Lezione 3 - 05/03/2025

| Algebra Relazionale

> [!TIP] Che cos'è l'algebra relazionale ? L'algebra relazionale è un insieme di operazioni matematiche che definiscono le manipolazioni di dati in un database relazionale. Ogni operazione agisce su una o più relazioni (tabelle) e restituisce un'altra relazione come risultato. L'algebra relazionale è la base teorica dei linguaggi di query, come SQL, ed è fondamentale per la gestione delle basi di dati relazionali.

Linguaggi per le basi di dati:

Esistono due tipi di categorie di operazioni:

1. Operazioni sullo **schema**
    1. ***Data Definition Language*** (DDL);
2. Operazioni sui **dati**
    1. ***Data Manipulation Language*** (DML)
        1. Interrogazione (query)
        2. Aggironamento (update)

Linguaggi di interrogazione per le basi di dati

1. > [!TIP] Cosa si intende per interrogazione? L'interrogazione è un'***operazione di lettura*** sulla base di dati che può richiedere l'accesso a **più di una tabella**.

2. > [!TIP] Cosa è necessario fare per specificare il significato di una interrogazione? Due formalismi
    >
    > 1. ***Modo dichiarativo***: si specificano le proprietà del risultato ("che cosa");
    > 2. ***Modo procedurale***: si specificano le modalità di generazione del risultato ("come").

Si definisce il **comportamento** delle interrogazioni in **modo preocedurale** utilizzando le espressioni dell'algebra relazionale.

Si definisce qual è il risultato di un'interrogazione in **modo dichiarativo** utilizzando le espressioni del calcolo relazionale

- Il calcolo relazionale è l'effettiva semantica del linguaggio
    - Le interrogazioni sono esprese ad alto livello
        - ***NESSUN concetto di costo***
- Con l'algebra relazionale si può invece **introdurre il concetto di costo**.

1. **Algebra = *dati + operatori***
2. **Algebra relazionale**:
    - *Dati*: relazioni
    - *Operazioni*:

    - su relazioni,
    - che producono relazioni,
    - e che possono essere scomposti

Operatori dell'Algebra Relazionale:

Esistono due tipologie di opeartori:

1. ***Operatori su insiemi***:
    1. Unione, Intersezione, Differenza
2. ***Operatori su relazioni***:
    1. Ridenominazione, Selezione, Proiezione, Join

> [!NOTE] (*Riguardo al join*) --> naturale, prodotto cartesiano, theta.

Notazione:

- R, R_1,R_2,... indicano nomi di relazioni
- A,B,C,A_1,A_2,... indicanon nomi di attributo
- X,Y,X_1,X_2... indicano insiemi di attributi
- XY è un'abbreviazione di X U Y
- Una relazione con n-uple t_1, t_2,...,t_n è indicata con l'insieme { t_1, t_2,...,t_n }
- $t_i[A_j]$ indica il valore dell'attributo A_j nella n-upla t_i
- ☑ indica l'n-upla ottenuta da t considerando solo gli attributi in X

Esempio:

ID	Nome	Età	Corso
1	Maria	22	Ingegneria
2	Luigi	23	Matematica
3	Sara	21	Informatica

- t_2[Nome] = "Luigi"
- t_1[Corso] = "Ingegneria"
- t_3 ☑ = {21} (se X = {Età})
- t_2 ☑ = {Luigi, Matematica} (se X = {Nome, Corso})

Opeartori su insiemi (con esempi):

- ***Unione***
    - **L'operazione di unione** permette di combinare i dati di due relazioni (tabelle) che hanno lo stesso schema, ossia le stesse colonne (attributi). L'unione restituisce tutte le righe che appaiono in entrambe le relazioni, eliminando i duplicati. (**R1** ∪ **R2**)
    - **Esempio:**

Tabella Studenti_Corso_A

ID	Nome
1	Maria
2	Luigi

Tabella Studenti_Corso_B

ID	Nome
2	Luigi
3	Sara

Tabella Studenti_Corso_A ∪ Studenti_Corso_B (Unione)

ID	Nome
1	Maria
2	Luigi
3	Sara

- ***Intersezione***
    - **L'operazione di intersezione** restituisce le righe che sono comuni a entrambe le relazioni. A differenza dell'unione, l'intersezione mantiene solo gli elementi che si trovano in entrambe le tabelle. (**R1 ∩ R2**)
    - **Esempio:**

