

Operatori aggregati

- Nella target list possiamo avere anche espressioni che calcolano valori a partire da insiemi di ennuple e che restituiscono una tabella molto particolare, costituita da un singolo valore scalare.
- SQL-2 prevede 5 possibili operatori aggregati:
 - Conteggio (COUNT),
 - Minimo (MIN),
 - Massimo (MAX),
 - Media (AVG),
 - Somma (SUM)

Espressioni aritmetica e valori NULL

- Se un argomento è NULL, lo è anche l'intera espressione
- Esempio: Prezzo * 1,22 - Sconto
- Nelle funzioni di gruppo, in generale i valori NULL sono ignorate.
- Esempio: somma(prezzi)
- Quindi, può accadere che:
 - Sum(Prezzo + IVA) è diversa da
 - Sum(Prezzo)+sum(IVA)

Prezzo	IVA
1000	21
NULL	15
2000	30

Operatori aggregati: COUNT

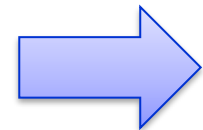
COUNT restituisce il **numero di righe** della tabella o il **numero di valori** di un particolare attributo

Esempio: Il numero di figli di Franco:

```
SELECT count(*) as NumFigliDiFranco  
FROM Paternita  
WHERE Padre = 'Franco'
```

l'operatore aggregato (**count**) viene applicato al risultato dell'interrogazione:

```
SELECT *  
FROM Paternita  
WHERE Padre = 'Franco'
```



Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

```
SELECT count(*)  
as NumFigliDiFranco  
FROM Paternita  
WHERE Padre = 'Franco'
```

```
SELECT *  
FROM Paternita  
WHERE Padre = 'Franco'
```

Padre	Figlio
Franco	Andrea
Franco	Aldo

count

NumFigliDiFranco
2

(*), ALL e DISTINCT

Mediante le specifiche (*), ALL e DISTINCT è possibile contare

(*): tutte le righe selezionate;

ALL: tutti i valori non nulli delle righe selezionate;

DISTINCT: tutti i valori non nulli distinti delle righe selezionate.

Se la specifica viene omessa, il default è ALL.

EsamiBD

Contare il numero di studenti iscritti al corso di BD e di Laboratorio di Basi di Dati

```
SELECT count(*) as "NumStud"
```

```
FROM EsamiBD
```

NumStud
5

Studente	BD	LBD
012345	27	NULL
032456	25	23
035221	NULL	NULL
033445	28	30
032441	NULL	30

Contare il numero di esami di BD superati positivamente

```
SELECT count([ALL] BD) "ContaBD"
```

```
FROM EsamiBD
```

ContaBD
3

Numero di voti distinti dati all'esame di LBD

```
SELECT count(distinct LBD) "ContDistLBD"
```

```
FROM EsamiBD
```

ContDistLBD
2

Max e Min

Le funzioni **MAX** e **MIN** calcolano rispettivamente il maggiore e il minore degli elementi di una colonna. L'argomento delle funzioni max e min può anche essere un'espressione aritmetica.

Esempio

L'età della persona più anziana nella tabella persone

```
SELECT max(eta)
FROM Persone
```

Persone

NOME	ETA	REDDITO
------	-----	---------

Il più basso dei voti assegnati all'esame di BD

```
SELECT min(BD)
FROM EsamiBD
```

Studente	BD	LBD
----------	----	-----

Sum

La funzione **SUM** calcola la somma dei valori di una colonna.

Le specifiche **ALL** e **DISTINCT** permettono di sommare tutti i valori non nulli o tutti i valori distinti.

Il default in mancanza di specifiche è **ALL**.

Esempio:

Calcolare la somma degli stipendi mensili degli impiegati del settore Produzione.

Impiegati

Matricola	Cognome	Dipart	Stip
-----------	---------	--------	------

```
SELECT SUM (ALL stipendio)
FROM Impiegati
WHERE Dipart= 'Produzione'
```


AVG

La funzione **AVG** calcola la media (average) dei valori **non nulli** di una colonna.

Le specifiche **ALL** e **DISTINCT** servono a calcolare la media fra tutti i valori o tra i valori distinti. Il default è **ALL**.

Esempio: Calcolare la media degli stipendi degli impiegati del dipartimento di Produzione e che hanno meno di 30 anni

Impiegati

Matricola	Cognome	Dipart	Stip
-----------	---------	--------	------

```
SELECT AVG(stipendi)
FROM Impiegati
WHERE Dipart= 'Produzione' AND eta<30
```

Operatori aggregati e valori nulli

```
SELECT AVG(reddito) AS Redditomedio  
FROM persone
```

Persone

Nome	Eta	Reddito
Andrea	27	30
Aldo	25	NULL
Maria	55	36
Anna	50	36

Redditomedio
34

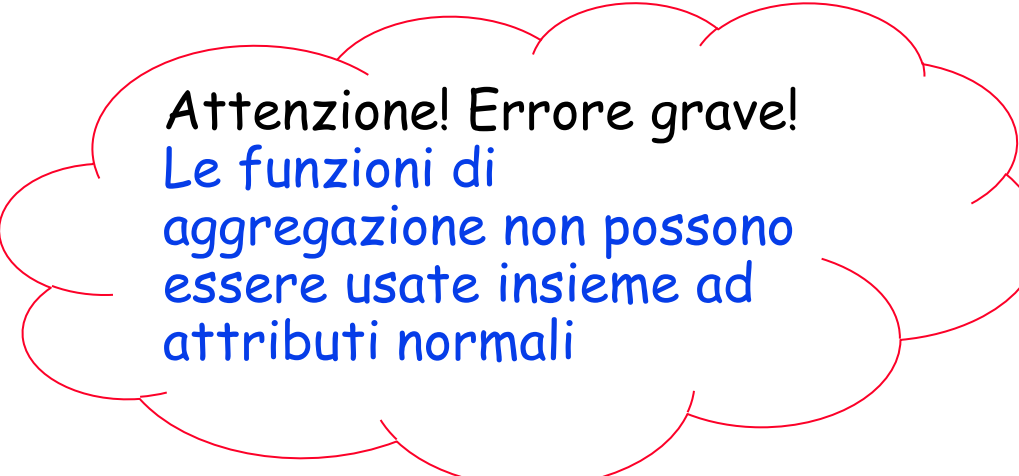
I valori nulli non vengono considerati nella media

Operatori aggregati e target list

Non è possibile utilizzare in una stessa select una proiezione su alcuni attributi della tabella considerata e operatori aggregati sulla stessa tabella.

