

Trigger / 24-05

DBMS tradizionali → possibili, esistono una quantità

→ spesso si necessita di) : l'DBMS esegue può → diversi
composte reattive) copiose da delle eseguire

Qui possiamo definire delle regole attive delle trigger

es. si vuole che se sotto un certo valore di un prodotto in magazzino, se ne ordino 100

Si aggiunge alla logica applicativa. Le funzionalità implementate dei trigger sarebbero altrimenti delegati ai programmi applicativi
→ basta definire un'unica regola, che non deve essere ripetuta e può essere condivisa, evitando i controlli da parte delle app. corrispondenti (interrogazioni → polling) per vedere lo stato del dato di interesse e agire di conseguenza

I sistemi sono spessoeterogenei sui trigger

Rispetto allo standard

↓
= meno
efficienza

L'ottimizzazione quindi c'è estendere il codice del programma

applicativo → ciò fa perdere il utilizzo del codice

→ se userà viene inserita la regola o un problema

I trigger permettono di specificare un'azione automatica data dalla verifica di certi eventi (inserimento, modifica, cancellazione) o combinatori di stati del DB

Il trigger descrive la reazione del sistema (specifico: i DDL) appartenente alla logica applicativa del DBMS

Per diagramma ECA evento, condizione, azione

ON evento

IF condizione

THEN azione

- evento: qualcosa che si verifica in un certo istante di tempo (comandi solo su tabella / visto)
- condizione: controllo in più (oltre la registrazione dell'evento)
- azione: sequenza di operazioni vengono eseguite il trigger è solo considerato (attivato) e la condizione è vera (es. comandi solo)

Verificare l'evento è più rapido che controllare una condizione, sui dati già inseriti, da sola. Inoltre per stesse condizioni ma eventi diversi possono imporre azioni diverse

CREATE TRIGGER <name>

E { BEFORE | AFTER | INSTEAD OF } { <evento> [OR <evento>] }

ON <tabella soggetto>

C [WHEN (<condizioni>)]

A EXECUTE function <diverse funzioni>

insert etc...

UPDATE (aggiorna etc...)

futuramente li modificherò tutti

- precedo / seguo (^{eseguo} o ^{non}) rispetto all'evento (e condizionare del trigger)
- esprimibile booleana scruplice con BEFORE / AFTER (in caso no solo query)

L'azione nei trigger in Postgre necessita la creazione di una funzione

Per il BEFORE: se non vero la condizione del trigger, non eseguo il comando nell'evento!

Per INSTEAD OF: al posto dell'evento eseguo la funzione del trigger → vale solo ~~viene~~ !

Così BEFORE si consiglia l'affornamento dei dati con il trigger

Modalità di esecuzione

- esecuzione orientata all'istante: viene eseguita una volta per ogni tuple coinvolta (variabile o tuple di transazione)
FOR EACH ROW
- esecuzione " " all'insieme: " " una sola volta per l'insieme di tuple coinvolte (tabelle di transazione)
FOR EACH STATEMENT (default)

All'interno della condizione del trigger posso far riferimento alle versioni delle tuple coinvolte nell'evento (prima e dopo il trigger)

\rightarrow	OLD	NEW	OLDTABLE	NEWTABLE
\downarrow	\downarrow	\downarrow	\downarrow	\downarrow
insert	delete			una x update

alias dichiarati con REFERENCES (v.10)

Sintesi: completo - slide 24

Trigger function

f. senza parametri che restituisce il tipo trigger
creata con CREATE FUNCTION. Posso comunque passare argomenti
(parametri attuali, no formal). Queste trigger fanno essere solo
più trigger

es

CREATE TRIGGER modificaValNull
AFTER INSERT ON Film ^{una sera TABEUS!}
↳ non mi interessa cancellare (o insert per qualche motivo)
FOR EACH ROW

eseguite quando
è inserito
il tuple

WHEN (NEW.valore IS NULL)
EXECUTE FUNCTION modificaVal()

CREATE FUNCTION modificaVal ()

RETURNS trigger AS

\$\$ [c'è un errore di commento] \$\$

UPDATE Film

SET valore = (SELECT AVG(valore)*1,1 FROM Film)

↳ provabile cambiare fra le tuple

WHERE filo = NEW.filto AND regista = NEW.regista
over riferis
allo tuple
open insert

DB & LANGUAGE plpgsql

funzione che prende A tuple

es

CREATE TRIGGER modificaVolNull
AFTER INSERT ON Film ^{ma se non inserisco!}

X.1.10 REFERENCING NEWTABLE AS NT

FOR EACH STATEMENT

! WHEN EXISTS (SELECT * FROM film WHERE valutazione IS NULL) ^{NT}
EXECUTE FUNCTION modificaVol()

V. < 10, non potendo agire sulle singole tuple
dell'insieme, posso solo vedere "se esiste" una
tuple nella sottoquery

ma PostgreSQL non ammette sottoquery nella condizione

quindi la condizione la verifico nella funzione: (a cura della funzione)

CREATE FUNCTION modificaVol ()

RETURNS trigger AS

\$[c stringa di commento]\$

IF (EXISTS [...])

THEN UPDATE film

SET (come prima)

WHERE valutazione IS NULL

DB & LANGUAGE plpgsql

(v.10) → le condizioni sono quelle dell'es precedente
ma i from sono su NT

Per ottenere info sui trigger vi sono altre variabili:
(nome, n° argomenti etc...)

Vedere di ritorno del trigger

→ NULL (default)

→ TUPA : deve avere lo schema del trigger da lui
disegnato, generato sulle tabella

→ NEW/OLD sicuramente ha lo stesso schema le tuple

utilizzati solo x trigger BEFORE EACH ROW (^{oltremodo} non si forma l')

→ return NULL, l'evento non viene eseguito

→ altrimenti, la tupla dev'essere inserita e' poi ritornata

<slide 31>

I trigger permettono d. implementare i vincoli d. integrità non
effettuabili col control

/ 26 - 05

Modello di esecuzione

Come si comporta il DBMS di fronte a trigger "a colpo"?

È un processo selettivo, in base allo standard vengono attivati dopo l'esecuzione del comando SQL.

In che ordine vengono attivati? In base a:

- tipo di trigger (before/after)
- mod di esecuzione (row/statement)
- priorità (Postgre in base all'ordine alfabetico)

Bisogna tenere conto anche delle violazioni dei vincoli, vengono verificati prima, ottenendo prima di non violare

d'esecuzione di un trigger "estende" sempre la transazione da lui generato l'eventuale ha attivato il trigger.

modelli di esecuzione

In caso di troppe operazioni sullo stesso
transazione viene segnalato

Esecuzioni in cascata

il nuovo trigger:

- modalità iterativa → viene messo invece agli altri trigger da considerare
- modalità ricorsiva → esecuzione rispetto all'ultimo attivato

Trigger per vincoli di integrità

CONSTRAINT TRIGGER, viene attivato ma eseguito a fine transazione (detelt). Sono solo FOR EACH ROW
Può essere immediato o differito.

Accoppiamento esterno

- Connessione al DB
- Esecuzione dei comandi SQL
- Chiusura della connessione (i server hanno un limite di connessioni aperte)

Nella fase di preparazione il DBMS prepara il piano di esecuzione, compilando e ottimizzando il comando, utile se viene chiesto più volte. Infine viene chiamata l'applicazione e il comando viene eseguito. Il risultato può essere poi manipolato dal linguaggio di programmazione.