Editions ENI

Oracle 12c
SQL, PL/SQL, SQL*Plus

Collection
Ressources Informatiques

Extrait

Chapitre 5
Objets de la base utilisant PL/SQL

1. Introduction

En plus des blocs PL/SQL anonymes utilisés par SQL*Plus ou par les outils de développement (Oracle*Forms, Oracle*Reports...), on peut utiliser le PL/SQL dans des objets de la base, comme les procédures stockées (PROCEDURE, FUNCTION, PACKAGE) et les déclencheurs de bases de données.

2. Les déclencheurs de bases de données

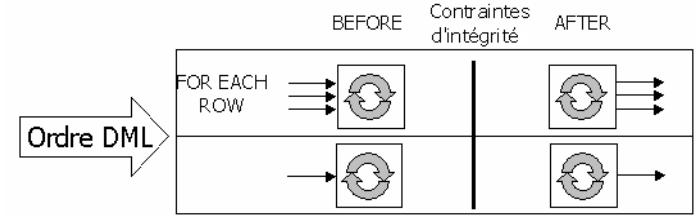
Un déclencheur ou trigger est un bloc PL/SQL associé à une table. Ce bloc s'exécutera lorsqu'une instruction du DML (INSERT, UPDATE, DELETE) sera demandée sur la table.

Le bloc PL/SQL qui constitue le trigger peut être exécuté avant ou après la mise à jour et donc avant ou après vérification des contraintes d'intégrité.

Les triggers offrent une solution procédurale pour définir des contraintes complexes ou qui prennent en compte des données issues de plusieurs lignes ou de plusieurs tables. Par exemple, pour garantir le fait qu'un client ne peut pas avoir plus de deux commandes non payées. Toutefois, les triggers ne doivent pas êtres utilisés lorsqu'il est possible de mettre en place une contrainte d'intégrité. En effet, les contraintes d'intégrité étant définies au niveau de la table et faisant partie de la structure de la table, la vérification du respect de ces contraintes est beaucoup plus rapide.

De plus, les contraintes d'intégrité garantissent que toutes les lignes contenues dans la tables respectent ces contraintes, tandis que les triggers ne prennent pas en compte les données déjà présentes dans la table lorsqu'ils sont définis.

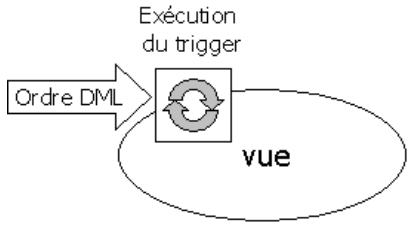
Le bloc PL/SQL associé au trigger peut être exécuté pour chaque ligne affectée par l'ordre DML (option FOR EACH ROW), ou bien une seule fois pour chaque commande DML exécutée (option par défaut).



Exécution avant ou après vérification des contraintes d'intégrité pour chaque ligne ou chaque ordre.

Dans les triggers BEFORE et FOR EACH ROW, il est possible de modifier les données qui vont être insérées dans la table pour qu'elles respectent les contraintes d'intégrité. Il est également possible de faire des requêtes de type SELECT sur la table sur laquelle porte l'ordre DML uniquement dans le cadre d'un trigger BEFORE INSERT. Toutes ces opérations sont impossibles dans les triggers AFTER car après vérification des contraintes d'intégrité, il n'est pas possible de modifier les données et comme la modification (ou ajout, ou suppression) de la ligne n'est pas terminée, il n'est pas possible d'effectuer des SELECT sur la table.

Il est également possible de poser des triggers sur les vues (VIEW) afin d'intercepter les ordres DML que l'on peut y exécuter. Ces triggers permettent de maîtriser toutes les opérations qui sont effectuées sur les vues et pour l'utilisateur final, la vue est en tout point similaire à une table puisqu'il peut y faire les opérations INSERT, UPDATE et DELETE. Ces triggers sont de type INSTEAD OF, c'est-à-dire que leur exécution va remplacer celle de la commande DML à laquelle ils sont associés. Ce type de trigger n'est définissable que sur les vues, et seul ce type de trigger peut être mis en place sur les vues.



Principe de fonctionnement des triggers instead of

Syntaxe

```
CREATE [OR REPLACE] TRIGGER nom_trigger
{BEFORE/AFFTER/INSTEAD OF}
{INSERT/UPDATE[OF col,...]/DELETE}
ON nom_table [FOR EACH ROW]
[FOLLOWS nom_autre_trigger[,...]]
[ENABLE/DISABLE]
[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 tables et la mise à jour des données dans la table.

`AFTER`

Le bloc PL/SQL est exécuté après 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 a déclenché le trigger (pour une vue uniquement).

`INSERT/UPDATE [OF col, ...]/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.

`FOLLOWS nom_autre_trigger[,...]`

Oracle permet de définir plusieurs triggers pour la même table et le même événement. Dans ce cas, l'ordre relatif de déclenchement de ces triggers est indéterminé. Si l'ordre de déclenchement de ces triggers est important pour votre application, vous pouvez utiliser la clause FOLLOWS apparue en version 11. Cette clause permet d'indiquer que le trigger doit être déclenché après les triggers mentionnés.

`ENABLE/DISABLE`

Cette clause permet d'indiquer si le trigger est actif ou non dès sa création ; par défaut, un trigger nouvellement créé est actif. Créer un trigger désactivé permet de vérifier qu'il se compile correctement avant de le mettre réellement en service. Un trigger créé désactivé peut ensuite être activé par un ordre ALTER TRIGGER … ENABLE.

`WHEN (condition)`

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

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 RECORD, OLD et NEW, qui reprennent la structure de la TABLE ou de la VIEW associée. Ces variables peuvent être utilisées dans la clause WHEN du trigger ou dans le bloc d'instructions. Dans ce dernier cas, elles sont référencées comme des variables hôtes avec le préfixe "**:**" (:OLD.nom_champs, :NEW.nom_champs).

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 un trigger UPDATE.

Les termes OLD et NEW sont fixés par défaut et il est possible d'utiliser d'autres termes en précisant la clause `REFERENCING OLD AS nouveau_nom NEW AS nouveau_nom`. Cette clause prend place juste avant la clause FOR EACH ROW (si elle existe) dans la définition du trigger.

Exemple

Exécution d'un bloc PL/SQL avant une suppression dans la table CLIENTS du user FLORIAN :

