Basi Di Dati e di conoscenza

MySQL- Viste e accessi

Contenuti della lezione

- Viste
- Operazioni con le viste
- Funzioni scalari e condizionali
- Gestione degli accessi

Viste

- Le viste sono tabelle virtuali ricavate da informazioni contenute in altre tabelle.
- Nella definizione possono essere contenute anche altre viste purché non vi siano dipendenze ricorsive o immediate (una vista non può dipendere da se stessa), né transitive.

`create view` *NomeVista* **`[(`***ListaAttributi***`)] as`** *SelectSQL*

`[ with [ local | cascaded] check option ]`

- La query SQL deve restituire un numero di attributi pari a quelli contenuti nello schema;
- l'ordine degli attributi nella target list deve rispettare quello dello schema

Viste: Esempio

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

Su certe viste è possibile fare modifiche che alterano le tabelle che le compongono. Ci sono problemi se la vista è definita tramite un join.

SQL permette la modifica di una vista solo se una sola riga di ciascuna tabella di base corrisponde a una riga della vista. Di solito si richiede che sia definita su una sola tabella e/o che contenga almeno una chiave primaria.

Viste

Le viste consentono anche di creare interrogazioni altrimenti impossibili da definire.

Esempio: Trovare il dipartimento che ha la massima somma degli stipendi

```
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

Vantaggi:

1. non occupano memoria
2. sicurezza (si possono non far vedere certi campi)
3. convenienza: si creano view anche per eseguire query più semplici, spesso con ottimizzazione dei tempi di risposta

Svantaggi:

1. `update` e `delete` sulle view diventano azioni pericolose per le tabelle.
2. Possibilità di inconsistenza tabelle e DB

Contenuti della lezione

- Viste
- Operazioni con le viste
- Funzioni scalari e condizionali
- Gestione degli accessi

Interrogazioni sulle viste

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

- 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
```

Aggiornamenti sulle viste

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

```
create view ImpiegatiAmminPoveri as
   select *
   from ImpiegatiAmmin
   where Stipendio < 50
   with check option
```

- **check option** permette modifiche, ma solo a condizione che la ennupla continui ad appartenere alla vista (non posso modificare lo stipendio portandolo a 60)

Aggiornamenti sulle viste

```
create view ImpiegatiAmminPoveri as
   select *
   from ImpiegatiAmmin
   where Stipendio < 50
   with check option

update ImpiegatiAmminPoveri
   set stipendio = 60
   where nome = 'Paola'
```

Un'interrogazione non standard

- Interrogazione scorretta

```
select avg(count(distinct Ufficio))
from Impiegato
group by Dipart
```

- Con una vista

```
create view DipartUffici(NomeDip,NroUffici) as
  select Dipart, count(distinct Ufficio)
  from Impiegato
  group by Dipart;

select avg(cast(NroUffici as decimal(5,2))) as NumeroMedioUffici
from DipartUffici;
```

Ancora sulle viste

- La nidificazione nella **having** non è ammessa in alcuni sistemi

```
select Dipart
from Impiegato
group by Dipart
having sum(Stipendio) >= all
    (select sum(Stipendio)
     from Impiegato
     group by Dipart)
```

Soluzione con le viste

```
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 ricorsive

- Per ogni persona, trovare tutti gli antenati, avendo
  `Paternita (Padre, Figlio)`

- Serve la ricorsione; in Datalog:

```
Discendenza (Antenato: p, Discendente: f) ←
  Paternita (Padre: p, Figlio: f)

Discendenza (Antenato: a, Discendente: d) ←
  Paternita (Padre: a, Figlio: f) ,
  Discendenza (Antenato: f, Discendente: d)
```

Viste ricorsive in MySQL 8+

```
with Discendenza(Antenato,Discendente) as
  (   select Padre, Figlio
          from Paternita
      union all
      select D.Antenato, Figlio
          from Discendenza D, Paternita
          where D.Discendente = Padre);
select *
from Discendenza;
```

Contenuti della lezione

- Viste
- Operazioni con le viste
- Funzioni scalari e condizionali
- Gestione degli accessi

Funzioni scalari

- Funzioni a livello di ennupla che restituiscono singoli valori
- Temporali
  - `current_date, extract(year from …)`
- Manipolazione stringhe
  - `char_length, lower`
- Conversione
  - `cast`
- Condizionali
  - …

Funzioni condizionali `Nullif ()`

- **`Nullif():`** confronta due espressioni e restituisce NULL se sono uguali. Altrimenti, viene restituita la prima espressione.