- un'interrogazione scorretta:

```
SELECT nome, max(reddito)  
FROM persone
```



Attenzione! Errore grave!
Le funzioni di
aggregazione non possono
essere usate insieme ad
attributi normali

- di chi sarebbe il nome? La target list deve essere omogenea.
- E' **corretta** invece la seguente:

```
SELECT min(eta), avg(reddito)  
FROM persone
```

ESEMPI: ordinamenti e funzioni di aggregazione

- Studenti ordinati per Nome

```
SELECT      *  
FROM        Studenti  
ORDER BY    Nome;
```

- Numero di elementi di Studenti

```
SELECT count(*)  
FROM    Studenti;
```

- Anno di nascita minimo, massimo e medio degli studenti:

```
SELECT min(AnnoNascita), max(AnnoNascita), avg(AnnoNascita)  
FROM    Studenti;
```

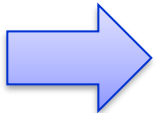
Group by

A volte può essere richiesto di calcolare operatori aggregati non per l'intera tabella, ma raggruppando le righe i cui valori coincidono su un certo attributo.

Per esempio, vogliamo sapere la media degli stipendi degli impiegati per ogni dipartimento. In tal caso si può utilizzare la clausola **GROUP BY**.

```
SELECT Dipart, AVG(stipendio)  
FROM Impiegati  
GROUP BY Dipart
```

Dipart	AVG(stipendio)
Produzione	1330
Amministrazione	1505
Distribuzione	1810
Direzione	2500



Semantica degli operatori di raggruppamento (1)

- La query è innanzitutto eseguita senza operatori aggregati e senza GROUP BY:

```
SELECT Dipart, stipendio  
FROM Impiegati
```

Quindi il risultato è diviso in sottoinsiemi aventi gli stessi valori per gli attributi indicati nel GROUP BY (Dipart nel nostro caso)

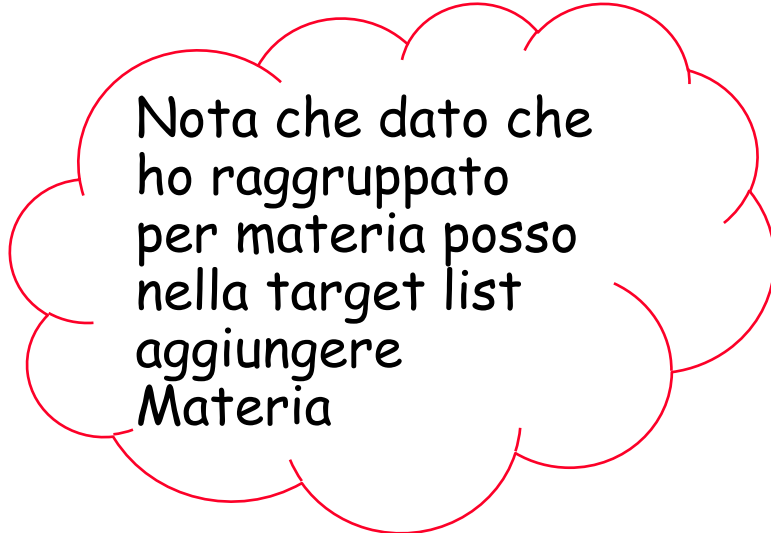
Quindi l'operatore aggregato è calcolato su ogni sottoinsieme:



IL RAGGRUPPAMENTO

- Per ogni materia, trovare nome della materia e voto medio:
 - Per ogni materia:
 - Un attributo della materia
 - Una funzione aggregata sugli esami della materia
- Soluzione:

```
SELECT e.Materia, avg(e.Voto)
FROM Esami e
GROUP BY e.Materia
```



Nota che dato che
ho raggruppato
per materia posso
nella target list
aggiungere
Materia

Osservazione

- Quando si effettua un raggruppamento, questo deve essere effettuato su tutti gli elementi della target list che non sono operatori aggregati (ossia sull'insieme degli attributi puri).
- Questo ha senso perché nel risultato deve apparire una riga per ogni «gruppo»

Esempio:

```
SELECT Dipart, AVG(stipendio)
```

```
FROM Impiegati
```

```
GROUP BY Dipart
```


Raggruppamenti e target list

- scorretta

```
SELECT padre, avg(f.reddito), p.reddito
FROM persone f JOIN paternita ON figlio = nome
JOIN persone p ON padre = p.nome
GROUP BY padre
```

- corretta

```
SELECT padre, avg(f.reddito)
FROM persone f JOIN paternita ON figlio = nome JOIN
persone p ON padre = p.nome
GROUP BY padre
```

Persone

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

IL RAGGRUPPAMENTO

- Per ogni studente, nome e voto medio:

```
SELECT s.Nome, avg(e.Voto)
FROM Studenti s, Esami e
WHERE s.Matricola = e.Matricola
GROUP BY s.Matricola, ...
```



Matricola non è
proiettato.