      Tabella Studenti_Corso_A

ID	Nome
1	Maria
2	Luigi

      Tabella Studenti_Corso_B

ID	Nome
2	Luigi
3	Sara

      Tabella Studenti_Corso_A ∩ Studenti_Corso_B (Intersezione)

ID	Nome

ID	Nome
2	Luigi

- ***Differenza***
    - **L'operazione di differenza** restituisce le righe che sono presenti in una relazione ma non nell'altra. In altre parole, restituisce gli elementi che appartengono alla prima relazione ma non alla seconda. (**R1 - R2**)
    - **Esempio:**

      Tabella Studenti_Corso_A

ID	Nome
1	Maria
2	Luigi

      Tabella Studenti_Corso_B

ID	Nome
2	Luigi
3	Sara

      Tabella Studenti_Corso_A ∩ Studenti_Corso_B (Intersezione)

ID	Nome
1	Maria

- (***Unione Impossibile?***) (esempio con Paternità e Meternità (attributi Padre figlio e Madre figlio))
    - L'operazione di unione è possibile **SOLO se le due relazioni hanno lo stesso schema** (cioè lo stesso numero di colonne e gli stessi nomi di attributi).

      Quindi, se le due tabelle hanno attributi diversi, l'unione non è possibile.
    - **Esempio:**

      Tabella Paternità

Padre	Figlio
Marco	Luca
Giovanni	Sara

      Tabella Maternità

Madre	Figlio
Laura	Luca
Maria	Sara

- Se provi a fare un'unione tra queste due tabelle, non sarebbe possibile perché hanno lo stesso attributo (Figlio), ma l'altro attributo è diverso: Padre nella prima tabella e Madre nella seconda. Quindi, l'unione tra queste due tabelle non è valida senza una trasformazione, come rinominare gli attributi o aggiungere un attributo che rappresenti la relazione in modo uniforme.

Per risolvere questa unione entra in gioco la ***Ridenominazione***:

- Operatore con un solo opearando ("monadico");
- Modifica lo schema dell' operando, lasciandone inalterata l'istanza
- Data una relazione R, in genereale, questo operatore si scrive come:

$$ \rho_{B_1, B_2, \dots, B_n \leftarrow A_1, A_2, \dots, A_m}(R) $$

da leggersi:

1. L'attributo A_1, viene sostituido dall'attributo B_1
2. L'attributo A_2, viene sostituido dall'attributo B_2
3. e così via...

Esempio

riprendendo l'esempio delle due tabelle maternità & paternità...

Tabella Paternità (modificata)

Genitore	Figlio
Marco	Luca
Giovanni	Sara

Tabella Maternità (modificata)

Genitore	Figlio
Laura	Luca
Maria	Sara

Tabella Unione (Paternità ∪ Maternità)

Genitore	Figlio
Marco	Luca
Giovanni	Sara

Genitore	Figlio
Laura	Luca
Maria	Sara

Operatori su UN SOLO insieme:

- ***Operatore di Selezione***:
    - L'*operatore di selezione* è un operatore monadico che permette di estrarre un sottoinsieme di tuple che soddisfano una determinata condizione (predicato).

      L'operatore di selezione viene scritto come:

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

        - σ è *l'operatore di selezione*.
        - **condizione** è un *predicato che definisce le tuple da selezionare (ad esempio, "Età > 21")*.
        - R è la *relazione su cui l'operatore di selezione viene applicato*.
    - Esempio:

      Immagina di avere una relazione Studenti con gli attributi ID, Nome ed Età:

ID	Nome	Età
1	Maria	22
2	Luigi	23
3	Sara	21

      L'operatore di selezione per ottenere tutti gli studenti con Età > 21 sarebbe:

      $$ \sigma_{\text{Età > 21}}(Studenti) $$

      Il risultato sarebbe:

ID	Nome	Età
1	Maria	22
2	Luigi	23

      L'operatore di selezione estrae solo le tuple che soddisfano la condizione specificata.

