Secteur Tertiaire Informatique
Filière étude - développement

Activité « Développer la persistance des données »

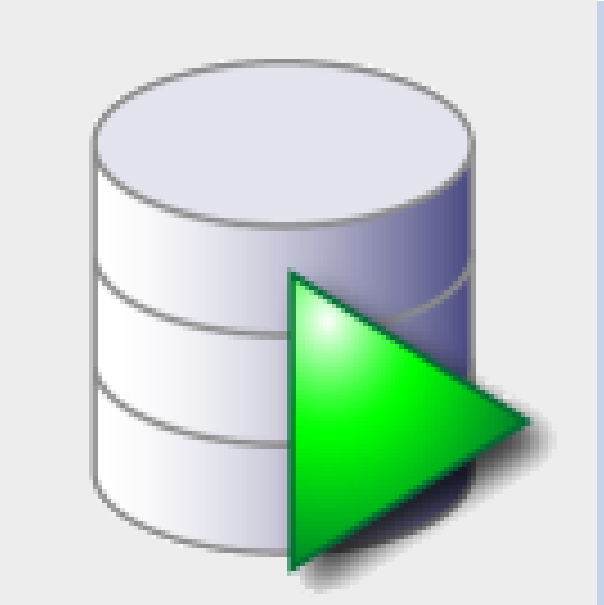
PL/SQL - Mise en œuvre des déclencheurs LMD

Accueil

Apprentissage

Période en entreprise

Evaluation



Code barre

SOMMAIRE

I DEFINITION DES DECLENCHEURS LMD

Un déclencheur LMD (Langage de Manipulation des donnéers) ou DML (Data Manipulation Language) est un type spécial de procédure stockée qui présente trois caractéristiques :

- Il est associé à une table
- Il s'exécute automatiquement lorsque un utilisateur essaie de modifier des données par une instruction LMD(UPDATE, DELETE, INSERT) sur la table où il est défini.
- Il ne peut pas être appelé directement.

Le déclencheur et l'instruction qui a provoqué son exécution sont traités comme une seule transaction.

Le bloc PL/SQL qui constitue le déclencheur (trigger) peut être exécuté avant ou après vérification des contraintes d'intégrité.

Les triggers offrent une solution procédurale pour définir des contraintes complexes non exprimables avec le CREATE TABLE ou qui prennent en compte des données issues de plusieurs lignes ou de plusieurs tables.

Exemple : Garantir qu'un client ne peut avoir que deux factures non payées.

Lorsque la contrainte d'intégrité peut être exprimée au niveau de la définition de la table, c'est cette solution qui sera privilégiée car la vérification est plus rapide que l'exécution du trigger. De plus, les contraintes d'intégrités garantissent que toutes les lignes de la table respectent ces contraintes, tandis que les triggers ne prennent pas en compte les données déjà présentes dans la table lorsqu'ils sont définis.

II UTILISATION DES DECLENCHEURS LMD

Les déclencheurs servent à maintenir une intégrité référentielle de bas niveau et non à envoyer des résultats de requête. Leur avantage principal réside dans le fait qu'ils peuvent contenir une logique de traitement complexe. Ils doivent être employés lorque les contraintes n'offrent pas les fonctionnalités nécessaires.

Par exemple, vous pourrez utiliser les déclencheurs pour :

- **Modifier en cascade les tables liées dans une base de données**
- **Mettre en œuvre une intégrité des données plus complexe qu'une contrainte CHECK**
  A la différence des contraintes CHECK, les déclencheurs peuvent référencer des colonnes d'autres tables. Par exemple, dans une gestion commerciale, lorsque la commande d'un article est passée, une ligne est insérée dans la table Lignes de Commandes. Un déclencheur INSERT sur cette table pourra déterminer si la commande peut être livrée ou non, en examinant la colonne quantité en stock dans la table Stock. Si cette valeur est insuffisante, il pourra générer automatiquement un ordre de commande fournisseur et avertir le gestionnaire.
- **Renvoyer des messages d'erreur personnalisés**
  Les règles, les contraintes et les valeurs par défaut ne peuvent communiquer des erreurs que par l'intermédiaire des messages d'erreur système standards.

