Azzolini Riccardo 2019-11-12

Viste

1 Viste

Una **vista** è una *relazione virtuale*: il suo contenuto (tuple)

- è definito mediante un'interrogazione sulla base di dati, e dipende quindi dal contenuto di altre relazioni;
- non è memorizzato fisicamente ("materializzato"), bensì viene ricalcolato ogni volta che si usa la vista.

Si dice invece **relazione di base** una relazione che non è una vista.

A differenza di una query, una vista è un oggetto della base di dati, sul quale è possibile effettuare operazioni, usandola (quasi a tutti gli effetti) come una relazione di base.

Il meccanismo delle viste è utile per:

- semplificare l'accesso ai dati (ad esempio, per evitare la ripetizione di operazioni di join particolarmente frequenti);
- fornire indipendenza logica, cioè nascondere agli utenti della base di dati eventuali modifiche dello schema logico;
- garantire la protezione dei dati, autorizzando agli utenti ad accedere a una vista (contenente solo, ad esempio, delle statistiche, oppure un sottoinsieme ristretto di tuple e/o colonne), ma non alla tabella sottostante.

2 Creazione di una vista

Il comando per la creazione di una vista ha la sintassi:

```
CREATE VIEW <nome vista> [(<lista nomi colonne>)]
AS <interrogazione>
[WITH [{LOCAL | CASCADED}] CHECK OPTION];
```

- `<nome vista>` è, appunto, il nome della vista.

- `<interrogazione>` è l'**interrogazione di definizione** della vista, la cui clausola di proiezione (`SELECT`) determina il numero e i domini delle colonne della vista. Qui si possono usare tutte le funzionalità del linguaggio di query; ad esempio, sono spesso utili:
  - join, per facilitare l'accesso ai dati, consentendo agli utenti di lavorare su una singola relazione;
  - funzioni di gruppo, per permettere a determinati utenti di lavorare solo su dei dati aggregati, piuttosto che sui dati di dettaglio, fornendo così una garanzia di riservatezza.
- `<lista nomi colonne>` è una lista di nomi da assegnare alle colonne della vista. Essa può anche non essere specificata: vengono allora utilizzati i nomi degli attributi restituiti dall'interrogazione. L'unico caso in cui diventa obbligatoria è quando la clausola di proiezione contiene colonne virtuali (espressioni, funzioni di gruppo, ecc.) a cui non è stato assegnato un nome (ma, in alternativa, si può usare la sintassi `AS` direttamente nell'interrogazione: ciò è più conveniente se devono essere rinominate solo alcune delle colonne).

2.1 Esempi

Vista contenente il codice cliente, la data di inizio noleggio, e la collocazione dei video in noleggio da più di tre giorni:

```
CREATE VIEW Nol3gg AS
SELECT codCli, dataNol, colloc
FROM Noleggio
WHERE dataRest IS NULL
  AND (CURRENT_DATE - dataNol) DAY > INTERVAL '3' DAY;
```

Vista che, per ogni cliente, contiene il codice, il numero di noleggi effettuati, e la durata massima in giorni di tali noleggi:

```
CREATE VIEW InfoCli (codCli, numNol, durataM) AS
SELECT codCli, COUNT(*), MAX((dataRest - dataNol) DAY)
FROM Noleggio
GROUP BY codCli;
```

3 Cancellazione di una vista

Una vista esistente può essere cancellata con il comando:

```
DROP VIEW <nome vista>;
```

Esso non cancella alcun dato: viene eliminata solo la definizione della vista, che quindi non potrà più essere usata nelle interrogazioni.

4 Operazioni sulle viste

Una volta definita, una vista è parte dello schema della base di dati, e, idealmente, dovrebbe poter essere manipolata dall'utente in modo analogo a una relazione di base. In pratica, su una vista si possono eseguire:

- interrogazioni, con tutte le funzionalità del linguaggio di query, e anche la possibilità di creare ulteriori viste basate su di essa;
- aggiornamenti, ma solo *sotto opportune condizioni*.

4.1 Aggiornamento

Quando si effettua un aggiornamento su una vista, le modifiche devono poter essere propagate alla relazione di base su cui essa è definita. In alcuni casi, però, la realizzazione dell'operazione richiesta sulla vista attraverso operazioni sulla relazione di base non esiste, oppure non è univoca. Ad esempio:

- In generale, una modifica su una colonna di una vista viene realizzata tramite una modifica sulla colonna corrispondente della relazione di base. Se, invece, la colonna della vista è virtuale (definita da un'espressione), non è possibile stabilire quali valori assegnare alle (una o più) colonne corrispondenti della relazione di base, per ottenere come risultato dell'espressione il valore specificato nella modifica sulla vista.
- Un inserimento in una vista viene realizzato tramite un inserimento nella relazione di base. Se la relazione di base ha una colonna obbligatoria (`NOT NULL`) e senza valore di default che non è inclusa nella vista, l'inserimento nella vista non specifica un valore per tale colonna, e quindi non può essere effettuato.
- Una cancellazione su una vista viene realizzata tramite una cancellazione sulla relazione di base. Se la vista è definita come join di più relazioni di base, la cancellazione di una tupla della vista può essere ottenuta:
  - cancellando la tupla corrispondente in una delle relazioni di base;
  - cancellando le tuple corrispondenti in tutte le relazioni di base;
  - ponendo a `NULL` il valore dell'attributo di join di una o più delle tuple corrispondenti nelle relazioni di base.