ATTENZIONE
occorre
raggruppare
anche per nome

- È necessario scrivere:
 - GROUP BY s.Matricola, **s.Nome**
- **Gli attributi espressi non aggregati nella select (s.Nome) devono essere inclusi tra quelli citati nella GROUP BY (s.Matricola, s.Nome)**
- Gli attributi aggregati (avg(e.Voto)) vanno scelti tra quelli non raggruppati

Esempio - Parte I - Due tabelle

Clienti

	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1	IDCLIENTE	NUMBER(38,0)	No	(null)	1	(null)
2	USERNAME	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	PASSWORD	VARCHAR2(10 BYTE)	No	(null)	3	(null)
4	NOME	VARCHAR2(20 BYTE)	No	(null)	4	(null)
5	COGNOME	VARCHAR2(20 BYTE)	No	(null)	5	(null)
6	DATANASCITA	DATE	No	(null)	6	(null)

idCliente è
chiave primaria

Username è
univoca

Cognome non
può essere
NULL

```
select idCliente, username, cognome, nome
from clienti
group by idCliente, username, cognome, nome
```

	❖ IDCLIENTE	❖ USERNAME	❖ COGNOME	❖ NOME
1	192	g.zavoli	Zavoli	Gasperino
2	42	l.zavoli	Zavoli	Leano
3	1	l.chiricozzi	Chiricozzi	Livio
4	2	b.gubinelli	Gubinelli	Brillante
5	3	a.borrelli	Borrelli	Assuntino
6	4	g.malagigi	Malagigi	Gilma
7	12	s.zavoli	Zavoli	Signorina

Esempio - Parte II - Raggruppamento per cognome

Clienti

	IDCLIENTE	USERNAME	COGNOME	NOME
1	192	g.zavoli	Zavoli	Gasperino
2	42	l.zavoli	Zavoli	Leano
3	1	l.chiricozzi	Chiricozzi	Livio
4	2	b.gubinelli	Gubinelli	Brillante
5	3	a.borrelli	Borrelli	Assuntino
6	4	g.malagigi	Malagigi	Gilma
7	12	s.zavoli	Zavoli	Signorina

Contiamo quante persone (cognomi e conteggio) hanno lo stesso cognome (stampare cognome e count(*))

```
select clienti.Cognome, count(*)  
from clienti  
group by clienti.Cognome
```

	COGNOME	COUNT(*)
1	Zavoli	3
2	Gubinelli	1
3	Malagigi	1
4	Chiricozzi	1
5	Borrelli	1

Esempio - Parte III - Raggruppiamo per chiavi

idCliente è chiave primaria
Username è univoco

	IDCLIENTE	USERNAME	COGNOME	NOME
1	192	g.zavoli	Zavoli	Gasperino
2	42	l.zavoli	Zavoli	Leano
3	1	l.chiricozzi	Chiricozzi	Livio
4	2	b.gubinelli	Gubinelli	Brillante
5	3	a.borrelli	Borrelli	Assuntino
6	4	g.malagigi	Malagigi	Gilma
7	12	s.zavoli	Zavoli	Signorina

Clienti

Contiamo quante persone hanno lo stesso idCliente (o username) e quanti hanno lo stesso username

```
select clienti.idCliente, count(*)  
from clienti  
group by clienti.idCliente
```

	IDCLIENTE	COUNT(*)
1	42	1
2	1	1
3	2	1
4	12	1
5	4	1
6	3	1
7	192	1

```
select username, count(*)  
from clienti  
group by username
```

	USERNAME	COUNT(*)
1	b.gubinelli	1
2	l.zavoli	1
3	s.zavoli	1
4	l.chiricozzi	1
5	g.zavoli	1
6	g.malagigi	1
7	a.borrelli	1

Posso fare un'unica query con idCliente e username nella target list?

Esempio - Parte IV

Clienti

	IDCLIENTE	USERNAME	COGNOME	NOME
1	192	g.zavoli	Zavoli	Gasperino
2	42	l.zavoli	Zavoli	Leano
3	1	l.chiricozzi	Chiricozzi	Livio
4	2	b.gubinelli	Gubinelli	Brillante
5	3	a.borrelli	Borrelli	Assuntino
6	4	g.malagigi	Malagigi	Gilma
7	12	s.zavoli	Zavoli	Signorina

Attenzione:
Errore grave

Contiamo quante persone hanno lo stesso idCliente, ma stampiamo anche il suo username

```
select idCliente, count(*), username
from clienti
group by idCliente
```

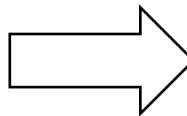
```
select idCliente, count(*), username
from clienti
group by idCliente, username
```

ORA-00979: non è un'espressione GROUP BY
00979. 00000 - "not a GROUP BY expression"

*Cause:

*Action:

Errore alla riga: 1, colonna: 29



	IDCLIENTE	COUNT(*)	USERNAME
1	1	1	l.chiricozzi
2	3	1	a.borrelli
3	4	1	g.malagigi
4	192	1	g.zavoli
5	42	1	l.zavoli
6	12	1	s.zavoli
7	2	1	b.gubinelli

Esempio - Parte V

Clienti

	IDCLIENTE	USERNAME	COGNOME	NOME
1	192	g.zavoli	Zavoli	Gasperino
2	42	l.zavoli	Zavoli	Leano
3	1	l.chiricozzi	Chiricozzi	Livio
4	2	b.gubinelli	Gubinelli	Brillante
5	3	a.borrelli	Borrelli	Assuntino
6	4	g.malagigi	Malagigi	Gilma
7	12	s.zavoli	Zavoli	Signorina

```
select idCliente, count(*), username
from clienti
group by idCliente, username
```

	IDCLIENTE	COUNT(*)	USERNAME
1	1	1	l.chiricozzi
2	3	1	a.borrelli
3	4	1	g.malagigi
4	192	1	g.zavoli
5	42	1	l.zavoli
6	12	1	s.zavoli
7	2	1	b.gubinelli

Che valori ottengo su count se sostituisco username con cognome?

```
select idCliente, count(*), cognome
from clienti
group by idCliente, cognome
```

	IDCLIENTE	COUNT(*)	COGNOME
1	4	1	Malagigi
2	1	1	Chiricozzi
3	42	1	Zavoli
4	12	1	Zavoli
5	2	1	Gubinelli
6	192	1	Zavoli
7	3	1	Borrelli

