09-11-2022

Ripasso trigger

- `for each statement`, data un'istruzione per un evento, tutto il blocco di record che viene interessato da quell'istruzione viene considerato un'**unica volta nel trigger**.
- `for each row`, per ogni riga aggiornata, il trigger sarà **eseguito per ogni riga** e si ha `old` o `new`, considerate come vere e proprie tabelle virtuali utilizzabili.
- **TEC**: struttura che **racchiude le info per l'esecuzione dei trigger**. Ogni volta che un trigger esegue operazioni, e queste operazioni scatenano altri trigger, questi altri trigger vengono valutati in un nuovo `TEC`, figlio del `TEC` di partenza.

> Se **tutti i TEC si chiudono** si è in stato quiescente e i **trigger terminano correttamente**.
> Se invece si raggiunge una **profondità di ricorsione elevata** (dettata dal DB in base al tempo necessario) allora si ha **errore**.

Proprietà formali

E' importante garantire che l'interferenza tra trigger in una qualunque loro attivazione non produca comportamenti anomali. Il *rollback* è fatto parzialmente quindi serve garantire che il database non sia in uno stato parzialmente valido:

- **Terminazione**: per un qualunque stato iniziale e una qualunque transazione, **si produce uno stato finale (*stato quiescente*)**;
- **Confluenza**: L'esecuzione dei trigger termina e produce un **unico stato finale**, indipendente dall'ordine di esecuzione dei trigger. Se eseguo un trigger e questo causa una catena di trigger, comunque vado ad eseguire la catena, l'ordine deve dare sempre lo stesso risultato. Il database riordina la sequenza dei trigger per rendere efficiente l'operazione.
- **Univoca osservabilità**: I trigger sono confluenti e producono verso l'esterno lo stesso effetto (messaggi, azioni di display)..

Grafo di triggering

Per vedere **se una serie di trigger terminerà**, si usa grafo di triggering. Esso è formato da:

- Un nodo per ogni trigger
- Un arco dal nodo t_i al nodo t_j se l'esecuzione dell'azione di t_i può attivare il trigger tj (ciò può essere dedotto con una semplice analisi sintattica)
- Se il **grafo è aciclico**, l'**esecuzione termina** ma questa non è l'unica condizione che si ha a disposizione
- Non possono esservi sequenze infinite di triggering
- Se il **grafo ha cicli**, esso può avere problemi di terminazione: lo si capisce guardando i cicli uno per uno.

Esempio:

```
T1: create trigger AdjustContributions
        after update of Salary on Employee
        referencing new table as NewEmp
        update Employee
        set Contribution = Salary * 0.8
        where RegNum in ( select RegNum from NewEmp)

T2: create trigger CheckBudgetThreshold
        after update on Employee
        referencing new_table as NewEmp1
        when 50000 < ALL (select (Salary+Contribution)
                                        from NewEmp1)
        update Employee
        set Salary = 0.9*Salary
```

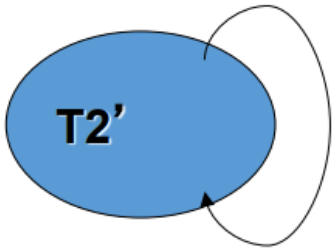
> In questo caso, **si ha un ciclo MA** esso finirà prima o poi. Ad un certo punto non soddisfo più la condizione della `when 50000 < ALL {...}` e **termina** perchè il salario lo vado a ridurre ogni volta.

```
![[Pasted image 20221109114456.png|380]]
```

Se `T2'` fosse stato il seguente:

```
T2': create trigger CheckBudgetThreshold
         after update on Employee
         for each row when New.Salary < 50000
         update Employee
         set Salary = 0.9*Salary
```

In questo caso, nonostante abbiamo un ciclo, il **trigger non termina**



Esercizio

Dato lo schema relazionale:

- **IMPIEGATO** *(Nome, Salario, DipNum)*
- **DIPARTIMENTO** *(DipNum, NomeManager)*
- Definire le seguenti regole attive in Oracle e DB2:

1. una regola, che quando il dipartimento cancellato, mette ad un valore di default (99) il valore di *DipNum* degli impiegati appartenenti a quel dipartimento;
2. una regola che cancella tutti gli impiegati appartenenti a un dipartimento quando quest'ultimo cancellato;
3. una regola che, ogni qual volta il salario di un impiegato supera il salario del suo manager, pone tale salario uguale al salario del manager;
4. una regola che, ogni qual volta vengono modificati i salari, verifica che non vi siano dipartimenti in cui il salario medio cresce più del tre per cento, e in tal caso annulla la modifica.

```
create trigger T1
after delete on DIPARTIMENTO
        for each row
        when (exists (select *
                                from IMPIEGATO
                                where DipNum=Old.DipNum))
udpade IMPIEGATO set DipNum = 99
where DipNum=Old.DipNum
```

```
create trigger T2
after delete on DIPARTIMENTO
for each row
        when (exist (select *
                                from IMPIEGATO
                                where DipNum=Old.DipNum))
delete from IMPIEGATO where DipNum=Old.DipNum
```

```
create trigger T3
after update of Salario on IMPIEGATO
for each row
        declare x number;
         begin
                 select Salary into x
                 from IMPIEGATO join DIPARTIMENTO on
                 Nome = NomeManager
                 Where DIPARTIMENTO.DipNum = New.DipNum
                 if new.Salario > x then
                         update IMPIEGATO set Salario = x
                         where Nome = New.Nome
        end
```

```
create trigger T4
after update of Salario on IMPIEGATO
for each row
        declare x number;
        declare y number;
        declare l number;
        begin
                select avg(salario) into x, count(*) into l
                from IMPIEGATO
                where DipNum=new.DipNum;
                y=((x*l)-new.Salario+old.Salario)/l;
                if (x>(y*1.03)) then
                        update IMPIEGATO set Salario=old.Salario
                        where DipNum=new.DipNum;
        end
```