- ***Operatore di Proiezione***
    - L'*operatore di proiezione* è un operatore monadico che seleziona un sottoinsieme di attributi (colonne) da una relazione, eliminando gli altri attributi. La proiezione restituisce una nuova relazione che contiene solo gli attributi specificati.

      L'operatore di proiezione viene scritto come:

$$ \pi_{A_1, A_2, \dots, A_n}(R) $$

- π è l'operatore di proiezione.
- A_1, A_2, ..., A_n sono gli attributi da selezionare dalla relazione.
- R è la relazione da cui si estraggono gli attributi.

- Esempio:

  Immagina di avere una relazione Studenti con gli attributi ID, Nome, ed Età:

ID	Nome	Età
1	Maria	22
2	Luigi	23
3	Sara	21

  L'operatore di proiezione per selezionare solo gli attributi Nome e Età sarebbe:

  $$ \pi_{Nome,Età}(Studenti) $$

  Il risultato sarebbe:

Nome	Età
Maria	22
Luigi	23
Sara	21

> [!TIP] Cardinalità delle Proiezioni Una proiezione può contenere al più tante n-uple quante ne ha l'operando e contenerne di meno Se X è una superchiave di R allora $\pi_X(R)$ contiene esattamente tante tuple quante ne ha R.

> [!IMPORTANT] RICORDA! I valori degli altributi di quella tupla la identificano univocamente!

Proiezione e Selezione:

- Selezione $\boldsymbol{\sigma}$: decomposizione **orizzontale**
- Proiezione π: decompozione **verticale**

Esempio:

> [!TIP]Calcola matricola e cognome degli iimpiegati che guadagnano più di 50000:

Matricola	Cognome	Filiale	Stipendio
1001	Rossi	Milano	60000
1002	Bianchi	Roma	45000

> | Matricola | Cognome | Filiale | Stipendio |
> |---|---|---|---|
> | 1003 | Verdi | Milano | 55000 |
> | 1004 | Neri | Torino | 70000 |

> [!WARNING] Eseguiamo prima una selezione per la condizione "stipendio > 50000" e poi in seguito una proiezione per estrarre solo le colonne "Matricola" e "Cognome".

Passo 1. Selezione

La selezione per gli impiegati che guadagnano più di 50.000 è espressa come:

$$ \sigma_{\text{Stipendio} > 50000}(\text{Impiegati}) $$

Possiamo scrivere sotto l'espressione:

$$

$$

e rappresentarla con la seguente tabella:

Matricola	Cognome	Filiale	Stipendio
1001	Rossi	Milano	60000
1003	Verdi	Milano	55000
1004	Neri	Torino	70000

Passo 2: Proiezione

Dopo aver effettuato la selezione, applichiamo la proiezione per estrarre solo le colonne "Matricola" e "Cognome". L'operazione di proiezione è espressa come:

$$ \pi_{\text{Matricola}, \text{Cognome}}(\sigma_{\text{Stipendio} > 50000}(\text{Impiegati})) $$

Il risultato della proiezione sarà:

Matricola	Cognome
1001	Rossi
1003	Verdi
1004	Neri

> [!NOTE] Note - Possiamo dunque dire che...
>
> - Combinando selezione e proiezione possiao estrarre informazioni da **una sola** relazione
> - **NON** possiamo **combinare informazioni** presenti in **relazioni diverse**
> - **NON** possiamo combinare informazioni presenti in **n-uple diverse della stessa relazione**

- **Esempio:**
    - Prove scritte in un concorso pubblico:
        - I compiti sono anonimi e a ognuno è associata una busta chiusa con il nome del candidato
        - Ogni compito e la relativa busta è contrassegnato con uno stesso numero

---

| Join Naturale

> [!TIP] Definizione Il **join naturale** è un'operazione dell'algebra relazionale utilizzata nei database relazionali per combinare due tabelle in base agli attributi con lo stesso nome e dominio. È un tipo di **join equi-join** (*join basato sull'uguaglianza*), ma con la differenza che non richiede di specificare esplicitamente la condizione di join: il database riconosce automaticamente gli attributi comuni tra le due tabelle e li usa per eseguire l'unione.

