

Ad ogni colonna si può anche associare un valore indice tramite il simbolo del dollaro:

es. $\Pi_{\text{codice}, \text{nome}} [\delta_{\text{merca}=\$1} (\text{Merca})]$

$\Pi_{\$1, \$2} [\delta_{\$3=\$2} (\text{Merca})]$

UGUALE \wedge

Nell'algebra relazionale è anche possibile utilizzare gli operatori insiemistici quali \cup, \cap, X e \setminus .

Ad esempio per estrarre i nomi sia dei fornitori che delle merci posso scrivere:

$\Pi_{\$2} (\text{Fornitore}) \cap \Pi_{\$2} (\text{Merca})$

↓ RISULTATO

Nome
A
B
C

∩

Nome
x
y
z

↓ RISULTATO

Nome

→ insieme vuoto

GAIA BERTOLINO

Le tabelle devono avere la stessa struttura e a pari di struttura lo stesso nome per le colonne in modo da essere compatibili.

Se hanno stessa struttura ma nome delle colonne diverso vanno riannodate prima di elaborarle

Gli operatori insiemistici sono:

- $\cup \rightarrow A \cup B$ vuol dire elementi di A unisci a quelli di B
- $\cap \rightarrow A \cap B$ vuol dire elementi sia di A che di B
- $\setminus \rightarrow A \setminus B$ vuol dire elementi di A meno quelli di B
- $X \rightarrow A \times B$ vuol dire concatenazione in una tabella di A e B,

Ogni tupla sarà una concatenazione delle due tuple

Il numero delle tuple è il prodotto delle due cardinalità.

Ridenominazione

La rideconominazione viene fatta attraverso l'operatore ρ

$$\rho_{\text{nu}} \rightarrow \text{nuova}$$

es. $\rho_{\$1} \rightarrow \text{nuovaName}$

L'operazione di prodotto cartesiano è detta anche di **PROLUNGAMENTO** in quanto provoca il prolungamento della Tabella di partenza.

Tale operazione si può anche indicare con l'operatore di **JOIN**:

es. Fornitore $\bowtie_{\$3=\$3}$ Fornitura $\iff \delta_{\$3=\$3} (\text{Fornitore} \times \text{Fornitura})$
 \wedge
 $\$1 \neq \4

Esempio

Fornitore		
	$\$2$	$\$3$
F_1	A	CS
F_2	B	RC
F_3	C	CS

Fornitura		
	$\$1$	$\$3$
F_1	π_1	10€
F_2	π_2	5€
F_3	π_2	6€

Se si vuole selezionare i codici delle merci fornite da A le operazioni da seguire sono:

$$\pi_{\$5} [\delta_{\$1=\$4} (\text{Fornitore} \times \text{Fornitura})]$$

\wedge
 $\$2 = 'A'$

OPPURE

$$\pi_{\$5} [\delta_{\$2 = 'A'} (\text{Fornitore}) \bowtie_{\$1=\$1} \text{Fornitura}]$$

OPPURE

$$\Pi_{\$5} [\delta_{\$1=\$4} [\delta_{\$2=1_A} (\text{Fornitore}) \times \text{Fornitore}]$$

OPPURE

$$\Pi_{\$5} [\delta_{\$2=1_A} (\text{Fornitore} \bowtie_{\$1=\$2} \text{Fornitore})]$$

es. I nomi delle merci fornite dai fornitori che sfanno a cosenza

$$\Pi_{\$8} \left\{ [\delta_{\$3=cs} (\text{Fornitore}) \bowtie_{\$1=\$2} \text{Fornitore}] \bowtie_{\$1=\$1} \text{Merce} \right\}$$

es. copie nome_fornitore e nome_merce tali che il fornitore fornisce la merce

$$\Pi_{\$5, \$8} [(\text{Fornitore} \bowtie_{\$1=\$3} \text{Fornitore}) \bowtie_{\$2=\$4} \text{Merce}]$$

Esercizi svolti

Fornitore		
F ₁	A	RC
F ₂	B	CS
F ₃	C	RC

Fornitore		
F ₁	H ₁	5
F ₂	H ₂	6
F ₂	H ₂	10

es. Codici dei fornitori che forniscono almeno una merce

$$\Pi_{\$2} [\text{Fornitore} \bowtie_{\$1=\$1} \Pi_{\$1} (\text{Fornitura})]$$

OPPURE

$$\Pi_{\$2} (\text{Fornitore} \bowtie_{\$1=\$2} \text{Fornitura})$$

Ad una query è possibile assegnare un nome:
miaQuery = $\Pi_{\$2} (\text{Fornitori})$

es. Nomi dei fornitori che forniscono almeno due merci

L'almeno due si realizza accoppiando la tupla con lo stesso fornitore ma diversa merce

$$\Pi_{\$1} [(\text{Fornitore} \bowtie \text{Fornitura}) \bowtie \text{Fornitore}]$$

$\begin{matrix} \$1 = \$1 \\ \wedge \\ \$2 \neq \$2 \end{matrix}$

es. Copie di fornitori della stessa città

$$\text{Fornitore} \bowtie \text{Fornitore}$$

$\begin{matrix} \$3 = \$3 \\ \wedge \\ \$1 < \$1 \end{matrix}$

es. Codici dei fornitori che vendono almeno tre merci

$$\text{Almeno Tre} = \Pi_{\$1} [(\text{Fornitura} \bowtie \text{Fornitura}) \bowtie \text{Fornitura}]$$

$\begin{matrix} \$1 = \$1 \\ \wedge \\ \$2 \neq \$2 \\ \wedge \\ \$3 = \$3 \\ \wedge \\ \$2 \neq \$2 \\ \wedge \\ \$5 \neq \$2 \end{matrix}$

