Declansatoare

Un declansator defineste o actiune care trebuie executata in baza de date la aparitia unei comenzi de tip insert, update sau delete.
Declansatoarele sunt similare procedurilor - sunt tot blocuri PL/SQL denumite. **Diferenta intre proceduri si declansatoare** consta in aceea ca o procedura este executata explicit dintr-un alt bloc printr-un apel care ii furnizeaza parametri in timp ce un declansator este executat implicit ori de cate ori apare evenimentul pentru care a fost definit (INSERT, UPDATE, DELETE). Declansatoarele nu accepta argumante.

Declansatoarele se utilizeaza pentru:

- impunerea respectării unor condiţii complexe de integritate (referenţială sau de comportament); dacă aceste constrângeri sunt prea complicate şi nu pot fi definite prin intermediul constrângerilor de integritate ale tabelelor, ele pot fi implementate cu ajutorul triggerelor;
- popularea coloanelor redundante în cazul tabelelor denormalizate (de exemplu, calculul unui total);
- transformarea într-un anumit format standard a datelor ce urmează a fi inserate în tabel; de exemplu, dacă se doreşte conversia tuturor numelor într-un format predefinit înainte de a fi inserate într-un tabel, triggerele pot furniza o conversie centralizată a acestora;
- culegerea de informaţii statistice în legătură cu accesarea tabelelor;
- întreţinerea sincronizată a copiilor tabelelor situate în diferite noduri ale unei baze de date distribuite; deşi există opţiuni de replicare Oracle, triggerele se pot folosi în mod extensiv pentru duplicarea datelor.

```
CREATE [OR REPLACE] TRIGGER nume_declansator
   {BEFORE | AFTER}  {INSERT | UPDATE [OF coloane] | DELETE}
    ON tabel_referit

  [REFERECING {NEW AS nume_calificativ | OLD AS nume_calificativ } ]

   [FOR EACH ROW  [WHEN conditii]]

 [DECLARE]                -- optional, pentru declararea  variabilelor locale

corpul_declansatorului
```

Denumirea declansatoarelor

Declansatoarele exista intr-un spatiu de nume(namespace) separat de proceduri, pachete, tabele (care partajeaza acelasi spatiu de nume), ceea ce inseamna ca un declansator poate avea acelasi nume cu un tabel sau procedura.

Tipuri de declansatoare

In Oracle exista doua tipuri de declansatoare:

- *la nivel de rând (FOR EACH ROW)*– declanşatorul se execută pentru fiecare rând în parte ce este afectat de instrucţiunea declanşatoare;
- *la nivel de comanda* – declanşatorul se execută o singură dată pentru instrucţiunea declanşatoare indiferent de numărul de linii afectate;

Declansatoarele pentru comenzi multiple INSERT, UPDATE, DELETE asupra unui tabel pot fi combinate intr-un singur declansator, cu conditia ca toate sa corespunda aceluiasi nivel (*la nivel de rând* sau *la nivel de comanda*) (ex. INSERT OR UPDATE OR DELETE).

Utilizarea pseudovariabilelor :old si :new pentru valorile noi şi vechi ale coloanelor
In cadrul unui declanşator la nivel de rand, se pot accesa noile şi vechile valori ale coloanelor rândului curent procesat de trigger. În acest context există două nume de corelare pentru valorile fiecărei coloane ale tabelului ce este actualizat: un nume pentru vechile valori şi un nume pentru noile valori. Dacă o instrucţiune are nevoie de o valoare dintr-o înregistrare nouă sau actualizată, se utilizează :new. Dacă o instrucţiune are nevoie de valoarea unei coloane înainte ca ea să fi fost modificată, se utilizează :old. În funcţie de tipul comenzii declanşatoare, anumite nume de corelare nu pot avea nici o semnificaţie. În cazul unei instrucţiuni `INSERT`, vechile valori ale coloanelor au valoarea *null*, în timp ce în cazul instrucţiunii `DELETE` sunt *null* noile valori ale coloanelor.

