

# Laboratorio di Basi di Dati

Turni T3 e T4

a.a. 2018/2019

Ruggero Pensa - Fabiana Venero

# In questa lezione

- SQL come DML:
  - > Funzioni aggregate
  - > Interrogazioni con raggruppamento

# Database di esempio

**S**

<u>SNum</u>	SName	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

S=Supplier (fornitore)  
P=Parts (parti)  
QTY=quantity

**P**

<u>PNum</u>	PName	Color	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

**SP**

<u>SNum</u>	<u>PNum</u>	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

nut=dado, bolt=bullone, screw=vite, cam=camma, cog=ruota dentata

# Funzioni aggregate - 1

- Finora abbiamo considerato esclusivamente condizioni in cui i predicati vengono valutati sulle singole tuple, indipendentemente le une dalle altre.
- In SQL è possibile valutare proprietà e condizioni su gruppi di righe.
  - Le *funzioni aggregate* prendono in considerazione gruppi di righe e restituiscono un (unico) valore per ogni gruppo di righe.
  - Le funzioni aggregate più usate sono quelle di **conteggio, somma, massimo, minimo**.

# Funzioni aggregate - 2

- **Esempio:** contare il numero di parti rosse.

> Posso usare la funzione aggregata **count** in questo modo:

```
select count(*)  
from P  
where Color='Red';
```

COUNT(*)
3

- **select count(\*)** conta il numero di righe (compresi eventuali valori nulli).
- Posso ovviamente anche rinominare la colonna:

```
select count(*) RedPartCount  
from P  
where Color='Red';
```

RedPartCount
3

# Funzioni aggregate - 3

- **Esempio:** contare il numero di città in cui vi sono fornitori.

```
select count(City) NCities  
from S;
```

NCities
5

- **select count(Attr)** conta il numero di valori non nulli dell'attributo **Attr**.
- Per contare valori distinti posso usare la parola chiave **distinct**:

```
select count(distinct City) NCities  
from S;
```

NCities
3

# Funzioni aggregate - 4

- Posso anche contare i valori su più attributi.
- Esempio:** estrarre il numero di nomi (distinti) dei fornitori e il numero di città (distinte).

```
select count(distinct SName) NNames,  
       count(distinct City) NCities  
from S;
```

NNames	NCities
5	3

# Funzioni aggregate - 5

- Funzioni disponibili:

- > **count()**: conteggio degli elementi della colonna *attr*
  - **count(\*)**: conteggio delle righe di una tabella
  - **count(attr)**: conteggio dei valori non nulli di *attr*
- > **sum(attr)**: somma dei valori nella colonna *attr*
- > **avg(attr)**: media dei valori della colonna *attr*
  - **sum** e **avg** si applicano solo a valori numerici
- > **max(attr)**: massimo valore nella colonna *attr*
- > **min(attr)**: minimo valore nella colonna *attr*

- I DBMS forniscono di solito diverse altre funzioni (es. deviazione standard)

- L'argomento della funzione può essere preceduto da **distinct**

- > (PostgreSQL non supporta **count(distinct \*)**).



# Funzioni aggregate - 6

- **Esempio 1:** estrarre la quantità totale di parti *P2* fornite.

```
select sum(QTY)
from SP
where PNum='P2';
```

SUM(QTY)
1000

- **Esempio 2:** estrarre il primo fornitore in ordine alfabetico.

```
select min(SName)
from S;
```

MIN(SNAME)
Adams

# Funzioni aggregate - 7

- È possibile utilizzare più funzioni anche sullo stesso attributo.
  - > **Esempio:** estrarre la quantità media, minima e massima di tutte le forniture

```
select avg(QTY), min(QTY), max(QTY)
from SP;
```

AVG(QTY)	MIN(QTY)	MAX(QTY)
258,333333333	100	400

# Funzioni aggregate - 8

- Una funzione aggregata viene valutata su un **gruppo di righe** e restituisce un **unico valore per ogni gruppo**.
  - **Conseguenza 1**: Non è possibile usare funzioni aggregate (che considerano gruppi di righe) direttamente nella **clausola where** (che viene valutata per ogni riga).

```
select SName  
from S
```

```
where Status=max (Status) ;
```



**errore!**

# Funzioni aggregate - 9

- Una funzione aggregata viene valutata su un **gruppo di righe** e restituisce un **unico valore per ogni gruppo**.
  - **Conseguenza 2:** Non è possibile combinare funzioni aggregate (che restituiscono un unico valore) con attributi non aggregati che possono assumere valori diversi nello stesso gruppo.

```
select SName, max(Status)  
from S;
```



**errore!**

# Funzioni aggregate - 10

- SQL non gestisce la possibile differenza di cardinalità tra le funzioni aggregate e le altre espressioni.
- Per evitare possibili errori, query di questo tipo vengono considerate dai DBMS scorrette *a priori*.

