04-11-2022

Le viste = tabelle virtuali. Nel DB non sono fisicamente disponibili e sono definite a partire da una query. Quindi al posto della vista mette la query che la origina.
Utilizzate per vari scopi:

- Semplificazione
- Protezione dati
- Scomposizione query complesse
- Riorganizzazione dati secondo nuovi schemi
  Sintassi per la creazione:

```
CREATE VIEW NomeVista
        [“(” Attributo {,Attributo} “)”]
        AS Query-Select
```

Esempio:

```
CREATE VIEW MediaVoti (Matricola,Media) AS
        SELECT Matricola, AVG(Voto)
        FROM Esami
        GROUP BY Matricola
```

La prima colonna della select corrisponderà alla prima colonna dell'attributo della vista specificato sopra.
La view si usa come una tabella normale quindi con una select.

Le VIEW possono essere distrutte alla pari di tabelle

- DROP (TABLE | VIEW) Nome [RESTRICT|CASCADE]
- Con RESTRICT non viene cancellata se è utilizzata in altre viste
- Con CASCADE verranno rimosse tutte le viste che usano la View o la Tabella rimossa
- La distruzione di una VIEW non altera le tabelle su cui la VIEW si basa

- … utile nel caso di accesso dati controllato
- Esempio:
  - Impiegato(Nome, Cognome, Dipart, Ufficio, Stipendio)

- Il personale della segreteria non puo' accedere ai dati sullo stipendio ma puo' modificare gli altri campi della tabella, aggiungere e/o cancellare tuple

- Si puo' controllare l'accesso tramite la definizione della VIEW:
  - CREATE VIEW Impiegato2 AS
    SELECT Nome, Cognome, Dipart, Ufficio
    FROM Impiegato

- INSERT INTO Impiegato2 VALUES (…)
  - Stipendio verra' inizializzato a Null
  - Se Null non e' permesso per Stipendio l'operazione fallisce

- Immaginiamo la seguente VIEW:

```
CREATE VIEW ImpiegatoRossi
  AS
  SELECT *
  FROM Impiegato
  WHERE Cognome='Rossi'
```

- La seguente operazione ha senso:
  - INSERT INTO ImpiegatoRossi (…'Rossi',…)

- Ma che succede nel caso di:
  - INSERT INTO ImpiegatoRossi (…'Bianchi',…)
  - In genere e' permesso, finisce nella tabella base ma non e' visibile dalla VIEW

  - Si puo' controllare tramite l'opzione "WITH CHECK OPTION":

```
CREATE VIEW ImpiegatoRossi AS
  SELECT *
  FROM Impiegato
  WHERE Cognome='Rossi'
  WITH CHECK OPTION
```

- Adesso l'insert con 'Bianchi' fallisce, quella con 'Rossi' viene invece eseguita.

- Consideriamo il seguente caso:

```
Impiegato( Nome, Cognome, Dipart, Ufficio, Stipendio)
Dipartimenti( Dipart, Indirizzo)

CREATE VIEW IMP_IND AS
SELECT Nome, Cognome, d.dipart, indirizzo
FROM Impiegato i join Dipartimenti d ON
i.Dipart=d.Dipart
```

- Un INSERT sulla VIEW IMP_IND dovrebbe inserire su entrambe le tabelle base

- In alcuni casi potrebbe inserire in una ma non nell'altra
- In genere quest'operazione non è consentita

- Alcuni DBMS consentirebbero l'INSERT se "Impiegati.Dipart" fosse una foreign key su "Dipartimenti.Dipart" e quest'ultima fosse chiave primaria

- In genere una VIEW definita su una singola tabella è modificabile se gli attributi della VIEW contengono la chiave primaria (e altre chiavi)

- In genere VIEW definite su piu' tabelle non sono aggiornabili
  - Alcuni DBMS, come discusso prima, lo permettono nel caso certe condizioni, molto restrittive, siano rispettate

- VIEW che usano funzioni di aggregazione non sono aggiornabili

- PRINCIPIO di base per l'aggiornamento delle VIEW:
  - Ogni riga ed ogni colonna della VIEW deve corrispondere ad una ed una sola riga ed una ed una sola colonna della tabella base

1. Vincolo a livello di intera tabella "*check*"
2. vincolo a livello di intero database "*assertion*"

- Definire a livello di schema il vincolo che il massimo degli stipendi degli impiegati di dipartimenti con sede a Firenze sia minore dello stipendio di tutti gli impiegati del dipartimento Direzione.

```
CREATE ASSERTION ControlloSalari
CHECK ( NOT EXISTS ( SELECT *
FROM Impiegato I JOIN Dipartimento D ON I.Dipartimento=D.Nome
WHERE D.Città='Firenze'
   AND Stipendio > ( SELECT MIN(Stipendio)
                     FROM Impiegato
                     WHERE Dipartimento='Direzione') ) )
```

Limiti algebra relazionale

- L'insieme di interrogazioni esprimibili con l'algebra relazionale è significativo; il concetto è robusto

- Ci sono però interrogazioni interessanti non esprimibili:
  - **calcolo di valori derivati**: possiamo solo estrarre valori, non calcolarne di nuovi; calcoli di interesse:
    - a livello di ennupla o di singolo valore (conversioni somme, differenze, etc.)
    - su insiemi di ennuple (somme, medie, etc.)
  - le estensioni sono ragionevoli, sono state viste in SQL
  - interrogazioni inerentemente ricorsive, come la chiusura transitiva

In SQL si possono definire usare query ricorsive, con viste ricorsive.

```
WITH RECURSIVE factorial (n, fact) AS
(
  SELECT 0, 1 -- Initial Subquery
    UNION ALL
  SELECT n+1, (n+1)*fact
  FROM factorial -- Recursive Subquery
  WHERE n < 9
)
SELECT * FROM factorial;
```

```
WITH RECURSIVE Capi(Impiegato,Superiore) A(
        SELECT Impiegato, Capo -- caso base
        FROM Supervisione
                UNION ALL -- caso ricorsivo
        SELECT c.Impiegato,s.Capo
        FROM Capi c, Supervisione s
        WHERE c.Superiore = s.Impiegato
)
SELECT * FROM Capi
```

Funzioni scalari

Funzioni a livello di ennupla che restituiscono singoli valori

Temporali

current_date, extract(year from data)

Manipolazione stringhe

char_length, lower

Conversione

CAST(X AS TIPO)