Règles lors de l'utilisation de déclencheurs :

- Les déclencheurs sont réactifs alors que les contraintes ont un caractère préventif. Les contraintes sont contrôlées en premier, les déclencheurs sont exécutés en réponse à une instruction INSERT, UPDATE ou DELETE.
- Il est possible de créer des déclencheurs sur des vues, ceux-ci sont de type INSTEAD OF.
- Les déclencheurs ne doivent pas renvoyer de jeux de résultats.
- Le propriétaire de la table et les rôles admin peuvent créer et supprimer des déclencheurs sur une table. De plus, le créateur du déclencheur doit avoir la permission d'exécuter toutes les instructions sur toutes les tables. Si l'une des permissions est refusée, la transaction est annulée totalement.

- Les données de la table à laquelle est associé le TRIGGER sont inaccessibles depuis les instructions du bloc. Seule la ligne en cours de modification est accessible à l'aide de deux variables de type RECORD, **OLD** et **NEW**, qui reprennent la structure de la table ou de la vue associée. Ces variables peuvent être utilisées dans la clause WHEN du trigger ou dans le bloc d'instruction. Dans ce dernier cas, elles sont référencées comme des variables hôtes avec le préfixe « : »

  Exemple : :OLD.nom_champ, : NEW.nom_champ

  Le terme OLD permet de connaître la ligne en cours de suppression dans un trigger DELETE ou la ligne avant modification dans un trigger UPDATE.

  Le terme NEW permet de connaître la nouvelle ligne insérée dans un trigger INSERT ou la ligne après modification dans le trigger UPDATE.

- Les prédicats INSERTING, UDATING et DELETING permettent au sein d'un déclencheur commun pour les instructions INSERT, UPDATE et DELETE de savoir si le bloc PL/SQL est exécuté à la suite d'une insertion une modification ou d'une suppression. Ces prédicats retournent une valeur booléenne et sont utilisés dans la condition de test d'une instruction IF. Il est ainsi possible d'écrire un déclencheur commun aux instruction INSERT et UPDATE, par exemple, tout en conservant l'exécution conditionnelle de certaines instructions.

III CREATION DE DECLENCHEURS

La création de déclencheurs s'effectue à l'aide de l'instruction CREATE TRIGGER. Cette instruction spécifie la table sur laquelle le déclencheur est défini, l'événement provoquant son exécution et les instructions qu'il contient.

```
CREATE [or REPLACE] TRIGGER nom_trigger
[BEFORE/AFTER/INSTEAD OF]
{INSERT/UPDATE[OF col,…]/DELETE}
ON Nom_Table [FOR EACH ROW]
[WHEN (condition)] Bloc PL/SQL;
```

OR REPLACE

Remplace la description du TRIGGER s'il existe déjà.

BEFORE

Le bloc PL/SQL est exécuté AVANT la vérification des contraintes de la table et la mise à jour des données.

AFTER

Le bloc PL/SQL est exécuté APRES la mise à jour des données dans la table.

INSTEAD OF

Le bloc PL/SQL qui suit remplace le traitement standard associé à l'instruction qui à déclenché le trigger.

INSERT/UPDATE …/DELETE

Instruction associée au déclenchement du trigger. Plusieurs instructions peuvent déclencher le même trigger. Elles sont combinées par l'opérateur OR.

FOR EACH ROW

Le trigger s'exécute pour chaque ligne traitée par l'instruction associée.

WHEN (condition)

La condition donnée doit être vérifiée pour que le code s'exécute.

IV PRINCIPE DE FONCTIONNEMENT

IV.1 DECLENCHEUR SUR EVENEMENT AFTER

Les étapes suivantes montrent comment un déclencheur **AFTER** est lancé lors d'un événement sur la table où il est défini.

