Les triggers (déclencheurs)

- Définitions:

Ce sont des procédures stockées qui s'exécutent automatiquement lorsqu'un événement se produit sur la BD. Cet événement est une opération DML.

- Rôle:

Un trigger permet de contrôler les accès à la base de données.

  - Éviter certains accès à la BD
  - Contrôler les valeurs de certaines colonnes lors des mises à jour
  - Tenir un journal des logs et des opérations
  - Garantir l'intégrité référentielle.

Les triggers (déclencheurs)

- Syntaxe simplifiée

CREATE [OR REPALCE] TRIGGER nomtrigger

[BEFORE | AFTER] [OR][DELETE |INSERT| UPDATE [OF nomcolonne]

ON Nomdetable [FOR EACH ROW] [WHEN condition]

BLOC PL/SQL

Les triggers (déclencheurs)

- L'option BEFORE /AFTER indique le moment du déclenchement du trigger. Une seule option à la fois.
- Les instructions SQL INSERT OR UPDATE OR DELETE peuvent être toutes présentes comme on peut en avoir juste une.
- Pour un UPDATE, on peut spécifier une liste de colonnes séparées par une virgule. Dans ce cas, le trigger ne se déclenchera que s'il porte sur l'une des colonnes précisées dans la liste.
- FOR EACH ROW : indique que le trigger contrôle chaque ligne de la table. Ce type de trigger qui se déclenche individuellement pour chaque ligne est appelé : Trigger LIGNE.
- WHEN: permet de poser une condition.

Les triggers (déclencheurs)

Lors de la création d'un trigger, si celui-ci est un trigger ligne et qu'il manipule les valeurs d'une ou plusieurs colonnes de la table en question alors:

L'ancienne valeur de la colonne est dite →:OLD.nomcolonne (:OLD.salaire)

La nouvelle valeur de la colonne est dite → :NEW.nomcolonne (:NEW.salaire).

Les deux points sont obligatoires sauf dans le **WHEN**

Les triggers (déclencheurs)

```
create or replace trigger empsal
before insert or update of salaireemp on employesclg
for each row
declare sal number;
begin
select avg(salaireemp) into sal from employesclg
group by codedep having codedep =:new.codedep;
if :new.salaireemp is null
then
:new.salaireemp:=sal;
end if;
end;
```

Les triggers (déclencheurs)

```
create or replace
trigger ctrlnotes
before update of note on resultats
for each row
WHEN (NEW.CODECOURS !='KED')
begin
if (:new.note <:old.note) then
raise_application_error (-20000, 'sauf le cours KED peut
    être révisé à la baisse.');
end if;
end;
```

Les triggers (déclencheurs)

RAISE_APPLICATION_ERROR vous permet d'émettre des messages d'erreur ORA définis par l'utilisateur à partir de de triggers. Une excellente façon de gérer vos exceptions

raise_application_error(
error_number, message[, {TRUE | FALSE}]);

Le nombtre se situe entre -20000 et -20999

True et false, indique la façon dont la pile des erreurs est gérée (False par défaut→ l'erreur n'est pas empilée, elle remplace les erreurs précedentes)

Les triggers (déclencheurs)

```
create or replace
trigger ctrlnotes
before update of note on resultats
for each row
WHEN (NEW.CODECOURS !='KED')
DECLARE
MESSAGE EXCEPTION;
begin
if (:new.note <:old.note) then RAISE MESSAGE; END IF;
EXCEPTION
WHEN MESSAGE THEN
raise_application_error (-20000, 'sauf le cours KED peut etre révisé à
    la baisse.');
end;
```

Les triggers (déclencheurs)

Remarque:

Lorsque toutes les opérations DML sont présentes alors on pourra indiquer pour chaque opération DML les opérations à entreprendre. Il suffit d'utiliser les prédicats:

INSERTING

UPDATING

DELETING

Les triggers (déclencheurs)

Triggers au niveau de la table: ce sont des triggers pour lesquels l'option FOR EACH ROW n'est pas requise. Ils sont déclenchés dès qu'il y a une opération DML sur la base de données.

Les triggers (déclencheurs)

```
CREATE OR REPLACE TRIGGER TRIGGER1
AFTER INSERT OR DELETE OR UPDATE  of note ON RESULTATS
BEGIN
IF INSERTING THEN
INSERT INTO logs VALUES ( USER, 'Insertion', sysdate);
ELSIF DELETING THEN
INSERT INTO logs VALUES ( USER, 'suppresion',sysdate);
ELSIF UPDATING THEN
INSERT INTO logs VALUES ( USER, 'Modification', sysdate);
END IF;
END;
```

Les triggers (déclencheurs)

Activer et désactiver de triggers.

Pour qu'un trigger soit déclenché celui-ci doit être ACTIVÉ. Par défaut, les triggers d'oracle sont activés dès leur création. On peut activer un triggers par SQL Développer ou par la commande **ALTER TRIGGER** nomtrigger ENABLE.

- Si un trigger est désactivé, alors il est stocké mais ignoré
- Pour désactiver un trigger, utiliser la même commande **ALTER TRIGGER** avec l'option DISABLE
- On peut désactiver TOUS les triggers associés à une table par la commande **ALTER TABLE** DISABLE ALL TRIGGERS.
- Pour les réactiver de nouveau, utiliser **ALTER TABLE** ENABLE ALL TRIGGERS.
- Pour supprimer un trigger, **utiliser DROP TRIGGER nomduTrigger**

Important:

Un trigger ligne ne peut pas lire (SELECT) et/ou modifier la table concernée (appelée table mutante) par l'instruction (INSERT, UPDATE ou DELETE) qui a déclenché ce trigger.