

Le Viste

- SQL permette di specificare tabelle virtuali in cui le righe non sono esplicitamente memorizzate nella base di dati, ma sono calcolate quando necessario: le Viste.
- Le viste vengono definite associando un nome ed una lista di attributi al risultato dell'esecuzione di un'interrogazione.
- L'interrogazione interna (che può contenere anche altre viste) deve restituire un insieme di attributi pari a quelli contenuti nello schema della vista, nello stesso ordine.
- ➤ Una vista è una "relazione di cui viene memorizzata solo la definizione, piuttosto che l'insieme delle tuple".

Le Viste

Create view NomeVista [(ListaAttributi)] as SelectSQL [with [local | cascaded] check option]

create view ImpiegatiAmmin
(Nome, Cognome, Stipendio) as
select Nome, Cognome, Stipendio
from Impiegato
where Dipart = 'Amministrazione' and
Stipendio > 10

- La vista è una tabella virtuale il cui contenuto dipende dal contenuto della altre tabelle. Vengono definite associando un nome e una lista di attributi al risultato di una interrogazione.
- Esistono viste virtuali (quelle supportate da tutti I sistemi moderni) e materializzate (non supportate) che invece salvano effettivamente il risultato della query in una tabella.
- Il problema è che non è semplice né efficiente mantenere l'allineamento, mentre le interrogazione sono piu efficienti.

Le Viste

➤ Possono fare riferimento alle viste come se fossero relazioni di base

select * from ImpiegatiAmmin

equivale a (e viene eseguita come)

select Nome, Cognome, Stipendio from Impiegato where Dipart = 'Amministrazione' and Stipendio > 10

Le Viste

CREATE VIEW NomeVista [(ListaAttributi)]
AS SelectSQL

[with check option] politica di inserimento controllato

ESEMPIO: A partire dalle Tabelle Impiegato (Nome, Cognome, Dipart, StipAnn) e Dipartimento (Nome, Città) Definire una vista **ImpiegatiAmmin** che contiene tutti gli impiegati del dipartimento Amministrazione con uno stipendio superiore a 10 mila euro

CREATE VIEW ImpiegatiAmmin (Nome, Cognome, Dipart, StipAnn) as SELECT Nome, Cognome, Dipart, StipAnn FROM Impiegato

WHERE Dipart = 'Amministrazione' AND StipAnn > 10

Le Viste

CREATE VIEW NomeVista[(ListaAttributi)]
AS SelectSQL
[with check option]

ESEMPIO: A partire dalle Tabelle Impiegato (Nome, Cognome, Dipart, StipAnn) e Dipartimento (Nome, Città), definire una vista ImpiegatiAmminPoveri definita a partire dalla vista ImpiegatiAmmin, che contiene tutti gli impiegati amministrativi con uno stipendio compreso tra 10 mila e 50 mila euro

CREATE VIEW ImpiegatiAmminPoveri as

SELECT *

FROM ImpiegatiAmmin WHERE StipAnn < 50

Viste Aggiornabili

- SQL permette di aggiornare solo quelle viste definite su una sola tabella di base e che per la loro definizione non usano operatori di aggregazione o il comando distinct.
- ➤ Queste restrizioni relative alla politica aggiornamenti sono FONDAMENTALI per rendere l'aggiornamento alla tabella sottostante non ambiguo.
- A ciascuna riga della tabella di base corrisponderà una sola riga della vista (le viste mantengono i duplicati).
- ➤ Un'operazione di modifica\cancellazione nella vista deve soddisfare tutti i vincoli di integrità della tabella sottostante, altrimenti non viene eseguita.

Viste Aggiornabili

CREATE VIEW ImpiegatiAmmin (Nome, Cognome, Dipart, StipAnn) as SELECT Nome, Cognome, Dipart, StipAnn FROM Impiegato

WHERE Dipart = 'Amministrazione' AND StipAnn > 10

INSERT INTO ImpiegatiAmmin (Nome, Cognome, Dipart, StipAnn) VALUES('Marco', 'Grigi', 'Amministrazione', '40')

La tupla ('Marco', 'Grigi', 'Amministrazione', '40') viene inserita correttamente nella vista ImpiegatiAmmin e anche nella relazione sottostante Impiegato.