> [!WARNING] Formalmente... Date due relazioni $R_1(X_1)$ e $R_2(X_2)$, in generale questo operatore si scrive come:
>
> $$ R_1⋈R_2 $$ Il risultato è una relazione $R(X_1 \cup X_2)$ definita come: $$ R(X_1 \cup X_2) = R_1(X_1)⋈R_2(X_2) $$

- **Esempio 1:**

  Relazione 1:

Impiegato	**Reparto**
Rossi	A
Neri	B
Bianchi	B

  Relazione 2:

Reparto	**Capo**
A	Mori
B	Bruni

  Facendo $R_1⋈R_2$:

Impiegato	**Reparto**	**Capo**
Rossi	A	Mori
Neri	B	Bruni
Bianchi	B	Bruni

  > [!IMPORTANT] Ogni n-upla contribuisce al risultato: join completo

- **Esempio 2:**

Impiegato Reparto

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto Capo

Reparto	Capo
B	Mori
C	Bruni

Impiegato Reparto Capo

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

- **Esempio 3:**

Impiegato Reparto

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto Capo

Reparto	Capo
D	Mori
C	Bruni

Impiegato Reparto Capo

Impiegato	Reparto	Capo

(Nessun dato disponibile)

Regole per il JOIN

Il numero di tuple nel risultato di un **JOIN** tra due relazioni (R1) e (R2) dipende dalla presenza di chiavi e vincoli di integrità referenziale.

1. Caso Generale

Il numero di tuple nel join (R1 \bowtie R2) è compreso tra:
[0 \leq |R1 \bowtie R2| \leq |R1| \times |R2|]

- **Caso minimo (0 tuple)**: se non ci sono corrispondenze tra le due tabelle, il risultato è vuoto.
- **Caso massimo ((|R1| \times |R2|))**: se ogni tupla di (R1) si abbina con tutte le tuple di (R2), otteniamo il prodotto cartesiano.

---

2. Caso in cui l'attributo di Join è una chiave in (R2)

Se il join coinvolge un attributo che è **chiave primaria in (R2)**, allora il numero di tuple sarà:
[0 \leq |R1 \bowtie R2| \leq |R1|]

- Questo accade perché ogni tupla di (R1) può al massimo avere **una corrispondenza unica** in (R2) (poiché (B) è chiave in (R2)).
- Se una tupla di (R1) non trova corrispondenza in (R2), il risultato avrà meno di (|R1|) tuple.

Esempio

Immaginiamo due tabelle:

Tabella `Studenti` (R1)

ID_Studente	Nome	Cognome
1	Luca	Rossi
2	Marco	Bianchi
3	Anna	Verdi

Tabella `Esami` (R2)

ID_Studente	Corso	Voto
1	Matematica	28
2	Fisica	25

Eseguendo il join, il risultato sarà:

ID_Studente	Nome	Cognome	Corso	Voto

ID_Studente	Nome	Cognome	Corso	Voto
1	Luca	Rossi	Matematica	28
2	Marco	Bianchi	Fisica	25

Qui (|R1 \bowtie R2| = 2), perché la tupla con `ID_Studente = 3` non ha un corrispondente in `Esami`.

---

3. Caso in cui l'attributo di Join è una chiave in (R1) e c'è un vincolo di integrità referenziale

Se (B) è **chiave primaria in (R1)** e c'è un vincolo di integrità referenziale che garantisce che ogni valore di (B) in (R2) esista in (R1), allora: [|R1 \bowtie R2| = |R2|]

- Ogni tupla in (R2) ha esattamente una corrispondenza in (R1), quindi il numero di tuple nel join è esattamente uguale a (|R2|).

Esempio

Immaginiamo due tabelle:

Tabella `Professori` (R1, con `ID_Prof` come chiave primaria)

ID_Prof	Nome	Cognome
10	Mario	Bianchi
20	Laura	Verdi
30	Paolo	Neri

Tabella `Corsi` (R2, con `ID_Prof` che fa riferimento a `Professori.ID_Prof`)

ID_Corso	Nome_Corso	ID_Prof
101	Matematica	10
102	Fisica	20
103	Informatica	30

Se facciamo il join, il risultato è:

ID_Prof	Nome	Cognome	ID_Corso	Nome_Corso
10	Mario	Bianchi	101	Matematica
20	Laura	Verdi	102	Fisica
30	Paolo	Neri	103	Informatica

Qui il numero di tuple nel join è **esattamente** (|R2| = 3), perché ogni riga di `Corsi` ha un valore di `ID_Prof` che esiste sicuramente in `Professori`.