Il numero di "almeno" esprime quante join bisogna fare

es. Nomi dei fornitori che forniscono esattamente una merce

Esso è dato dalla differenza fra le due query

Almeno 1 e Almeno 2 :

$$\text{Almeno 1} = \Pi_{\$1} (\text{Fornitore})$$

$$\text{Almeno 2} = \Pi_{\$1} [(\text{Fornitore} \bowtie \text{Forniture})]$$

$\begin{matrix} \$1 = \$1 \\ \wedge \\ \$2 \neq \$2 \end{matrix}$

$$\text{Esattamente 1} = \text{Almeno 1} - \text{Almeno 2}$$

$$\text{Query} = \Pi_{\$3} (\text{Esattamente 1} \bowtie \text{Fornitore})$$

$\begin{matrix} \$1 = \$1 \\ \wedge \\ \$2 = \$2 \end{matrix}$

es. Fornitori che forniscono al più due merci

$$\text{Query} = \text{Fornitori} - \text{Almeno 3} = \Pi_{\$1} (\text{Fornitore} - \text{Almeno 3})$$

es. Fornitori che forniscono almeno due merci e al più 4

$$\text{Query} = \text{Almeno 2} - \text{Almeno 5}$$

es. Fornitori che vengono forniscono merci

$$\text{Query} = \text{Fornitori} - \text{Almeno 1}$$

OPPURE

$$\text{Query} = \Pi_{\$1} (\text{Fornitore}) - \Pi_{\$1} (\text{Fornitura})$$

es. Prezzo minimo al quale viene fornita M1

In questo caso non si può ragionare cercando il prezzo minimo ma vado alla ricerca di quei numeri che sono assolutamente minimi.

$$\text{Prezzi Non Minimi} = \Pi_{\$2=\text{M1}'} (\text{Fornitura}) \Delta\text{ Fornitura}$$



$$\text{Query} = \text{Tutti} - \text{Prezzi Non Minimi} = \Pi_{\$3 \atop \$2=\text{M1}} [6 (\text{Fornitura})] - \text{Prezzi Non Minimi}$$

es. Nomi dei fornitori che forniscono tutte le merci

Il termine "tutte" e "per ogni" non possono essere direttamente tradotti ma necessitano di essere tradotti con il "per ogni" e l'"esiste". Solitamente $= \forall$ e $\Delta = \exists$

Dunque la query diventa:

Nomi dei fornitori tali che esiste almeno una merce che essi vengono forniscono.

$$\text{Non Fornita} = \Pi_{\$1} (\text{Fornitore}) \times \Pi_{\$1} (\text{Merca}) - \Pi_{\$1, \$2} (\text{Fornitore})$$

$$\text{Query} = \Pi_{\$3} \left\{ \left[\Pi_{\$1} (\text{Fornitore}) - \Pi_{\$1} (\text{Non Fornita}) \right] \Delta \text{ Fornitore} \right\}_{\$1=\$2}$$

es. Marche che producono almeno due merci, ciascuna delle quali è fornita da almeno due fornitori

$$\text{ForniteDaDueFornitori} = \Pi_{\$2} (\text{Fornitura} \Delta\text{ Fornitura})$$

\downarrow lo chiamo Recl12P

$$\text{Query} = \Pi_{\$4} [(\text{Merco2F} \bowtie \text{Merco}) \bowtie (\text{Merco2F} \bowtie \text{Merco})]$$

$\begin{matrix} \$1=\$4 & \$4=\$4 & \$1=\$1 \\ \wedge & \wedge & \\ \$1 \neq \$2 & & \end{matrix}$

es. Merco prodotte a CS e fornite da esattamente un fornitore

$$\text{Almeno2F} = \Pi_{\$2} (\text{Fornitura} \bowtie \text{Fornitura})$$

$\begin{matrix} \$1 \neq \$1 \\ \wedge \\ \$2 = \$2 \end{matrix}$

$$\text{Esattamente1F} = \Pi_{\$2} (\text{Fornitura}) - \text{Almeno2F}$$

$$\text{Query} = \Pi_{\$2} \left\{ \delta_{\$2=CS} [(\text{Esattamente1F} \bowtie \text{Merco}) \bowtie \text{Merco}] \right\}$$

EQUIVALENTE A

$$\text{Query} = \Pi_{\$3} \left\{ [\delta_{\$2=CS} (\text{Merco}) \bowtie \text{Merco}] \bowtie \text{Esattamente1F} \right\}$$

es. Coppie $\langle \text{NMP}, P \rangle$ dove NMP è il nome di una merce prodotta a costanza e P è il prezzo massimo al quale la merce viene fornita a RC.

Per ogni merce corso il prezzo al quale viene fornita a RC

$$\text{FornituraRC} = \Pi_{\$1, \$2, \$3} [\text{Fornitura} \bowtie \delta_{\$3=RC} (\text{Fornitura})]$$

$$\text{CoppieNouMax} = \Pi_{\$2, \$3} (\text{FornituraRC} \bowtie \text{FornituraRC})$$

$\begin{matrix} \$3 < \$3 \end{matrix}$

$$\text{CoppieMax} = \Pi_{\$2, \$3} (\text{FornituraRC}) - \text{CoppieNouMax}$$

$$\text{Query} = \Pi_{\$4, \$2} [(\text{CoppieMax} \bowtie \text{Merco}) \bowtie \delta_{\$2=CS} (\text{Merco})]$$

$\begin{matrix} \$1=\$4 & \$5=\$1 \end{matrix}$

es. Merco fornite da tutti i fornitori

Così equivale a "merci fatte da uno o più fornitori che non le forniscono"

$$\text{MercoNefornite} = \Pi_{\$1} (\text{Merco}) \times \Pi_{\$1} (\text{Fornitura}) - \Pi_{\$2, \$1} (\text{Fornitura})$$

$$\text{Query} = \Pi_{\$1} (\text{Merce}) - \Pi_{\$1} (\text{Fornitore})$$

es. **Nomi dei fornitori che forniscono tutte le loro merci a più di 10€**

Ciò equivale a dire "nomi dei fornitori tali che non esiste alcuna merce da essi fornita a meno di 10€"

$$\text{Fornitore} = \Pi_{\$1} [G_{\$3 < 10} (\text{Fornitore})]$$

$$\text{Query} = \Pi_{\$3} \left\{ [\Pi_{\$1} (\text{Fornitore}) - \text{Fornitore}] \right\}_{\$1=\$2} \text{ Fornitore}$$

es. **Merce che non sono fornite a RC**

Ciò equivale a dire "codici merce tali che il fornitore non è di raggio"

$$\text{Fornitori RC} = \Pi_{\$5} [G_{\$3=RC} (\text{Fornitore})] \right\}_{\$1=\$2} \text{ Fornitore}$$

$$\text{Query} = \Pi_{\$3} \left\{ [\Pi_{\$1} (\text{Merce}) - \text{Fornitori RC}] \right\}_{\$1=\$2} \text{ Merce}$$