Viste Aggiornabili

CREATE VIEW ImpiegatiAmmin (Nome, Cognome, Dipart, StipAnn) as SELECT Nome, Cognome, Dipart, StipAnn FROM Impiegato

WHERE Dipart = 'Amministrazione' AND StipAnn > 10

INSERT INTO ImpiegatiAmmin (Nome, Cognome, Dipart, StipAnn) VALUES('Marco', 'Grigi', 'Amministrazione', '7')

La tupla ('Marco', 'Grigi', 'Amministrazione', '7') viene inserita correttamente nella relazione sottostante Impiegato, ma non nella vista ImpiegatiAmmin, che non accetta Stipendi < 10.

Viste Aggiornabili

CREATE VIEW ImpiegatiAmmin (Nome, Cognome, Dipart, StipAnn) as SELECT Nome, Cognome, Dipart, StipAnn FROM Impiegato

WHERE Dipart = 'Amministrazione' AND StipAnn > 10

INSERT INTO ImpiegatiAmmin (Nome, Cognome, Dipart, StipAnn) VALUES('Marco', 'Grigi', 'Amministrazione', '7')

WITH CHECK OPTION

La tupla ('Marco', 'Grigi', 'Amministrazione', '7') NON viene inserita nella vista ImpiegatiAmmin, che non accetta Stipendi < 10 e NEPPURE nella relazione sottostante Impiegato.

Viste Aggiornabili

CREATE VIEW NomeVista[(ListaAttributi)]
AS SelectSQL
[with check option]

ESEMPIO: A partire dalla Tabella Dipartimento (Nome, Città), Definire una vista DipartimentiMilano che contiene tutti i nomi dei dipartimenti che si trovano a Milano, con una politica di inserimento controllato

CREATE VIEW DipartimentiMilano(Nome) as

SELECT Nome FROM Dipartimento WHERE Città='Milano' with check option

Viste Aggiornabili

Data la Tabella Dipartimento (Nome, Città)

CREATE VIEW DipartimentiMilano(Nome) as

SELECT Nome FROM Dipartimento WHERE Città='Milano' with check option

INSERT INTO DipartimentiMilano (Nome) VALUES ('Sviluppo')

L'inserimento fallisce, perché la riga verrebbe correttamente inserita all'interno di Dipartimento (con il valore di Città posto a NULL), ma tale risultato non sarebbe visibile in DipartimentiMilano (data la presenza di with check option nella creazione della vista).

Viste e Interrogazioni

- Le viste in SQL possono anche servire per formulare interrogazioni che non sarebbero altrimenti esprimibili, aumentando il potere espressivo del linguaggio.
- Ad esempio, permettono di definire in SQL query che richiedono di utilizzare diversi operatori aggregati in cascata, diversamente non utilizzabili in quanto la sintassi di SQL non permette di combinare in cascata la valutazione di diversi operatori aggregati.
- ➤ ES: Estrarre il numero medio di Dipartimenti per città

 SELECT AVG(COUNT'nome')

 FROM Dipartimento

 GROUP BY Città

 NO !!!

Viste e Interrogazioni

SESTITUTE Estrarre il numero medio di Dipartimenti per ogni città

Data la Tabella Dipartimento (Nome, Città)

CREATE VIEW DipartCittà (NomeCittà, NroDipart) as

SELECT Città, count(Nome)
FROM Dipartimento
GROUP BY Città

SELECT AVG(NroDipart) FROM DipartCittà

Viste e Interrogazioni

 Estrarre il dipartimento caratterizzato dal massimo della somma per gli stipendi

Data la Tabella Dipartimento (Nome, Città) e la Tabella Impiegato (Nome, Cognome, Dipart, Stipendio)

CREATE VIEW BudgetStipendi(Dip, TotaleStipendi) as

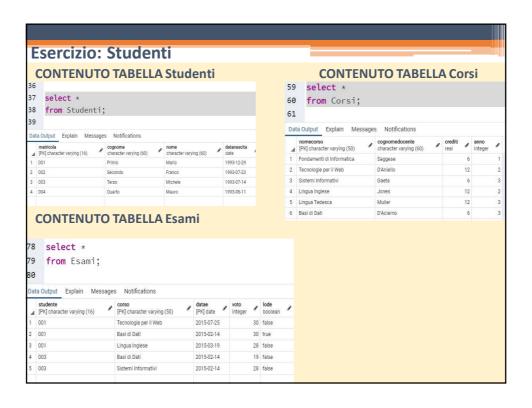
SELECT Dipart, sum(Stipendio)
FROM Impiegato
GROUP BY Dipart