```
SELECT NULLIF(25, 26);
   >>  25
SELECT NULLIF(25, 25);
   >>  NULL
```

Funzioni condizionali `Coalesce()`:

- `Coalesce ()`: riporta il primo valore non nullo in una lista

```
select Nome, Cognome, coalesce(Dipart,'Ignoto')
from Impiegato
```

Funzioni condizionali `Coalesce()`:

- `Coalesce ()`: riporta il primo valore non nullo in una lista

```
select Nome, Cognome, coalesce(Dipart,'Ignoto')
from Impiegato
```

Funzioni condizionali **Case**

- **Case:**
  - valuta delle condizioni e restituisce un valore quando la prima condizione è verificata (simile a un'istruzione IF-THEN-ELSE). Una volta che una condizione è vera, smetterà di valutare le altre condizioni e restituirà il risultato.
  - Se nessuna delle condizioni è vera, verrà restituito il valore specificato nella clausola ELSE.

```
select Targa,
    case Tipo
          when 'Auto' then 2.58 * KWatt
          when 'Moto' then (22.00 + 1.00 * KWatt)          else null
    end as Tassa
 from Veicolo
 where Anno > 1975;
```

Contenuti della lezione

- Viste
- Operazioni con le viste
- Funzioni scalari e condizionali
- Gestione degli accessi

Controllo degli accessi

- SQL prevede la definizione di **utenti e ruoli**, per assegnare diversi **privilegi**.
- Gli utenti possono essere gli stessi del sistema su cui è attivo il server SQL, oppure indipendenti dal sistema.
- Ogni componente del sistema è proteggibile, di solito si proteggono le tabelle.

- Il controllo degli accessi è basato sul concetto di **privilegio**, caratterizzato da:
  - **risorsa cui si riferisce**
  - **utente che concede il privilegio**
  - **utente che lo riceve**
  - **azione che viene permessa**
  - **possibilità di trasmettere o meno il privilegio ad altri utenti**

Ruoli

- Un ruolo è un meccanismo che può essere utilizzato per consentire l'autorizzazione.
- Una **persona** o un **gruppo** di persone può essere autorizzato a un **ruolo** o a un **gruppo di ruoli**.
- Con molti ruoli, il responsabile può gestire facilmente i privilegi di accesso. I ruoli sono forniti dal sistema di gestione del database per una gestione dei privilegi facile e controllata.
- Per creare o cancellare un ruolo:

  `Create role` e `drop role`

Gestione degli accessi

- MySQL consente la creazione di account che permettono agli **utenti client** di connettersi al server e accedere ai dati gestiti dal server.
- La funzione principale del sistema di privilegi di MySQL è autenticare un utente che si connette da un determinato host e associare a quell'utente dei privilegi su un database.
- Tra le funzionalità aggiuntive vi è la possibilità di concedere privilegi per operazioni amministrative.
- Per controllare quali utenti possono connettersi, a ciascun account possono essere assegnate credenziali di autenticazione come una password.
- L'interfaccia utente per gli account MySQL è costituita da istruzioni SQL come **`CREATE USER`**, **`GRANT`** e **`REVOKE`**.

Controllo degli accessi

- Il creatore di una risorsa ha tutti i privilegi, al momento della sua creazione.
- Esiste anche un utente "speciale" `_system`, che ha tutti i privilegi su tutte le risorse in qualsiasi momento.
- I privilegi sono:
  - `insert` (applicabile a tabelle o viste) inserisce un nuovo oggetto nella risorsa
  - `update` (tabelle viste attributi) aggiorna il valore di un oggetto
  - `delete` (tabelle e viste) rimuove un oggetto
  - `select` (tabelle, viste, attributi) permette di leggere la risorsa
  - `references` (tabelle ed attributi) permette che venga fatto riferimento ad una risorsa nell'ambito della definizione dello schema di una tabella.
  - `usage` (domini) permette che venga usata la risorsa
  - `drop` e `alter` sono riservati al creatore degli oggetti cui si applicano

Privilegi

- I comandi per concedere o revocare privilegi sono **grant** e **revoke**

```
grant tipo privilegio
on oggetto(DB,tabelle)
to nome_utente;

revoke tipo privilegio
on oggetto
from nome utente;
```

Privilegi: esempio

```
grant select on Dipartimento to Stefano
```

- concede all'utente Stefano il privilegio di `select` sulla tabella `Dipartimento`.
- Se si specifica anche with `grant option` l'utente può propagare i diritti anche ad altri.
- La parola chiave `all privileges` specifica tutti i possibili privilegi.

Controllo accessi

- A volte è meglio costruire delle **`view`** con le sole colonne che possono essere viste e dare accesso a queste, invece di dare accesso alle singole colonne delle tabelle.
- **Troppi privilegi, o troppi privilegi a grana fine, o troppi privilegi separati sulle colonne possono creare problemi di performance.**