APPUNTI DI INGEGNERIA

es. **Copie F', F'' di fornitori che forniscono gli stessi codici di merce**

Ciò equivale a dire "Copie F' e F'' di fornitori tali che tutte le merci fornite da F' sono fornite da F'' e viceversa" a sua volta equivalente a "Copie F', F'' di fornitori tali che non esiste alcuna merce fornita da F' ma non da F'' e viceversa.

$$\text{Copie NO} = \Pi_{\$1, \$3} (\text{Fornitore}) \right\}_{\$2=\$3} \text{ Fornitore NO}$$

$$\text{Fornitore NO} = \Pi_{\$1} (\text{Fornitore}) \times \Pi_{\$1} (\text{Merce}) - \Pi_{\$1, \$2} (\text{Fornitore})$$

$$\text{Query} = [\Pi_{\$1} (\text{Fornitore}) \right\}_{\$1 < \$2} \Pi_{\$1} (\text{Fornitore}) - \text{Copie NO} - \Pi_{\$2, \$1} (\text{Copie NO})$$

Quando non viene segnalato il campo da restituire allora si restituisce la chiave della tabella.

es. Fornitori di CS che forniscono prodotti di almeno due marche, dove ciascuna di tali marche produce almeno due merci fornite a più di 10€ da qualche fornitore

es. Marche con sede a RC che producono solo merci fornite in città diverse da RC

es. Nome dei fornitori che forniscono merci di tutte le marche

es. Copie di fornitori di città diverse che forniscono gli stessi insiemi di marche (v.b. i ve fornitore fornisce una marca se fornisce almeno un prodotto della marca)

es. Fornitori che forniscono tutte le merci di una marca (fornitori tali che esiste una marca di cui forniscono tutte le merci)

APPUNTI DI INGEGNERIA INFORMATICA

GAIA BERTOLINO

Esercizi

mercoledì 26 gennaio 2022 23:45

- I) A) Copie Piva, nome dei fornitori
- B) Copie Piva, nome dei fornitori che sono stanziate a RC
- C) Codice delle merci che si chiamano X o che sono prodotte dalla marca Y
- D) Copie di nomi che rappresentano fornitori (pro. cart.)
- E) Copie di nomi di fornitori che stanno nella stessa città (pro. cart.)

Fornitore

Piva	Nome	Città
F ₁	A	CS
F ₂	B	RC
F ₃	C	CS

Rerci

Codice	Nome	Marca
R ₁	X	α
R ₂	Y	β

A) $\Pi_{\substack{\text{Piva}, \\ \text{nome}}} (\text{fornitori})$

B) $\Pi_{\$1, \$2} [\delta_{\$3 \neq RC} (\text{fornitori})]$

↳ altra soluzione

↳ $\Pi_{\$1, \$2} [\text{Fornitore} \setminus \delta_{\$3 = RC} (\text{fornitori})]$

$\Pi_{\substack{\text{Piva}, \\ \text{nome}}} (\text{fornitori}) \setminus \delta_{\text{Città} = RC} (\text{fornitori})$

C) $\Pi_{\$1} [\delta_{\$2 = X} (\text{Rerci})]$

↳ altra soluzione

↳ $\delta_{\text{Nome} = X} (\text{Rerci}) \cup \delta_{\text{Marca} = Y} (\text{Rerci})$

D) $\Pi_{\text{Nome}} (\text{fornitori}) \times \Pi_{\text{Nome}} (\text{fornitori})$

↳ altra soluzione

U) "uone (fornitori) x "uone (fornitori)

altra soluzione

$\Pi_{\$2, \$5} (\text{fornitori} \times \text{fornitori})$

E) $\Pi_{\$2, \$5} [\delta_{\$3=\$6} (\text{oratori} \times \text{oratori})]$

\wedge

$\$2 \neq \4

prendo solo le colonne $\$3$ (città fornitori) che sono uguali alle colonne $\$6$ (città fornitori 2)

elimino le coppie che hanno lo stesso fornitore ovvero quelle coppie che sono il prodotto di se stesse e rimango solo le coppie di elementi distinti



2)

1. nomi delle merci prodotte dalla marca alpha
2. Codici delle merci che sono fornite da almeno un fornitore a meno di 10euro
3. nomi delle merci che sono prodotte a milano (la marca che le produce si trova a mi)
4. nomi delle merci che sono fornite a più di 20euro da fornitori di Catanzaro
5. Coppie di città tali che la prima delle due è quella di un fornitore che fornisce almeno una merce prodotta presso la seconda città
6. Nomi dei fornitori che forniscono almeno una merce
7. Nomi dei fornitori che forniscono almeno due merci

Sono dati:

Fornitore (più, nome, città)

Merce (codice nome, marca)

Fornitura (fornitore, merce, marca)

Marca (nome, sede)

1) $\Pi_{\$2} [\delta_{\$3=\$2} (\text{merce})]$

2) $\pi_{\$2} [G_{\$3 < 10} (\text{Formulare})]$

$$3) \quad \Pi_{\$2} \left\{ G_{\$5='hi'} [G_{\$3=\$u} (\text{Herc} \times \text{Kara})] \right\}$$

