Les Triggers (déclencheurs)

Introduction

Traditionnellement les bases de données sont des systèmes s'occupant de données statiques. Les bases de données actuelles sont capables de comportements dynamiques, donc d'être actives.
Ainsi, une base de données peut stocker un traitement déclenché par un événement. Exemples :

- Réapprovisionnement d'un stock de pièces à partir d'un seuil prédéfini,
- Log des mises-à-jour et des suppressions effectuées …

Dans Oracle, tout comme dans les SGBD les plus répandus du marché, cet aspect dynamique s'implante à l'aide de déclencheurs ou « triggers ». Ceux-ci peuvent être programmés avec le langage PL/SQL, en Java ou encore en C.

Les triggers (déclencheurs) - Définition

Un trigger est un objet de schéma. Il correspond à un traitement stocké dans la base et déclenché par un événement. Il est généralement associé à une table ou une vue, mais peut aussi être lié à tout un schéma ou même à toute la base de données.

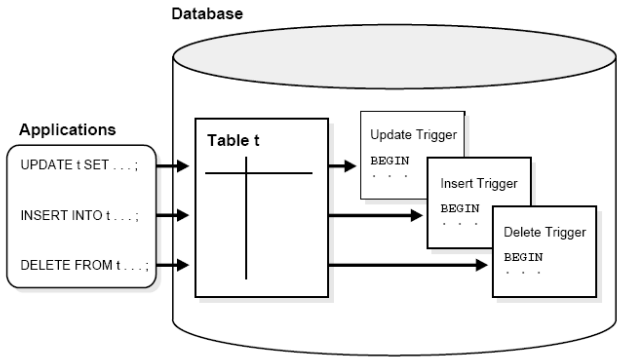
Le traitement (suite d'instructions) est exécuté automatiquement par le système de gestion de base de données, chaque fois que l'événement auquel il est associé se produit.

Cet événement peut être de différents types :

- Requête de type LMD sur un objet de schéma (table ou vue)
- Requête de type LDD dans un schéma ou dans la base de données
- Connexion ou déconnexion d'utilisateurs
- Erreurs sur le serveur
- Démarrage ou arrêt d'une base de données.

La différence principale entre une procédure stockée et un trigger est que

- la **procédure stockée** est **explicitement lancée** par un utilisateur, une application ou un trigger,
- un **trigger** est **déclenché automatiquement** par Oracle lorsque survient l'événement lié, et ce quel que soit l'utilisateur connecté ou l'application utilisée.



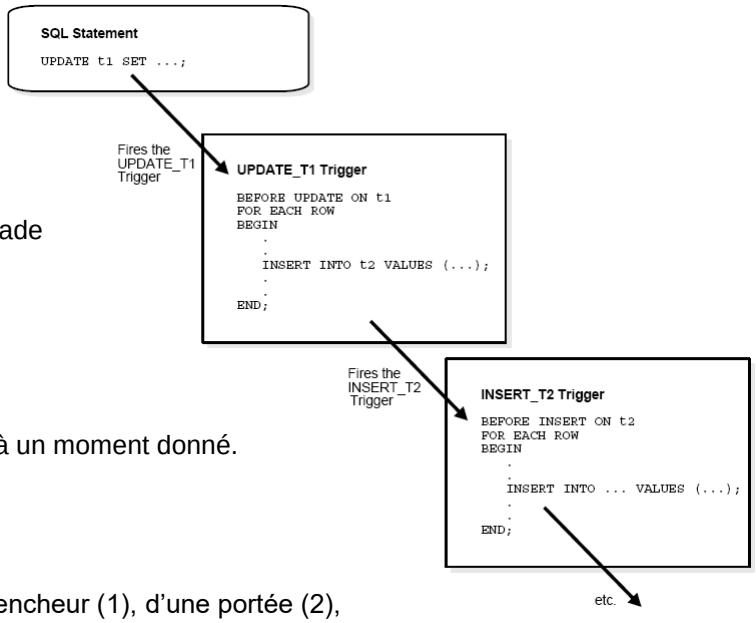
Le type de trigger le plus courant est le trigger qui porte sur les modifications des données se trouvant dans une table (INSERT, UPDATE, DELETE).