Esempio - Parte VI

Clienti

IDCLIENTE	COGNOME	NOME
192	Zavoli	Luigi
97	Grassi	Maria
114	Di Santo	Luigi
42	Zavoli	Luigi
5	Di Santo	Luigi
138	Grassi	Maria
12	Zavoli	Maria

```
select cognome, nome, count(*)  
from Clienti  
group by cognome, nome
```

	COGNOME	NOME	COUNT(*)
1	Zavoli	Luigi	2
2	Di Santo	Luigi	2
3	Zavoli	Maria	1
4	Grassi	Maria	2

```
select nome, count(*)  
from Clienti  
group by nome
```

NOME	COUNT(*)
Maria	3
Luigi	4

```
select cognome, count(*)  
from Clienti  
group by cognome
```

	COGNOME	COUNT(*)
1	Zavoli	3
2	Grassi	2
3	Di Santo	2

Concettualmente diversi:
prima si raggruppa per
cognome (risp. Nome) e la
partizione ottenuta si
partiziona ulteriormente
per nome (risp. Cognome).
Il risultato è uguale
perché la partizione
finale (cognome-nome) è
uguale a quella (nome-
cognome)

```
select nome, cognome, count(*)  
from Clienti  
group by nome, cognome
```

NOME	COGNOME	COUNT(*)
Maria	Grassi	2
Luigi	Zavoli	2
Luigi	Di Santo	2
Maria	Zavoli	1

Condizioni sui gruppi, clausola HAVING

- Si possono applicare condizioni sul valore aggregato per ogni gruppo. Si può realizzare mediante la clausola HAVING.
- Esempio: I dipartimenti la cui media degli stipendi è maggiore di 1700 euro

```
Select dipart, AVG(stipendio)  
FROM Impiegati  
Group by Dipart  
HAVING AVG(stipendio)>1700
```

Dipart	AVG(stipendio)
Amministrazione	1505
Distribuzione	1810
Direzione	2500
Produzione	1330

Dipart	AVG(stipendio)
Distribuzione	1810
Direzione	2500

HAVING AVG(stipendio)>1700

Where o Having

- In generale se la condizione coinvolge un attributo, si usa la clausola **where**, mentre se coinvolge un operatore aggregato si usa la clausola **having**.

EsamiBD (matricola, nome, cognome, città, voto, età)

- Le città per cui la media dei voti dei suoi studenti di meno di 21 anni è maggiore di 26

SELECT città, avg(voto)


FROM EsamiBD

WHERE età < 21

GROUP BY città

HAVING avg(voto) > 26

Attenzione! Having solo in presenza di un group by, e dopo di esso



SelectSQL ::=

select ListaAttributi O Espressioni

from ListaTabelle

[where CondizioniSemplici]

[group by ListaAttributiDiRaggruppamento]

[having CondizioniAggregate]

[order by ListaAttributiDiOrdinamento]

IL RAGGRUPPAMENTO

- `SELECT ... FROM ... WHERE ... GROUP BY A1,...,An [HAVING condizione]`
- Semantica:
 - Esegue le clausole `FROM - WHERE`
 - Partiziona la tabella risultante rispetto all'uguaglianza su tutti i campi `A1...An` (solo in questo caso, si assume `NULL = NULL`)
 - Elimina i gruppi che non rispettano la clausola `HAVING`
 - Da ogni gruppo estrae una riga usando la clausola `SELECT`

LA CLAUSOLA HAVING: IMPORTANTE

- Attenzione:
 - Se la `SELECT` contiene sia espressioni aggregate (`MIN`, `COUNT...`) che attributi non aggregati, allora **DEVE** essere presente la clausola `GROUP BY`
 - Le clausole `HAVING` e `SELECT` citano solo:
 - espressioni su attributi di raggruppamento;
 - funzioni di aggregazione applicate ad attributi non di raggruppamento.

ESECUZIONE DI GROUP BY

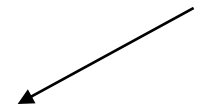
```
SELECT Matricola, count(*) AS NEsami, min(Voto), max(Voto), avg(Voto)
FROM Esami
GROUP BY Matricola
HAVING count(*) > 1;
```

Materia	Matricola	Voto	Docente
DA	1	20	10
LFC	2	30	20
MTI	1	30	30
LP	2	20	40



Materia	Matricola	Voto	Docente
DA	1	20	10
MTI	1	30	30
LFC	2	30	20
LP	2	20	40

Matricola	NEsami	min(Voto)	max(Voto)	Avg(Voto)
1	2	20	30	25
2	2	20	30	25



SQL E ALGEBRA RELAZIONALE

SQL -> ALGEBRA

SELECT DISTINCT S_A, S_{FA}

FROM R, S

WHERE W_C

GROUP BY G_A

HAVING H_C

ORDER BY O_A ;

