Il linguaggio SQL: viste e tabelle derivate

Sistemi Informativi T

Versione elettronica: 04.5.SQL.viste.pdf

DB di riferimento per gli esempi

Imp

CodImp	Nome	Sede	Ruolo	Stipendio
E001	Rossi	S01	Analista	2000
E002	Verdi	S02	Sistemista	1500
E003	Bianchi	S01	Programmatore	1000
E004	Gialli	S03	Programmatore	1000
E005	Neri	S02	Analista	2500
E006	Grigi	S01	Sistemista	1100
E007	Violetti	S01	Programmatore	1000
E008	Aranci	S02	Programmatore	1200

Sedi

Sede	Responsabile	Citta
S01	Biondi	Milano
S02	Mori	Bologna
S03	Fulvi	Milano

Prog

CodProg	Citta
P01	Milano
P01	Bologna
P02	Bologna

Definizione di viste

- Mediante l'istruzione CREATE VIEW si definisce una vista, ovvero una "tabella virtuale"
- Le tuple della vista sono il risultato di una query che viene valutata dinamicamente ogni volta che si fa riferimento alla vista

```
CREATE VIEW ProgSedi(CodProg,CodSede)
AS      SELECT P.CodProg,S.Sede
        FROM   Prog P, Sedi S
        WHERE  P.Citta = S.Citta
```

ProgSedi

CodProg	CodSede
P01	S01
P01	S03
P01	S02
P02	S02

```
SELECT *
FROM   ProgSedi
WHERE  CodProg = 'P01'
```

CodProg	CodSede
P01	S01
P01	S03
P01	S02

Uso delle viste

- Le viste possono essere create a vari scopi, tra i quali si ricordano i seguenti:
  - Permettere agli utenti di avere una visione personalizzata del DB, e che in parte astragga dalla struttura logica del DB stesso
  - Far fronte a modifiche dello schema logico che comporterebbero una ricompilazione dei programmi applicativi
  - Semplificare la scrittura di query complesse
- Inoltre le viste possono essere usate come meccanismo per il controllo degli accessi, fornendo ad ogni classe di utenti gli opportuni privilegi
- Si noti che nella definizione di una vista si possono referenziare anche altre viste

Indipendenza logica tramite VIEW

- A titolo esemplificativo si consideri un DB che contiene la tabella

  EsamiSIT(Matr,Cognome,Nome,DataProva,Voto)

- Per evitare di ripetere i dati anagrafici, si decide di modificare lo schema del DB sostituendo alla tabella **EsamiSIT** le due seguenti:

  StudentiSIT(Matr,Cognome,Nome)
  ProveSIT(Matr,DataProva,Voto)

- È possibile ripristinare la "visione originale" in questo modo:

```
CREATE VIEW EsamiSIT(Matr,Cognome,Nome,DataProva,Voto)
AS      SELECT S.*,P.DataProva,P.Voto
        FROM   StudentiSIT S, ProveSIT P
        WHERE  S.Matr = P.Matr
```

Query complesse che usano VIEW (1)

- Un “classico” esempio di uso delle viste si ha nella scrittura di query di raggruppamento in cui si vogliono confrontare i risultati della funzione aggregata

La sede che ha il massimo numero di impiegati

- La soluzione senza viste è:

```
SELECT   I.Sede
FROM     Imp I
GROUP BY I.Sede
HAVING   COUNT(*) >= ALL (SELECT   COUNT(*)
                          FROM     Imp I1
                          GROUP BY I1.Sede)
```

Query complesse che usano VIEW (2)

- La soluzione con viste è:

```
CREATE VIEW NumImp(Sede,Nimp)
AS      SELECT   Sede, COUNT(*)
        FROM     Imp
        GROUP BY Sede

SELECT Sede
FROM   NumImp
WHERE  Nimp = (SELECT MAX(NImp)
               FROM   NumImp)
```

NumImp

Sede	NImp
S01	4
S02	3
S03	1

che permette di trovare “il MAX dei COUNT(*)”, cosa che, si ricorda, non si può fare direttamente scrivendo MAX(COUNT(*))

Query complesse che usano VIEW (3)

- Con le viste è inoltre possibile risolvere query che richiedono "piu' passi di raggruppamento", ad es:

  Per ogni valore (arrotondato) di stipendio medio, numero delle sedi che pagano tale stipendio

- Occorre aggregare **prima** per sede, **poi** per valore di stipendio medio

```
CREATE VIEW StipSedi(Sede,AvgStip)
AS      SELECT   Sede, AVG(Stipendio)
        FROM     Imp
        GROUP BY Sede

SELECT AvgStip, COUNT(*) AS NumSedi
FROM   StipSedi
GROUP BY AvgStip
```

StipSedi

Sede	AvgStip
S01	1275
S02	1733
S03	1000

AvgStip	NumSedi
1275	1
1733	1
1000	1

Aggiornamento di viste

- Le viste possono essere utilizzate per le interrogazioni come se fossero tabelle del DB, ma per le operazioni di aggiornamento ci sono dei limiti

```
CREATE VIEW NumImp(Sede,NImp)
AS      SELECT    Sede,COUNT(*)
        FROM        Imp
        GROUP BY  Sede
```

NumImp

Sede	NImp
S01	4
S02	3
S03	1

```
UPDATE NumImp
SET    NImp = NImp + 1
WHERE  Sede = 'S03'
```

- Cosa significa? Non si può fare!