SELECT Dip FROM BudgetStipendi

WHERE TotaleStipendi = (SELECT max (TotaleStipendi) FROM BudgetStipendi)

Viste Aggiornabili

- Le viste possono essere usate per presentare le informazioni necessarie (o un loro riassunto), nascondendo al contempo i dettagli delle relazioni sottostanti.
- ➤I comandi GRANT o REVOKE possono essere usati per controllare l'accesso alle relazioni e alle viste.
- Le viste possono essere interrogate proprio come relazioni ordinarie, ma sono consentite solo forme limitate di aggiornamento.





Operatori di Aggregazione

Calcolare la media dei Voti dello Studente Mario Primo

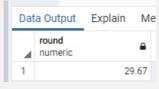
Ovviamente dobbiamo collegare le due Tabelle Studenti ed Esami con un JOIN sulla Matricola.

Esercizio: Studenti

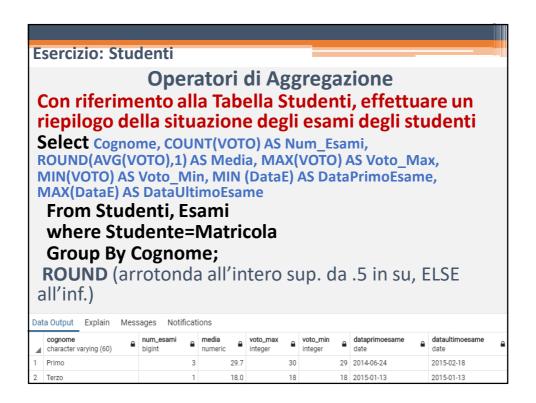
Calcolare la media dei Voti dello Studente Mario Primo Select matricola, AVG(Voto) From Esami, Studenti Where (Studente=Matricola) AND Cognome='Primo' AND Nome='Mario';



Select ROUND ((AVG(Voto)),2) From Esami, Studenti Where (Studente=Matricola) AND Cognome='Primo' AND Nome='Mario';



Operatori di Aggregazione Con riferimento alla Tabella Studenti, effettuare un riepilogo della situazione degli esami degli studenti Numero Esami; Media; Voto Max; Voto Min; Data primo esame; Data secodo esame.



Operatori di Aggregazione

Calcolare la media ponderata dei Voti dello Studente Mario Primo

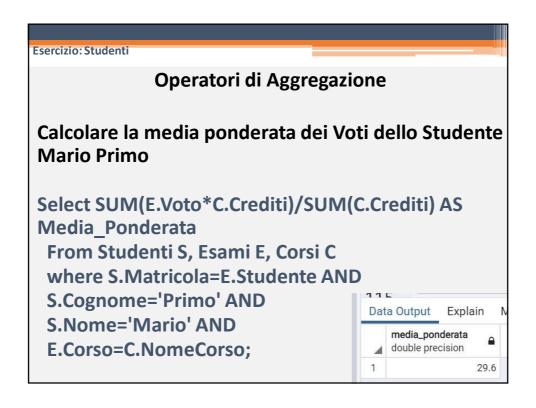
Quindi SUM(Voto ei x Crediti ei) / SUM(Crediti ei)

Esercizio: Studenti

Operatori di Aggregazione

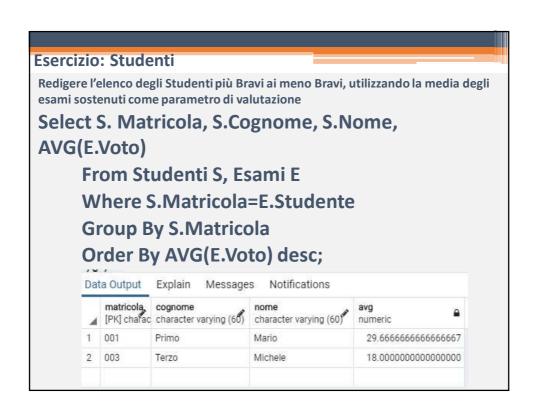
Calcolare la media ponderata dei Voti dello Studente Mario Primo

Ovviamente dobbiamo collegare le tre Tabelle Studenti, Esami sulla Matricola selezionare Mario Primo e ancora in JOIN con Corsi sul nome del corso.