> [!TIP] Riassumendo...

Caso	**Formula per il numero di tuple ($\|R1 \bowtie R2\|$)**
Caso Generale (nessuna chiave specifica)	$0 \leq \|R1 \bowtie R2\| \leq \|R1\| \times \|R2\|$
Se l'attributo di join è una chiave in (R2)	$0 \leq \|R1 \bowtie R2\| \leq \|R1\|$
Se l'attributo di join è chiave in (R1) e c'è un vincolo di integrità >referenziale	$\|R1 \bowtie R2\| = \|R2\|$

> [!WARNING] Attenzione... Alcune n-uple non contribuiscono al risultato: vengono “tagliate fuori”

| Join Esterno

> [!TIP]Definizione Il join esterno è una variante del join che include tutte le tuple di una o entrambe le relazioni coinvolte, anche se non trovano corrispondenza nell’altra tabella. Quando non c’è un match, i valori nelle colonne della tabella senza corrispondenza vengono riempiti con NULL.

Esistono 3 tipi di Join Esterno:

1. ***Sinistro***: mantiene tutte le -uple del primo operando, estendendole con valori nulli se necessario
2. ***Destro***: mantiene tutte le -uple del secondo operando, estendendole con valori nulli se necessario
3. ***Completo***: mantiene tutte le -uple di entrambi gli operandi, estendendole con valori nulli se necessario

- **Esempio Join Sinistro**

  Clienti R1:

ID_Cliente	**Nome**
1	Anna
2	Marco
3	Luca

  Ordini R2:

ID_Cliente	**Prodotto**
1	Laptop
2	Smartphone

  Risultato del **Join Sinistro**:

ID_Cliente	**Nome**	**Prodotto**
1	Anna	Laptop
2	Marco	Smartphone

ID_Cliente	Nome	Prodotto
3	Luca	NULL

- **Esempio Join Destro**

  Clienti R1:

ID_Cliente	Nome
1	Anna
2	Marco

  Ordini R2:

ID_Cliente	Prodotto
1	Laptop
2	Smartphone
3	Tablet

  Risultato del **Join Destro**:

ID_Cliente	Nome	Prodotto
1	Anna	Laptop
2	Marco	Smartphone
NULL	NULL	Tablet

- **Esempio Join Completo**

  Clienti R1:

ID_Cliente	Nome
1	Anna
2	Marco
3	Luca

  Ordini R2:

ID_Cliente	Prodotto
1	Laptop
2	Smartphone
4	Tablet

Risultato del **Join Completo**:

ID_Cliente	Nome	Prodotto
1	Anna	Laptop
2	Marco	Smartphone
3	Luca	NULL
4	NULL	Tablet

Join e Proiezioni

Dobbiamo sapere 3 concetti:

1. Date due relazioni R e 1(X_1) R2(X_2): $$ πX_1 (R_1 ⋈ R_2) ⊆ R_1 $$
2. Date due relazioni R_1(X_1) e R_2(X_2) $$ πX_1 (R_1 ⋈ R_2) ⊆ R_1 $$
3. Data una relazione R(X) con X = X_1 ∪ X_2 $$ (πX1(R) ⋈ πX2(R)) ⊇ R $$

| Prodotto Cartesiano

Il prodotto cartesiano (o join cartesiano) di due relazioni, chiamate R1 e R2, è una nuova relazione che contiene tutte le possibili combinazioni di tuple di R1 e R2. Ogni coppia di tuple (una proveniente da R1 e l'altra da R2) viene unita per formare una nuova tupla, che contiene tutte le colonne di entrambe le relazioni.

Formula

Se:

- R1 ha m colonne e n1 tuple
- R2 ha p colonne e n2 tuple

Il risultato del prodotto cartesiano R1 × R2 avrà:

- m + p colonne
- n1 × n2 tuple

(1) Esempio - immaginiamo 2 tabelle...

Clienti R1:

ID_Cliente	Nome
1	Anna
2	Marco
3	Luca

> [!NOTE] R1 ha m = 2 colonne (ID_Cliente e Nome) e n1 = 3 tuple.

