Université Ibn Tofeil
Ecole Nationale des Sciences Appliquées
Kenitra

Année Universitaire 2017-2018

Le language PL / SQL

Références:

- ORACLE 11g Rel. 2 – Concepts
- ORACLE 11g Rel. 2 – PL/SQL Language Reference

Ce cours est élaboré sur le support officiel

Trigger

Trigger

- Un trigger définit une action événementielles que la database doit activer automatiquement quel que soit l'utilisateur ou l'application qui les déclenche.

- Un déclencheur est un bloc PL/SQL associé à une table ou une vue et qui sera exécuté quand une instruction du langage de manipulation des données (DML) est invoquée.

- Un Trigger est utilisé:

  – Pour améliorer l'intégrité référentielle déclarative
  – Pour imposer des règles complexes liées à l'activité de la base de données
  – Pour effectuer des vérifications sur les modifications des données.

Trigger

•L'exécution des déclencheurs est donc transparente à l'utilisateur.

•Les déclencheurs sont automatiquement exécutés par la base de données lorsque des types spécifiques de commandes (événements) de manipulation de données sont exécutés sur des tables ou vues spécifiques. Ces commandes incluent les commandes d'insertion, de mise à jour et de suppression DML ,

  • Les derniers SGBD fournissent également des déclencheurs sur les instructions DDL telles que Créer une vue, etc.

•Les mises à jour de colonnes spécifiques peuvent également être utilisées comme déclencheurs d'événements.

Privilèges

•Pour créer un déclencheur sur une table, vous devez pouvoir modifier la table. Vous devez donc être le premier propriétaire de la table, puis le privilège ALTER pour la table ou le privilège système ALTER ANY TABLE. De plus, vous devez avoir le privilège système CREATE TRIGGER.

•De même, pour pouvoir modifier un déclencheur d'une table, vous devez posséder la table et également disposer de privilèges ALTER ANY TRIGGER.

Déclencheurs au niveau des lignes

Les déclencheurs au niveau de la ligne sont exécutés une fois pour chaque ligne modifiée dans une transaction; ils sont souvent utilisés dans les applications de révision de données et sont utiles pour les opérations d'audit de données et pour synchroniser les données distribuées.

Pour créer un déclencheur au niveau de la ligne, vous devez spécifier la clause FOR EACH ROW dans l'instruction CREATE TRIGGER.

Déclencheurs au niveau des instructions

Les déclencheurs de niveau instruction ne sont exécutés qu'une seule fois pour chaque transaction, quel que soit le nombre de lignes modifiées. Ils sont donc utilisés pour des activités liées aux données; ils sont généralement utilisés pour imposer des mesures de sécurité supplémentaires sur les types de transactions pouvant être effectuées sur une table.

Il s'agit du type de déclencheur par défaut dans la commande create trigger (en d'autres termes, vous n'avez pas besoin de spécifier qu'il s'agit d'un déclencheur de niveau instruction).

Type de Trigger

- BEFORE et AFTER: les trigger peuvent être éxécutés avant ou aprés l'utilisation des commandes insert, update et delete; à l'interieur du trigger, il est possible de faire référence aux anciennes et nouvelle valeurs utilisées dans la transaction.

- Il faut utiliser la clause:
  BEFORE/AFTER <type d'évenement> (insert, delete, update).

Type de Trigger

INSTEAD OF: pour spécifier quoi faire au lieu d'exécuter les actions qui ont déclenché le Trigger.

Par exemple, vous pouvez utiliser un déclencheur INSTEAD OF pour rediriger des insertions dans une table vers une autre table ou pour mettre à jour avec update plusieurs tables faisant partie d'une vue.

Types de Trigger

- BEFORE INSERT la ligne
- BEFORE INSERT l'instruction
- AFTER INSERT la ligne
- AFTER INSERT l'instruction
- BEFORE UPDATE la ligne
- BEFORE UPDATE l'instruction
- AFTER UPDATE la ligne
- AFTER UPDATE l'instruction
- BEFORE DELETE la ligne
- BEFORE DELETE l'instruction
- AFTER DELETE la ligne
- AFTER DELETE l'instruction
- INSTEAD OF la ligne
- INSTEAD OF l'instruction

