VISTE

Una *vista* è una relazione *derivata* dalle relazioni di *base* o da altre viste per mezzo di una definizione basata sul costrutto Select From Where. Come le relazioni di base, le viste sono dotate di uno schema. Pertanto, Vista = Schema + Definizione. Una vista è utilizzabile nelle interrogazioni alla stessa stregua delle relazioni di base.
Due tipologie di viste:

- *virtuali*: prive di estensione (solo definizione)
- *materializzate*: tabelle derivate il cui contenuto (tuple) viene temporaneamente memorizzato nella base di dati (problema: mantenimento dell'allineamento con le tabelle di base)

Schema Logico

- ALUNNO(**CF**, nome, cognome, data_nascita, sostegno)
- INSEGNANTE(**CF**, nome, cognome, data_nascita, di_ruolo, materia)
- CLASSE(**id**, anno, sezione, id_scuola*)
- AULA(**id**, mq, #max:stud, nome_fasciaEtà*, piano, id_plesso*)
- PLESSO(**id**, indirizzo, #piani, ascensore)
- FASCIAETÀ(**nome**, min, max)
- SCUOLA(**id**, nome, telefono, tipo, tempoPieno)
- INSEGNA(**id_ins*, id_classe*, AS**, ore)
- ASSEGNAMENTO(**id_classe*, id_aula*, AS,** numerosa**)**
- STORICOISCRIZIONE(**id_alunno*, id_classe*, AS,** data)

Aggiungiamo allo schema le seguenti viste:

1. PLESSOAULE(plesso_id, #aule, #max:stud)
2. INSEGNATESTUDENTE(CF_ins, cognome_Ins, materia, CF_Stu, cognome_Stu, AS, annoClasse, sezione)
3. ScuolaClassi(scuola_id, AS, #classi, #alunni)

//per ogni plesso, numero di aule e numero max di studenti che può ospitare

1. **Define View** PlessoAule(**plesso_id**, #aule, #max:stud) As
   Select P.id, count(*), sum(#max:stud)
   From Plesso P, Aula A
   Where A.id_plesso=P.id
   GroupBy P**.**id

Esempi di interrogazioni su PlessoAule
Q_0= Il plesso col max numero di aule

```
Select plesso_id
From PlessoAule
Where  #aule = (Select max(#aule)
                From PlessoAule)
```

// Studenti avuti da un insegnante in un anno scolastico

2. **Define View** InsegnateStudente(**<u>CF_ins, CF_Stu, AS</u>**, cognome_Ins, materia, cognome_Stu, annoClasse, sezione) As
   Select IS.CF, AL.CF, IN.AS, IS.cognome, IS.materia, AL.cognome, CL.anno, CL.sezione
   From Insegnante as IS, Insegna as IN, StoricoIscrizione as SI, Alunno as AL, Classe as CL
   Where IS.CF=IN.ind_ins and IN.id_classe=SI.id_classe and SI.id_alunno=AL.CF and IN.Id_classe=CL.id and IN.AS=SI.AS

Esempi di interrogazioni su InsegnateStudente

Q1: Quali insegnati ha avuto lo studente Verdi con CF=A10C nell'anno scolastico 2016?

```
Select CF_ins, cognome_Ins
From InsegnateStudente
Where AS=2016 and CF_Stu= A10C
```

Q2: Quanti studenti ha avuto l'insegnate con CF=BBB nell'anno scolastico 2016?

```
Select count(*)
From InsegnateStudente
Where CF_Ins= BBB and AS=2016
```

Q3: Insegnante che ha avuto il max numero di studenti nel 2016

….

I suddetti esempi dimostrano come la definizione di viste semplifichi la formulazione delle interrogazioni. Tuttavia, vi è un altro vantaggio fondamentale: l'incremento del potere espressivo del linguaggio di interrogazione, in quanto ci sono query esprimibili solo grazie alle viste (le sole tabelle di base non sono sufficienti). In particolare, si tratta dei seguenti casi

- vi è una applicazione in cascata di funzioni di aggregazione
- query ricorsive

Applicazione di funzioni in cascata. Siccome SQL non supporta l'uso di funzioni che hanno funzioni come argomento (funzioni di funzioni), per poter applicare una funzione sui risultati di un'altra funzione è necessario definire una vista che calcoli i valori della prima funzione. Ad esempio, si consideri la seguente interrogazione

- Q4 - per ogni scuola, il numero medio di alunni negli ultimi 5 anni

Per calcolare il numero medio di alunni negli ultimi 5 anni è necessario preventivamente calcolare il numero di alunni per ogni anno. A tal fine definiamo la seguente vista:

Define View ScuolaClassi (scuola_id, AS, #classi , #alunni) As

```
Select id_scuola, AS, count(distict classe_id), count(*)
From StoricoIscrizione S, Classe C
Where S.id_classe=C.id
GroupBy id_scuola, AS
```

A questo punto, è possibile formulare sulla vista la seguente query, dove la funzione avg si applica all'attributo #alunni calcolato tramite la funzione count nella vista ScuolaClassi.

