenerlo
- Il DBMS decide la strategia di esecuzione

## Ricorda

SQL è il linguaggio che studieremo nella prossima unità!

# Dichiarativo vs Procedurale

## Linguaggio Dichiarativo

- ✓ Specifica il risultato
- ✓ Non i passaggi
- ✓ Più semplice

## Linguaggio Procedurale

- Specifica i passaggi
- Sequenza di operazioni
- Più dettagliato

## SQL

SQL è dichiarativo: dici **cosa** vuoi, non **come** ottenerlo

# I 5 Operatori Fondamentali

Simbolo	Nome	Tipo
$\sigma$	Selezione	Unario
$\pi$	Proiezione	Unario
$\times$	Prodotto cartesiano	Binario
$-$	Differenza	Binario
$\cup$	Unione	Binario

**Proprietà importante:** L'algebra è **chiusa**  
⇒ Ogni operazione restituisce una relazione!

# Operatori Unari vs Binari

## Operatori Unari

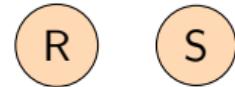
Si applicano a **una sola** relazione e restituiscono una relazione

## Operatori Binari

Si applicano a **due** relazioni e restituiscono una relazione



Unario



Binario

# Operatori Derivati

Dai 5 operatori di base derivano:

- **Intersezione ( $\cap$ )**
- **Congiunzione (Join) ( $\bowtie$ )**
  - Natural Join
  - Left Join
  - Right Join
  - Full Join

## Importante

Questi operatori sono molto utili nella pratica!

## Concetto chiave

Le tabelle relazionali sono **insiemi** e le righe sono **elementi**

## Conseguenze:

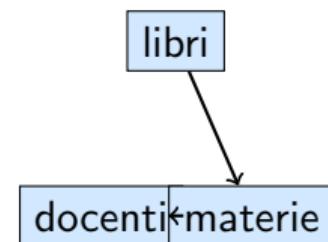
- Possiamo applicare operazioni insiemistiche
- Unione, intersezione, differenza
- Il risultato è sempre una relazione

Tabella = Insieme di tuple

# Database di Esempio

## Scenario

Database scolastico con tre tavole: libri, docenti, materie



# Tabella LIBRI

isbn	titolo	autore	editore	id_materia	anno
11223344	Il linguaggio C	Camagni	Hoepli	MA01	2020
11223355	Il Novecento	Verdi	Perinazzi	MA04	2021
11223366	Java	Nikolassy	Hoepli	MA01	2022

**Chiave primaria:** isbn

**Chiave esterna:** id\_materia

# Tabella DOCENTI

<b>matricola</b>	<b>cognome</b>	<b>nome</b>	<b>id_materia</b>	<b>telefono</b>
10100	Verdi	Mario	MA01	111.1234567
20100	Bianchi	Filippo	MA02	222.1234567
30100	Neri	Antonio	MA01	333.1234567

**Chiave primaria:** matricola

**Chiave esterna:** id\_materia

# Tabella MATERIE

<b>id_materia</b>	<b>nome</b>
MA01	Informatica
MA02	Fisica
MA03	Italiano
MA04	Storia

**Chiave primaria:** id\_materia

# Selezione ( $\sigma$ ) - Introduzione

## Definizione

La **selezione** restituisce un sottoinsieme di tuple che soddisfano una condizione

## Caratteristiche:

- Operatore unario (una relazione  $\rightarrow$  una relazione)
- Seleziona righe (tuple) in base a una condizione
- Simbolo:  $\sigma$  (sigma)

$$\sigma_{\text{condizione}}(R)$$

## Struttura generale:

$$\sigma_{\text{condizione booleana}_i}(R)$$

## La condizione può essere:

- attributo = valore
- attributo > valore
- attributo <> valore
- Condizioni complesse con AND, OR, NOT

# Esempio di Selezione

**Richiesta:** Selezionare i libri con editore "Hoepli"

**In algebra relazionale:**

$$\sigma_{\text{editore} = \text{"Hoepli"}}(\text{libri})$$

**Risultato:**

isbn	titolo	autore	editore	id_materia	anno
11223344	Il linguaggio C	Camagni	Hoepli	MA01	2020
11223366	Java	Nikolassy	Hoepli	MA01	2022

## Operatori logici disponibili:

- AND (congiunzione)
- OR (disgiunzione)
- NOT (negazione)

## Esempio:

$$\sigma_{autore = "Camagni" \text{ AND } editore = "Hoepli"}(\text{libri})$$

## Proprietà Commutativa

Più selezioni consecutive possono essere combinate con AND

**Le selezioni godono della proprietà commutativa:**

$$\sigma_{\text{cond1}}(\sigma_{\text{cond2}}(R)) = \sigma_{\text{cond1 AND cond2}}(R)$$