Esercizio

Dato lo schema relazionale:

- **DOTTORANDO** *(Nome, Disciplina, Relatore)*
- **PROFESSORE** *(Nome, Disciplina)*
- **CORSO** *(Titolo, Professore)*
- **ESAMI** *(NomeStud, TitoloCorso)*
- Descrivere i trigger che gestiscono i seguenti vincoli di integrità (*business rules*):
    1. ogni dottorando deve lavorare nella stessa area del suo relatore;
    2. ogni dottorando deve aver sostenuto almeno 3 corsi nell'area del suo relatore;
    3. ogni dottorando deve aver sostenuto l'esame del corso di cui responsabile il suo relatore

MYSQL

- MySQL, definito Oracle MySQL, è un **Relational Database Management System**, composto da un client con **interfaccia a riga di comando e un server**, entrambi disponibili sia per sistemi *Unix* o *Unix-like* come *GNU/Linux* che per *Windows*, anche se prevale un suo utilizzo in ambito *Unix*.
- Oggi l'ultima versione disponibile è la 8.0.x (*MariaDB 10.5*)
- Dal 1996 supporta la maggior parte della sintassi SQL e si prevede in futuro il pieno rispetto dello standard ANSI.
- Possiede delle interfacce per **diversi linguaggi**, compreso un driver *ODBC*, due driver *Java*, un driver per *Mono* e *.NET* ed una libreria per *Python*.
- Può gestire anche tabella con attributi di **tipo non relazionale**, quindi *non strutturati*.

MySQL mette a disposizione diversi tipi di tabelle (**"storage engine**") per la memorizzazione dei dati e ognuno presenta proprietà e caratteristiche differenti
Esiste una API che si può utilizzare per creare nuovi tipi di tabella che si possono installare senza necessità di riavviare il server.

Ci sono **due sistemi principali**:

1. **Transazionali**: sono più sicuri, permettono di recuperare i dati anche in caso di crash, e consentono di effettuare modifiche tutte insieme;
2. **Non transazionali**: sono più veloci, occupano meno spazio su disco e minor richiesta di memoria

Engine: MyIsam

- **MyISAM** era lo storage engine di default dal MySQL 3.23 fino al MySQL 5.4.
- **MyISAM** utilizza la struttura **ISAM** e deriva da un tipo più vecchio, oggi non più utilizzato, che si chiamava appunto **ISAM**.
- È un motore di immagazzinamento dei dati estremamente veloce e richiede poche risorse, sia in termini di memoria RAM, sia in termini di spazio su disco.
- Il suo limite principale rispetto ad alcuni altri SE consiste nel mancato supporto delle transazioni e alle foreign key.

- Ogni tabella MyISAM è memorizzata all'interno del disco con tre file:
  - un file **.frm** che contiene la definizione della tabella,
  - un file **.MYD** per i dati
  - un file **.MYI** per gli indici

Engine: InnoDB

- **InnoDB** è un motore per il salvataggio di dati per MySQL, fornito in tutte le sue distribuzioni (Default dalla versione 5.5).
- La sua caratteristica principale è quella di supportare le transazioni di tipo **ACID**.

- **InnoDB** mette a disposizione le seguenti funzionalità:
  - transazioni SQL con savepoint e transazioni XA;
  - lock a livello di record;
  - foreign key;
  - integrità referenziale;
  - colonne AUTOINCREMENT;
  - tablespace.
- InnoDB offre delle ottime performance in termini di prestazioni e utilizzo della CPU specialmente quando si ha a che fare con una grande quantità di dati.
- InnoDB può interagire tranquillamente con tutti gli altri tipi di tabelle in MySQL.

> Per **tablespace** si intende l'area di `innoDB` che astrae dal filesystem del sistema operativo. Esso maschera il `filesystem`.

- Le tabelle InnoDB sono soggette alle seguenti limitazioni:
  - Non è possibile creare più di **1000 colonne** per tabella;
  - Su alcuni sistemi operativi le **dimensioni del tablespace** non possono superare i **2 Gb**;
  - La grandezza di tutti i **file di log** di InnoDB deve essere inferiore ai **4 Gb**;
  - La grandezza minima di un tablespace è di **10 MB**;
  - Non possono essere creati indici di tipo **FULLTEXT** con **MySQL 5.5 o precedente**;
  - Le **SELECT COUNT(*)** su tabelle di grandi dimensioni possono essere molto lente.

> Gli indici `FULLTEXT` sono delle chiavi speciali, **permettono di cercare all'interno del testo**.

INSTALLAZIONE CON DOCKER

```
mkdir data #dove verranno memorizzati i dati
cd data
#Creo nuovo container
        docker run --name esempiodb -e MYSQL_ROOT_PASSWORD=password -d -p
3306:3306 -v /users/mestesso/data:/var/lib/ymsql mysql:latest

#-e variabile ambiente
#-d eseguito come demone, va in backround e lavora in backround
#-p porta 3306 mappata sulla 3306 del mio pc
#-v dove vengono salvati i dati, quindi cartella locale data.
```

--> il cui risultato è una stringa.

```
docker ps #fa vedere container in esecuzione
docker

docker exec -it esempiodb mysql -u root -p
show databases; #vede i db disponibili

#-it comando interattivo, mouse+keyboard
```

Come si accede al database

`USE "nome_db"`

`SHOW TABLES` mostra tutte le tabelle del database

> MYSQL è case sensitive.(?)