OPPURE

$\pi_{\$2} [G_{\$3=\$6} \text{ (Rece x Reca)}]$

OPPURE

$\pi_{\$2} [\odot_{\$5='hi'} (\text{Merca} \bowtie \text{Merca})]$

OPPURE

$\prod_{\substack{S_2 \\ S_3 = S_2}} \left(\text{Hence } \bigwedge S_{t_2} = 'ni' \text{ (Hence)} \right)$

$$4) \quad \Pi_{\$2} \left\{ G_{\$6 > '20'} [(\text{Herc} \bowtie \text{Formulare}) \bowtie \text{Formulare}] \right\}$$

\uparrow
 $\$q = 'C2'$

$\$1 = \2 $\$u = \1

OPPURE

$$\pi_{\$g} \left\{ [G_{\$g_3 > 20} (\text{Fertilized}) \bowtie_{\$g_3 = \$g_2} G_{\$g_3 = \$g_2} (\text{Fertilized})] \bowtie_{\$g_2 = \$g_1} \text{Recept} \right\}$$

$$5) \quad \Pi \$3, \$11 \quad \left\{ \begin{array}{l} [(\text{Forisfora} \bowtie \text{Forisfora}) \bowtie \text{Hercé}] \bowtie \text{Morca} \\ \$1 = \$1 \qquad \qquad \$5 = \$1 \qquad \qquad \$9 = \$1 \end{array} \right.$$

3)

Formifore

$\$1$	$\$2$	$\$3$
F_1	A	RC
F_2	B	CS
F_3	C	RC

COD. NOME CITTÀ
FORN. FORN. FORN.

Formura

$\$1$	$\$2$	$\$3$
F_1	M_1	5
F_2	M_2	6
F_3	M_3	10

COD. merce presso
FORN.

- Codici forniscono che forniscono almeno una mappa

$$\text{Aluno 1} = \text{PI}_{\$2} (\text{Formula} \bowtie \text{Formula})$$

$\$1 = \1

- Nome dei fornitori / almeno 2 merci

$$\text{Almeno 2} = \Pi \$_2 \left(\begin{array}{c} \text{Fornitura} \bowtie \text{Fornitura} \\ \$_2 = \$_1 \end{array} \right) \bowtie \text{Fornitore}$$

\wedge
 $\$_2 \neq \$_2$

- Copie stessa città

$$\text{Copie} = \Pi \$_1 \left(\begin{array}{c} \text{Fornitore} \bowtie \text{Fornitore} \\ \$_3 = \$_3 \end{array} \right)$$

\wedge
 $\$_1 < \$_2$

- Codici / Almeno 3 merci

$$\text{Almeno 3} = \Pi \$_1 \left[\left(\begin{array}{c} \text{Fornitura} \bowtie \text{Fornitura} \\ \$_4 = \$_4 \end{array} \right) \bowtie \text{Fornitura} \right]$$

\wedge
 $\$_2 \neq \$_2$ \wedge
 $\$_5 \neq \$_2$

- Esattamente 1 = Fornitori - Almeno 2

$$\text{Almeno 2} = \Pi \$_1 \left(\begin{array}{c} \text{Fornitura} \bowtie \text{Fornitura} \\ \$_1 = \$_1 \end{array} \right)$$

\wedge
 $\$_2 \neq \$_2$

- Al più 2 = Fornitori - Almeno 3

$$\text{Almeno 3} = \text{Fornitura}$$

- Almeno 2 e al più 4 = $\underbrace{\text{Almeno 2} - \text{Almeno 5}}$

Rassimoli

- Non forniscono merci
=

Fornitori - Quelli che forniscono

- Prezzo minimo di fornitura di $\pi_{\$1}$

$$NO = 6_{\$2 = "NO"} (\text{Fornitura}) \wedge \text{Fornitura}$$

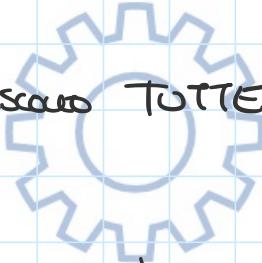
$$\quad \quad \quad \$2 = \$2$$

$$\quad \quad \quad \wedge$$

$$\quad \quad \quad \$3 > \$3$$

$$Ris = \pi_{\$3} (6_{\$2 = "NO"} \text{ Fornitura} - NO)$$

- Fornitori che forniscono TUTTE le merci



$$\text{Fornitori NO} = \pi_{\$1} (\text{Fornitore}) \times \pi_{\$1} (\text{merce}) - \pi_{\$1} (\text{Fornitura})$$

$$Ris = \pi_{\$2} \left\{ [\pi_{\$1} (\text{Fornitura}) - \text{Fornitori NO}] \wedge \text{Fornitore} \right\}$$

$$\quad \quad \quad \$1 = \$1$$

- Marche (Almeno 2 merci e ogni merce da almeno due fornitori)

$$\text{Almeno 2F} = \pi_{\$2} \left[\text{Fornitura} \wedge \text{Fornitura} \right]$$

$$\quad \quad \quad \$2 = \$2$$

$$\quad \quad \quad \wedge$$

$$\quad \quad \quad \$1 = \$1$$

$$\text{Almeno 2M} = \text{Almeno 2F} \wedge \text{Merce}$$

$$\quad \quad \quad \$1 = \$1$$

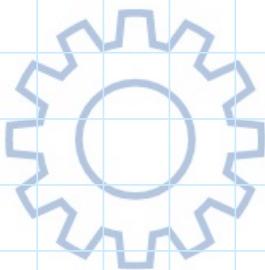
$$\quad \quad \quad \wedge$$

:

- Merce /cs + fornite da un fornitore

$$E_1 = \text{Alimento 1} - \text{Alimento 2} = \text{Fornitore} - \text{Alimento 2}$$

:



APPUNTI DI INGEGNERIA INFORMATICA

GAIA BERTOLINO

SQL → Structured Query Language

Al suo interno esistono due linguaggi chiamati

- DDL (Data Definition Language)
 - ↳ creazione, modifica e altre operazioni su tabella
- DML (Data Manipulation Language)
 - ↳ creazione, modifica e altre operazioni su tuple