ATTENZIONE Provare il seguente comando: Select 5/2; Come potrete osservare il risultato è: 2 e non 2,5 come ci aspettavamo. Se dividiamo due interi il risultato è ancora un intero, per avere il risultato corretto, almeno uno dei due operandi deve essere in virgola mobile. La Conversione di un'espressione verso un altro dominio prende il nome di CAST e si ottiene mediante la seguente funzione **CAST (Espressione AS Dominio)** Select CAST(SUM(E.Voto*C.Crediti) AS Real)/SUM(C.Crediti) AS Media Ponderata From Studenti S, Esami E, Corsi C where S.Matricola=E.Studente AND S.Cognome='Primo' AND Data Output Explain S.Nome='Mario' AND media_ponderata E.Corso=C.NomeCorso; real 29.6

Redigere l'elenco degli Studenti più Bravi ai meno Bravi, utilizzando la media degli esami sostenuti come parametro di valutazione



Redigere l'elenco degli Studenti più Bravi ovvero con Voto di Media > 26

Esercizio: Studenti

Redigere l'elenco degli Studenti più Bravi ovvero con Voto di Media > 26

Spunto: dobbiamo prima realizzare le partizioni degli esami degli Studenti e poi sulla media di ogni singola partizione effettuare il controllo.

Redigere l'elenco degli Studenti più Bravi ovvero con Voto di Media > 26

Select S. Matricola, S.Cognome, S.Nome, AVG(E.Voto)

From Studenti S, Esami E
Where S.Matricola=E.Studente
Group By S.Matricola
HAVING AVG(E.Voto) >26;

Esercizio: Studenti

Determinare gli esami di Basi di Dati svolti nel gennaio 2015

Determinare gli esami di Basi di Dati svolti nel gennaio 2015

Select E.Corso, E.DataE from Esami E where E.Corso='Basi di Dati' and E.DataE between '2015-01-01' and '2015-01-30';

Esercizio: Studenti

Elencare i Corsi, stampando in caratteri MAIUSCOLI i corsi da 12 crediti, in caratteri MINUSCOLI i corsi da 6 crediti, lasciando inalterati gli altri.

Elencare i Corsi, stampando in caratteri

MAIUSCOLI i corsi da 12 crediti, in caratteri MINUSCOLI i corsi da 6 crediti, lasciando inalterati gli altri

Select C.Cognomedocente, CASE C.Crediti when 12 then UPPER(C.NomeCorso) when 6 then LOWER(C.NomeCorso)

ELSE C.NomeCorso

END

From Corsi C

Order by C.Cognomedocente; 4 Jones Muller

 Data Output
 Explain
 Messages
 Notifications

 cognomedocente character varying (60)
 nomecorso character varying
 nomecorso character varying

 1
 D'Acierno
 basi di dati

 2
 D'Aniello
 TECNOLOGIE PER IL W...

 3
 Gaeta
 sistemi informativi

 4
 Jones
 LINGUA INGLESE

 5
 Muller
 LINGUA TEDESCA

 6
 Saggese
 fondamenti di informat...

Esercizio: Ancora su CASE

Obiettivo

Elencare Matricola, Cognome e Media di tutti gli Studenti, ma con il seguente vincolo

IF AVG(E.Voto)>28 then 'Eccellente'

IF AVG(E.Voto)>26 then 'Ottimo'

IF AVG(E.Voto)>24 then 'Buono'

IF AVG(E.Voto)>22 then 'Dicreto'

IF AVG(E.Voto) IS NULL then 'Nessun Esame Svolto'