Trigger, structure

- L'instruction Create suivie du nom attribué au déclencheur
- Type de déclencheur, Before/After (avant / après)
- Evénement qui déclenche le trigger Insert/Delete/Update
- [For each row] [Pour chaque ligne], si vous souhaitez spécifier des déclencheurs au niveau de la ligne (sinon rien pour les déclencheurs au niveau de l'instruction)
- Spécifiez à quelle table il s'applique
- Une condition qui doit être vérifiée pour que le déclencheur soit exécuté
- Action, définie par le code à exécuter si la condition est remplie

Trigger: syntaxe

create trigger [OR REPLACE] <NomTrigger>
Mode Evenement {, Evenement}
ON <TabelTarget>
for each row
[when <Predicat SQL>]
Block PL/SQL

Mode Evenement before ou after
Evenement: insert, update, delete
for each row spécifie la granularité. En l'absence de cette clause est destiné à chaque instruction

Anciennes et nouvelles valeurs

Étant donné que la plupart des déclencheurs ont trait à des modifications de ligne, il est important de pouvoir se référer aux valeurs de la ligne avant insertion, suppression, modification et aux valeurs après de tels événements.

En particulier si, par exemple, il s'agit d'un déclencheur BEFORE UPDATE, nous entendons par anciennes valeurs les valeurs de la table que nous souhaitons modifier et pour les nouvelles valeurs, nous souhaitons les insérer à la place des anciennes. Si à l'inverse est un déclencheur AFTER UPDATE, par anciennes nous entendons celles qui existaient avant la mise à jour et les nouvelles dans le table à la fin de la modification.

Anciennes et nouvelles valeurs

Dans Oracle, vous pouvez automatiquement faire référence aux anciennes valeurs (c'est-à-dire les valeurs avant la mise à jour) et aux nouvelles valeurs (c'est-à-dire celles après la mise à jour) respectivement à l'aide des mots-clés. "old" et "new".

Alors que dans la condition (when) les mots-clés old et new apparaissent sans aucun autre symbole, dans l'action les mots-clés old et new sont précédés de deux points (:old, :new).

Exemple

Construire le déclencheur TrigProd qui insère un nom de produit dans une table Produits lorsqu'un produit qui ne figurait pas déjà dans la table Produits est placé dans la table Ventes.

Produits

Nom

Vente

Prod	Cod_bar	prix

Exemple

Prod

Nom

Vente

Prod	Cod_bar	prix

```
CREATE OR REPLACE TRIGGER TrigProd
AFTER INSERT ON Vente                          → Evenement
FOR EACH ROW
WHEN  new.prod NOT IN                          } → Condition
        (SELECT nom  FROM Produits))
BEGIN                                          } → Action
        INSERT INTO Produits(nom) VALUES(:new.prod);
```

Exemple

Créer le TrigPrix qui mémorise dans la table Ripoffournisseur(cod_bar) le nom de chaque fournisseur qui augmente le prix de n'importe quel produit de plus de 2Dh.

ListeFournisseur

Cod_bar	Nom_four	Adresse_four	Telephone_four

Vente

Prod	Cod_bar	ix

Reapofournisseur

Cod_bar

ListeFournisseur

Cod_bar	Nom_four	Adresse_four	Telephone_four

Vente

Prod	Cod_bar	prix

Reapofournisseur

Cod_bar

```
CREATE OR REPLACE TRIGGER TrigPrix
AFTER UPDATE OF prix ON Vente
FOR EACH ROW
WHEN  new.prix > old.prix + 2.00)

        INSERT INTO Reapofournisseur VALUES(:new.cod_bar);
```

Exemple

Library

Titre	Editeur	Nom_cat	Classification

Audit_Library

Titre	Editeur	Nom_cat	Ancienne_Class	Nouvelle_class	Date_audit

Nous voulons que la table Audit_Library assure le suivi de tous les changements dans la classification de la table Library lorsque la valeur de classification est réduite.

Exemple

Create or replace Trigger Biblioteque_bef_upd_Row → Attribuer un nom trigger

BEFORE UPDATE on Library → Avant la mise à jour

FOR EACH ROW → Pour chaque ligne modifiée

WHEN (new.Classification<old.Classification) → Lorsque la nouvelle valeur de classification réduit la précédente

```
BEGIN
   INSERT INTO Audit_Library (Titre, Editeur,
      Nom_Cat, Ancienne_Class, Nouvelle_Class, Date_Audit)
    VALUES(:old.Titre, :old.Editeur, :old:Nomcat,
        :old.Classification, :new.Classification, Sysdate)
END
```

→ Action

Remarque

Le fonctionnement du déclencheur est complètement transparent pour l'utilisateur qui met à jour la table Library.