Operazione di creazione di un DB con tabelle:

`CREATE DATABASE NomeDelDatabase;` → crea il DB

`USE NomeDelDatabase;` → definisce che i vari comandi si riferiscono ad esso

`CREATE TABLE NomeTabella AS`

(colonna1 CHAR(numeroCaratteri) PRIMARY KEY,
 esattamente definisce una chiave primaria
 max)
 colonna2 VARCHAR(numeroCaratteri) UNIQUE, definisce una chiave secondaria
 colonna3 VARCHAR(numeroCaratteri) NOT NULL
);

} crea una tabella nel database

I vincoli di unicità e chiave primaria possono essere anche inseriti in coda

es. `CREATE DATABASE MiDB;`

`USE MiDB;`

`CREATE TABLE Fornitore GAS BERTOLINO`

(Piva CHAR(20),
 Nome VARCHAR(40) NOT NULL,
 Città VARCHAR(20),
 PRIMARY KEY Piva);

`CREATE TABLE Merce (`

Codice NUMBER(10),
 Nome VARCHAR(40),
 Marca VARCHAR(40),
 PRIMARY KEY Codice);

`CREATE TABLE Fornitore (`

Fornitore CHAR(20) REFERENCES Fornitore (Piva),
 Merce NUMBER(10) REFERENCES Merce (Codice),

Fornitore CHAR(20) REFERENCES Fornitore (PIVA),
 Merce NUMBER(10) REFERENCES Merce (Codice),
 Prezzo NUMBER(4,2) NOT NULL,
 PRIMARY KEY (Fornitore, Merce),

chiave doppia

→ ciò può anche essere scritto come

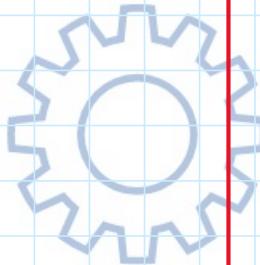
```

CREATE TABLE Fornitore (
    Fornitore CHAR(20),
    Merce NUMBER(10),
    Prezzo NUMBER(4,2) NOT NULL,
    PRIMARY KEY (Fornitore, Merce),
    FOREIGN KEY Fornitore REFERENCES Fornitore (PIVA),
    Merce REFERENCES Merce (Codice));
  
```

Altre operazioni:

Cancellazione:

DROP TABLE nome;



Modifica:

ALTER TABLE (descrizione modifica);

Inserimento di tuple:

INSERT INTO Nome VALUES (tupla);

Eliminazione Tuple:

DELETE ...

Modifica tuple:

UPDATE ...

Selezione di Tuple:

SELECT ...

SELECT

Select ha lo scopo di visualizzare / stampare qualsiasi valore a video

SELECT valori

{ ciò corrisponde, in algebra relazionale, a:
T valori (Nome tabella)

SELECT valori

FROM NomeTABELLA;

Ciò corrisponde, in algebra relazionale, a:

Π valori (NomeTABELLA)

es. SELECT PiVA, nome

FROM Fornitore;

Π PiVA, nome (Fornitore)

es. SELECT nome

FROM Fornitore;

Π nome (Fornitore)

es. SELECT *

FROM Fornitore;

Fornitore

Ridenominazione

SELECT nome AS nuovoNome

FROM NomeTABELLA;

Riconosce le colonne

selezionate dalla tabella

es. SELECT M.nome AS NomeM

FROM Merce AS M

WHERE M.marca = 'A'

Il prefisso evita la disambiguità
nel caso di riferimento di più tabella

Riconosce sia la tabella

merce che la colonna M

con lo scopo di semplificare

il riferimento delle stesse

Condizioni

GAIA BERTOLINO

SELECT nome

FROM NomeTABELLA

WHERE condizione1 AND condizione2 OR condizione3 ...

es. SELECT nome

FROM Fornitore

WHERE città = 'CS' OR (PiVA = 'F1' AND nome <> 'A');

Connettori: = > < >= <= <> != #

Diverso

es. IP nome delle merci che sono della marca A

SELECT nome

FROM Merce

Π \$₁ (\$₃=A (Merce))

```
SELECT movie
FROM Merce
WHERE marca = 'd'
```

$\pi_{\$1} (\exists_{\$3=\alpha} \text{ (here)})$

Esempi completi

es. Coppie di valori di formatori della stessa città

ALGEBRA RECAZIONALE I

TI \$2, \$3 (Fornitore Δ Fornitore)

EQUIVALENTE A

$$\pi_{\$2,\$5} \left[\begin{matrix} 6_{\$3=\$6} \\ \$2+\$4 \end{matrix} \right] (\text{Forniture} \times \text{Fornitura})$$

SQL : SELECT F!.name, F!.name
FROM Formulare AS F¹, Formulare AS F²
WHERE F¹.citta = F².citta
AND F¹.piva ≠ F².piva;

es. Copie nomeFornitore e nomeMerca effettiva (= il fornitore fornisce la merce)

ALGEBRA RELAZIONALE: $\Pi_{\$2, \$8} [(\text{Formulare } \bowtie_{\$1=\$2} \text{ Formulare}) \bowtie_{\$5=\$4} \text{ Merce }]$

GAIA BERTOLINO

EQUIVALENTE A

$$\text{Tr} \$2, \$8 \left[\begin{matrix} 6_{\$1=\$4} \\ 1 \\ \$5=\$7 \end{matrix} \quad (\text{Fornitore} \times \text{Fornitura} \times \text{Merca}) \right]$$

SQL : SELECT F.name , M.name
FROM Formulare F , Formulare X , Merce M
WHERE F.Piva = X.Formulare
AND M.codici = X.merce .

es. con una JOIN o più il codice è complesso se non fatto con una proiezione

SELECT F.name, H.name
FROM (Formulare F INNER JOIN Formulare X
ON F.id = X.id) INNER JOIN Kunde

`FROM (Fornitore F INNER JOIN Fornitura X
ON F.PIVA = X.Fornitore) INNER JOIN Merce
RE ON(...);`

Le joins si distribuiscono a cascata e ciò causa
poca comprensione.

