Oracle : Triggers (déclencheurs)

Déclencheur (Trigger)

- Traitement implicite déclenché par un événement (INSERT/UPDATE/DELETE)
- Utilisé souvent pour implémenter des règles de gestion complexes
- Un trigger est associé en général à une table
- Un trigger est défini au moyen du langage PL/SQL.

Caractéristiques d'un Trigger

- Son code est stocké dans la base de données.
- Il est déclenché sur un événement complété par un prédicat
- Un déclencheur peut être actif ou non (enable/disable)
- Si un déclencheur réussit, la transaction qui l'a appelé peut se poursuivre

Description d'un trigger

- Deux types de triggers :
  - **Instruction** : le trigger ne se déclenche qu'une fois, même s'il concerne plusieurs lignes (comportement par défaut)
  - **Pour chaque ligne** (for each row) : à chaque ligne de la table concernée par l'événement, dans ce cas, le trigger s'applique autant de fois que nécessaire
  - Par défaut : si l'option FOR EACH ROW est spécifiée, c'est un trigger ligne, sinon c'est un trigger instruction (global)

Structure d'un déclencheur

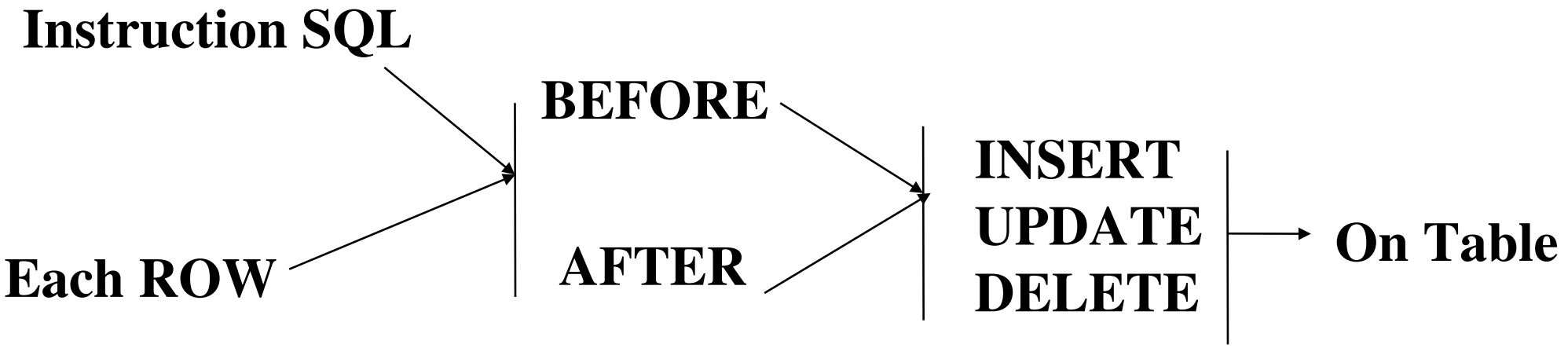

Illustration : EX_Cours_1

```
set serveroutput on
CREATE OR REPLACE TRIGGER mon_Trigger
    BEFORE UPDATE OF ename ON emp -- on peut faire une maj de col.
    FOR EACH ROW
  BEGIN
     DBMS_OUTPUT.PUT_LINE(:NEW.ENAME|| '/// ' ||:OLD.ENAME);
  EXCEPTION
     WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('SQLERRM : ' || SQLERRM);
 END;
 /

Trigger created.

SQL>
```

Exécution

SQL> update sadeg.emp set ename = 'SADEG' where ename like 'SMITH';

SADEG/// SMITH

1 row updated.

SQL> select empno, ename from sadeg.emp where ename like 'SADEG';

EMPNO	ENAME
7369	SADEG

SQL>

Remarques

OLD et NEW : pour un trigger ligne :

- **Si l'instruction de déclenchement du trigger est** INSERT**, seule la** nouvelle valeur **a un sens.**
- **Si l'instruction de déclenchement du trigger est** DELETE**, seule l'**ancienne valeur **a un sens.**

La nouvelle valeur est appelée :new.colonne

L'ancienne valeur est appelée :old.colonne

AUTRES

Métabase : Les définitions des triggers sont stockées dans les tables de la métabase (tables USER_TRIGGERS, ALL_TRIGGERS et

DBA_TRIGGERS)

La procédure raise_application_error(numero_erreur, message_erreur)

- numero_erreur doit être un entier dans [-20000 , -20999]

- message_erreur doit être une chaîne de 500 caractères maximum.