Esercizio: Ancora su CASE Select S.Matricola, S.Cognome, CASE when AVG(E.Voto)>28 then 'Eccellente' when AVG(E.Voto)>26 then 'Ottimo' when AVG(E.Voto)>24 then 'Buono' when AVG(E.Voto)>22 then 'Dicreto' when AVG(E.Voto) IS NULL then 'Nessun Esame Svolto' **ELSE 'Sufficiente' FND** from Studenti S LEFT JOIN Esami E ON S.Matricola=E.Studente Group by S.Matricola order by S.Matricola; Data Output Explain Messages Notifications matricola cognome case [PK] character varying (16) character varying (60) text Eccellente 001 002 Secondo Nessun Esame Svolto 4 004 Ouarto Nessun Esame Svolto

Ancora su CASE

- L'espressione CASE ha due formati supportano un argomento facoltativo ELSE :
 - L'espressione CASE semplice confronta un'espressione con un set di espressioni semplici per determinare il risultato.
 - L'espressione CASE avanzata valuta un set di espressioni booleane per determinare il risultato.
- L'espressione CASE può essere utilizzata in qualsiasi istruzione o clausola che consenta un'espressione valida. È possibile, ad esempio, utilizzare CASE in istruzioni quali SELECT, UPDATE, DELETE e SET e in clausole quali select_list, IN, WHERE, ORDER BY e HAVING
- Simple CASE expression:

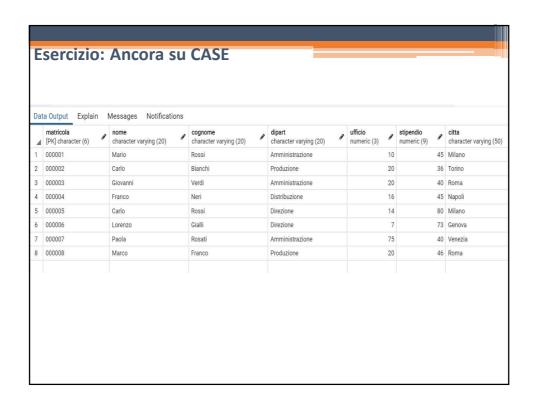
```
CASE input expression WHEN
     when expression THEN result expression [...n]
     [ ELSE else_result_expression ]
END
```

Searched CASE expression:

END

```
CASE
      WHEN Boolean_expression THEN result_expression [ ...n ]
      [ ELSE else_result_expression ]
```

```
Esercizio: Ancora su CASE
/* Creare una tabella DIPARTIMENTI */
CREATE TABLE DIPARTIMENTI (
       CODICE CHAR(4) NOT NULL,
       NOME VARCHAR(20) NOT NULL UNIQUE,
       DIP_IND varchar (50),
       DIP CITTA varchar(20),
       CONSTRAINT PK_DIPARTIMENTI PRIMARY KEY (CODICE) );
/* Creare una tabella IMPIEGATI */
CREATE TABLE IMPIEGATI (
       matricola char(6) primary key,
       nome varchar(20),
       cognome varchar (20),
       dipart varchar (20) references DIPARTIMENTI(nome),
       ufficio numeric (3),
       stipendio numeric (9) default 0,
       citta character varying(50),
       unique (cognome, nome));
```



Ancora su CASE ...

Select Matricola, Nome, Cognome,

CASE

when dipart='Amministrazione' then 'Impiegato di Concetto' when dipart='Produzione' then 'Operaio' when dipart='Ricerca' then 'Ricercatore' when dipart='Distribuzione' then 'Autista'

ELSE 'Direzione'

END as Ruolo

from Impiegati;

4	matricola [PK] character (6)	nome character varying (20)	cognome character varying (20)	ruolo text
1	000001	Mario	Rossi	Impiegato di Concetto
2	000002	Carlo	Bianchi	Operaio
3	000003	Giovanni	Verdi	Impiegato di Concetto
4	000004	Franco	Neri	Autista
5	000005	Carlo	Rossi	Direzione
6	000006	Lorenzo	Gialli	Direzione
7	000007	Paola	Rosati	Impiegato di Concetto
8	000008	Marco	Franco	Operaio

Ancora su CASE ... Le insiemistiche

Select matricola, nome, cognome, 'Impiegato di Concetto' as Ruolo

from Impiegati where Dipart='Amministrazione'

UNION

Select matricola, nome, cognome, 'Operaio' as Ruolo

from Impiegati where Dipart='Produzione'

UNION

Select matricola, nome, cognome, 'Ricercatore' as Ruolo

from Impiegati where Dipart='Ricerca'