Dunque NON si usa mai

EXCEPT

Serve ad implementare la sottrazione. Esso si applica a due SELECT

es. $\Pi_{\$1}(\text{Fornitore}) - \Pi_{\$1}(\text{Fornitura})$

`SELECT PIVA
FROM Fornitore
EXCEPT
SELECT Fornitore
FROM Fornitura;`

~~`SELECT nome
FROM Merce
EXCEPT
Fornitore;`~~

↳ non può essere
applicato ad una tabella
senza select

UNION

Serve a realizzare l'unione di due selezioni

es. `SELECT PIVA`

GAIA BERTOLINO

`FROM Fornitore`

`WHERE nome = 'x'`

`UNION`

`SELECT Fornitore`

`FROM Fornitura`

`WHERE Merce = 'y'`

INTERSECT

Implementata l'intersezione. Tuttavia è spesso non presente nei DB poiché può essere implementata con un join.

es. `Fornitori che forniscono esattamente una merce`

ALGEBRA RELAZIONALE: Almeno 1 = $\Pi_{\$1}(\text{Fornitura})$

$\text{Almeno } 2 = \Pi_{\$1} (\text{Fornitore } \Delta \text{ Fornitura})$

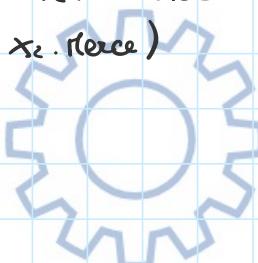
$$\begin{array}{c} \$1 = \$1 \\ \wedge \\ \$2 \neq \$2 \end{array}$$

Query = Almeno 1 - Almeno 2
Esattamente uno

SQL : **CREATE VIEW** Almeno1 (Fornitore) AS
 (SELECT Fornitore
 FROM Fornitura);

CREATE VIEW Almeno2 (Fornitore) AS
 (SELECT X1.Fornitore
 FROM Fornitura X1, Fornitura X2
 WHERE X1.Fornitore = X2.Fornitore
 AND X2.Merce \neq X1.Merce)

SELECT *
 FROM Almeno1
 EXCEPT
 SELECT *
 FROM Almeno2;



APPUNTI DI INGEGNERIA INFORMATICA

VISTA \Rightarrow Crea una tabella data da
 un risultato parziale

es. Fornitori di CS che non forniscono merce di marca d

ALGEBRA RELAZIONALE :

Fornitore NO = $\Pi_{\$4} [G_{\$3=d} (\text{Merce}) \Delta \text{ Fornitura}]$

$$\begin{array}{c} \$1 = \$1 \\ \wedge \\ \$3 = \$2 \end{array}$$

$Q = \Pi_{\$1} [G_{\$3=CS} (\text{Fornitore})] - \text{Fornitore NO}$

SQL : SELECT piuA
 FROM Fornitore
 WHERE citta = 'CS'
 EXCEPT

```
WHERE citta = 'CS'
```

```
EXCEPT
```

```
SELECT *
```

```
FROM FornitoreNO;
```

```
CREATE VIEW FornitoreNO ( PiVA ) AS
```

```
( SELECT X.Fornitore  
FROM Fornitura X, Marca M  
WHERE X.Merce = M.Codice  
AND M.Marca = 'A' ),
```

ALTRI COSTRUTTI :

IN → Serve ad implementare la ricerca incrociata

es. ALGEBRA RELAZIONALE : $\Pi_{\$2}[(\text{Fornitore} \bowtie \text{Fornitura})]$

$\$1=\2

EQUIVALENTE A

$$\Pi_{\$2}[\delta_{\$1-\$4}((\text{Fornitore} \times \text{Fornitura}))]$$

SQL : 1) **versione iniziale**

```
SELECT F.nome  
FROM Fornitore F, Fornitura X  
WHERE F.PiVA = X.Fornitore
```

versione con IN

```
2) SELECT F.nome  
FROM Fornitore F  
WHERE F.PiVA IN (SELECT Fornitore FROM Fornitura)
```

NOT IN → Viene utilizzato come la IN ma col significato opposto

es. **Copie nomeFornitore e nomeMerce Tali che il primo fornisce la seconda**

ALGEBRA RELAZIONALE : $\Pi_{\$2,\$8}[(\text{Fornitore} \bowtie \text{Fornitura}) \bowtie \text{Merce}]$

$\$2=\1

$\$5=\1

SQL : 1) **SELECT F.nome, M.nome**

```
FROM Fornitore F, Fornitura X, Merce M
```

```
WHERE X.Fornitore = F.PiVA AND M.Codice = X.Merce
```

Riscriviamo la richiesta :

Copie di nomi di Fornitori e merci tali che le corrispondenti coppie (PiVA, codice) appartengono a fornitora

2) `SELECT F.nome, M.nome
FROM Fornitore F, Merce M
WHERE (F.PiVA, M.codice) IN (SELECT Fornitore, merce
FROM Fornitura)`

A destra di una IN / NOT IN deve sempre esserci una tabella

Quando vengono unite due tabelle ad esse si associano delle intestazioni (es. F.nome, M.nome) che permettono di non avere problemi di sovrapposizione di nomi nella tabella

es. Copie di nomi di Fornitori e Merci tali che il fornitore NON fornisce la merce
ALGEBRA RELAZIONALE:

$$\{ [\pi_{\$3, \$4} (\text{Fornitore} \times \text{Merce}) - \pi_{\$1, \$2} (\text{Fornitore})] \bowtie_{\$1=\$2} \text{Fornitore} \bowtie_{\$2=\$1} \text{Merce}$$

SQL: `CREATE VIEW CopieCodiceNO (...) AS
(SELECT F.PiVA, M.codice
FROM Fornitore F, Merce M
EXCEPT
SELECT Fornitore, Merce
FROM Fornitore)`

SELECT
FROM CopieCodiceNO C, Fornitore F, Merce M
WHERE C.Fornitore = F.PiVA AND C.Merce = M.codice

es. Copie di nomi di fornitore e merci tali che le coppie (PiVA, codici) corrispondenti non appartengono a Fornitura

SQL: `SELECT
FROM Fornitore F, Merce M
WHERE (F.PiVA, M.Codice) NOT IN (SELECT Fornitore,`

Un linguaggio è detto **DICHIARATIVO** ovvero esprime direttamente il risultato. Invece, l'algebra relazionale è detta **INTRINSICAMENTE PROCEDURALE** in quanto il risultato si può comprendere osservando la procedura.