Numele de corelare se pot utiliza atât în corpul declanşatorului cât şi în clauzele opţionale `WHEN` şi `REFERENCING`. Cele două puncte care preced cuvintele cheie `new` şi `old` sunt obligatorii atunci când acestea se utilizează în interiorul declanşatorului. Cele două puncte nu sunt permise atunci când se utilizează cuvintele cheie `new` şi `old` în clauzele opţionale `WHEN` şi `REFERENCING`. Cuvintele cheie `new` şi `old` nu pot fi utilizate cu coloane de tip `LONG` şi `LONGRAW`.

Opţiunea `REFERENCING` poate fi specificată în interiorul unui declanşator la nivel de rând pentru a evita conflictele de nume ce apar între numele de corelare şi o tabelă a bazei de date ce este denumită `new` sau `old`. Conflictele de nume ce apar se datorează faptului că `new` şi `old` nu sunt cuvinte rezervate.

```
REFERENCING new AS nou
```

`Acum,` noile valori ale coloanelor rândului curent procesat de trigger sunt referite utilizând numele de corelare `nou`, în loc de `new`.

Condiţia de declanşare

Clauza WHEN

In clauza WHEN (opţionala) se specifică, intre paranteze rotunde, o expresie booleană care trebuie să fie adevărată pentru ca declanşatorul să fie activat. Această condiţie nu poate să fie o condiţie PL/SQL, ci trebuie să fie o condiţie SQL ce nu poate conţine o subinterogare. Spre deosebire de celelalte secţiuni ale declanşatorului, această secţiune poate lipsi.

```
Exemplu: (WHEN new.nume is not null)
```

În momentul în care clauza `WHEN` este inclusă, expresia este evaluată pentru fiecare înregistrare afectată. Dacă restricţia de declanşare are valoarea `false` sau este necunoscută acţiunile conţinute de un declanşator nu sunt executate.

Clauza WHEN este validă doar pentru declanșatoarele la nivel de rând. Dacă aceasta este prezentă, corpul declanșatorului va fi executat doar pentru acele rânduri care corespund condiției specificate în clauză.

Numele de corelare new si old se pot utiliza şi în clauzele opţionale `WHEN` şi `REFERENCING`, însă fără a fi prefixate de cele două puncte(:).

Utilizarea predicatelor de declanșare: INSERTING, UPDATING și DELETING
Se pot utiliza trei funcții booleene pentru a determina ce operație se efectuează (deoarece un declanșator se poate adresa mai multor evenimente, uneori este necesară determinarea cărei operații se executa în momentul curent).

Elementele din dictionarul de date corespunzatoare declansatoarelor

USER_TRIGGERS

Comenzi corespunzatoare declansatoarelor
• DROP TRIGGER nume_decl
• ALTER TRIGGER nume_decl [{DISABLE | ENABLE }]
• ALTER TABLE nume_tabel [{DISABLE | ENABLE } [ALL TRIGGERS]]

APLICATIE

SQL>create tablespace ts_student datafile ‘d:\date\fd_student.dbf’ size 10M;
SQL> create temporary tablespace ts_temp_std tempfile ‘d:\date\temp_student.dbf’ size 5M;
SQL>create user student identified by student

```
        default tablespace ts_student
        temporary tablespace ts_temp_std
        quota unlimited on ts_student;

SQL> grant resource to student;
```

Rolul **RESOURCE** acordă utilizatorilor următoarele privilegii de sistem:

```
 CREATE CLUSTER
 CREATE PROCEDURE
 CREATE SEQUENCE
 CREATE TABLE
 CREATE TRIGGER
 CREATE TYPE
```