Cependant, la transaction effectuée sur la table Library dépend du succès de l'exécution du déclencheur.

Drop Trigger

Un Trigger est supprimé à l'aide de l'instruction:

DROP Trigger <nom trigger>

Différentes vues du dictionnaire de données Oracle traitent des triggers.
Vous pouvez y accéder en fonction des droits dont vous disposez pour visualiser les triggers portant sur une table et leur code:

- USER_TRIGGERS ;
- USER_TRIGGER_COLS ;
- ALL_TRIGGERS ;
- ALL_TRIGGER_COLS ;
- DBA_TRIGGERS ;
- DBA_TRIGGER_COLS.

Combinaison de types de déclencheurs

Il est possible de combiner différents déclencheurs sur différentes commandes d'insertion, de mise à jour et de suppression, à condition qu'ils soient tous au même niveau.

Dans ce cas, l'événement peut être indiqué par exemple comme suit

Before insert OR update

Et les différents cas sont sélectionnés à travers les différents types de transactions, qui sont INSERTING, DELETING et UPDATING.

Exemple

On considère le déclencheur Library_Bef_Upd_Ins_Row, qui modifie la table Audit_Library, dans lequel, si une nouvelle insertion est effectuée dans la table Library, les valeurs du titre, de l'éditeur, de la catégorie, de la nouvelle classification et de la date d'insertion sont insérées dans la table Audit_Library; si une modification est apportée, en plus de ces valeurs, l'ancienne classification est également enregistrée dans la table Audit_Library.

Library

Titre	Editeur	Nom_cat	Classification

Audit_Library

Titre	Editeur	Nom_cat	Ancienne_Class	Nouvelle_class	Date_audit

Exemple

```
Create or replace trigger Library_Bef_Upd_Ins_Row
Before Insert or Update of Classification on Library
For each Row
Begin
  If INSERTING then
    Insert into audit_Library (Titre, Editeur, Nom_cat,
                                Nouvelle_Class, Date_Audit)
    Values(:new.Titre, :new.Editeur, :new.Nom_cat,
           :new.Classification, Sysdate);
  ELSE
   if UPDATING then
    Insert into audit_Library (Titre, Editeur, Nom_cat,
                               Ancienne_Class, Nouvelle_Class, Date_Audit)
    Values(:new.Titre, :new.Editeur, :new.Nom_cat,
           :old.Classification, :new.Classification, Sysdate);
   End if;
  End if;
End;
```

Magazin

Prod	QteDisp	Seuil	QteComd

Command_Prev

Prod	Qte	Date

Créez un déclencheur qui vérifie chaque modification de la quantité disponible d'un produit en stock, si cette quantité disponible est passée en dessous du seuil.

Dans ce cas, si le produit n'est pas déjà présent dans les commandes en attente, une commande de produit est ajoutée dans la table Command_Prev, d'une quantité égale à la quantité de réapprovisionnement présente dans la table Magazin.

.

Magazin

Prod	QteDisp	Seuil	QteComd

Command_Prev

Prod	Qte	Date

```
create trigger Réorganiser
after update of     QteDisp on Magazin
  when (new.QteDisp < new.Seuil)
  for each row
  declare X number;
  begin
          select count(*) into X
          from Command_Prev
          where Prod = :new.Prod;
          if X = 0  then
                  insert into Command_Prev
                  values (:new.Prod, :new.QteComd, sysdate);
          end if;
  end
```

Clients

Cod_client	Nom	Ville	Remise

Commandes

Num_ord	Cod_client	Cod_agent	Produit	Date	Montant

Supposons que nous voulions imposer la contrainte que nous n'acceptons pas les nouvelles commandes de clients avec un découvert supérieur à 10 000 Dh. Créer un déclencheur pour satisfaire cette contrainte

Clinets

Cod_client	Nom	Ville	Remise

Commandes

Num_ord	Cod_client	Cod_agent	Produit	Date	Montant

```
Create trigger ControleFidelite
Before insert on Commandes
For each row
Declare  A_Payer Number;
BEGIN
   Select sum(Montant) into A_Payer
                From  Commandes
                Where cod_client=:new.cod_client);
   IF A_Payer+:new.Montant >=10000
   THEN
      RAISE_APPLICATION_ERROR(20001, 'Fidelite depassée') ;
   ENDIF;
END
```

Exeption

Trigger actifs et passifs

•Un déclencheur est actif lorsque, à certains événements, il modifie le statut de la base de données.