**Vantaggio:**

- Possiamo combinare più condizioni
- L'ordine non importa
- Ottimizzazione delle query

# Proiezione ( $\pi$ ) - Introduzione

## Definizione

La **proiezione** restituisce un sottoinsieme di attributi (colonne)

## Caratteristiche:

- Operatore unario
- Seleziona colonne (attributi)
- Simbolo:  $\pi$  (pi greco)
- Elimina duplicati automaticamente

$$\pi_{\text{lista attributi}}(R)$$

## Struttura generale:

$$\pi_{\text{attr}_1, \text{attr}_2, \dots, \text{attr}_n}(R)$$

## Note importanti:

- Il numero di attributi è il **grado della proiezione**
- Gli attributi sono separati da virgola
- L'ordine degli attributi è significativo

# Esempio di Proiezione (1)

**Richiesta:** Selezionare solo il titolo dei libri

**In algebra relazionale:**

$$\pi_{\text{titolo}}(\text{libri})$$

**Risultato:**

titolo
Il linguaggio C
Il Novecento
Java

Grado della proiezione: 1

## Esempio di Proiezione (2)

**Richiesta:** Selezionare titolo e autore dei libri

**In algebra relazionale:**

$$\pi_{\text{titolo, autore}}(\text{libri})$$

**Risultato:**

titolo	autore
Il linguaggio C	Camagni
Il Novecento	Verdi
Java	Nikolassy

Grado della proiezione: 2

**Possiamo combinare i due operatori!**

**Esempio:** Titolo e anno dei libri di Camagni

$$\pi_{\text{titolo, anno}}(\sigma_{\text{autore} = \text{"Camagni"}}(\text{libri}))$$

**Esecuzione:**

- ① Prima: selezione (filtrà le righe)
- ② Poi: proiezione (seleziona le colonne)

**Importante**

L'ordine delle operazioni è significativo!

# Esempio Completo

## Passi dell'elaborazione:

### ① Selezione:

$\text{LibriDiCamagni} \leftarrow \sigma_{\text{autore} = \text{"Camagni"}}(\text{libri})$

### ② Proiezione:

$\text{Risultato} \leftarrow \pi_{\text{titolo, anno}}(\text{LibriDiCamagni})$

## Risultato finale:

titolo	anno
Il linguaggio C	2020

# Ridenominazione ( $\rho$ )

## Definizione

La **ridenominazione** consente di rinominare gli attributi per eliminare ambiguità

## Sintassi:

$$\rho_{\text{nuovoNome} \leftarrow \text{vecchioNome}}(\text{tabella})$$

## Utilizzo:

- Chiarire risultati intermedi
- Evitare conflitti di nomi
- Preparare join complesse

# Esempio di Ridenominazione

**Scenario:** Rinominare "nome" in "disciplina" nella tabella materie

$$\rho_{\text{disciplina} \leftarrow \text{nome}}(\text{materie})$$

<b>Prima:</b>	<table border="1"><tr><td>id_materia</td><td>nome</td></tr></table>	id_materia	nome
id_materia	nome		
<b>Dopo:</b>	<table border="1"><tr><td>id_materia</td><td>disciplina</td></tr></table>	id_materia	disciplina
id_materia	disciplina		

## Requisito

Le due relazioni devono avere la **stessa struttura**

### I tre operatori insiemistici:

- ① **Unione** ( $\cup$ )
- ② **Intersezione** ( $\cap$ )
- ③ **Differenza** ( $-$ )

Più l'operatore:

- **Prodotto Cartesiano** ( $\times$ )

# Tabelle di Esempio

Per gli esempi useremo:

alunni\_ripetenti

matr.	cognome	nome
10100	Verdi	Mario
20100	Bianchi	Filippo
30100	Neri	Antonio

alunni\_motorizzati

matr.	cognome	nome
10100	Verdi	Mario
40100	Gialli	Anna
50100	Rossi	Pina

Stesso schema: *matricola, cognome, nome*

# Unione ( $\cup$ )

## Definizione

Restituisce **tutte le tuple** presenti in almeno una delle due relazioni

## Sintassi:

$$R \cup S$$

## Proprietà:

- Elimina automaticamente i duplicati
- Commutativa:  $R \cup S = S \cup R$
- Le relazioni devono avere la stessa struttura

# Esempio di Unione

**Operazione:**

`alunni_ripetenti ∪ alunni_motorizzati`

**Risultato:**

<b>matricola</b>	<b>cognome</b>	<b>nome</b>
10100	Verdi	Mario
20100	Bianchi	Filippo
30100	Neri	Antonio
40100	Gialli	Anna
50100	Rossi	Pina

5 righe totali (Verdi Mario compare una sola volta)