Exists → Richiede che l'operazione che ha come argomento restituisca almeno un elemento

WHERE ...

AND Exists (SELECT ...)

es. **Nomi di fornitori che forniscono almeno 1 merce.**

SELECT F.nome

FROM Fornitore F

WHERE Exists (SELECT *

FROM Fornitore X

WHERE X.Fornitore = F.Piva);

EQUIVALENTE A

WHERE F.Piva IN (SELECT Fornitore

FROM Fornitore).

NOT EXISTS → opposto dell'exists

es. Per ogni merce il prezzo minimo di fornitura

Cioè equivale a "per ogni merce, il prezzo è tale che non
ne esiste uno più piccolo"

ALGEBRA RELAZIONALE:

Copie MercePrezzoNo = $\Pi_{\$2,\$3} (\text{Fornitura} \bowtie \text{Fornitura})$

$\begin{matrix} \$2 = \$2 \\ \uparrow \\ \$3 > \$3 \end{matrix}$

Query = $\Pi_{\$2,\$3} (\text{Fornitura}) - \text{Copie MercePrezzoNo}$

SQL: SELECT

FROM Fornitura X₁

WHERE NOT EXISTS (SELECT

FROM Fornitore x2
 WHERE x2.Riceve = x1.Riceve
 AND x2.prezzo < x1.prezzo)

es. Nomi di fornitori che forniscono tutte le merci

Ciò equivale a "nomi di fornitori tali che non esistono merci M tali che F1.M non è in Fornitura"

ALGEBRA:

$\text{Fornitore No} = \Pi_{\$1}(\text{Fornitore}) \times \Pi_{\$1}(\text{Mer}) -$
 $- \Pi_{\$1, \$2}(\text{Fornitura})$

$Q = \Pi_{\$3} \left\{ \left[\Pi_{\$1}(\text{Fornitore}) - \Pi_{\$1}(\text{Fornitore No}) \right] \right\}_{\$1=\$2} \text{Fornitore}$

SQL:

SELECT
 FROM Fornitore F
 WHERE NOT EXISTS (SELECT *
 FROM Merce M
 WHERE (F.PIVA, M.codice) NOT IN
 (SELECT Fornitore, Merce
 FROM Fornitura));

es. Coppe di nomi nomeF1, nomeF2 tali che l'insieme delle merci fornite da F1 è uguale all'insieme delle merci fornite da F2.
 Corrisponde a: Non esiste alcuna merce fornita da F2 ma non da F1 e viceversa

SQL:

SELECT F1.nome, F2.nome
 FROM Fornitore F1, Fornitore F2
 WHERE NOT EXISTS (SELECT *
 FROM Fornitura x1
 WHERE x1.Fornitore = F1.PIVA
 AND (F2.PIVA, x1.Merce) NOT IN
 (SELECT Fornitore, Merce
 FROM Fornitura))
 AND NOT EXISTS (SELECT *

```

FROM Fornitore x
WHERE x.Fornitore = Fz.PIVA
AND (Fz.PIVA, x.Nezce) NOT IN
( SELECT Fornitore, Nezce
  FROM Fornitura )
AND Fz.Nome ≠ Fz.Nome;

```

Operatori matematici:

Essi sono SUM, COUNT, MIN, MAX, AVG...

es. Clienti (CF, nome, città, età)

Età massima:

```

SELECT max(età) AS etàmassima
FROM Clienti

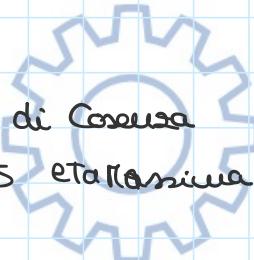
```

Età massima dei clienti di Cesena

```

SELECT max(età) AS etàmassima
FROM Clienti
WHERE città = 'CS'

```



Età media dei clienti diversi da 'N' :

```

SELECT Avg(età) AS etàmedia
FROM Clienti
WHERE nome ≠ 'N'

```

Categoria clienti:

```

SELECT count(CF)
FROM Clienti

```

Numeri righe della Tabella:

```

SELECT count(*) AS numerorighe
FROM Clienti

```

COUNT ha la particolarità di contare gli elementi: non nulli
COUNT-DISTINCT invece calcola i valori di elementi distinti

es. Numero di clienti che risiede a RC

```
SELECT COUNT(*)
```

```
FROM Cliente
```

```
WHERE citta = 'RC'
```

GROUP BY → Crea tabella per ogni valore della colonna di raggruppamento

es. SELECT citta, COUNT(*)

```
FROM Cliente
```

```
WHERE nome ≠ 'Pasquale'
```

```
GROUP BY citta
```

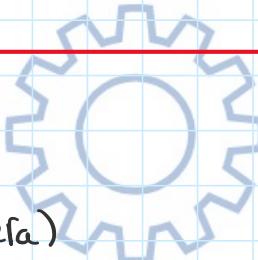
RECIA SINTATTICA

Nella SELECT, si possono mettere SOLO gli attributi della Group by se si usano gli aggregatori

esempi sbagliati:

```
SELECT nome, SUM(eta)
```

```
FROM Cliente;
```



SELECT ~~citta~~, ~~nome~~, COUNT(*)

```
FROM Cliente
```

```
GROUP BY citta;
```

Esempi

SCHEMI DI RELAZIONI

Cliente (CF, nome, citta, eta)

Prodotto (codice, nome, marca)

Fattura (numero, anno, cliente, data)

Composizione (numeroF, annoF, prodotto, qta, prezzoU)

1) Calcolare l'età media dei clienti che vivono a RC

```
SELECT AVG(eta)  
FROM Cliente  
WHERE city != 'Rc'
```

~~SELECT AVG(eta), nome~~
FROM Cliente SBALCIATO!
WHERE citta ≠ 'RC' Nov fa parte di una Group by

- 2) Calcolare il numero complessivo di esemplari del prodotto 'Pz' venduto a clienti di CS

ALGEBRA !

