ORAE505
Conception et administration de B.D.

5. Le langage PL/SQL
e. Les triggers

Carina Roels

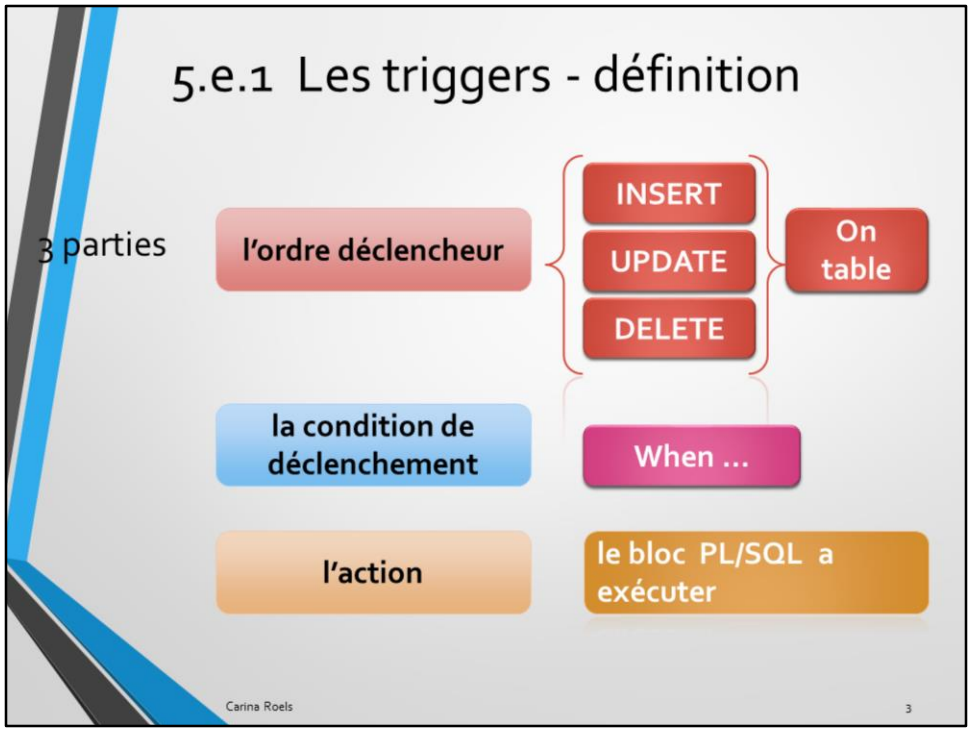
5.e.1 Les triggers - définition

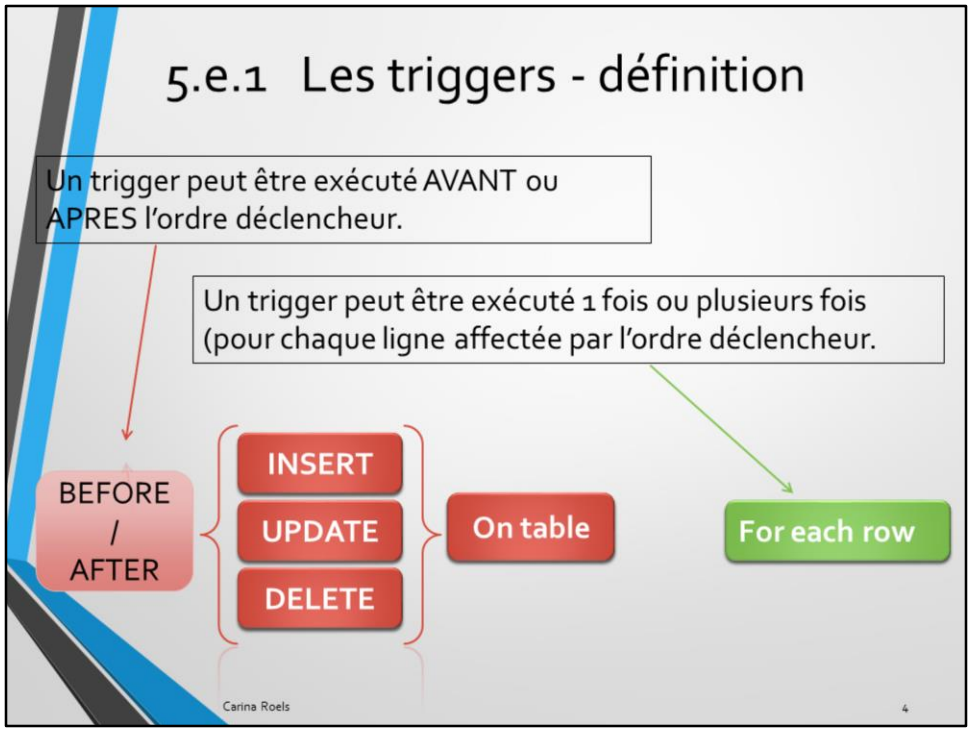
Un bloc PL/SQL stocké - associé à une table.