•Un déclencheur est passif s'il sert à provoquer l'échec de la transaction sous certaines conditions.

•Seul le dernier exemple parmi ceux vus est un déclencheur passif, tandis que les autres sont tous actifs.

Trigger Actif

1. Pour définir les règles métier ou les actions à mener pour garantir la bonne évolution du système d'information

2. Pour stocker des événements sur la basé des données pour des raisons de contrôle (audit et journalisation)

3. Pour propager les effets de certaines opérations de table sur d'autres tables

4. Pour garder alignées les données dupliquées lorsque vous modifiez l'une d'entre elles

Trigger Passif

1. Pour définir des contraintes d'intégrité qui ne peuvent pas être exprimées dans le modèle de données utilisé

2. Pour vérifier les opérations admissibles des utilisateurs en fonction des valeurs des paramètres des commandes SQL

Personnaliser les conditions d'erreur

Les numéros et les messages d'erreur affichés par l'utilisateur sont définis par la procédure.

RAISE_APPLICATION_ERROR

Qui peut être appelé dans chaque trigger

Raise_Application_Error est une procédure qui prend deux paramètres en entrée

- Le numero de l'erreur (qui doit être un nombre entre -20001 et -20999)
- Le message d'erreur à afficher

Exemple

Lorsqu'un utilisateur tente de supprimer une ligne de la table Library, le déclencheur Trigger Library_Bef_Del doit vérifier que l'opération n'est pas effectuée pendant le week-end et que le nom d'utilisateur du compte de suppression est "LIB".

Le code de déclenchement correspondant est le suivant:

```
Create or Replace Trigger Library_Bef_Del
Before deletion on Library
Declare
Weekend_error EXCEPTION;
Not_library _user EXCEPTION;

BEGIN
   to_char(sysdate, 'DY')='Sam' or to_char(Sysdate, 'DY')='Dim'
                Weekend_error
      if;
   upper(Utilisateur)<> 'LIB'  THEN  RAISE not_library_error;
End if;

EXCEPTION
          weekend_error THEN
 Raise_Application_Error
     (-20001, 'Suppression non permise pendant le weekend');
          not_library_user THEN
 Raise_Application_Error
     (-20002, 'Les suppressions sont autorisées seulement aux employés');
END
```

Déclaration d'exceptions
Donnez un nom aux exceptions

Definition de weekend_error

Definition de not_library_user

Trigger: Exemple

Registre

Date	Action	Article	Quantité	Montant	Personne

Audit_Registre

Date	Action	Article	Quantité	Montant	Personne	Augmentation

Écrivez un déclencheur sur la table Registre qui insère les anciennes valeurs des données insérées dans la table Registre_audit à chaque fois que la modification du montant est supérieure à 10%, en plus de l'augmentation du montant.

Exercice

```
create trigger registre_bef_upd_row
  before update on REGISTRE
  for each row
  when (new.Montant/old.Montant>1.1)
  begin
    insert into REGISTRE_AUDIT
      values (:old.DateAction, :old.Action, :old.Article,
 :old.Quantite, :old.TypeQuantite, :old.Taxe,
 :old.Montant, :old.Personne, :new.Montant – :old_Montant);
end;
```

Trigger: exemple

Créer un trigger sur la table Registre qui avant d'insérer une ligne dans la table Registre, insert le nom dans la table Personne, avec tous les caractères en minuscule.

```
create trigger registre_bef_upd_ins_row
  before insert or update of Personne on REGISTRE
  for each row
  begin
      :new.MiniscPersonne := UPPER(:new.Personne);
end;
```

Trigger: activation et désactivation

alter trigger Library_Bef_Del enable;

Active le trigger Library_Bef_Del

alter table Library enable all triggers;

Active tous les trigger Sur la table Library

alter trigger Library _Bef_Del disable;

Desactive le trigger Library _Bef_Del

alter table Library disable all triggers;

Desactive tous les trigger Sur la table Library