Appunti Basi di Dati

Luca Seggiani

15 Maggio 2024

1 Database attivi

Abbiamo visto finora le metodistiche dall'SQL procedurale, che permette di implementare funzionalità in linguaggio, appunto, procedurale, sebbene il paradigma dell'SQL sia in teoria dichiarativo. Adesso vedremo alcune caratteristiche dell'SQL che permettono di implementare comportamenti "attivi" del database: per la precisione, **trigger** e **event**.

Trigger

Un trigger è una procedura che viene eseguita sulla base di eventi di istruzione DML (*data manipulation language*). Un trigger è fornito di una **parte reattiva** che reagisce a eventi DML, o a determinati istanti temporali, che causa l'esecuzione di un'azione (operazione) sui dati. Gli eventi DML scatenanti possono ad esempio essere modifiche, cancellazioni, inserzioni, timer ecc...

La **condizione** è un predicato booleano che viene valutato dopo l'evento. Un risultato positivo della condizione comporta l'esecuzione dell'azione. Abbiamo quindi:

1. Evento;
2. Condizione;
3. Azione.

Facciamo un'esempio: voglliamo gestire un'attributo ridondante nella tabella Paziente contenente la data nella quale un paziente è stato visitato l'ultima volta. Sarà un'attributo ridondante perché lo potremo già tovare nella tabella Visita. D'altronde, ad ogni successiva visita, l'attributo sulla tabella paziente andrà aggiornato. In ogni caso, il comportamento che vogliamo è che, all'inserzione dell'ultima visita nella tabella visita da parte di un'operatore, il database possa automaticamente (*database attivo*) inserire l'informazione

necessaria nella tabella paziente attraverso un trigger.
Individuiamo evento, condizione ed azione:

- **Evento:** l'evento sarà l'inserimento di una nuova visita nella tabella Visita;
- **Condizione:** qui la condizione non c'é: ogni nuova visita è un'ultima visita.
- **Azione:** rintraccia il paziente della visita, e aggiorna la sua ultima visita.

in SQL, la sintassi di un trigger sarà:

```
DROP TRIGGER IF EXISTS nome_trigger
CREATE TRIGGER nome_trigger
[BEFORE | AFTER][INSERT |UPDATE|DELETE] ON target
FOR EACH ROW blocco_istruzioni
```

Vediamo i dettagli. Con BEFORE si indica un'azione di *preprocessing* quindi di esecuzione prima dell'evento, mentre con AFTER si indica un'azione *a posteriori*, o "collaterale", che viene eseguita dopo l'evento. La sintassi che vorremo nell'esempio appena sopra sarà quindi:

```
DROP TRIGGER IF EXISTS aggiorna_ultima_visita;
CREATE TRIGGER aggiorna_ultima_visita
AFTER INSERT ON Visita FOR EACH ROW
  UPDATE Paziente
  SET UltimaVisita = CURRENT_DATE
  WHERE CodFiscale = NEW.Paziente
```

dove NEW indica il record che è stato già inserito (in un trigger AFTER; in un trigger BEFORE sarebbe stato quello che sta per essere inserito)..

Trigger multi-statement

Un trigger *multi-statement* ("multi-blocco") è formato da più blocchi separati da punti e virgola. Si rende quindi necessario, come avevamo già visto, l'uso di un delimitatore ausiliario:

```
DROP TRIGGER IF EXISTS nome_trigger;
DELIMITER $$
CREATE TRIGGER nome_trigger
BEFORE ... ON ...
FOR EACH ROW

BEGIN
-- blocchi di istruzioni
END $$
DELIMITER ;
```