Ordini R2:

ID_Ordine	Prodotto
A1	Laptop
A2	Smartphone

> [!NOTE] R2 ha p = 2 colonne (ID_Ordine e Prodotto) e n2 = 2 tuple.

> [!TIP] Utilizziamo la formula del Prodotto Cartesiano: Numero di colonne nel risultato: **m + p = 2 + 2 = 4**
>
> Numero di tuple nel risultato: **n1 × n2 = 3 × 2 = 6**
>
> Quindi, il prodotto cartesiano avrà **4 colonne** e **6 tuple**.

Riscrivendo il prodotto cartesiano (R_1 X R_2):

ID_Cliente	Nome	ID_Ordine	Prodotto
1	Anna	A1	Laptop
1	Anna	A2	Smartphone
2	Marco	A1	Laptop
2	Marco	A2	Smartphone
3	Luca	A1	Laptop
3	Luca	A2	Smartphone

| Theta-Join

> [!TIP] Definizione Il theta-join è un'operazione di join tra due tabelle (relazioni) che combina le tuple delle tabelle in base a una condizione arbitraria che coinvolge uno o più attributi. A differenza di un equi-join (dove la condizione è una semplice uguaglianza), il theta-join permette di utilizzare qualsiasi tipo di relazione (come =, <, >, <=, >=, ≠, etc.).

> [!WARNING] Formula logica: $$ R1 ⋈θ R2 $$ Dove θ è il predicato (condizione) che può essere qualsiasi operatore logico tra i valori delle colonne nelle due tabelle.

> [!IMPORTANT] In poche parole... Il theta-join è una forma generale di join che consente di specificare condizioni personalizzate tra le colonne di due tabelle. È utile quando vogliamo combinare tabelle basandoci su relazioni diverse dall'uguaglianza, come le disuguaglianze o altre condizioni logiche.

- **(1) Esempio:**

  Clienti R1:

ID_Cliente	Nome	Età

ID_Cliente	Nome	Età
1	Anna	25
2	Marco	30
3	Luca	35

Ordini R2:

ID_Ordine	ID_Cliente	Prodotto	Prezzo
A1	1	Laptop	1000
A2	2	Smartphone	500
A3	1	Tablet	400
A4	3	Laptop	1200

La domanda che ci facciamo è:

> [!WARNING]Supponiamo di voler fare un theta-join sulle tabelle Clienti e Ordini in cui l'ID_Cliente sia uguale a 1 e il Prezzo degli ordini sia maggiore di 500.
>
> Sottoforma logica: $$ R_1 ⋈ (ID_{\text Cliente} = ID {\text Cliente} \space AND \space Prezzo > 500) R_2 $$

Il risultato del theta-join sarà il seguente, poiché solo le tuple che soddisfano entrambe le condizioni (ID_Cliente uguale a 1 e Prezzo maggiore di 500) vengono combinate:

ID_Cliente	Nome	Età	ID_Ordine	ID_Cliente	Prodotto	Prezzo
1	Anna	25	A1	1	Laptop	1000
1	Anna	25	A3	1	Tablet	400

- Numero di colonne: Le colonne di R_1 (ID_Cliente, Nome, Età) e R_2 (ID_Ordine, ID_Cliente, Prodotto, Prezzo) sono combinate, risultando in 7 colonne.
- Numero di tuple: Solo le tuple che soddisfano la condizione (ID_Cliente = 1 e Prezzo > 500) sono incluse nel risultato. Pertanto, otteniamo solo 2 tuple (una per Laptop e una per Tablet, anche se quest'ultima sarebbe stata esclusa in un caso pratico).

> [!IMPORTANT] Lista esempi delle theta-join

Operatori Relazionali Fondamentali

Sorprendentemente, **cinque operatori relazionali** sono sufficienti per qualsiasi interrogazione:

1. **Selezione** (σC(R))
2. **Proiezione** (πX(R))
3. **Prodotto cartesiano** (R1 × R2)
4. **Unione** (R1 ∪ R2)

5. **Differenza** (R1 − R2)

Tutti gli altri sono **operatori derivati/di convenienza**.