- Quand cette procédure est appelée, elle termine le trigger, défait la transaction (ROLLBACK), renvoie un numéro d'erreur défini par l'utilisateur et un message d'erreur à l'application.

EXEMPLE : EX_Cours_2

/* vérifier le salaire d'un employé : s'assurer qu'il est compris dans les bornes correspondant au grade de l'employé */

-- on ajoute la colonne GRADE à la table EMP

SQL> Alter table sadeg.emp add grade varchar2(10);

Table altered.

SQL>

Trigger

```
CREATE or replace TRIGGER verif_grade_salaire
   BEFORE INSERT OR UPDATE OF sal, grade ON emp
   FOR EACH ROW
DECLARE
  minsal number;
  maxsal number;
BEGIN
/* retrouver le salaire minimum et maximum du grade */
SELECT losal, hisal INTO minsal, maxsal
    FROM salgrade WHERE grade = :new.grade;
```

```
   /* s'il y a un problème, on provoque une erreur */
  IF (:new.sal<minsal OR :new.sal>maxsal)
  THEN
       raise_application_error (-20300,'Salaire ' || TO_CHAR(:new.sal)
       || 'incorrect pour ce grade');
   END IF;
EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20301,  'Grade incorrect');
END;
/
```

```
SQL> select empno, ename, sal  from emp where empno=7900;

     EMPNO ENAME      SAL

---------- ---------- ----------

      7900 JAMES       950

SQL> update  emp set sal=sal+6000, grade=3 where empno=7900;

min=1401   max=2000

NEW SAL=6950

update emp set sal=sal+6000, grade=3 where empno=7900

       *

ERROR at line 1:

ORA-20300: Salaire 6950  incorrect pour ce grade

ORA-06512: at "SADEG.VERIF_GRADE_SALAIRE", line 15

ORA-04088: error during execution of trigger
'SADEG.VERIF_GRADE_SALAIRE'
```



```
SQL>
```

Si Plusieurs événements

- Un Trigger peut répondre à plusieurs événements
- => On utilise dans ce cas :

  les prédicats **INSERTING**, **UPDATING** ou **DELETING** pour exécuter un traitement en fonction du type d'événement.

Exemple : EX_Cours_3

```
SET SERVEROUTPUT ON
CREATE OR REPLACE TRIGGER mon_trigger_2
   AFTER UPDATE OR INSERT ON sadeg.emp
      FOR EACH ROW
BEGIN
       IF INSERTING THEN
          DBMS_OUTPUT.PUT_LINE(' INSERTION ');
       END IF;
       IF UPDATING('ENAME') THEN
          DBMS_OUTPUT.PUT_LINE(' maj_nouveau nom ' || :NEW.ENAME);
       END IF;
END;
/
Trigger created.

SQL> UPDATE emp SET ename = 'TOTO' WHERE empno = 7839;
maj_nouveau nom  TOTO
1 row updated.
```

Exécution

```
SQL> update sadeg.emp set ename = 'TOTO' where ename = 'FORD';
```

Nouveau nom maj : TOTO

1 row updated.

SQL> insert into sadeg.emp (empno, ename, sal, deptno) values (1000, 'SADEG', 1500, 20);

INSERTION

1 row created.

SQL> ALTER TABLE sadeg.emp DISABLE ALL TRIGGERS;

Exemple : contrôle d'un traitement EX_Cours_4

```
SQL> CREATE OR REPLACE TRIGGER ajout
  BEFORE INSERT ON sadeg.dept
  BEGIN
     IF (USER != 'TOTO') THEN
        RAISE_APPLICATION_ERROR(-20001,'Vous n''etes pas '||'TOTO');
     END IF;
   END;
   /
Trigger created.

SQL> insert into dept values (60, 'LIMOUSIN', 'LIMOGES');
insert into dept values (60, 'LIMOUSIN', 'LIMOGES')
            *
ERROR at line 1:
ORA-20001: Vous n'etes pas U_AUTORISE
ORA-06512: at "SADEG.AJOUT", line 3
ORA-04088: error during execution of trigger 'SADEG.AJOUT'
```

Mise hors-service (non actif) d'un Trigger

ALTER TRIGGER mon_trigger disable ;
-- sous SYS, car le trigger appartient à SYS)

SQL> ALTER TRIGGER sys.ajout DISABLE;

Trigger altered.

SQL> insert into sadeg.dept (deptno, dname) values (55, 'D1');
-- maintenant, on peut insérer

1 row created.

