Laurea in Informatica
A.A. 2021-2022

Corso "Base di Dati"

Concetti Avanzati di SQL

Prof. Massimiliano de Leoni

Università degli Studi di Padova

Vincoli di integrità generici: CHECK

- Specifica di vincoli:
  - di tupla
  - fra tuple della stesso o diversa relazione (quasi mai supportati)

CHECK (Condizione)

Check, esempio

```
CREATE TABLE Imp
(
Matricola INTEGER PRIMARY KEY,
Nome CHARACTER(20),
Eta INTEGER CHECK (Eta > 16 AND Eta<100),
Stipendio INTEGER CHECK (Stipendio > 0) ,
Capo INTEGER,
CHECK (Stipendio <= (SELECT Stipendio
                     FROM Imp J
                     WHERE Capo = J.Matricola) )
)
```

Non supportato da quasi tutti i DBMS

Matricola	Nome	Età	Stipendio	Capo
7309	Rossi	34	45	5698
5998	Bianchi	37	38	5698
9553	Neri	42	35	4076
5698	Bruni	43	42	4076
4076	Mori	45	50	8123

Check, esempio

```
CREATE TABLE Imp
(
Matricola INTEGER PRIMARY KEY,
Nome CHARACTER(20),
Eta INTEGER CHECK (Eta > 16 AND Eta<100),
Stipendio INTEGER CHECK (Stipendio > 0) ,
Capo INTEGER,
CHECK (Stipendio <= (SELECT Stipendio
                     FROM Imp J
                     WHERE Capo = J.Matricola) )
)
```

Non supportato da quasi tutti i DBMS

Matricola	Nome	Età	Stipendio	Capo
7309	Rossi	34	45	5698
5998	Bianchi	37	38	5698
9553	Neri	42	35	4076
5698	Bruni	43	42	4076
4076	Mori	45	50	8123

Check, Esempio che funziona!

```
CREATE TABLE Imp
(
      Matricola INTEGER PRIMARY KEY,
      Nome CHARACTER(20),
      Eta INTEGER CHECK (Eta > 16 AND Eta<100),
      Stipendio INTEGER CHECK (Stipendio > 0) ,
      Capo INTEGER,
      Ritenute INTEGER,
      Netto INTEGER,
      CHECK (Stipendio =Netto+Ritenute )
)
```

Si assume che Stipendio sia il lordo

Vincoli di integrità generici: asserzioni

- Specifica vincoli a livello di schema

```
CREATE ASSERTION NomeAss
   CHECK ( Condizione )

CREATE ASSERTION AlmenoUnImpiegato
     CHECK (1 <= (       SELECT COUNT(*)
                         FROM Impiegato ))
```

- Raramente supportato per ora

Viste

- Una vista è rappresentata da una query (SELECT)
- Il risultato può essere utilizzato come se fosse una tabella.
- Le viste generalmente vengono utilizzate per semplificare le query.
- Le viste possono essere aggiornate via INSERT, UPDATE e DELETE.

CREATE VIEW NomeVista [(ListaAttributi)]
AS SelectSQL

Viste: Esempio

```
CREATE VIEW ImpiegatiNonCapo(Nome, Eta, Stipendio) AS
    SELECT Nome, Eta, Stipendio
    FROM Impiegato
    WHERE Nome NOT IN
                Matricola   (SELECT Capo FROM Impiegato)
```

Matricola	Nome	Età	Stipendio	Capo
7309	Rossi	34	45	5698
5998	Bianchi	37	38	5698
9553	Neri	42	35	4076
5698	Bruni	43	42	4076
4076	Mori	45	50	8123

Impiegato

Interrogazioni sulle viste

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

```
SELECT * FROM ImpiegatiNonCapo
```

equivale a (e viene eseguita come)

```
SELECT Nome, Eta, Stipendio
FROM Impiegato
WHERE Nome NOT IN
                    (SELECT Capo FROM Impiegato)
```

Aggiornamenti sulle viste / 1

- Ammessi (di solito) solo su viste definite su una sola relazione
- Alcune verifiche possono essere imposte

Aggiornamenti sulle viste / 2

```
CREATE VIEW ImpiegatiNonCapoPoveri as
    SELECT *
    FROM ImpiegatiNonCapo
    WHERE Stipendio < 50
    WITH CHECK OPTION
```

- CHECK OPTION permette modifiche, ma solo a condizione che la tupla continui ad appartenere alla vista (non posso modificare lo stipendio portandolo a 60)

Aggiornamenti sulle viste / 3

```
CREATE VIEW ImpiegatiNonCapoPoveri as
    SELECT *
    FROM ImpiegatiNonCapo
    WHERE Stipendio < 50
    WITH CHECK OPTION
```

- Per esempio, non è ammesso:

```
UPDATE ImpiegatiNonCapoPoveri
    SET Stipendio = 60
    WHERE Nome = 'Paola'
```

Esercizio (Senza Viste)

Dato il seguente schema:

ESECUZIONE(CodiceReg,TitoloCanz,Anno)
AUTORE(Nome,TitoloCanzone)
CANTANTE(NomeCantante,CodiceReg)

Restituire il/i nome/i dello/degli autore/i che ha scritto la canzone con più esecuzioni, insieme con il nome della canzone stessa

```
SELECT *
FROM Autore
WHERE TitoloCanz
IN
(
        SELECT TitoloCanz
        FROM ESECUZIONE
        GROUP BY TitoloCanz
        HAVING COUNT(*) >
                SELECT MAX(CONTA) FROM
                        (SELECT COUNT(*) AS CONTA
                        FROM ESECUZIONE
                        GROUP BY TitoloCanzone)
                                AS CONTEGGIO
)
```

