



Les Déclencheurs automatiques : Triggers

Définition & généralité

-Les déclencheurs (*triggers*) existent depuis la version 6 d'Oracle.

Rq :Un trigger = objet de BD qui déclenche un programme quand un événement (insertion, modification...) se produit dans la BD.

Définition : Un trigger est une séquences d'actions, définis par le programmeur, qui se déclenche lors de l'occurrence d'un événement particulier

- le déclencheur « se déclenche » automatiquement

-La majorité des déclencheurs sont programmés en PL/SQL, mais il est possible d'utiliser un autre langage

- C est un objet stocké

- Les événement déclencheurs :
 - une instruction INSERT, UPDATE, DELET sur une table ou une vue
on parle de déclencheurs LMD
 - une instruction CREATE, ALTER, ou DROP sur un objet (table ...). On
parle de déclencheur LDD
 - ...

- **Les étapes :**

- coder le déclencheur
- le compiler

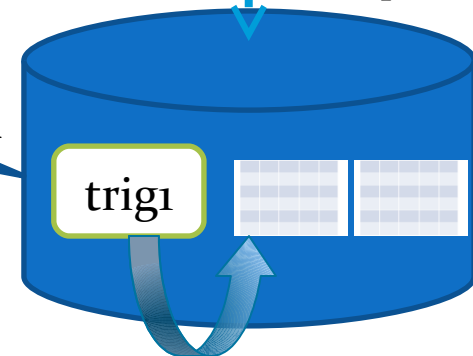
➔ si le déclencheur est actif
chaque événement qui caractérise
le déclencheur aura pour
conséquence son exécution

1- **déclaration**

```
CREATE OR REPLACE TRIGGER  
trig1  
{BEFORE|AFTER| INSTEAD OF}  
Nature de l'événement e1  
Bloc PL/SQL
```

2- **compilation**

3- **événement e1**



4- **Action codé dans trig1**

Conditions pour créer et modifier un trigger

- **Conditions nécessaires pour créer un trigger dans votre schéma :**

- disposer des privilège **CREATE TRIGGER**

- **Pour créer un déclencheur dans un autre schéma :**

le privilège **CREATE ANY TRIGGER** est requis.

sur la table sur laquelle on veut définir le trigger il faut :

- ➔ être propriétaire de la table

- ➔ ou posséder le privilège ALTER sur la table

- ➔ ou posséder le privilège ALTER ANY TABLE

- **Modification de triggers**

- on refait une instruction CREATE OR REPLACE TRIGGER

- ou bien on supprime le trigger : DROP TRIGGER nomtrigger
et on le crée à nouveau.

Création de triggers

- Création du trigger

```
CREATE [OR REPLACE] TRIGGER nom_trigger
```

```
{BEFORE|AFTER | INSTEAD OF}
```

précise le moment de l'exécution du trigger

```
{INSERT|DELETE|UPDATE [OF col1 ..]} ON <[schéma.]nomTable|nomVue>
```

```
[FOR EACH ROW ]
```

```
[WHEN (<condition>)]
```

```
DECLARE ..... <<<<déclarations>>>>
```

```
BEGIN
```

```
..... <<<< bloc d'instructions PL/SQL>>>>
```

```
END;
```

nom-trigger :doit être unique dans un même schéma

Pour les vue

table ou vue associé au déclencheur

Cœur du trigger

nature de l'événement

- **BEFORE | AFTER | INSTEAD OF** précise la chronologie entre l'action à réaliser par le déclencheur LMD et la réalisation de l'événement (exemple BEFORE INSERT programmera l'exécution du déclencheur avant de réaliser l'insertion).
- **DELETE | INSERT | UPDATE** précise la nature de l'événement pour les déclencheurs LMD.
- **ON {[schéma.] nomTable | nomVue}** spécifie la table, ou la vue, associée au déclencheur LMD.
- **WHEN** conditionne l'exécution du déclencheur.

- la taille d'un déclencheur ne peut excéder 32 ko == > conseille : limiter la taille (partie instructions) d'un déclencheur à soixante lignes de code PL/SQL

=> contournement : appeler des sous- programmes dans le code du déclencheur.

- Un déclencheur ne peut valider aucune transaction

=> les instructions suivantes sont interdites :

COMMIT, ROLLBACK, SAVEPOINT et SET CONSTRAINT.