Select matricola, nome, cognome, 'Autista' as Ruolo

from Impiegati where Dipart='Distribuzione'

UNION

Select matricola, nome, cognome, 'Direzione' as Ruolo

from Impiegati where (Dipart<>'Amministrazione') AND (Dipart<>'Produzione')

AND (Dipart<>'Ricerca') AND (Dipart<>'Distribuzione');

4	matricola character (6)	nome character varying (20)	cognome character varying (20)	ruolo text
1	000004	Franco	Neri	Autista
2	000001	Mario	Rossi	Impiegato di Concetto
3	000002	Carlo	Bianchi	Operaio
4	000008	Marco	Franco	Operaio
5	000005	Carlo	Rossi	Direzione
6	000003	Giovanni	Verdi	Impiegato di Concetto
7	000006	Lorenzo	Gialli	Direzione
8	000007	Paola	Rosati	Impiegato di Concetto

Funzioni condizionali

≻Coalesce

Valuta gli argomenti seguendo l'ordine e restituisce il valore corrente della prima espressione che inizialmente non restituisce NULL.

Ad esempio, SELECT COALESCE(NULL, NULL, 'third_value', 'fourth_value'); restituisce il terzo valore perché il terzo valore è il primo non Null

≻Nullif

Restituisce un valore Null se le due espressioni specificate sono uguali.

Ad esempio, SELECT NULLIF(4,4) AS Same, NULLIF(5,7) AS Different:

restituisce NULL per la prima colonna (4 e 4) perché i due valori di input sono uguali.

La seconda colonna restituisce il primo valore (5) perché i due valori di input sono diversi.

BASI DI DATI

Materiale utilizzato e bibliografia

- > Le slide utilizzate dai docenti per le attività frontali sono in gran parte riconducibili e riprese dalle slide originali (con alcuni spunti parziali ripresi dai libri indicati) realizzate da:
- √ autori del libro Basi di Dati (Atzeni e altri) testo di riferimento del corso Basi di Dati e sono reperibili su internet su
 molteplici link oltre che laddove indicato dagli stessi autori del libro;
- ✓ Prof.ssa Tiziana Catarci e dal dott. Ing. Francesco Leotta corso di Basi di Dati dell'Università degli Studi La Sapienza di Roma al seguente link ed altri: http://www.dis.uniroma1.it/~catarci/basidatGEST.html (molto Interessanti anche le lezioni su YouTube).
- ✓ Proff. Luca Allulli e Umberto Nanni, Libro Fondamenti di basi di dati, editore HOEPLI (testo di facile lettura ed efficace).
- Diverse slide su specifici argomenti utilizzate dai docenti per le attività frontali sono anche in parte riconducibili e riprese dalle slide originali – facilmente reperibili e accessibili su internet - realizzate da:

Prof.ssa Roberta Aiello – corso Basi di Dati dell'Università di Salerno

Prof. Dario Maio - corso Basi di Dati dell'Università di Bologna al seguente link ed altri: <u>http://bias.csr.unibo.it/maio</u>

Prof. Marco Di Felice - corso Basi di Dati dell'Università di Bologna al seguente link ed altri: http://www.cs.unibo.it/difelice/dbsi/

Prof Marco Maggini e prof Franco Scarselli - corso Basi di Dati dell'Università di Siena ai seguenti link ed altri: http://staff.icar.cnr.it/pontieri/didattica/LabSI/lezioni/_preliminari-DB1%20(Maggini).pdf

Prof.ssa Raffaella Gentilini - corso Basi di Dati dell'Università di Perugia al seguente link ed altri: http://www.dmi.unipg.it/raffaella.gentilini/BD.htm

Prof. Enrico Giunchiglia - corso Basi di Dati dell'Università di Genova al seguente link ed altri: http://www.star.dist.unige.it/~enrico/BasiDiDati/

Prof. Maurizio Lenzerini - corso Basi di Dati dell'Università degli Studi La Sapienza di Roma al seguente link ed altri http://didatticainfo.altervista.org/Quinta/Database2.pdf

- The PostgreSQL Global Development Group PostgreSQL nn.xx Documentation
- > PostgreSQL (appendice scaricabile dal sito del libro (area studenti) e www.postgresql.org