Secondo lo standard SQL, è possibile eseguire aggiornamenti su una vista solo se una sola riga di ciascuna tabella di base corrisponde a una sola riga di tale vista, cioè se si ha una corrispondenza univoca tra le tuple della vista e delle tabelle su cui è definita. In questo caso, infatti, è sempre possibile propagare le modifiche alle tabelle di base, senza ambiguità.

Quindi, perché una vista sia aggiornabile, secondo lo standard, la sua interrogazione di definizione deve:

- comprendere le chiavi primarie delle tabelle di base;
- non contenere `DISTINCT`;
- non contenere funzioni di gruppo;
- non contenere join;
- ecc.

Concretamente, i criteri per l'aggiornamento delle viste variano in base al DBMS (ad esempio, alcuni lo permettono anche in presenza di join).

5 Check option

L'interrogazione di definizione di una vista può contenere (nella clausola `WHERE`, ecc.) delle condizioni sul contenuto delle tuple. Allora, quando si effettuano degli inserimenti (se la definizione della vista li consente), i valori inseriti dovrebbero, in teoria, rispettare tali condizioni: altrimenti, poi, le tuple inserite non si ritroverebbero nelle interrogazioni sulla vista. Analogamente, quando vengono modificate delle tuple esistenti, sarebbe opportuno che esse continuassero a soddisfare le condizioni di appartenenza alla vista.

Per assicurare che le tuple inserite/modificate tramite una vista siano accettate solo se verificano le condizioni specificate dall'interrogazione di definizione di tale vista, si usa la clausola `WITH CHECK OPTION` del comando `CREATE VIEW`.

Come caso particolare, se la vista è realizzata in termini di altre viste, ciascuna di esse potrebbe a sua volta essere definita con `CHECK OPTION`. In questa situazione, è possibile scegliere di verificare:

- solo le condizioni della vista corrente (`WITH LOCAL CHECK OPTION`);
- anche, ricorsivamente, le condizioni delle viste "sottostanti" (`WITH CASCADED CHECK OPTION`, che comunque è l'opzione di default, qualora non si specifichi esplicitamente né `LOCAL` né `CASCADED`).

5.1 Esempio 1

La vista

```
CREATE VIEW Nol3gg AS
SELECT codCli, dataNol, colloc
FROM Noleggio
WHERE dataRest IS NULL
  AND (CURRENT_DATE - dataNol) DAY > INTERVAL '3' DAY;
```

contiene solo tuple relative a noleggi che sono in corso da più di tre giorni. Se si inserisse la tupla `(1128, CURRENT_DATE, 6635)`, corrispondente a un noleggio che inizia oggi, questa non verrebbe ritrovata dalle interrogazioni su `Nol3gg`.

Invece, definendo `Nol3gg` con `CHECK OPTION`,

```
CREATE VIEW Nol3gg AS
SELECT codCli, dataNol, colloc
FROM Noleggio
WHERE dataRest IS NULL
  AND (CURRENT_DATE - dataNol) DAY > INTERVAL '3' DAY
WITH CHECK OPTION;
```

l'inserimento della tupla `(1128, CURRENT_DATE, 6635)` verrebbe rifiutato.

5.2 Esempio 2

La vista

```
CREATE VIEW ProdottiTagliaMediaOGrande AS
SELECT CodP, NomeP, Taglia
FROM Prodotti
WHERE Taglia >= 42
WITH CHECK OPTION;
```

non accetta modifiche/inserimenti con valori di taglia minori di 42.

Se, su di essa, si definisce la vista

```
CREATE VIEW ProdottiTagliaMedia AS
SELECT CodP, NomeP, Taglia
FROM ProdottiTagliaMediaOGrande
WHERE Taglia <= 46
WITH CASCADED CHECK OPTION;
```

quest'ultima accetta solo aggiornamenti del contenuto che specificano taglie comprese tra 42 e 46. Invece, con `LOCAL CHECK OPTION`,

```sql
CREATE VIEW ProdottiTagliaMedia AS
SELECT CodP, NomeP, Taglia
FROM ProdottiTagliaMediaOGrande
WHERE Taglia <= 46
WITH LOCAL CHECK OPTION;
```

gli aggiornamenti devono rispettare solo la condizione `Taglia <= 46`, quindi è possibile specificare taglie minori di 42, che, però, non verranno poi ritrovate nelle interrogazioni (per le quali si applicano sempre e comunque le condizioni di tutte le viste coinvolte).