Funzioni definite dall'utente e trigger

Nicola Vitacolonna

Corso di Basi di Dati
Università degli Studi di Udine

26 novembre 2015

Trigger

- Azione eseguita automaticamente dal sistema in conseguenza di un'operazione di modificazione della base di dati
- Un trigger è attivato indipendentemente dall'utente o dall'applicazione che effettua l'operazione
- L'attivazione di un trigger causa l'esecuzione di una UDF
- Sintassi semplificata:

```
create trigger ⟨nome⟩ { before | after } ⟨evento⟩
on ⟨tabella⟩
for each { row | statement }
[ when ⟨condizione⟩ ]
execute procedure ⟨funzione⟩ ();
```

↳ UDF

- Ciascun DBMS ha le proprie varianti sintattiche

Caratteristiche dei trigger

- L'evento può essere **`insert`**, **`update`** o **`delete`**
- Un trigger può essere attivato prima (**`before`**) o dopo (**`after`**) l'evento
- È possibile specificare una condizione per l'attivazione (**`when`**)
- Un trigger può essere attivato una volta sola per evento (**`for each statement`**) o una volta per ciascun record modificato dall'evento (**`for each row`**)
- Può essere eseguito immediatamente dopo l'evento che lo attiva, ovvero essere posposto al termine della transazione corrente (**esecuzione differita**)
  - In PostgreSQL, solo i **`constraint trigger`** sono differibili
- Se più trigger sono attivati dal medesimo evento, l'ordine d'esecuzione dei trigger dipende dall'implementazione
  - Può coincidere con l'ordine di definizione
  - Può essere determinato automaticamente dal sistema
  - Può essere definito dall'utente
  - In PostgreSQL, in ordine alfabetico

Trigger: osservazioni

- Un trigger può effettuare controlli sui dati e modificare lo stato della base di dati
  - Registrazione degli eventi in un log (**auditing**, replicazione, allineamento di dati duplicati„ gestione di viste materializzate, etc…)
- È preferibile usare altri meccanismi, se possibile
  - Ad esempio, la clausola `on delete cascade` per una cancellazione in cascata
  - Funzionalità *built-in* per replicazione, auditing e viste materializzate
- Un trigger può attivarne altri in cascata, il che pone problemi di coerenza e terminazione
  - Ad esempio, un trigger A attiva un trigger B che riattiva A
  - Alcuni sistemi (e.g., Oracle) vietano attivazioni ricorsive
- Bisogna evitare attivazioni non volute
  - Ad esempio, in procedure di ripristino da backup o replicazione mediante ripetizione di comandi SQL (**statement-based replication**)

Funzioni per trigger nel linguaggio PL/pgSQL

```
create or replace function <nome_funzione>()
returns trigger
language plpgsql as $$
  declare
    <dichiarazioni di variabili>
  begin
    <istruzioni>
  end;
$$;
```

- La funzione non ha parametri
- Al posto del valore di ritorno c'è la clausola **`returns trigger`**

Trigger PL/pgSQL: variabili `new` e `old`

Due variabili speciali sono disponibili nel corpo della funzione:

- **`new`**: in operazioni **`insert`** o **`update`**, rappresenta il record che si vuole inserire o aggiornare
- **`old`**: in operazioni **`delete`** o **`update`**, rappresenta il record che si vuole cancellare o modificare
- **`new`** e **`old`** sono di tipo **`record`**
- Un campo `c` di un record è denotato da **`new`**`.c` o **`old`**`.c`
- L'uso di tali variabili ha senso solo in funzioni invocate da trigger definiti **`for each row`**

Trigger PL/pgSQL: valore di ritorno

- Una funzione invocata da un trigger deve restituire un valore attraverso l'istruzione **`return`** (≠ returns all'inizio, infatti all'inizio di una UDF per trigger metto returns trigger)
- In trigger **`before`** e **`for each row`**, il valore restituito può essere **`null`** (operazione annullata) o un record, che dev'essere **`old`** per trigger di cancellazione
- In trigger di inserimento/aggiornamento, il record restituito è quello che viene effettivamente scritto nella base di dati (sarà **`new`** se non è modificato dalla funzione)
- Se il trigger è definito **`after`** oppure **`for each statement`**, il valore di ritorno è ignorato, quindi si può usare **`return null`**
- Per ulteriori dettagli, fare riferimento al Cap. 36 del manuale di PostgreSQL

Trigger: esempio

"Nessun dipartimento può avere più di sei impiegati"

```
create or replace function controlla_num_dip()
returns trigger language plpgsql as
$$
  declare
    n integer;
  begin
    select count(*) into n from Impiegato I
     where new.dip = I.dip;

    if n >= 6 then
      raise notice 'Numero massimo raggiunto';
      return null; -- Annulla l'operazione
    end if;

    return new;
  end;
$$;
```

Trigger: esempio (cont.)

La funzione precedente va invocata

- quando si inserisce un nuovo impiegato
- quando si modifica il dipartimento d'afferenza di un impiegato (perché non lo si può spostare in un dipartimento che ha già il numero massimo d'impiegati)

```
create trigger controlla_impiegato
before insert or update on Impiegato
for each row
execute procedure controlla_num_dip();
```