Trigger e gestione delle eccezioni con PL/pgSQL

Laboratorio di basi di dati
Stefano Montanelli
Dipartimento di Informatica
Università degli Studi di Milano
http://islab.di.unimi.it/bdlab1

Definizione di trigger

- Un TRIGGER è uno strumento del DBMS per rendere *attivo* il comportamento di una base di dati, innescando l'esecuzione di una funzione al verificarsi di uno specifico evento
- PL/pgSQL aderisce allo standard SQL-3 per la definizione di trigger pur implementando un sottoinsieme delle funzionalità in esso previste

```
CREATE TRIGGER <nome_trigger>
{ BEFORE | AFTER } <evento>
ON <nome_tabella>
[ FOR [ EACH ] { ROW | STATEMENT } ]
EXECUTE PROCEDURE <nome_funzione>
```

Clausole dell'istruzione

- La clausola <evento> specifica l'evento che innesca il trigger

  evento = { INSERT | UPDATE | DELETE }

  - E' possibile associare il trigger a più di un evento combinando gli eventi mediante l'operatore OR
- L'opzione `{ROW | STATEMENT}` definisce quante volte l'azione associata al trigger (`<nome_funzione>`) deve essere eseguita
  - `FOR EACH ROW`. <nome_funzione> viene eseguita per ogni tupla interessata dall'evento che ha sollevato il trigger
  - `FOR EACH STATEMENT` (default). <nome_funzione> viene eseguita per ogni istruzione SQL derivata dall'evento che ha sollevato il trigger

Visualizzazione ed eliminazione

- Per visualizzare la definizione di un trigger:

```
\d <nome_tabella>
```

  dove `<nome_tabella>` è la tabella su cui il trigger è definito

- Per eliminare un trigger:

```
DROP TRIGGER <nome_trigger>
ON <nome_tabella>
```

Azioni associate ai trigger

- Le azioni associate al trigger possono essere specificate mediante una funzione
  - La funzione non può avere parametri
  - Il tipo di dato restituito è TRIGGER

```
CREATE [OR REPLACE] FUNCTION
      <nome_funzione>() RETURNS
TRIGGER
AS $$
      <corpo_funzione>;
$$ LANGUAGE plpgsql
```

- La funzione termina con un'istruzione che restituisce NULL o una tupla del medesimo tipo di quella che ha sollevato il trigger

Le variabili NEW e OLD

- PostgreSQL predispone due variabili speciali che possono essere utilizzate nel corpo di una funzione di tipo trigger
  - **NEW**: è una variabile di tipo RECORD. E' disponibile in caso di trigger sollevati da un evento INSERT/UPDATE. E' NULL in caso di trigger sollevati da altri eventi
  - **OLD**: è una variabile di tipo RECORD. E' disponibile in caso di trigger sollevati da un evento UPDATE/DELETE. E' NULL in caso di trigger sollevati da altri eventi

Contenuto delle variabili NEW e OLD

- Trigger sollevato da evento INSERT
  - NEW contiene il record della tupla inserita
  - OLD è NULL
- Trigger sollevato da evento UPDATE
  - NEW contiene il record della tupla DOPO le modifiche di update
  - OLD contiene il record della tupla PRIMA delle modifiche di update
- Trigger sollevato da evento DELETE
  - NEW è NULL
  - OLD contiene il record della tupla eliminata

Messaggi ed eccezioni

- Mediante il comando RAISE è possibile visualizzare messaggi e sollevare eccezioni

```
RAISE <livello> '<messaggio>' [, <espressione> [, ...]];
```

  - `<livello>` indica una sequenza di azioni da eseguire che vanno dalla visualizzazione di specifici messaggi fino all'interruzione della procedura che solleva l'eccezione
  - `<messaggio>` è la stringa associata all'evento sollevato
  - `<espressione>` è una (eventuale) lista di parametri per il messaggio

Livello dei messaggi

- Livelli di RAISE
  - **DEBUG**: messaggio loggato sullo standard output solo se il DBMS è in debug mode
  - **LOG**: messaggio loggato solo sul file di log di PostgreSQL
  - **INFO**: messaggio loggato solo su standard output
  - **NOTICE**: messaggio loggato su file di log di PostgreSQL e su standard output
  - **WARNING**: messaggio loggato su periferiche di debug, file di log di PostgreSQL e standard output
- Questi livelli di RAISE non causano l'interruzione della procedura in esecuzione

Gestione di eccezioni

- La sezione `EXCEPTION WHEN` può essere utilizzata per gestire le eccezioni sollevate durante l'esecuzione del codice
- La clausola WHEN specifica il tipo di eccezione che si intende gestire
  - E' possibile specificare una clausola WHEN per ogni tipologia di eccezione che si intende gestire
  - Si veda il manuale del PostgreSQL per un elenco completo delle tipologie di errore (PostgreSQL Error Codes)
  - Esempi: `unique_violation`, `foreign_key_violation`, `not_null_violation`