ORDER BY O_A

DISTINCT

SELECT S_A, S_{FA}

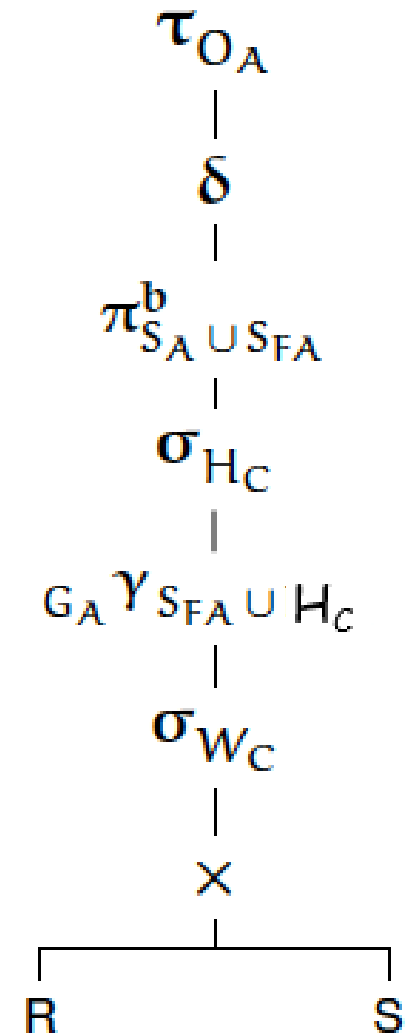
HAVING H_C

GROUP BY G_A

WHERE W_C

FROM R, S

Comando **SELECT**

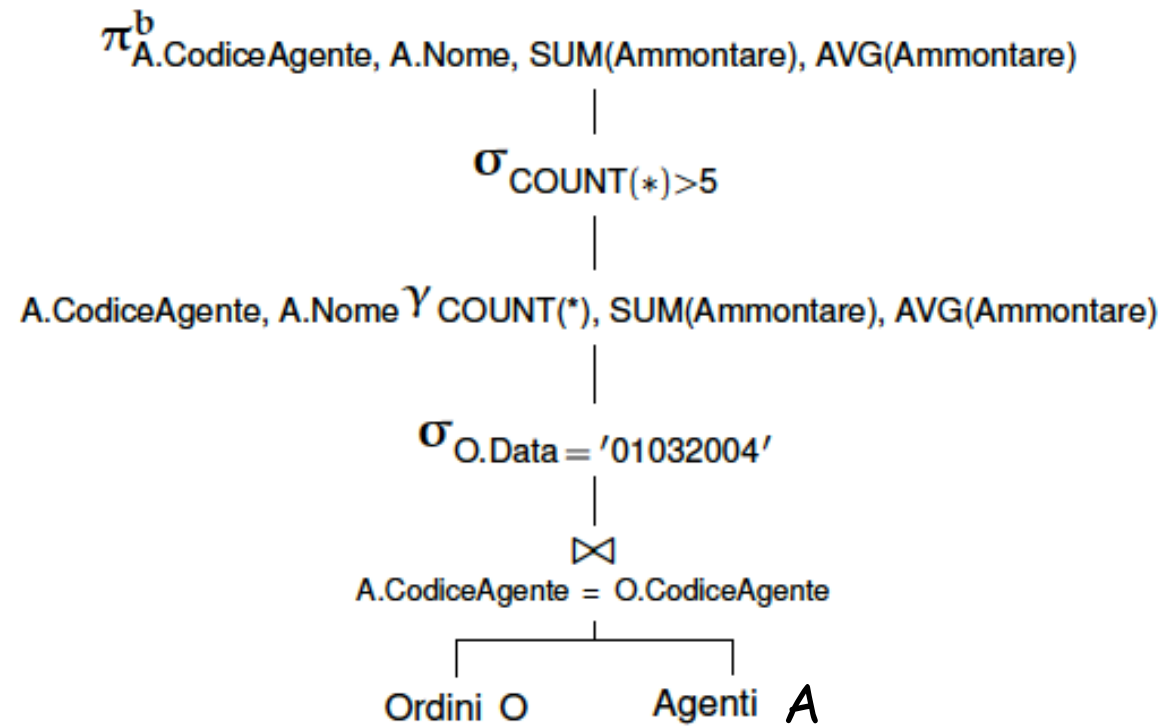


Albero logico

Esempio

Trovare il codice e nome degli agenti con più di cinque ordini in data 1/3/2004 e, degli ordini fatti, il totale e la media dell'ammontare:

```
SELECT A.CodiceAgente, A.Nome,  
       SUM(Ammontare), AVG(Ammontare)  
FROM Ordini O, Agenti A  
WHERE A.CodiceAgente = O.CodiceAgente  
      AND  
      O.Data = '01032004'  
GROUP BY O.CodiceAgente, A.Nome  
HAVING COUNT(*) > 5;
```



Riferimento alle colonne

- Spesso nel riferimento alle colonne selezionate nel join è necessario specificare da quale delle tabelle la colonna è stata estratta, al fine di evitare ambiguità.
- La sintassi usata è

NomeTabella.NomeColonna

Di solito in una select che definisce una join possono essere necessarie ridenominazioni

- nel prodotto cartesiano (ossia ridenominare le tabelle coinvolte)
- nella target list (ossia ridenominare gli attributi)

```
SELECT X.A1 AS B1, Y.A2 AS B2 ...  
FROM   Tab1 X, Tab2 Y, Tab1 Z  
WHERE  X.A2 = Y.A3 AND ...
```

→ Ridenominazioni
di colonne

→ Ridenominazione
di tabelle

JOIN

Cross Join

- Il cross-join implementa il prodotto cartesiano.
- Si realizza semplicemente mediante una select che utilizza le due (o più) tabelle coinvolte, senza specificare nessuna condizione di join.

```
Select categorie.*, Fabbriche.*  
From Categorie, Fabbriche
```

Join fra due tabelle

- Se due tabelle del database contengono dei dati in comune, possono essere correlate mediante un'operazione di **JOIN**.
- Le colonne delle due tabelle che creano la correlazione rappresentano la stessa entità, ossia i loro valori appartengono allo **stesso dominio**.
- In genere le colonne delle due tabelle considerate sono legate da un vincolo di chiave esterna (ma non è obbligatorio)

Join

- IL **join** (o **equi-join**) fra due tabelle è una terza tabella le cui righe sono tutte e sole quelle ottenute dal prodotto cartesiano delle righe delle due tabelle di partenza i cui valori delle colonne espresse dalla condizione di join sono uguali.
- In SQL il join viene realizzato mediante una particolare forma del **SELECT**, in cui
- Nella clausola **FROM** vengono indicate le due tabelle coinvolte
- Nella clausola **WHERE** viene espresso il collegamento fra le due tabelle, mediante la condizione di join