Cas 1 : événement INSERT

- Une instruction INSERT est exécutée sur une table comportant un déclencheur INSERT
- L'instruction INSERT est journalisée dans le journal des transactions. Le record **NEW** reçoit la copie de la ligne ajoutée à la table
- Le déclencheur est lancé et ses instructions s'exécutent

Cas 2 : événement DELETE

- Une instruction DELETE est exécutée sur une table comportant un déclencheur DELETE
- L'instruction DELETE est journalisée dans le journal des transactions. Le record **OLD** reçoit la copie de la ligne supprimée de la table.
- Le déclencheur est lancé et ses instructions s'exécutent

Cas 3 : événement UPDATE

- Une instruction UPDATE est exécutée sur une table comportant un déclencheur UPDATE.
- L'instruction UPDATE est journalisée dans le journal des transactions sous la forme INSERT et DELETE.
  Le record **OLD** reçoit la copie de la ligne de la table représentant **l'image avant** la modification.
  Le record **NEW** reçoit la copie de la ligne de la table **représentant l'image après** la modification.
- Le déclencheur est lancé et ses instructions s'exécutent.

Exemple 1: Création d'un déclencheur AFTER **Vente_Insert** sur l'instruction **INSERT** de la table Ventes de structure

VENTES (vnt_art, vnt_cli, vnt_qte, vnt_prix)

Lorsqu'une ligne est insérée dans la table VENTES, le déclencheur décrémente la colonne quantité en stock dans la table ARTICLES de la quantité vendue ;

ARTICLES (art_num, art_nom, art_coul, art_pa, art_pv, art_qte, art_frs)

```
CREATE TRIGGER Vente_Insert
AFTER INSERT
ON Ventes
For each row
BEGIN
        UPDATE Article SET Art_Qte = Art_Qte – :new.Vnt_Qte
        Where article.Art_Num. = :new.Vnt_Art
END ;
```

Dans l'exemple ci-dessus, le record new contient la ligne de la table VENTES qui vient d'être ajoutée, et les colonnes de la table VENTES sont manipulables à travers le record new**.**

Un déclencheur peut être défini avec l'instruction IF UPDATE, qui contrôle la mise à jour d'une colonne donnée.

IV.2 DECLENCHEURS INSTEAD OF

Le principe de fonctionnement des déclencheurs **INSTEAD OF** est simple.

L'instruction appelante est interceptée, donc non réellement exécutée, et le code du déclencheur la remplace. Il est ainsi possible de tester les valeurs insérées, mises à jour ou supprimées pour décider de la suite des opérations.

Les déclencheurs **INSTEAD OF** peuvent être définis sur des vues : un déclencheur sur une vue permet d'étendre la possibilité de mise à jour de vues multi tables ; un seul déclencheur **INSTEAD OF** par instruction est autorisé sur une table ou vue.

Exemple 2 : Création d'un déclencheur INSTEAD OF **Insert_Multiple** sur l'instruction **INSERT** de la vue multi tables, Vue_TousClients qui regroupent les clients français et étrangers.

Lorsqu'une ligne est insérée, le déclencheur met à jour les tables concernées ClientsF et ClientsE.

```
CREATE TRIGGER Insert_Multiple
INSTEAD OF INSERT
ON Vue_TousClients
FOR EACH ROW
BEGIN
If (select payC from new) = 'F'
      Insert ClientsF select * from new
ELSE
      Insert ClientsE select * from new
END
```

Etablissement référent
Direction de l'ingénierie Neuilly

Equipe de conception
Groupe d'étude de la filière étude - développement

Remerciements :

Date de mise à jour 30/07/2014
afpa © Date de dépôt légal juillet 14

afpa / Direction de l'Ingénierie13 place du Générale de Gaulle / 93108 Montreuil Cedex
association nationale pour la formation professionnelle des adultes
Ministère des Affaires sociales du Travail et de la Solidarité