Basterebbe solo l'uguale, piuttosto che max

Esercizio (Con Viste)

Dato il seguente schema:

ESECUZIONE(CodiceReg,TitoloCanz,Anno)
AUTORE(Nome,TitoloCanzone)
CANTANTE(NomeCantante,CodiceReg)

Restituire il/i nome/i dello/degli autore/i che ha scritto la canzone con più esecuzioni, insieme con il nome della canzone stessa

```
CREATE VIEW ESECUZIONE_X_CANZONE(Titolo,NumEsecuzioni) AS
SELECT TitoloCanz,COUNT(*)
FROM ESECUZIONE
GROUP BY TitoloCanz

SELECT *
FROM Autore
WHERE TitoloCanz
IN
(
        SELECT Titolo
        FROM ESECUZIONE_X_CANZONE
        WHERE NumEsecuzioni=
                (SELECT MAX(NumEsecuzioni)
                 FROM ESECUZIONE_X_CANZONE)
)
```

Funzioni condizionali: *Coalesce*

- *Coalesce* restituisce il primo valore non nullo in una serie di espression
- Esempio: coalesce(NULL,‘A’,’B’,'Ignoto') restituisce ‘A’

- Esempio: *Estrarre i nomi, e l’età degli impiegati. Scrivere 0 se non si conosce l’età (eta=null)*

select Nome, coalesce(Età,0)
from Impiegato

Matricola	Nome	Età	Stipendio	Capo
7309	Rossi	34	45	5698

Funzioni condizionali: *nullif*

- *nullIf(expr)* restituisce NULL se, data una costante *x*, *expr* = *x*.

- Esempio: *Estrarre i nomi, e l'età degli impiegati. Scrivere NULL se l'età = 0*

```
select Nome, nullif(Età,0)
from Impiegato
```

Matricola	Nome	Età	Stipendio	Capo
7309	Rossi	34	45	5698

Funzioni condizionali: *Case*

- Simile allo Switch-Case di C
- Esempio: *Estrarre i nomi e la classe di età degli impiegati (<30 giovani, >60 anziani, altrimenti media)*

```
select Nome, case
        when (Età<30) then 'Giovane'
        when (Età>60) then 'Anziani'
        else 'Medio'
        end
from Impiegato
```

Matricola	Nome	Età	Stipendio	Capo
7309	Rossi	34	45	5698

Funzioni Scalari

- Funzioni a livello di attributi di tuple che restituiscono singoli valori:
  - Temporali:
    current_date, extract(year from …)
  - Manipolazione stringhe:
    char_length, lower
  - Conversioni:
    cast
  - …
- In PostgreSQL 13.0: https://www.postgresql.org/docs/13/functions.html

Controllo dell'accesso

- In SQL è possibile specificare
  - chi (utente) e
  - come (lettura, scrittura, …)
  - dove (tabelle, viste, attributi, …)

  può utilizzare la base di dati (o parte di essa)

- Il creatore di una risorsa ha tutti i privilegi su di essa

Privilegi

Un privilegio è caratterizzato da:

- la risorsa cui si riferisce
- l'utente che concede il privilegio
- l'utente che riceve il privilegio
- l'azione che viene permessa
- la trasmissibilità del privilegio

Tipi di privilegi offerti da SQL

- INSERT: permette di inserire nuovi oggetti (tuple)
- UPDATE: permette di modificare il contenuto
- DELETE: permette di eliminare oggetti
- SELECT: permette di leggere la risorsa
- REFERENCES: permette la definizione di vincoli di integrità referenziale verso la risorsa (può limitare la possibilità di modificare la risorsa)
- USAGE: permette l'utilizzo in una definizione (per esempio, di un dominio)

Autorizzazioni: Osservazione

- La gestione delle autorizzazioni deve “nascondere” gli elementi cui un utente non può accedere, senza sospetti

- Per esempio, l’utente riceve lo stesso messaggio se:
    - Impiegati non esiste
    - Impiegati esiste, ma l’utente non è autorizzato

Come autorizzare a vedere solo alcune tuple di una relazione?

Attraverso una vista:

1. Definiamo la vista con una condizione di selezione
2. Attribuiamo le autorizzazioni sulla vista, anziché sulla relazione di base

Esempi

Impiegato

Matricola	Nome	Età	Stipendio	Capo
7309	Rossi	34	45	5698
5998	Bianchi	37	38	5698
9553	Neri	42	35	4076
5698	Bruni	43	42	4076
4076	Mori	45	50	8123

```
CREATE VIEW ImpiegatiNonCapo(Nome, Eta, Stipendio) AS
    SELECT Nome, Eta, Stipendio
    FROM Impiegato
    WHERE Nome NOT IN (SELECT Capo FROM Impiegato)
```

Concedere a tutti i privilegi di leggere a *ImpiegatiNonCapo*:

```
GRANT SELECT ON ImpiegatiNonCapo TO PUBLIC;
```

Dare a «Manuel» tutti i privilegi su *ImpiegatiNonCapo* :

```
GRANT ALL PRIVILEGES ON ImpiegatiNonCapo TO Manuel;
```

Dare a «Max» la possibilità di aggiungere *Impiegato*:

```
GRANT INSERT PRIVILEGES ON Impiegato TO Max;
```

Riferimenti

- Capitolo 5 del libro:
  - Sezione 5.1
  - Sezione 5.2
  - Sezione 5.5