Join, sintassi

Siano Tab1(A1,A2) Tab2(A3,A4) due relazioni

SELECT Tab1.A1, Tab2.A4

FROM Tab1, Tab2

WHERE Tab1.A2 = Tab2.A3

Tab1

A1	A2
A	B
C	D
E	F
G	H

Tab2

A3	A4
F	I
D	L
D	M
B	N

Tab1 ⋈_{Tab1.A2=Tab2.A3} Tab2



A1	A4
A	N
C	L
C	M
E	I

Join e algebra relazionale

Tab1(A1,A2) Tab2(A3,A4)

SELECT Tab1.A1, Tab2.A4

FROM Tab1, Tab2

WHERE Tab1.A2 = Tab2.A3

Traduce l'espressione dell'algebra relazionale

$\pi_{A1,A4} (\sigma_{A2=A3} (Tab1 \bowtie Tab2))$

- Quindi il join consiste di:
 - Un prodotto cartesiano (FROM)
 - Una selezione (WHERE)
 - Una proiezione (SELECT)

Esempio

Supponiamo che nel registro automobilistico siano presenti le seguenti tabelle:

Categorie

<u>Cod_cat</u>	Nome_cat
----------------	----------

Veicoli

<u>Targa</u>	Cod_mod	Categoria*	Cilindrata	Cod_comb	cav.Fis	Velocita	Posti	Imm
--------------	---------	------------	------------	----------	---------	----------	-------	-----

Vogliamo selezionare per ciascun veicolo la descrizione della relativa categoria. In questo caso devono essere coinvolte le due tabelle. Le due tabelle sono legate da un vincolo di chiave esterna tra gli attributi Cod_cat (in Categorie) e Categoria (in Veicoli)

```
SELECT targa, Veicoli.Categoria, nome_cat  
FROM Categorie, Veicoli  
WHERE Veicoli.Categoria = Categorie.Cod_cat  
      [Categoria=cod_cat]
```

Targa	Categoria	Nome_cat
-------	-----------	----------



Veicoli

Targa	Categoria
A24353Q	01
F63457T	03
D2343GH	01
T94756U	02
W34985U	02
L23843K	01

```
SELECT targa, Veicoli.categoria, nome_cat  
FROM Categorie, Veicoli  
WHERE Veicoli.Categoria = Categorie.cod_cat
```

Categorie

Veicoli.Categoria=
categorie.cod_cat

Cod_cat	Nome_cat
01	Autovettura
02	Rimorchio
03	Motociclo
04	Furgone

Targa	Categoria	Nome_cat
A24353Q	01	Autovettura
F63457T	03	Motociclo
D2343GH	01	Autovettura
T94756U	02	Rimorchio
W34985U	02	Rimorchio
L23843K	01	Autivettura

Join, Esempio

SELECT targa, categorie.cod_cat, nome_cat	→ Scelta colonne
FROM Categorie, Veicoli	→ Tabelle selezionate
WHERE Veicoli.Categoria =Categorie.cod_cat	→ Condizione di join

La condizione di join può essere presente assieme ad altre condizioni mediante il connettore logico AND. Per esempio

```
SELECT targa, Veicoli.categoria, nome_cat
FROM Categorie, Veicoli
WHERE Veicoli.categoria =Categorie.cod_cat
      AND Cilindrata>1600
```

Join, Esempio

Se si vogliono selezionare tutte le colonne delle due tabelle si può sempre usare la notazione nome_tabella.*

```
SELECT Categorie.*, Veicoli.*  
FROM Categorie, Veicoli  
WHERE Veicoli.categoria = Categorie.cod_cat
```

In tal caso figureranno entrambi i campi Veicoli.Categoria e Categorie.cod_cat (denominati rispettivamente Categoria e cod_cat)

Join, Esempio 2

- Elencare i padri di persone che guadagnano più di 2000 euro al mese (reddito/12)

Maternità

Madre	Figlio
-------	--------

Paternità

Padre	Figlio
-------	--------

Persone

Nome	Eta	Reddito
------	-----	---------

$\pi_{\text{Padre}}(\text{paternita} \bowtie_{\text{Figlio=Nome}} (\sigma_{\text{Reddito}/12 > 2000} (\text{persone})))$

```
SELECT distinct padre
FROM persone, paternita
WHERE figlio = nome AND
      reddito/12 > 2000
```

Inner-Join

L' **INNER JOIN** è un' operazione di join in cui la condizione non sia necessariamente una condizione di uguaglianza.

Modelli

<u>Cod_mod</u>	Nome_mod	Cod_Fab	Num_vers	Cil_media
----------------	----------	---------	----------	-----------

Veicoli

<u>Targa</u>	Cod_mod*	Cod_cat	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm
--------------	----------	---------	------------	-----------	----------	----------	-------	-----

Tutti i Veicoli la cui cilindrata è minore della cilindrata media del loro modello

```
SELECT Veicoli.*  
FROM Modelli, Veicoli  
WHERE Veicoli.cod_mod=Modelli.cod_mod  
and Cil_media > cilindrata
```

Inner-Join

L' INNER JOIN è un' operazione di join in cui la condizione non sia necessariamente una condizione di uguaglianza.

Esempio:

Supponendo di avere una tabella impiegati e una reparti, trovare il nome dei reparti in cui non lavora Mario Rossi.

```
select rep.nome  
from impiegati as imp join reparti as rep on  
        imp.IdReparto <> rep.IdReparto  
where imp.cognome = 'Rossi' and imp.nome = 'Mario';
```


Self-join

- Un caso molto particolare di Join è quello che mette in relazione una tabella con se stessa. Questo si può ottenere ridenominando due volte la tabella con due nomi diversi, trattando le due copie come se si trattasse di due tabelle diverse. In questo caso si parla di **SELF JOIN**

```
SELECT X.A1, Y.A4  
FROM   Tab1 X, Tab2 Y, Tab1 Z  
WHERE  X.A2 = Y.A3 AND X.A2 = Z.A1
```

Self Join, Esempio

Trovare tutte le coppie di veicoli che sono dello stesso modello.