- Attention à ne pas créer de déclencheurs récursifs

Application

Ecrire un trigger qui affiche un message **après** chaque suppression de la table **Employees** du compte hr.

SQL>

```
CREATE OR REPLACE TRIGGER aff_discount  
BEFORE INSERT OR UPDATE ON clients
```

```
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE(' bonjour ' );
```

```
END;
```

- **Déclencheur de ligne**

➔ déclaré avec la directive : **FOR EACH ROW**

exemple de trigger

-- trigger déclenché lors d'une insertion ou d'une modification de la table client

```
SQL> CREATE OR REPLACE TRIGGER aff_discount  
BEFORE INSERT OR UPDATE ON clients  
FOR EACH ROW  
--WHEN (no_cli > 0)  
BEGIN  
    DBMS_OUTPUT.PUT_LINE(' bonjour ' );  
END;
```

-- FOR EACH ROW signale qu'une modification de 4 lignes

-- par un seul UPDATE déclenche 4 fois le trigger.

-- Si on ne souhaite qu'un seul déclenchement ,

-- on omet simplement la clause FOR EACH ROW

Les directives

:NEW et :OLD

les colonnes de la ligne en cours de modification sont accessibles par l'intermédiaire de 2 directives de type enregistrement **OLD** et **NEW**

OLD représente la valeur avant modification

L'ancienne valeur est appelée :

:old.nomcolonne

OLD n'est renseignée que pour les ordres DELETE et UPDATE. Elle n'a aucune signification pour un ordre INSERT, puisqu'aucune ancienne valeur n'existe

NEW représente la nouvelle valeur

✓ La nouvelle valeur est appelée :

:new.nomcolonne

NEW n'est renseignée que pour les ordres INSERT et UPDATE. Elle n'a aucune signification pour un ordre DELETE, puisqu'aucune nouvelle valeur n'existe

Remarque : Ces 2 directives ne marchent qu'en présence de for reach row₁₁

Exemple d'utilisation de :NEW

- un pilote ne doit pas être qualifié sur plus de trois types d'avions

Pilote				
Brevet	nom	nbhvol	comp	nbqualif
PL-1	J.M toto	450	AF	3
PL-2	T. Guibert	3400	AF	1
PL-3	Michel	900	SING	1
	Tuf			

Qualifications

brevet	typeA	expire
PL-1	A340	22/06/2005
PL-1	A340	05/02/2005
PL-1	A320	16/01/2004
PL-2	A320	18/01/2004
PL-3	A330	22/01/2006

Programmons le déclencheur **TrigInsQualif** qui surveille les insertions arrivant sur la table Qualifications et incrémente de 1 la colonne nbQualif pour le pilote concerné, ou refuse l'insertion pour le pilote ayant déjà trois qualifications (cas du pilote de code 'PL-1' dans la figure suivante).

- question : comment gérer des insertion dans la table Qualifications pour répondre à la contrainte ci-dessus

```
insert into Qualification values('PL-2','A380','20-06-2006');
```

←
:New.brevet

Exemple d'utilisation de :NEW

L'exécution de cette instruction :

insert into Qualifications values('PL-2','A380','20-06-2006');
doit déclencher le trigger ci-après

CREATE OR REPLACE TRIGGER T_INSQualif

- (1) **BEFORE INSERT**
- (2) ON Qualifications
- (3) FOR EACH ROW

DECLARE

v_compteur Pilote.nbqualif %type;

v_nom Pilote.nom %type; Déclaration des variables
locales.

BEGIN

SELECT nbqualif, nom into v_compteur, v_nom from Pilote where brevet=:New.brevet;

if v_compteur < 3 THEN

Update Pilote set nbqualif= nbqualif + 1
where brevet=:New.brevet;

else

RAISE_APPLICATION_ERROR(-20100, 'le pilote' || v_nom || ' a déjà 3 qualifs');

END IF;

END;


Événement déclencheur est :

before insert car il faudra
s'assurer, avant de faire
l'insertion, que le pilote n'est
pas déjà qualifié sur trois types
d'appareils.

Corps du déclencheur.

Renvoi d'une erreur utilisateur.

On utilise un déclencheur FOR EACH ROW car
on désire qu'il s'exécute autant de fois qu'il y a de
lignes concernées par l'événement déclencheur.