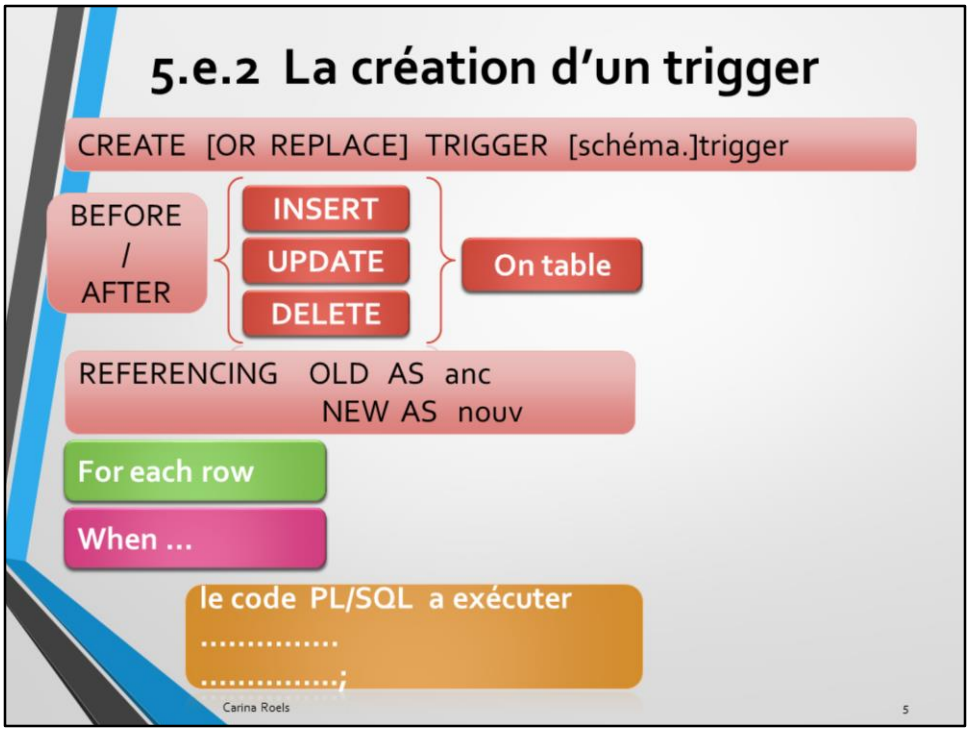
ORACLE exécute automatiquement le trigger quand une instruction SQL spécifiée est lancée sur la table.

UTILITÉ

Renforcer des autorisations complexes
Maintenir la réplication des tables
Génération automatique de valeurs de colonnes, lors d'une mise à jour


5.e.1 Les triggers - définition
3 parties
l'ordre déclencheur
INSERT
UPDATE
DELETE
On table
la condition de déclenchement
When ...
l'action
le bloc PL/SQL a exécuter
Carina Roels
3


5.e.1 Les triggers - définition
Un trigger peut être exécuté AVANT ou APRES l'ordre déclencheur.
Un trigger peut être exécuté 1 fois ou plusieurs fois (pour chaque ligne affectée par l'ordre déclencheur.
BEFORE / AFTER
INSERT
UPDATE
DELETE
On table
For each row
Carina Roels
4


5.e.2 La création d'un trigger
CREATE [OR REPLACE] TRIGGER [schéma.]trigger
BEFORE / AFTER
INSERT
UPDATE
DELETE
On table
REFERENCING OLD AS anc
NEW AS nouv
For each row
When ...
le code PL/SQL a exécuter
.....
.....;
Carina Roels
5


Exemples de triggers

Exemple 1 :

```
CREATE TRIGGER emp_auth_modifs
BEFORE  DELETE OR UPDATE  ON   emp
DECLARE
        bidon                   INTEGER;
BEGIN
        /* si jour  =  samedi ou dimanche,  alors  ERREUR */
        /* ------------------------------------------------------------- */
        IF  ( TO_CHAR( SYSDATE,'DY' ) = 'SAT'   OR
                TO_CHAR( SYSDATE,'DY' ) = 'SUN'  )
        THEN    RAISE_APPLICATION_ERROR( -20501 , 'Action impossible' );
        END IF;

        /* comparaison du jour avec les jours de vacances de l'entreprise */
        /* ------------------------------------------------------------------------- */
        SELECT  COUNT(*)  INTO  bidon
        FROM    vacances-entrep
        WHERE   jour  =  TRUNC( SYSDATE );

        IF  bidon > 0
        THEN   RAISE_APPLICATION_ERROR( -20501 , 'Action impossible' );
        END IF;
END;
/
```

Exemple 2 :

```
CREATE TRIGGER cde_fourn
AFTER UPDATE OF quantité ON inventaire
REFERENCING     OLD AS ancien
                NEW AS nouveau
FOR EACH ROW
WHEN ( nouveau.acder = 'T' )
 BEGIN
        IF :nouveau.quantité < :nouveau.seuil_cde
        THEN
            INSERT INTO commandes
            VALUES ( :nouveau.article , :nouveau.quantité_acder , SYSDATE );
        END IF;
 END;
 /
```

COLONNE

TABLE

nouveau = référence au tuple modifié.

nouveau.quantité = colonne quantité après modification

colonnes de la table inventaire après modification

Exemple 3 (avec traitement d'exception) :

```
CREATE TRIGGER verif_salaire
BEFORE INSERT OR UPDATE OF salaire, fonction ON emp FOR EACH ROW
WHEN ( new.fonction != 'PRESIDENT' )
DECLARE
        psalmin NUMBER;
        psalmax NUMBER;
BEGIN
        /* Recherche salaire minimum et maximum pour une fonction */
        SELECT salmin, salmax into psalmin, psalmax
        FROM   salaires WHERE fonction = :new.fonction;

        /* Erreur : Si salaire < salaire minimum ou salaire > salaire maximum
           ou encore si l'augmentation est négative ou si elle excède 10 % */
        IF ( :new.salaire < psalmin OR :new.salaire > psalmax)
        THEN    RAISE_APPLICATION_ERROR(-20225, 'Salaire incorrect !' );
        ELSIF ( :new.salaire < :old.salaire )
        THEN    RAISE_APPLICATION_ERROR(-20225, 'Augmentation négatif !' );
        ELSIF ( :new.salaire > 1.1 * :old.salaire )
        THEN    RAISE_APPLICATION_ERROR(-20225, 'Augmentation > 10 % !' );
        END IF;
END;
/
```