Query Q4

```
Select id_scuola,  avg(#alunni)
From ScuolaClassi
Where AS < 2023 and AS > 2017
GroupBy id_scuola
```

Come altro esempio, si consideri la seguente interrogazione che richiede la definizione di 2 viste

- Q5: Trovare, per ogni anno scolastico, gli insegnanti con #ore superiore alla media

```
DEFINE VIEW OrePerInsegnante(insegnante,AS,#ore) AS
        SELECT id_ins, AS, sum(ore)
        FROM Insegna
        GROUP BY id_ins, AS

DEFINE VIEW MediaOrePerAS(Anno,avgOre) AS
        SELECT AS, avg(#ore)
        FROM OrePerInsegnante
        GROUP BY AS

SELECT AS, insegnante
FROM OrePerInsegnante, MediaOrePerAS
WHERE Anno=AS and #ore>avgOre
```

Viste ricorsive **-** SQL consente di definire viste ricorsive. Ad esempio, possiamo definire la vista **Antenato** come *chiusura transitiva* della relazione di base **AntenatoDiretto** (gli antenati diretti sono i genitori): se **b** e un ascendente diretto (cioè, di primo livello) di **a** e **c** è un ascendente di **b**, allora **c** è un ascendente di **a.** La relazione **AntenatoDiretto** è definita come segue:

AntenatoDiretto(disc*, asc*)

dove **disc (**discendente*)* e **asc** (ascendente*)* sono chiavi secondarie definite su **Persona**.

```
CREATE VIEW Antenato(disc, asc, livello) AS
// AD = AntenatoDiretto, A = Antenato
    SELECT disc, asc, 1
    FROM AD
            UNION
    SELECT X.disc, Y.asc, livello+1
    FROM AD as X, A as Y
    WHERE X.asc=Y.disc
```

La prima sottoquery (quella in alto) calcola il passo base della ricorsione: ogni antenato diretto è un antenato di livello 1. La relazione Antenato coincide con AntenatoDiretto.
La seconda sottoquery implementa il passo induttivo: se X è una istanza di AntenatoDiretto, e Y è una istanza di Antenato tale che Y.asc è un antenato di livello L di Y.disc, e Y.disc è un antenato di livello 1 di X.disc, allora Y.asc è un antenato di livello L+1 di X.disc -- cioè, < X.disc, Y.asc, livello+1> è una istanza di Antenato.

Procedure puntoFisso per il calcolo della relazione A=Antenato
Input: AD= AntenatoDiretto
Output: A=Antenato
begin
i=0; A(i)= ∅;
repeat
i=i+1;
$A(i) = \pi_{AD.disc,A.acs}(AD \bowtie_{AD.asc=A.disc} A(i-1)) \cup AD)$
until $A(i) = A(i-1)$
end

dove A(i) contiene tutte le tuple di antenato al passo i-esimo (gli antenati di livello max pari a *i*). A(i) si ottiene mettendo in join gli antenati del livello precedente A(i-1) con gli antenati diretti AD (di livello 1). L'algoritmo termina quando le tuple della chiusura transitiva calcolate al passo i-esimo sono uguali a quelle calcolate al passo precedente – si è raggiunto il punto fisso.

ESECUZIONE DI QUERY

1) Vista virtuale: trasformazione della query sulla vista in una query equivalente sulle relazioni di base – a tempo di compilazione
2) Vista materializzata: quando una query sulla vista viene eseguita per la prima volta, le tuple generate vengono memorizzate come per le relazioni di base. Problema: aggiornamento necessario ogniqualvolta le relazioni di base, in funzione delle quali è definita la vista, vengono aggiornate

VANTAGGI USO VISTE

- Semplicità interrogazioni
- Maggiore potere espressivo - Viste ricorsive e composizione di funzioni
- Protezione dei dati: gruppi di utenti, per motivi di sicurezza/privacy, possono accedere alla BD attraverso opportune viste
- Indipendenze logica: se viene modificato lo schema logico della BD, si può definire un insieme di viste tali da mantenere inalterato lo schema iniziale – per gli utenti non cambia quindi la percezione della BD