Opeatore Divisione:

> [!TIP] Definizione La divisione tra due relazioni R1 e R2 (dove R2 è un sottoinsieme di R1) restituisce tutte le tuple di R1 che sono associate a tutte le tuple di R2.

> [!WARNING] Formalmente... Dati due insiemi di attributi disgiunti X_1 e X_2, una relazione r su $X_1 \cup X_2$ e una relazione r_2 su X_2, la divisione è una relazione su X_1 che contiene le tuple ottenute come "proiezione" di tuple di r che si combinano con tutte le tuple di r_2 per formare tuple di X_1.
>
> In altre parole, $$ r \div r_2 = \{ t_1 \text{ su } X_1 \mid \forall t_2 \in r_2, \exists t \in r \text{ con } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \} $$
>
> Questa definizione implica che una tupla di r_1 viene inclusa nel risultato della divisione solo se essa è associata a tutte le tuple di r_2 attraverso l'attributo comune X_2.

La divisione è solitamente utilizzata quando:

- Vogliamo trovare tutte le entità in R1 che sono collegate a tutte le entità di R2.
- La relazione R1 deve contenere una parte che può essere associata con tutte le tuple di R2.

Esempio:

Immagina due tabelle:

Studenti (R1):

ID_Studente	Corso
1	Matematica
1	Informatica
2	Matematica
3	Informatica
4	Matematica

Corsi (R2):

Corso
Matematica
Informatica

Domanda: Troviamo gli **studenti** che sono iscritti **a tutti i corsi** (quindi sia a **Matematica** che a **Informatica**)?

Procedura:

1. **Relazione R1** contiene gli studenti e i corsi a cui sono iscritti.
2. **Relazione R2** contiene l'elenco dei corsi da considerare.

Per trovare gli **studenti** che sono iscritti a **tutti** i corsi presenti in **R2**, dobbiamo fare la **divisione** tra le due tabelle.

Divisione tra R1 e R2:

1. Prendiamo le **tuple** di **R1** che hanno una **proiezione** sugli **attributi comuni** (**Corso**).
2. Una **tupla** di **R1** viene inclusa nel risultato della divisione se è associata a **tutte** le **tuple** di **R2**.

Risultato:

ID_Studente
1

Spiegazione:

- **ID_Studente = 1** è iscritto sia a **Matematica** che a **Informatica**, quindi è incluso nel risultato della divisione.
- **ID_Studente = 2** è iscritto solo a **Matematica**, quindi non appare nel risultato.
- **ID_Studente = 3** è iscritto solo a **Informatica**, quindi non appare nel risultato.
- **ID_Studente = 4** è iscritto solo a **Matematica**, quindi non appare nel risultato.

Conclusione:

La divisione restituisce solo gli **studenti** che sono iscritti **a tutti** i corsi presenti nella tabella **Corsi (R2)**. In questo caso, solo lo **Studente 1** soddisfa il criterio di essere iscritto a **entrambi** i corsi, quindi è incluso nel risultato.

| Chiusura Transitiva

Non esiste la possibilità di esprimere l'interrogazione che calcoli la chiusura transitiva di una relazione qualunque • In algebra relazionale l’operazione si simulerebbe con un numero di join illimitato

---

| Equivalenza di Espressioni

- Due **espressioni** sono **equivalenti** se producono lo stesso risultato qualunque sia l’istanza attuale della base di dati
- L’equivalenza è importante nella pratica perché i DBMS cercano di eseguire **espressioni equivalenti** a quelle date, ma **meno “costose”**
- Il costo dell’esecuzione di un’interrogazione viene valutato in termini delle dimensioni dei risultati intermedi della valutazione dell’espressione dell’algebra relazionale

Equivalenze I

1. ***Atomizzazione delle selezioni***: una congiunzione di selezioni può essere sostituita da una sequenza di selezioni atomiche (con F_1 e F_2 espressioni Booleane ed E espressione qualsiasi). $$ \sigma_{\text{F1} \wedge \text{F2}}(E) \equiv \sigma_{\text{F1}}(\sigma_{\text{F2}}(E)) $$

2. ***Idempotenza delle proiezioni***: una proiezione può essere trasformata in una sequenza di proiezioni che eliminano i vari attributi in varie fasi (con E espressione definita su un insieme di attributi che contiene X e Y). $$ \pi_{\text{X}}(E) \equiv \pi_{\text{X}}(\pi_{\text{XY}}(E)) $$

