Les Déclencheurs (Triggers)

1. Introduction (1/2)

- Traitement implicite déclenché par un événement
- Utilisé pour implémenter des règles de gestion complexes et pour étendre les règles d'intégrité référentielle associées à une relation
- Un déclencheur est associé à une relation
- Le traitement déclenché est défini au moyen de PL/SQL (ORACLE).
- Son code est stocké dans la base de données (exécution plus rapide)

1. Introduction (2/2)

- Déclencheur
  - Evènement – condition – action
    - Evènement dans la BD
    - Si la Condition est vérifiée alors déclenchement d'une action

2. Structure d'un déclencheur

CREATE [OR REPLACE] TRIGGER nom-trigger
BEFORE | AFTER]INSTEAD OF
INSERT OR UPDATE [OF column] OR DELETE ON nom-relation
[FOR EACH ROW [WHEN (condition)]]
bloc d'instructions PL/SQL

- **BEFORE | AFTER |INSTEAD OF**
  - spécifie le point de déclenchement comme avant ou après l'exécution de l'événement ou à la place de l'évènement (NB qui pourrait être annulée par le trigger)
- **INSERT OR UPDATE [OF column] OR DELETE ON nom-relation**
  - spécifie les événements qui feront déclencher l'action
    - vérification d'une contrainte d'intégrité que le trigger doit implémenter
- **[FOR EACH ROW [WHEN (condition)]]**
  - spécifie le type de déclencheur comme étant un déclencheur de tuple (si omis c'est un déclencheur de structure) et des conditions optionnelles pour son déclenchement
- **bloc d'instructions** PL/SQL
  - spécifie les actions à exécuter, programmées en PL/SQL

3. Suppression

- Drop Trigger nom_trigger

4. Option BEFORE/ AFTER

- Les déclencheurs before sont moins efficaces car il nécessite une double lecture
- Toute modification effectuée avant une modification (Before) est visible après modification (AFTER)

5. Déclencheur de tuple

- Dans les déclencheurs de tuple il est possible d'accéder aux valeurs des attributs avant modification et après modification
  - Nouvelle valeur d'un attribut
    - :new.attribut
  - Ancienne valeur d'un attribut
    - :old.attribut

6. Exemple

```
1 CREATE OR REPLACE TRIGGER monPremierTrigger
2 BEFORE UPDATE OF nom ON emp
3  FOR EACH ROW
4   BEGIN
5   DBMS_OUTPUT.ENABLE(20000);
6   DBMS_OUTPUT.PUT_LINE(:NEW.nom|| ` ` ||:OLD.nom);
7   EXCEPTION
8   WHEN OTHERS THEN      DBMS_OUTPUT.PUT_LINE(SQLERRM);
9  END;

SQL> UPDATE emp
     SET nom = 'Durand'
    WHERE noemp = 7839;
SQL>
```


7. Résultats d'un déclencheur

Le déclencheur se termine favorablement s'il ne soulève pas d'exceptions ou ne viole pas d'autres contraintes (contraintes déclarées e.g. check).

Dans ce cas, les actions effectuées par l'événement et le trigger sont acceptées. Sinon toutes les actions du trigger sont annulées.

8. Les prédicats conditionnels
INSERTING,DELETING et UPDATING

- Quand un déclencheur comporte plusieurs clause il est possible de tester la clause de déclenchement
  - If inserting then … endif
  - If deleting then … endif
  - If updating then … endif

9. Exemple 2

```
CREATE OR REPLACE TRIGGER secondTrigger
AFTER UPDATE OR INSERT ON emp
        FOR EACH ROW
        BEGIN
    DBMS_OUTPUT.ENABLE(20000);
                IF INSERTING THEN  DBMS_OUTPUT.PUT_LINE(' INSERT ');
                END IF;
                IF UPDATING('nom') THEN
                  DBMS_OUTPUT.PUT_LINE(' UPDATED ' || :NEW.nom);
                END IF;
        END;
/
SQL> UPDATE emp SET nom = 'Dulong' WHERE noemp = 7839;
UPDATED Dulong
1 row updated.
```


10. Activation/Désactivation d'un trigger

- ALTER TRIGGER monTrigger DISABLE;
  - Permet de ne plus tenir compte d'une contrainte temporairement
- ALTER TRIGGER tonTrigger ENABLE
  - Permet de rétablir la vérification d'une contrainte.
- ALTER TABLE maRelation DISABLE ALL TRIGGERS;
- ALTER TABLE maRelation ENABLE ALL TRIGGERS;
- DROP TRIGGER ceTrigger;

11. 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

14. Instructions autorisées

- Les instructions du LMD sont autorisées
- Les instructions commit et rollback sont interdites
  - Il est impossible de savoir si une transaction est terminée
- Les instructions du LDD sont interdites car elles déclenchent automatiquement un commit