Screva Tabella:

[\$6 = \$3 = \$5 (Cliente)] \times \$1 = \$3 Fattura] \times \$6 = \$3 = \$1 (composizione)

SQL:

```
SELECT sum(qta)
FROM Clienti C, Fattura F, Composizione X
WHERE C.citta = 'CS' AND X.Prodotto = 'P1'
      AND F.cliente = C.CF
      AND F.numero = X.numeroF
      AND F.anno = X.annoF
```

Piccola variazione: per ogni coppia città, prodotto associa anche il numero totale di prodotto venduto:

~~SELECT SUM(qta), prodotto~~
FROM Clienti C, Fattura F, Composizione X
WHERE F.cliente = C.CF
AND F.numero = X.numeroF
AND F.anno = X.annoF
GROUP BY citta, prodotto

Tale codice NON considera le coppie con qualisiasi paria zero

- 3) Per ogni fattura, il numero di prodotti distinti in fattura (cioè la somma delle quantità)

```
SELECT numeroF, annoF, COUNT (prodotto)
FROM Composizione
GROUP BY numeroF, annoF.
```

FROM Composizione

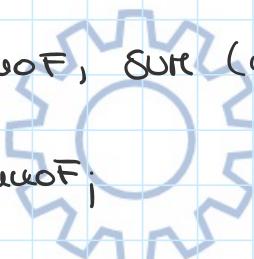
GROUP BY numeroF, annoF;

In questo caso usare count o count-distinct è uguale poiché per realizzazione la composizione ha elementi distinti.

In una select posso inserire più aggregati ma non posso inserire a cascata gli aggregati es. sum(max(...)) se si riferiscono a righe diverse. Invece, se l'aggregato fa riferimento alla stessa riga allora è possibile eseguirlo es. sum(qta * prezzoU) perché non si stanno inserendo aggregatori.

4) Per ogni fattura, l'importo Totale.

SELECT numeroF, annoF, sum(qta * prezzoU)
FROM Composizione
GROUP BY numeroF, annoF;



5) Per ogni città l'età minima dei clienti della città

SELECT citta, min(eta)
FROM cliente
GROUP BY citta

APPUNTI DI INGEGNERIA

INFORMATICA

GAIA BERTOLINO

6) La coppia CF, età del cliente più giovane

~~SELECT CF, min(eta)~~
~~FROM cliente~~

REGOLA SINTATTICA

↓
In una query dove è presente un aggregatore non è possibile inserire un attributo nella select che non è argomento di una group by

Una versione correttamente posta corrisponde alla ricerca di tutti i clienti che hanno età pari all'età minima

Un operatore di aggregazione

SELECT CF, min(eta)

... , ...

```
SELECT CF, eta  
FROM cliente  
WHERE eta = HIN(HIN)
```

Un operatore di aggregazione
non può stare in una
where poiché SQL non sa
se è specificata una group by
(che di default è fatta dopo)

Versione corretta:

1)

```
SELECT CF, eta  
FROM cliente  
WHERE eta IN (SELECT min(eta)  
               FROM cliente);
```

2)

```
SELECT CF, eta  
FROM cliente  
WHERE eta = (SELECT min(eta)  
               FROM cliente);
```

?) Per ogni fattura che ha almeno 10 prodotti, realizza
l'impiego complessivo

~~SELECT numeroF, annoF, SUM(qta * prezzoU)
FROM Composizione
WHERE COUNT(prodotto) ≥ 10
GROUP BY numeroF, annoF~~

1)

```
SELECT numeroF, annoF, SUM(qta * prezzoU)  
FROM Composizione C1  
WHERE 10 ≤ (SELECT COUNT(prodotto)  
            FROM Composizione C2  
            WHERE C1.numeroF = C2.numeroF  
              AND C1.annoF = C2.annoF)  
GROUP BY numeroF, annoF
```

2)

```
CREATE VIEW A (...) AS  
(SELECT numeroF, annoF, COUNT(*)  
  FROM Composizione  
 GROUP BY numeroF, annoF);
```

FROM Composizione
GROUP BY numeroF, annoF;

SELECT numeroF, annoF, SUM(qta * prezzo)
FROM Composizione C, A
WHERE C.numeroF = A.numeroF
AND C.annoF = A.annoF
AND A.numero >= 10
GROUP BY numeroF, annoF;

HAVING → consente di scrivere delle condizioni che possono essere verificate dopo la group by. Dunque può contenere degli aggregati.

es. SELECT numeroF, annoF, SUM(qta * prezzo)
FROM Composizione
GROUP BY numeroF, annoF
HAVING COUNT(prodotto) >= 0

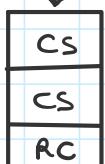
APPUNTI DI INGEGNERIA

SQL vero è esattamente riconducibile come l'algebra rispetto al concetto di tabella e insieme di tuple vera lo è rispetto al concetto di multinsieme ovvero uno stesso elemento può essere presente più volte

es.

C ₁	A	CS	10
C ₂	B	CS	20
C ₃	C	RC	15

SELECT citta
FROM cliente



$\pi_{\{3\}}(\text{clienti})$



costo quadratico
rispetto al numero
di tuple

SELECT Distinct citta → La Distinct si applica
FROM cliente

1...1... or ...n...