# Query con raggruppamento - 1

- È possibile suddividere le righe di una tabella in più gruppi e applicare la funzione aggregata a ogni gruppo.
  - **Ad esempio:** non mi interessa il numero totale di fornitori, ma il numero di fornitori per ogni città.
  - Per estrarre questo tipo di informazioni devo **raggruppare** le righe secondo l'attributo **City**.

# Query con raggruppamento - 2

- Esempio: estrarre il numero di fornitori con status maggiore o uguale a 10 di ogni città.
  - > Attributo discriminante: **City**.
  - > Attributo della funzione aggregata: **SNum** (oppure **\***).
  - > Query:

```
select City, count(SNum)
from S
where Status >= 10
group by City;
```

City	COUNT(SNum)
Athens	1
London	2
Paris	2

# Query con raggruppamento - 3

## ● Funzionamento:

1. Viene prima eseguita la query come se non ci fossero né la clausola **group by** né la funzione aggregata.

```
select City, SNum  
from S  
where Status>=10;
```

City	SNum
London	S1
Paris	S2
Paris	S3
London	S4
Athens	S5



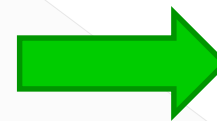
# Query con raggruppamento - 4

## ● Funzionamento:

2. Le righe vengono raggruppate in base ai valori dell'attributo nella clausola **group by**.

`group by City`

City	SNum
London	S1
Paris	S2
Paris	S3
London	S4
Athens	S5



City	SNum
Athens	S5
London	S1
London	S4
Paris	S2
Paris	S3

# Query con raggruppamento - 5

## ● Funzionamento:

3. Viene ora considerata la clausola **select** e la funzione aggregata viene calcolata sull'attributo **SNum** per ogni gruppo nella tabella.

```
select City, count(SNum)
```

City	SNum
Athens	S5
London	S1
London	S4
Paris	S2
Paris	S3




City	COUNT(SNum)
Athens	1
London	2
Paris	2

# Query con raggruppamento - 6

- Cosa succede se considero un attributo diverso da City nella select?

```
select SName, count(SNum)
from S
where Status >= 10
group by City;
```

 errore!

City	SName	SNum
Athens	Adams	S5
London	Smith	S1
London	Clark	S4
Paris	Jones	S2
Paris	Blake	S3



SName	COUNT(SNum)
Adams	1
?	2
?	2

# Query con raggruppamento - 7

- La sintassi SQL-92 è:

```
select SottinsiemeAttributiDiscriminanti,  
       FunzioneAggregata(AltroAttributo)  
from Tabelle  
where Condizione  
group by AttributiDiscriminanti;
```

- **La target list può comprendere esclusivamente attributi discriminanti e funzioni aggregate su attributi non discriminanti.**
  - > Dato che ogni gruppo ha lo stesso valore per gli attributi discriminanti, la funzione aggregata, che calcola un solo valore per ogni gruppo, non dà problemi di differenza di cardinalità.
  - > Una query con funzioni aggregate che non rispetta questa sintassi viene rifiutata dal parser SQL indipendentemente dalle cardinalità effettive.
  - > In altre parole, si deve considerare la sintassi dell'intero costrutto e non si può semplicemente aggiungere la clausola **group by** a una **select** espressa in qualsiasi forma.

# Query con raggruppamento - 8

- Esempio: la query seguente è scorretta:

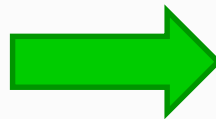
```
select City, Status, count(SNum)
from S
group by City;
```



**errore!**

- Infatti, sebbene **count (SNum)** restituisca un unico valore, per ogni città si possono avere valori diversi per Status.