SQL>

SQL> ALTER TABLE maTable
DISABLE ALL TRIGGERS;

SQL> ALTER TABLE sadeg.emp
DISABLE ALL TRIGGERS;
-- On désactive tous les triggers sur EMP
Table altered.

SQL> ALTER TRIGGER tonTrigger ENABLE ;

SQL> DROP TRIGGER ceTrigger;
-- Suppprimer un trigger

La table USER_TRIGGERS

SQL> desc user_triggers

Name	Null?	Type
TRIGGER_NAME		VARCHAR2(30)
TRIGGER_TYPE		VARCHAR2(16)
TRIGGERING_EVENT		VARCHAR2(227)
TABLE_OWNER		VARCHAR2(30)
BASE_OBJECT_TYPE		VARCHAR2(16)
TABLE_NAME		VARCHAR2(30)
COLUMN_NAME		VARCHAR2(4000)
REFERENCING_NAMES		VARCHAR2(128)
WHEN_CLAUSE		VARCHAR2(4000)
STATUS		VARCHAR2(8)
DESCRIPTION		VARCHAR2(4000)
ACTION_TYPE		VARCHAR2(11)
TRIGGER_BODY		LONG
CROSSEDITION		VARCHAR2(7)
BEFORE_STATEMENT		VARCHAR2(3)
BEFORE_ROW		VARCHAR2(3)
AFTER_ROW		VARCHAR2(3)
AFTER_STATEMENT		VARCHAR2(3)
INSTEAD_OF_ROW		VARCHAR2(3)
FIRE_ONCE		VARCHAR2(3)
APPLY_SERVER_ONLY		VARCHAR2(3) [1]

SQL>

Exemple : Auditer un traitement (EX_Cours_5)

```
-- Creer une table
SQL> CREATE TABLE emp2
(
   usr       VARCHAR2(15),
   when        DATE,
   previousVal  VARCHAR2(15),
   currentVal    VARCHAR2(15)
)
/
Table created.

SQL>
```

```
set serveroutput on
CREATE OR REPLACE TRIGGER auditer
       BEFORE UPDATE OF ename ON sadeg.emp
       FOR EACH ROW
DECLARE
--  Pour faire COMMIT dans un trigger : PRAGMA AUTONOMOUS_TRANSACTION;

  PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN
   DBMS_OUTPUT.PUT_LINE(USER||' '||SYSDATE||' '||:NEW.ENAME);
     INSERT INTO emp2 VALUES (USER,SYSDATE, :OLD.ename,
:NEW.ename);
     COMMIT;
EXCEPTION
     WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE(SQLCODE);
          DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
SQL>
```

SQL> update emp set ename ='DUVALLET' where ename = 'SADEG';
SADEG 12-JAN-16 DUVALLET
Nouveau nom maj : DUVALLET

1 row updated.

SQL> select * from emp2;

USR	WHEN	PREVIOUSVAL	CURRENTVAL
SADEG	12-JAN-16	SADEG	DUVALLET

SQL>

Table historique 1/2 : EX__Cours_6

SQL> Drop table emp3;

Table dropped.

CREATE TABLE emp3
AS SELECT * FROM emp;

Table created.

ALTER table emp3
ADD usr VARCHAR2(15);
Table altered.

SQL>

2/2

```
CREATE OR REPLACE TRIGGER histo
        BEFORE DELETE ON sadeg.emp
        FOR EACH ROW
BEGIN
  INSERT INTO emp3(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,
  COMM,DEPTNO,USR) values (SELECT emp.*, USER FROM
    sadeg.emp   WHERE emp.empno = :OLD.empno);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLCODE);
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/

Trigger created.
SQL>
```

```
SQL> select * from emp3;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO USR
---------- ---------- --------- ---------- --------- ---------- ---------- ---------- ----------
      7369 SMITH      CLERK            7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER          7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST          7566 09-DEC-82       3000                    20
      7839 Durand     PRESIDENT              17-NOV-81       5000                    10
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK            7788 12-JAN-83       1100                    20
      7900 JAMES      CLERK            7698 03-DEC-81        950                    30
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK            7782 23-JAN-82       1300                    10

14 rows selected.

SQL>
```

```
SQL> delete from sadeg.emp where sadeg.emp.empno = 7369;

1 row deleted.

SQL> select * from emp3;

     EMPNO ENAME      JOB              MGR HIREDATE     SAL       COMM     DEPTNO USR
---------- ---------- --------- ---------- --------- ---------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000                    20
      7839 Durand     PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 12-JAN-83       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      7369 sadeg      CLERK           7902 17-DEC-80        800                    20 SYS

15 rows selected.

SQL
```