Fondements des bases de données

Programmation en PL/SQL Oracle – les déclencheurs

Équipe pédagogique BD

`http://liris.cnrs.fr/~mplantev/doku/doku.php?id=lif10_2016a`

Version du 13 octobre 2016

Déclencheurs (triggers)

Les déclencheurs (triggers)

- Les contraintes prédéfinies ne sont pas toujours suffisantes
  Ex : *Tout nouveau prix d'un produit doit avoir une date de début supérieure à celle des autres prix pour ce produit*
- Exécuter des actions lors de certains événements :
  - `AFTER` ou `BEFORE`
  - `INSERT`, `DELETE` ou `UPDATE`
  - `FOR EACH ROW`
    - non (`STATEMENT`) : exécuté *une seule fois* pour la commande.
    - oui (`ROW`) : exécuté *à chaque ligne* concernée.

Syntaxe générale

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER} {INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name] ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
BEGIN
  -- sql statements
END;
```

Utilisation des déclencheurs

Cas d'usage

- Logging et audit (enregistrement des modifications)
- Contrôle d'accès
- Application de règles métier

Problèmes d'utilisation

- Ne jamais simuler de fonctionnalités existantes
  - Foreign keys
  - Primary keys
  - Checks
  - Attributs virtuels
  - Vues
  - Réplication
  - ...
- Flot d'exécution difficile à suivre
- Maintenabilité du code

Accès aux valeurs modifiées

Utilisation de `:NEW` et `:OLD`

- Si nous ajoutons un client dont le nom est `toto` alors nous récupérons ce nom grâce à la variable `:NEW.nom`
- Dans le cas de suppression ou modification, les anciennes valeurs sont dans la variable `:OLD.nom`

Exemple

Archiver le nom de l'utilisateur, la date et l'action effectuée dans une table `LOG_EMP` lors de l'ajout d'un clients dans la table `EMP`

- Créer la table `LOG_EMP` avec la même structure que `EMP`.
- Ajouter les colonnes `LOGGIN`, `DATE_M`, `ACTION`.
- Créer un trigger `AFTER INSERT ON emp`

Accès aux valeurs modifiées

Exemple

```
CREATE or REPLACE TRIGGER logadd
   AFTER INSERT
   ON emp
   FOR EACH ROW
BEGIN
   INSERT INTO log_emp VALUES(
      :new.matr,
      ...
      USER,
      SYSDATE,
      'INSERT');
END;
/

INSERT INTO EMP VALUES
   (3,'Tarjan, Robert', 2446.13, 'Princeton, US', 2);
SELECT * FROM log_emp;
```

Bug de sqldeveloper `Enter bindings` : exécuter le script *en intégralité* dans un nouveau fichier...

Prédicats conditionnels

- Lorsqu'un trigger a plusieurs opérations déclenchantes le corps peut avoir des prédicats conditionnels :
  - `IF INSERTING THEN ... END IF;`
  - `IF UPDATING THEN ... END IF;`
- On peut préciser les colonnes soumises aux opérations déclenchantes avec `OF`
- On peut préciser une condition de test dans le `WHEN`

Exemple

```
CREATE OR REPLACE TRIGGER t_emp_sal_increase
  BEFORE UPDATE OF sal, dep
  ON EMP
  FOR EACH ROW
  WHEN (NEW.dep = 1)
BEGIN
  IF UPDATING('sal') THEN
    IF (:NEW.sal < :OLD.sal) THEN
      raise_application_error(-20000,'Sallary cannot decrease');
    END IF;
  END IF;
END;
```

Important

Interdictions

- Les commandes de définition de données (LDD)
- les commandes de contrôle de transactions (`ROLLBACK`, `COMMIT`)

Ne doivent pas être utilisées dans le corps d'un trigger.

Remarques

- Pour éviter une modification de données dans un trigger `BEFORE`, il faut lever une exeception.
- Le dictionnaire de données a des vues sur les triggers : `USER_TRIGGERS`, `ALL_TRIGGERS`, `DBA_TRIGGERS`.
- La commande `DROP` permet de supprimer un trigger.

Tables mutantes et contraignantes

- Une table mutante est une table *en cours de modification* par une opération déclenchante (`UPDATE`, `DELETE`, `INSERT`) ou l'effet de `DELETE CASCADE` provenant de cette opération.
- Une table contraignante est une table qu'une *opération déclenchante doit lire*, soit directement via une commande SQL (`UPDATE SET ... WHERE`) ou indirectement pour une contrainte d'intégrité référentielle.

Les commandes SQL dans le corps d'un trigger ne peuvent pas

- Lire (par une requête) ou modifier un table mutante d'une opération déclenchante.
- Changer des valeurs sur les colonnes de clés (`PRIMARY`, `FOREIGN`, `UNIQUE`) d'une table contraignante.

Ces restrictions permettent d'éviter la consultation d'une table dans un état transitoire et donc incohérent.

Exemple d'erreur

```
CREATE OR REPLACE TRIGGER emp_count
  AFTER DELETE ON emp
  FOR EACH ROW
  DECLARE
    n integer;
  BEGIN
    SELECT COUNT(*) INTO n
    FROM emp;
    dbms_output.put_line(
      'On a ' || n || ' employes dans la base');
  END;

DELETE FROM emp WHERE matr = 3;
```

ORA-04091: table RTHION.EMP is mutating, trigger/function may not see it.

Dans ce cas là, il ne faut pas utiliser `FOR EACH ROW` pour pouvoir déterminer la valeur du `SELECT COUNT(*) FROM emp;`.

Fin du cours.