Veicoli

Targa	Cod_mod	Cod_cat	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm
-------	---------	---------	------------	-----------	----------	----------	-------	-----

```
Select V1.Targa, V2.Targa  
From Veicoli V1, Veicoli V2  
Where V1.cod_mod=V2.cod_mod  
and V1.Targa>V2.Targa
```

Condizione di
join

Per evitare che nel risultato ci siano coppie contenenti due volte la stessa macchina e le coppie che si ottengono scambiando l'ordine di coppie già esistenti

Esempio Self-join

Trovare i colleghi di Mario Rossi (cioè quelli che lavorano nello stesso reparto)

```
Select imp1.nome, imp1.cognome , Imp1.Reparto, Imp2.Nome, Imp2.Cognome, Imp2.Reparto
from impiegati imp1 join impiegati imp2
    on (imp1.idreparto = imp2.idreparto)
Where imp2.nome='Mario' and imp2.cognome = 'Rossi'
and imp1.matricola <> imp2.matricola;
```

Impiegati imp1

Nome	Cognome	Reparto
Mario	Rossi	Informatica
Luigi	Bianchi	Agraria
Mario	Rossi	Agraria
Marco	Verdi	Informatica
Luca	Viola	Agraria

Impiegati imp2

Nome	Cognome	Reparto
Mario	Rossi	Informatica
Luigi	Bianchi	Agraria
Mario	Rossi	Agraria
Marco	Verdi	Informatica
Luca	Viola	Agraria

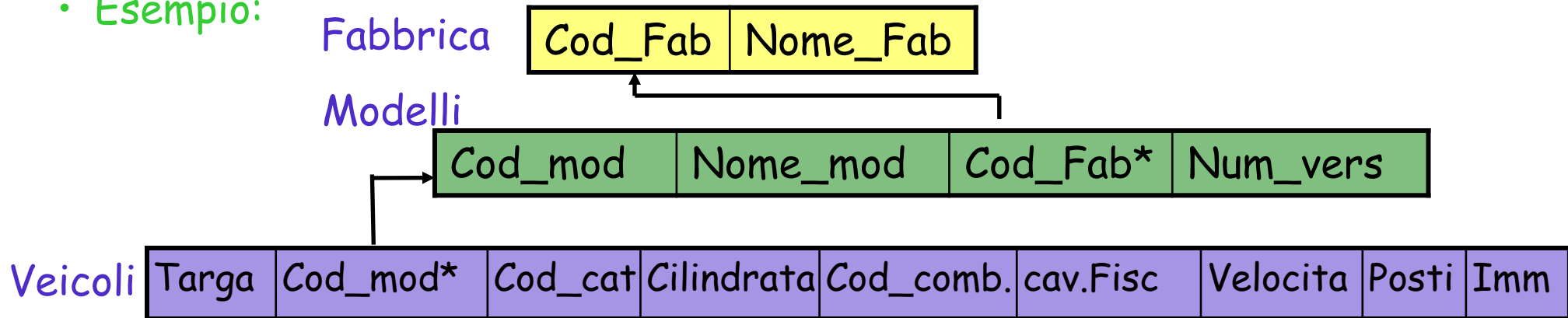
Imp1 join Imp2

Imp1.Nome	Imp1.Cognome	Imp1.Reparto	Imp2.Nome	Imp2.Cognome	Imp2.Reparto

Join su più tabelle

- Talvolta un'interrogazione può coinvolgere più di due tabelle.

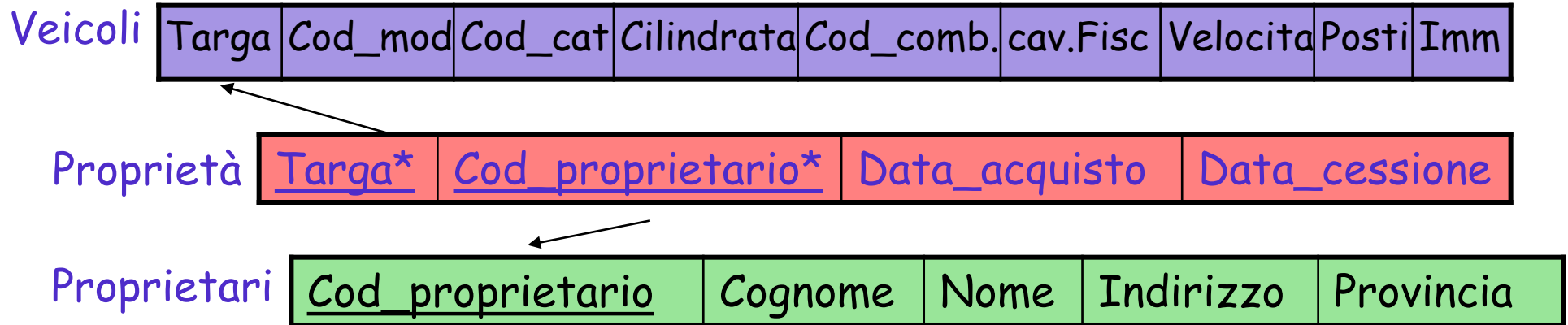
- Esempio:



Nome della fabbrica del modello dell'automobile con targa W34534R

```
Select Targa, Nome_mod, nome_fab
FROM Modelli M, Veicoli V, Fabbrica F
Where M.cod_mod=V.cod_mod and M.cod_fab=F.cod_fab
and V.targa= 'W34534R'
```

Esempio



Cognome e nome del proprietario e cilindrata della macchina la cui targa è W34534R.

```
Select nome, cognome, cilindrata
FROM Proprietari P, Proprietà PR, Veicoli V
Where V.targa=PR.targa
      and PR. Cod_Proprietario=P.cod_proprietario
      and targa = 'W34534R'
```

Join usando la clausola 'JOIN'