Pour que la cohérence soit plus complète, il faudrait aussi programmer le déclencheur qui décrémente la valeur de la colonne nbQualif pour chaque pilote concerné par une suppression de lignes dans la table Qualifications. Il faut raisonner ici sur la directive :OLD.

Programmons le déclencheur TrigDelQualif qui surveille les suppressions de la table Qualifications et décrémente de 1 la colonne nbQualif pour le pilote concerné par la suppression de sa qualification.

```
CREATE TRIGGER T_delQualif
AFTER DELETE ON Qualifications
FOR EACH ROW
BEGIN
    UPDATE Pilote set nbqualif =nbqualif-1
    where  brevet=:OLD.brevet;
END;
```

Tableau 7-32 Test du déclencheur

Événement déclencheur	Sortie SQL*Plus
SQL> DELETE FROM Qualifications WHERE typa = 'A320';	2 ligne(s) supprimée(s). SQL> SELECT * FROM Pilote; BREVET NOM NBHVOL COMP NBQUALIF ----- PL-1 J.M Misztela 450 AF 2 PL-2 Thierry Guibert 3400 AF 0 PL-3 Michel Tuffery 900 SING 1

Exemple de trigger avec exception :

```
CREATE TRIGGER T_INSQualif
  BEFORE INSERT ON Qualifications
  FOR EACH ROW
  DECLARE
    v_compteur Pilote.nbhvol%type;
    v_nom Pilote.nom%type;
  BEGIN
    SELECT nbqualif, nom into v_compteur, v_nom from Pilote where brevet=:New.brevet;
    if v_compteur <3 THEN
      Update Pilote set nqualif= nqualif + 1 where brevet=:New.brevet;
    else
      RAISE_APPLICATION_ERROR(-20100, 'le pilote' || v_nom ' a déjà 3 qualifs');
    END IF;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR(-20101, 'pas de pilote de code ' || :NEW.brevet);
    WHEN OTHERS
      RAISE_APPLICATION_ERROR(-20102, 'erreur oracle');
  END;
```

• Possibilité de restreindre l'exécution du trigger : **WHEN**

```
CREATE OR REPLACE TRIGGER T_INSQualif
  BEFORE INSERT ON Qualifications
  FOR EACH ROW
  WHEN (NEW.typeA='A320' OR NEW.typeA='A340')
  DECLARE
    v_compteur Pilote.nbhvol%type;
    v_nom Pilote.nom%type;
  BEGIN
    SELECT nbqualif, nom into v_compteur, v_nom from Pilote where
    brevet=:New.brevet;
    if v_compteur <3 THEN
      Update Pilote set nqualif= nqualif + 1 where brevet=:New.brevet;
    else
      RAISE_APPLICATION_ERROR(-20100, 'le pilote' || v_nom ' a déjà 3 qualifs');
    END IF;
  END;
```

Notez que dans la condition WHEN, les « pseudo enregistrements » **NEW** et **OLD** s'écrivent sans le symbole « : ».

Application : FOR EACH ROW & OLD

Créons un déclencheur qui ne fait qu'afficher le numéro et le nom d'un employé (avant son suppression) que l'on veut supprimer de la table employé

-Annuler la transaction, après suppression

Le résultat et message du test :

```
SQL> set serveroutput on
SQL> delete from emp where empno = 7369
  Suppression de l'employé n° 7369, son nom est SMITH 1 ligne supprimée.
SQL> rollback; Annulation (rollback) effectuée.
```

Application

La DRH annonce que désormais, tout nouvel employé devra avoir un numéro supérieur ou égal à 10000, créer un trigger qui interdit (déclenche un message d'erreur) toute insertion qui ne reflète pas cette nouvelle directive

Résultat du test

Tentons d'insérer un nouvel employé avec le numéro 9999

```
SQL> insert into emp (empno, ename, job) values( 9999, 'Burger', 'CLERK' ) ;  
insert into emp (empno, ename, job) values( 9999, 'Burger', 'CLERK' )
```

* ERREUR à la ligne 1 : ORA-20010: Numéro employé inférieur à 10000 ORA-06512: à
« hr.TRG_BIR_EMP", ligne 3 ORA-04088: erreur lors d'exécution du déclencheur
'hr.TRG_BIR_EMP'