Equivalenze II

1. ***Push selections down***: se la condizione F coinvolge solo attributi della espressione E_2: $$ \sigma F(E1 \bowtie E2) \equiv E1 \bowtie \sigma F(E2) $$

2. ***Push projections down***: dove X_1 sono gli attributi di E_1, X_2 sono gli attributi di E_2, Y2 ⊆ X2 e gli attributi X_2 - X_1 non sono coinvolti nel join (cioè $X_1 \cap X_2 \subseteq Y_2$): $$ \pi X1Y2 (E1 \bowtie E2) \equiv E1 \bowtie \pi Y2 (E2) $$

Ottimizzazione delle interrogazioni

- ***Query processor*** (od ottimizzatore): un modulo del DBMS
- Più importante nei sistemi attuali che in quelli "vecchi" (gerarchici e reticolari):
- Le interrogazioni sono espresse ad alto livello (ricordare il concetto di indipendenza dei dati):
    - Insiemi di n-uple
    - Poca proceduralità
- L'ottimizzatore sceglie la strategia realizzativa (di solito fra diverse alternative), a partire dall'istruzione SQL.

Profili delle Relazioni

> [!TIP]Definizione I profili delle relazioni sono un insieme di informazioni quantitative relative a una base di dati, che descrivono le caratteristiche e le proprietà delle relazioni (tabelle) presenti nel database.

Queste informazioni sono fondamentali per ottimizzare le operazioni di accesso ai dati e di elaborazione delle query. Vengono memorizzate nel catalogo del database, che è una struttura contenente metadati sulle tabelle, gli attributi e altre informazioni necessarie per l'ottimizzazione delle query.

Le principali informazioni quantitative nei profili delle relazioni includono:

- ***Cardinalità di ciascuna relazione***: Rappresenta il numero di tuple (righe) presenti nella relazione. Questo valore è importante per stimare la quantità di dati da processare in un'operazione, come un join o una selezione.
- ***Dimensioni delle tuple***: Indica il numero di attributi (colonne) che una tupla (riga) contiene. Ogni relazione ha una dimensione fissa per ogni tupla, che dipende dal numero di attributi e dal tipo di dati di ciascun attributo.

- ***Dimensioni dei valori***: Riferito alla quantità di spazio occupata da ciascun valore presente in una tupla. Questo include la lunghezza di stringhe, numeri, date, ecc.
- ***Numero di valori distinti degli attributi***: Fornisce la quantità di valori unici che un dato attributo può assumere in una relazione. Questa statistica è utile per capire la distribuzione dei dati e per pianificare come ottimizzare gli accessi ai dati.
- ***Valore minimo e massimo di ciascun attributo***: Indica i valori estremi (minimo e massimo) di un dato attributo. Queste informazioni sono utilizzate per determinare le condizioni di selezione più efficienti.

Memorizzazione e Utilizzo:

Questi profili vengono memorizzati nel catalogo del database, che è una sorta di "libro mastro" contenente informazioni sulle strutture e i dati nel database. La gestione e l'aggiornamento di questi profili sono cruciali per l'ottimizzazione delle query.

Utilizzo nell'ottimizzazione delle query:

Questi profili sono utilizzati dal **query optimizer** del database per stimare la dimensione dei risultati intermedi e pianificare l'esecuzione più efficiente di una query. Con informazioni precise sui dati, l'ottimizzatore può scegliere strategie di accesso ai dati più rapide, come l'ordine di esecuzione di join, l'uso di indici e la selezione dei migliori piani di esecuzione.

| Grafo

Un ***grafo G = (V,E)*** consiste in:

1. un insieme V di vertici (o nodi)
2. un insieme E di coppie di vertici, detti archi
    1. ogni arco connette due vertici Esistono 2 tipi di grafi:
3. ***Grafo orientato*** (o ***diretto***): ogni arco è orientato e rappresenta relazioni orientate tra coppie di oggetti.
4. ***Grafo non orientato*** (o ***non diretto***): gli archi non hanno un orientazione e rappreentano relazioni simmetriche tra coppie di oggetti.