Utilisation des triggers

Les triggers sont utilisés pour ajouter des fonctionnalités au SGBD, telles que :

- restreindre l'accès ou la modification des données à certaines heures
- calculer des valeurs à stocker dans des colonnes calculées
- interdire des transactions non-valides
- mettre en œuvre des règles de sécurité complexes
- mettre en œuvre des règles de gestion complexes
- mettre en œuvre des règles de gestion dans un environnement distribué
- fournir des méthodes d'audit et de suivi de transactions
- mettre en œuvre la réplication de base
- mettre à jour des tables quand des données de vues sont modifiées
- …

Attention cependant à ne pas abuser des triggers, afin, principalement, d'éviter le risque des triggers se déclenchant en cascade (voire en boucle !!!) :



ORACLE intègre un mécanisme de sécurité en autorisant jusqu'à 32 triggers en cascade à un moment donné.

Composants d'un trigger

Un trigger est composé d'un événement déclencheur (1), d'une portée (2), d'une restriction de déclenchement (3), et d'une action (4). Exemple :

```
CREATE OR REPLACE TRIGGER trg_update_emp_salaire
   AFTER UPDATE OF emp_salaire ON exe_employe      -- (1) événement déclencheur
   FOR EACH ROW                                    -- (2) portée (instruction/enregistrement)
   WHEN (OLD.emp_salaire > NEW.emp_salaire)        -- (3) restriction
   DECLARE
      v_diff NUMBER;
   BEGIN                                           -- (4) action, traitement
      v_diff := :OLD.emp_salaire - :NEW.emp_salaire;

      dbms_output.put('Ancien :' || :OLD.emp_salaire);
      dbms_output.put('Nouveau:' || :NEW.emp_salaire);
      dbms_output.put_line(' Différence:' || v_diff);
   END;
```

Activation et désactivation

Un trigger peut être activé ou désactivé par les instructions :

- `ALTER TRIGGER <nom_du_trigger> {ENABLE|DISABLE};`

et détruit par l'instruction :

- `DROP TRIGGER <nom_du_trigger>.`

Les types de triggers

Les principaux types de triggers sont les suivants :

- Trigger de ligne (row) ou d'instruction (statement)

  Cette option définit si le trigger est un trigger de ligne ou d'instruction.

  - S'il est défini comme **trigger de ligne (FOR EACH ROW)**, le corps du trigger sera **exécuté pour chaque enregistrement "touché"** par l'instruction qui a déclenché le trigger.
  - Si le trigger est un **trigger d'instruction**, il sera **exécuté une et une seule fois** par instruction, indépendamment du nombre d'enregistrements manipulés par l'instruction.

- Trigger BEFORE ou AFTER

  - Un trigger qui se déclenche **avant (BEFORE)** un événement, ne se déclenche en réalité qu'après que l'instruction ait été donnée à Oracle, mais avant que celle-ci ne soit exécutée. On peut ainsi « **intercepter** » un événement et agir en conséquence avant qu'Oracle n'effectue l'instruction. Il est, par exemple, possible de modifier les valeurs qui vont être écrites dans une table.
    Ex: … `BEFORE UPDATE` …
  - Par opposition, un trigger qui se déclenche **après (AFTER)** un événement ne pourra que suivre l'exécution de l'instruction. Il est, par exemple, **impossible de modifier** les valeurs car elles ont déjà été écrites dans la table. Ex : … `AFTER INSERT` …

- Trigger INSTEAD OF

  Un trigger INSTEAD OF s'exécute « **à la place** » d'une instruction qui **porte sur une vue**. On peut ainsi modifier les tables de base qui se cachent derrière la vue, tout en autorisant les utilisateurs ou les applications à accéder à la vue. Ex : … `INSTEAD OF INSERT OR UPDATE ON vw_xxx` …

- Trigger sur les événements utilisateurs ou système

  On peut enfin créer des triggers pour réagir aux événements causés par les utilisateurs (logon, logoff, …) ou par le système (startup, suspend, server error, …).
  Ex : … `BEFORE LOGOFF ON DATABASE` …

Exécution des triggers

En simplifiant, on peut dire qu'Oracle utilise le modèle suivant d'exécution de triggers lorsque survient un événement déclencheur (requête) :

- Tous les **triggers d'instruction** de type **BEFORE** sont exécutés
- Pour chaque enregistrement concerné par la requête, la boucle suivante est effectuée
  - Tous les **triggers de ligne** de type **BEFORE** sont exécutés
  - L'enregistrement est verrouillé et le changement effectué
  - Tous les **triggers de ligne** de type **AFTER** sont exécutés
- Tous les **triggers d'instruction** de type **AFTER** sont exécutés.

Les noms de corrélation (OLD et NEW)

Dans un **trigger de ligne** on doit pouvoir accéder aux anciennes et nouvelles valeurs de colonne de la ligne.
Les noms de corrélation permettant de désigner ces deux valeurs sont **:OLD** pour l'ancienne valeur et **:NEW** pour la nouvelle valeur.