City	Status	SNum
Athens	30	S5
London	20	S1
London	20	S4
Paris	10	S2
Paris	30	S3



City	Status	COUNT(SNum)
Athens	30	1
London	<b>20</b>	2
Paris	<b>?</b>	2

# Query con raggruppamento - 9

- È possibile considerare più di un attributo per formare raggruppamenti più fini
- Esempio:** estrarre il numero di fornitori per città e status

```
select City, Status, count(SNum)
from S
where Status >= 10
group by City, Status;
```

City	Status	COUNT(SNum)
Paris	10	1
Paris	30	1
Athens	30	1
London	20	2

- L'ordine degli attributi nella clausola **group by** non è importante (a differenza della **order by**).

# Passi di esecuzione

City	Status	SNum
London	20	S1
Paris	10	S2
Paris	30	S3
London	20	S4
Athens	30	S5

from S  
where Status >= 10

City	Status	SNum
London	20	S1
Paris	10	S2
Paris	30	S3
London	20	S4
Athens	30	S5

group by City, Status

City	Status	SNum
Paris	10	S2
Paris	30	S3
Athens	30	S5
London	20	S1
London	20	S4

select City,  
Status,  
count (SNum)

City	Status	count (SNum)
Paris	10	1
Paris	30	1
Athens	30	1
London	20	2

# Query con raggruppamento - 10

```
select Attr1, Attr2, sum(Attr3)
from Tab
group by Attr1, Attr2;
```

Attr1	Attr2	Attr3
A	X	2
A	X	3
A	Y	3
A	Y	7
B	X	4
B	X	1
B	Z	5
B	Z	2

Attr1	Attr2	Attr3
A	X	2
A	X	3
A	Y	3
A	Y	7
B	X	4
B	X	1
B	Z	5
B	Z	2

Attr1	Attr2	Attr3
A	X	2
A	Y	3
A	X	3
A	Y	7
B	Z	5
B	X	4
B	Z	2
B	X	1

Attr1	Attr2	Attr3
A	X	2
A	Y	3
B	Z	5
A	X	3
B	X	4
A	Y	7
B	Z	2
B	X	1

Attr1	Attr2	SUM(Attr3)
A	X	5
A	Y	10
B	X	5
B	Z	7



# Query con raggruppamento - 11

- Nota: SQL-99 attenua la restrizione sulla sintassi della select con **group by** rendendo possibile includere nella **target list**, al di fuori delle funzioni aggregate, anche attributi funzionalmente dipendenti da quelli discriminanti

- Quindi in SQL-99 è possibile scrivere

```
select PName, sum(Qty)
from SP join P on SP.Pnum=P.Pnum
group by P.PNum;
```

per ricavare il nome e il numero totale di parti fornite

- > PostgreSQL supporta la sintassi SQL-99 permettendo, come nell'esempio, di avere nella **target list**, al di fuori delle funzioni aggregate, attributi la cui chiave primaria è tra gli attributi discriminanti, Oracle supporta esclusivamente SQL-92, MySQL supporta SQL-99 solo recentemente (da 5.7.5)

# Predicati sui gruppi - 1

- È possibile specificare condizioni che devono valere, anziché sulle righe, su **gruppi** di righe.
  - > **Caso 1**: estrarre il numero di fornitori per città con status almeno uguale a 20.
  - > **Caso 2**: elencare le città con almeno due fornitori.

# Predicati sui gruppi - 2

- **Caso 1:** estrarre il numero di fornitori per città con status di almeno 20

> Si può controllare la condizione riga per riga, quindi si usa una semplice clausola **where**:

```
select City, count(SNum)
from S
where Status >= 20
group by City;
```

City	COUNT(SNum)
Athens	1
London	2
Paris	1

# Predicati sui gruppi - 3

- **Caso 2:** elencare le città con almeno due fornitori.
  - > Si può controllare la condizione solo sul risultato della funzione aggregata, non su singole righe!
  - > Si introduce la clausola **having**.

```
select City
from S
group by City
having count(*) >= 2;
```

City
London
Paris

- La clausola **having** viene verificata al termine del raggruppamento eseguito con la **group by**.

# Predicati sui gruppi - 4

- È importante comprendere la differenza tra i predicati della clausola **where** e quelli della clausola **having**: nella clausola **having** generalmente vanno messe solo condizioni in cui compaiono **funzioni aggregate**.
  - **Esempio:** elencare le città con almeno due fornitori con status almeno uguale a 20.

# Predicati sui gruppi - 5

- Esempio: elencare le città con almeno due fornitori con status almeno uguale a 20.

```
select City
from S
where Status >= 20
group by City
having count(*) >= 2;
```

City
London