- Oltre alla forma vista, nei DBMS più moderni, per effettuare il join di due tabelle è possibile utilizzare una forma più esplicita (standard ANSI).
- La sintassi è la seguente

```
SELECT Attributi  
FROM Tab1 JOIN Tab2  
ON CondizioneDiJoin
```

Esempio

Categorie

<u>Cod_cat</u>	Nome_cat
----------------	----------

Veicoli

<u>Targa</u>	Cod_mod	Categoria*	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm
--------------	---------	------------	------------	-----------	----------	----------	-------	-----

Selezionare per ciascun veicolo la descrizione della relativa categoria utilizzando l'operatore JOIN.

```
SELECT targa, Veicoli.Categoria, nome_cat  
FROM Categorie JOIN Veicoli  
ON Veicoli.Categoria = Categorie.Cod_cat
```

Targa	Categoria	Nome_cat
-------	-----------	----------

Equi-Join e Natural Join

- Se dobbiamo operare una **equi-join**, ossia un join la cui condizione sia una condizione di uguaglianza, e che sia anche un **Natural Join**, ossia un join creato su tutte le colonne che hanno il medesimo nome in entrambe le tabelle, possiamo utilizzare la seguente sintassi

```
SELECT listaAttributi  
FROM Tab1 NATURAL JOIN Tab2
```


Esempio

DEPUTATI (Codice, Cognome, Nome, CodCommissione*, Provincia*, Collegio*)

COLLEGI (Provincia*, Numero*, Nome)

PROVINCE (Sigla, Nome, Regione*)

REGIONI (Codice, Nome)

COMMISSIONI (CodCommissione, Nome, Presidente)

Cosa succede se facciamo il natural join fra Deputati e Commissioni?

```
SELECT * FROM DEPUTATI NATURAL JOIN COMMISSIONI
```

È equivalente a

```
SELECT * FROM DEPUTATI, COMMISSIONI where Deputati.CodCommissione =  
Commissioni.CodCommissione
```

???

Using

- Può succedere comunque che nelle tabelle coinvolte ci siano più attributi con lo stesso nome, e col Natural Join tutte queste coppie di attributi presi dalla due tabelle vengono identificate.
- Se invece vogliamo operare una join in cui la condizione riguarda solo una o alcune di queste coppie, si usa la clausola **USING** seguita dall'elenco degli attributi coinvolti nella condizione

```
SELECT lista attributi  
FROM Tab1 JOIN Tab2  
USING (attr1,attr2,...)
```

Esempio

Modelli

Cod_mod	Nome_mod	Cod_Fab	Num_vers	Cil_media
---------	----------	---------	----------	-----------

Veicoli

<u>Targa</u>	Cod_mod*	Cod_cat	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm
--------------	----------	---------	------------	-----------	----------	----------	-------	-----

Il nome del modello del veicolo di targa ZX2345

```
SELECT Nome_mod  
FROM Modelli JOIN Veicoli  
USING (Cod_mod)  
WHERE targa="ZX2345"
```

Outer Join

- Quando vengono correlate mediante una join delle tabelle con colonne contenenti dati in comune, è possibile che un valore sia presente in una delle colonne e non nell'altra.
- Effettuando un equi-join la riga corrispondente a tale valore viene scartato.
- In alcuni casi invece può essere necessario mantenere questi valori. Per fare questo si deve effettuare un **outer join**.

```
Select lista_attributi  
From Tab1 LEFT [OUTER] JOIN Tab2
```

```
Select lista_attributi  
From Tab1 RIGHT [OUTER] JOIN Tab2
```

```
Select lista_attributi  
From Tab1 FULL [OUTER] JOIN Tab2
```

Outer join, Esempio

Categorie

<u>Cod_cat</u>	Nome_cat
----------------	----------

Veicoli

<u>Targa</u>	Cod_mod*	Cod_cat	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm
--------------	----------	---------	------------	-----------	----------	----------	-------	-----

Ottenere i codici e nomi delle categorie di macchine della tabella veicoli e anche quelli per cui non esiste nessun veicolo.

```
Select Categorie.*, Veicoli.*  
From Categorie LEFT [OUTER] JOIN Veicoli  
ON Categorie.cod_cat=Veicoli.Cod_cat
```

```
Select Categorie.*, Veicoli.*  
From Veicoli RIGHT [OUTER] JOIN Categorie  
ON Categorie.cod_cat=Veicoli.Cod_cat
```

Join esterno: "outer join"

- Per ogni persona, elencare il padre e, se nota, la madre.

```
SELECT paternita.figlio, padre, madre  
FROM paternita LEFT JOIN maternita  
ON paternita.figlio = maternita.figlio
```

```
SELECT paternita.figlio, padre, madre  
FROM paternita LEFT OUTER JOIN maternita  
ON paternita.figlio = maternita.figlio
```

Maternità

MADRE	FIGLIO
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Join Esterno/Outer Join

L'operatore di OUTER JOIN può essere applicato usando la seguente sintassi:

- {LEFT | RIGHT | FULL} [OUTER] JOIN + ON <predicato>
- {LEFT | RIGHT | FULL} [OUTER] JOIN + USING <colonne>
- dove "outer" è opzionale

Esercizi

Libro(CodiceLibro, ISBN, Titolo, NomeAutore, CognomeAutore, Pagine, anno)

1. Il numero di libri scritti da Andrea Camilleri prima del 2010
2. Massimo numero di pagine fra i libri presenti nella tabella
3. Il massimo numero di pagine fra quelle dei libri scritti da Andrea Camilleri
4. Per ogni autore, il massimo numero di pagine dei suoi libri
5. I nomi e cognomi degli autori che hanno scritto più di 8 libri
6. Ordinare i libri scritti da Camilleri in ordine di anno decrescente e quelli dello stesso anno in ordine lessicografico per titolo
7. Per ogni anno, il numero di libri pubblicati
8. Per ogni autore il numero dei libri scritti ogni anno da quell'autore