- si l'instruction de déclenchement du trigger est INSERT, seule la valeur `:NEW` a un sens.
- si l'instruction de déclenchement du trigger est DELETE, seule la valeur `:OLD` a un sens.
- si l'instruction de déclenchement du trigger est UPDATE, les valeurs `:NEW` et `:OLD` ont un sens.

De plus, dans les cas `BEFORE INSERT` et `BEFORE UPDATE`, il est possible de modifier les valeurs `:NEW` avant de les inscrire dans les tables.

REFERENCING
Si une table s'appelle NEW ou OLD, on peut utiliser REFERENCING pour éviter l'ambiguïté entre le nom de la table et le nom de corrélation.

Voilà le tableau récapitulatif des accès et des modifications possibles de ces valeurs :

		Accès aux valeurs		Modification des valeurs	
		:OLD	**:NEW**	**:OLD := xx**	**:NEW := xx**
INSERT	BEFORE INSERT	Null	nouvelle valeur	*erreur de compil*	OK
	AFTER INSERT	Null	nouvelle valeur	*erreur de compil*	*erreur de compil*
UPDATE	BEFORE UPDATE	valeur avant modif	nouvelle valeur	*erreur de compil*	OK
	AFTER UPDATE	valeur avant modif	nouvelle valeur	*erreur de compil*	*erreur de compil*
DELETE	BEFORE DELETE	valeur avant modif	Null	*erreur de compil*	*erreur de compil*
	AFTER DELETE	valeur avant modif	Null	*erreur de compil*	*erreur de compil*

Exemple :

```
CREATE OR REPLACE TRIGGER trg_imprime_augmentation_salaire
   AFTER UPDATE ON exe_employe
   FOR EACH ROW
DECLARE
   v_difference NUMBER;
BEGIN
   v_difference := :NEW.emp_salaire - :OLD.emp_salaire;
   dbms_output.put_line(' Différence ' || v_difference);
END;
```

Prédicats conditionnels

Quand un trigger comporte plusieurs instructions de déclenchement (par exemple INSERT OR UPDATE OR DELETE), on peut utiliser les prédicats conditionnels (INSERTING, UPDATING et DELETING) pour exécuter des blocs de code spécifiques pour chaque instruction de déclenchement.

Exemple :

```
CREATE OR REPLACE TRIGGER trg_predicats
   BEFORE INSERT OR UPDATE OR DELETE ON exe_employe
   ...
BEGIN
   IF INSERTING THEN ... END IF;
   IF UPDATING  THEN ... END IF;
   IF DELETING  THEN ... END IF;
END;
```

Tables en mutation (mutating tables)

Lorsque l'on déclenche un trigger de ligne/d'instruction sur une table et que l'on cherche dans ce trigger à accéder aux données de la table, cet accès peut être refusé si la table est en mutation.

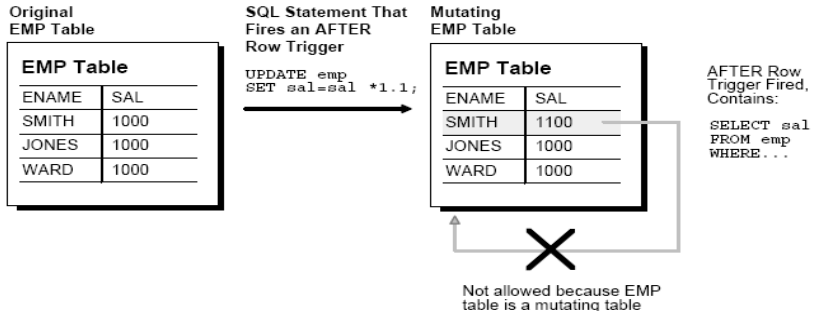
L'erreur que l'on obtient est la suivante :

```
ORA-04091 EXE_EMPLOYE IS MUTATING, TRIGGER MAY NOT READ OR MODIFY IT.
```

Une table en mutation est une table en train d'être modifiée par une requête INSERT, UPDATE ou DELETE.

En effet, si Oracle autorisait l'accès aux données en modification, on pourrait obtenir des lectures inconsistantes pendant l'exécution du trigger.

Ce cas de figure ne peut survenir que lorsque l'on exécute un trigger de type ligne (FOR EACH ROW). Ce problème peut être illustré de la façon suivante :



Il existe quelques solutions pour contourner ce problème. Nous en parlerons plus tard.