Remarque : rollback ne peut pas marcher dans trigger

Pour annuler insertion, utiliser une exception dans partie exception

Regroupement d'événement

- **Les prédicats conditionnels INSERTING, DELETING et UPDATING**

Possibilité de regrouper des événements (INSERT, UPDATE ou DELETE) au sein d'un même déclencheur **s'ils sont de même** type (BEFORE ou AFTER)

-> **Ainsi, un seul déclencheur est à coder**

✓ utiliser des prédicats conditionnels (INSERTING, DELETING et UPDATING) pour exécuter des blocs de code spécifiques pour chaque instruction de déclenchement.



```
CREATE TRIGGER ...  
BEFORE INSERT OR UPDATE ON nomTable  
.....  
BEGIN  
    .....  
IF INSERTING THEN ..... END IF;  
IF UPDATING THEN ..... END IF;  
.....  
END;
```

- éviter de programmer plusieurs fois la même action par l'intermédiaire des différents déclencheurs .
- si vous regrouper plusieurs déclencheurs mono-événement en 1 seul, supprimer les déclencheurs mono-événements (DROP TRIGGER...)

- Permet de mettre à jour une vue multitable qui ne pouvait pas être modifiée directement par INSERT, UPDATE ou DELETE
- ne font pas intervenir les options BEFORE et AFTER.
- Ne s'utilise que pour des vues
- Pas possible de spécifier une liste de colonnes dans un déclencheur INSTEAD OF UPDATE

• Exemple : Déclencheur INSTEAD OF

Brevet	nom	nbhvol	comp
PL-1	J.Mtoto	450	AF
PL-2	T. Guibert	3400	AF
PL-3	Michel Tuf	900	SING

comp	nrue	rue	ville	nomComp
AF	124	rue 1	Paris	Air France
SING	7	rue2	Singapour	Singapore AL

```
CREATE VIEW VueCompPil
AS SELECT c.comp, c.nomcomp, p.brevet,
p.nom, p.nbHvol
FROM Pilot p, Compagnie c
WHERE p.comp = c.comp
```

== > insert INTO Vue VueCompPil

Values ('AERIS', 'Aéris Toulouse','PL4','Pascal Larrazet', 5600);

== > le déclencheur qui gère les insertion dans la vue est chargé d'insérer , à chaque nouvel ajout, un enregistrement dans chacun des deux tables

Déclencheur INSTEAD OF

- Exemple :

```
CREATE OR REPLACE TRIGGER T_ VueCompPil
INSTEAD OF INSERT ON VueCompPil
FOR EACH ROW
DECLARE
    V_comp NUMBER;
    V_pil NUMBER;
BEGIN
    select count(*) into v_pil from Pilot where brevet=:new.brevet
    select count(*) into v_comp from compagnie where comp=:new.comp
    if(v_pil >0) then
        RAISE_APPLICATION_ERROR(-20103, 'le pilote existe déjà');
    else if v_pil=0 THEN
        insert into Pilote values(:new.brevet, :new.nom, ...);
    End if;

    IF v_comp=0 THEN insert into Compagnie values(:new.comp, null,null, ...,
    new.nomComp);
    ELSE RAISE_APPLICATION_ERROR(-20103, 'la compagnie existe déjà');

END;
```


Activer/désactiver

- Par défaut, un trigger est activé dès sa création.
- Pour désactiver le trigger :

`ALTER TRIGGER nomTrigger DISABLE;`

- Pour désactiver tous les trigger sur une table :

`ALTER TABLE nomTable DISABLE ALL TRIGGERS;`

- Pour activer un trigger :

`ALTER TRIGGER nomTrigger ENABLE;`

- Pour activer tous les trigger sur une table :

`ALTER TABLE nomTable ENABLE ALL TRIGGERS;`

Information sur les triggers

- **Recherche d'information sur les triggers**
 - Les définitions des triggers sont stockées dans :

USER_TRIGGERS

ALL_TRIGGERS et

DBA_TRIGGERS

Exercice :

Écrire un trigger qui permet, lors de la modification de la table employée, de mettre le salaire à 5000 si un employé n'a pas de salaire.

Solution:

```
CREATE TRIGGER DefaultSalaire  
BEFORE INSERT OR UPDATE OF salaire ON employe  
FOR EACH ROW WHEN (new.salaire is null)  
BEGIN  
SELECT 5000  
INTO :new.salaire FROM dual;  
END;
```