Sau

grant create trigger to *utilizator*

grant create sequence to *utilizator*

```
create table studenti
( id int primary key,
  nume varchar(20),
  codFac char(10),
  nota1 int check (nota1 between 1 and 10),
  nota2 int check(nota2 between 1 and 10),
  nota3 int check(nota3 between 1 and 10),
  media numeric(4,2)
)


create or replace trigger calcul_medie
  before insert or update  on studenti
 for each row
begin
  if ( :new.nota1>=5 and :new.nota2>=5 and :new.nota3>=5)
   then :new.media:=trunc(( :new.nota1+ :new.nota2+ :new.nota3)/3.0,2);
   else :new.media:=null;
  end if;
end;

insert into studenti(id,nume,codFac,nota1,nota2,nota3) values(1,'ion','MI',10,9,10);
insert into studenti(id,nume,codFac,nota1,nota2) values(2,'ionut','MI',10,9);
insert into studenti(id,nume,codFac,nota1,nota2,nota3) values(3,'dan','EC',10,4,10);
select * from studenti;
update studenti set nota2=7 where id=3;
select * from studenti;
```

```
create sequence seq_id_std increment by 1 start with 1;

create or replace trigger calcul_medie
  before insert or update  on studenti
for each row
begin
  if inserting  and  :new.id is null
   then
    :new.id:=seq_id_std.nextval;
   end if;
  if  :new.nota1>=5 and :new.nota2>=5 and :new.nota3>=5
   then :new.media:=trunc(( :new.nota1+ :new.nota2+ :new.nota3)/3.0,2);
   else :new.media:=null;
  end if;
end;


insert into studenti(nume,nota1,nota2,nota3) values('marius',10,9,10);


create or replace trigger calcul_medie
  before insert or update  on studenti  for each row
WHEN (new.nume is not null)
begin
  if inserting and :new.id is null
   then
    :new.id:=seq_id_std.nextval;
   end if;
  if ( :new.nota1>=5 and :new.nota2>=5 and :new.nota3>=5)
   then :new.media:=trunc(( :new.nota1+ :new.nota2+ :new.nota3)/3.0,2);
   else :new.media:=null;
  end if;
end;
```

```
create or replace trigger calcul_medie
  before insert or update   on studenti   for each row

DECLARE
  nume_eronat EXCEPTION;
begin
    if length(:new.nume)<3
       then  raise nume_eronat;  end if;
  else
  begin
    if inserting and :new.id is null
       then   :new.id:=seq_id_std.nextval;    end if;

   if ( :new.nota1>=5 and :new.nota2>=5 and :new.nota3>=5)
      then :new.media:=trunc(( :new.nota1+ :new.nota2+ :new.nota3)/3.0,2);
      else :new.media:=null;
   end if;
  end
  EXCEPTION   WHEN nume_eronat THEN
     dbms_output.put_line('Nume eronat');
end;
```

```
create or replace trigger nota_max
 after insert or update or delete on studenti
declare
  max1 int;
  max2 int;
  max3 int;
begin
  select max(nota1),max(nota2),max(nota3) into max1,max2,max3 from studenti;
  dbms_output.put_line('note maxime:'||max1||' '||max2||' '||max3);
end;
```

CURSOARE

I)

```
Declare
 cursor c1 is select * from studenti;
 rand c1%rowtype;
BEGIN
  open c1;
  LOOP
    fetch c1 into rand;
    exit when c1%notfound;
    dbms_output.put_line(rand.codFac||' '||rand.nume||' '|| rand.media);
  END LOOP;
  close c1;
END
```

II)

```
Declare
 cursor c2 is select * from studenti;
 BEGIN
   FOR rand in c2  LOOP
     dbms_output.put_line(rand.codFac||' '||rand.nume||' '|| rand.media);
  END LOOP
END
```

Observații

- rand este implicit de tipul c2%rowtype
- cursorul este deschisb automat de FOR
- cursorul este automat inchis la finalizarea lui FOR

III) Cursoare cu parametri

```
Declare
  Facultate studenti.codFac%type;

 cursor c3(cod_Fac studenti.codFac%type)
          is  select * from studenti where codFac=cod_Fac;

BEGIN
  Facultate:='MI';
  FOR rand in c3(Facultate)  LOOP
       dbms_output.put_line(rand.codFac||' '||rand.nume||' '|| rand.media);
  END LOOP;
END
```