# Intersezione ( $\cap$ )

## Definizione

Restituisce le tuple che sono presenti in **entrambe** le relazioni

## Sintassi:

$$R \cap S$$

## Proprietà:

- Solo le righe comuni
- Commutativa:  $R \cap S = S \cap R$
- Operatore derivato (non fondamentale)

# Esempio di Intersezione

**Operazione:**

`alunni_ripetenti ∩ alunni_motorizzati`

**Risultato:**

<b>matricola</b>	<b>cognome</b>	<b>nome</b>
10100	Verdi	Mario

Solo 1 riga: quella presente in entrambe le tabelle

# Differenza (-)

## Definizione

Restituisce le tuple presenti nella **prima** relazione ma **non** nella seconda

## Sintassi:

$$R - S$$

## Proprietà:

- **NON commutativa:**  $R - S \neq S - R$
- L'ordine è importante!

# Esempio di Differenza

**Operazione 1:** alunni\_ripetenti – alunni\_motorizzati

20100	Bianchi	Filippo
30100	Neri	Antonio

**Operazione 2:** alunni\_motorizzati – alunni\_ripetenti

40100	Gialli	Anna
50100	Rossi	Pina

Risultati diversi! L'ordine conta!

# Prodotto Cartesiano ( $\times$ )

## Definizione

Produce una nuova tupla dalla combinazione di **ogni tupla** di R con **ogni tupla** di S

## Sintassi:

$$R \times S$$

## Caratteristiche:

- Combina tutte le righe possibili
- Genera molte tuple "inutili"
- Base per la JOIN

# Esempio di Prodotto Cartesiano

**Operazione:** libri  $\times$  materie

**Risultato parziale:**

isbn	titolo	id_mat.	anno	ID_mat.	nome
11223344	Il linguaggio C	MA01	2020	MA01	Informatica
11223344	Il linguaggio C	MA01	2020	MA02	Fisica
...	...	...	...	...	...

Se libri ha 3 righe e materie ha 4 righe:

Risultato =  $3 \times 4 = 12$  righe!

# Problema del Prodotto Cartesiano

Attenzione!

Il prodotto cartesiano da solo non è molto utile!

**Problemi:**

- Genera troppe tuple
- La maggior parte sono "senza senso"
- Combinazioni non significative

**Soluzione:**

Prodotto cartesiano + Selezione = **JOIN**

# Congiunzione o Join ( $\bowtie$ )

## Definizione

La **Join** combina tuple di due relazioni che hanno **un attributo in comune**

## Sintassi:

$$R \bowtie_{\text{condizione}} S$$

**Pronuncia:** Il simbolo  $\bowtie$  si pronuncia "bowtie" (cravatta a farfalla)

# Esempio di Join

## Operazione:

$\text{libri} \bowtie_{\text{id\_materia} = \text{ID\_materia}} \text{materie}$

## Equivale a:

- ① Prodotto cartesiano
- ② Selezione con condizione di uguaglianza
- ③ Elimina colonne duplicate

## Equi-Join

Quando la condizione è un'uguaglianza, si chiama **equi-join**

# Natural Join (\*)

## Natural Join

Join automatica sugli attributi con lo **stesso nome**

Sintassi:

$$R * S$$

Esempio:

libri \* materie

Unisce automaticamente su `id_materia!`

**Nota:** Gli attributi comuni appaiono una sola volta

# Left Join

## Left Join

Conserva **tutte le righe** della tabella di **sinistra**

### Esempio:

```
linee = ⋈id_autista autisti
```

### Risultato:

- Tutte le linee, anche senza autista
- Campi mancanti: NULL

# Right Join

## Right Join

Conserva **tutte le righe** della tabella di **destra**

### Esempio:

```
linee  $\bowtie_{id\_autista}$  autisti
```

### Risultato:

- Tutti gli autisti, anche senza linea
- Campi mancanti: NULL

# Full Join

## Full Join

Conserva **tutte le righe** di **entrambe** le tabelle

### Sintassi:

$$\text{linee} = \bowtie_{\text{id\_autista}} \text{autisti}$$

### Risultato:

- Tutte le linee (anche senza autista)
- Tutti gli autisti (anche senza linea)
- Campi mancanti: NULL

# Riepilogo delle Join

<b>Tipo di Join</b>	<b>Cosa conserva</b>
Equi-Join / Natural Join	Solo righe corrispondenti
Left Join	Tutte le righe di sinistra
Right Join	Tutte le righe di destra
Full Join	Tutte le righe di entrambe

## In SQL

Queste operazioni corrispondono a:

INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN

# Grazie per l'attenzione!

Domande?

*IIS Fermi Sacconi Ceci  
Informatica - 5B Inf.  
A.S. 2025/26*