```
CREATE TRIGGER avant_sup_cli
      BEFORE    DELETE
      ON FLORIAN.CLIENTS
      DECLARE
         ...
      BEGIN
         ...
      END ;
```

Exécution d'un bloc PL/SQL après mise à jour de chaque ligne de la table ARTICLES si l'ancien prix est supérieur au nouveau :

```
create or replace trigger post_majprix
  after update of prix
  on articles
  for each row
  when (old.prix > new.prix)
declare
  ...
begin
  ...
end;
```

Pour chaque commande on souhaite connaître le nom de l'utilisateur Oracle qui a réalisé la saisie. La première étape consiste à ajouter une nouvelle colonne à la table des commandes. Cette colonne doit accepter la valeur NULL car pour les lignes de commande existantes, le nom de l'utilisateur Oracle est inconnu.

Modification de la table des commandes :

```
SQL> alter table commandes
  2    add (utilisateur varchar2(30));

Table modifiée.

SQL>
```

Dans le trigger, le nom de la nouvelle ligne est changé, il s'exécute avant vérification des contraintes d'intégrité pour chaque ligne insérée dans la table commandes.

Définition du trigger :

```
SQL> create or replace trigger bf_ins_commandes
  2    before insert
  3    on commandes
  4    referencing new as nouvelle
  5    for each row
  6  begin
  7    select user into :nouvelle.utilisateur from dual;
  8  end;
  9  /

Déclencheur créé.

SQL>
```

On souhaite connaître le nombre de commandes saisies par utilisateur Oracle. Pour éviter d'écrire une requête qui parcourt la totalité de la table des commandes, ce qui peut être très lourd, une table de statistiques va être alimentée par le trigger.

Création de la table de statistiques :

```
SQL> create table stat_util(
  2    utilisateur varchar2(30),
  3    nombre integer);

Table créée.

SQL>
```

Le trigger doit s'assurer que l'utilisateur existe dans la table des statistiques, et s'il n'est pas déjà présent alors il faut le créer. Le trigger s'exécute après chaque insertion de ligne, pour des raisons d'optimisation en cas de violation des contraintes d'intégrité.

Editions ENI

Oracle 12c
Administration

Collection
Ressources Informatiques

Extrait

Chapitre 7
Création d'une nouvelle base de données

1. Vue d'ensemble

1.1 Étapes de création d'une nouvelle base de données pour une application

Le processus complet de création d'une nouvelle base de données pour une application comporte les grandes étapes suivantes :

Conception du modèle physique

- Définir tous les objets (Oracle) de l'application : tables, contraintes d'intégrité (clés primaires/uniques/étrangères), index, vues, programmes stockés (triggers, procédures/ fonctions stockées, packages).
- Étudier la volumétrie de l'application (nombre d'utilisateurs, nombre de lignes attendues dans les tables).

Création de la base proprement dite (ce chapitre)

- Créer une nouvelle instance.
- Créer une nouvelle base de données (fichiers de contrôle, fichiers de journalisation et fichiers de données des tablespaces "techniques" d'Oracle).
- Rendre le dictionnaire de données exploitable.
- À ce stade, la base de données peut être vue comme une "enveloppe" (une "boîte vide") dans laquelle des structures vont être créées pour une ou plusieurs applications.

Création des structures de stockage adaptées (chapitre Gestion des tablespaces et des fichiers de données)

– Créer les tablespaces (avec leurs fichiers de données) destinés à stocker les données de l'application (tables et index).
– Les dimensionner en fonction de l'étude de volumétrie réalisée initialement.

Création du compte Oracle qui va contenir les objets de l'application (chapitre Gestion des utilisateurs et de leurs droits)

– Créer le compte.
– Lui donner les privilèges suffisants pour créer les objets.
– L'autoriser à utiliser de l'espace dans les tablespaces de l'application.

Création des objets de l'application dans ce compte Oracle (chapitre Gestion des tables et des index)

– Créer les objets Oracle de l'application (généralement sous la forme d'un ou de plusieurs scripts).

Création des utilisateurs finaux de l'application (chapitre Gestion des utilisateurs et de leurs droits)

– Créer les utilisateurs.
– Leur donner des droits adaptés sur les objets de l'application (i.e. sur les objets créés précédemment dans le compte propriétaire de l'application).

Sauvegarde de la base (chapitre Sauvegarde et récupération)

– Sauvegarde de référence de la base.

Comme vous pouvez le constater, la création de la base de données proprement dite présentée dans ce chapitre n'est qu'une petite étape du processus complet (mais une étape fondamentale).

1.2 Étapes de création de la base de données proprement dite

Les grandes étapes de la création de la base de données proprement dite sont les suivantes :

– Créer les répertoires sur les disques, si possible en respectant les recommandations du standard OFA.
– Préparer un nouveau fichier de paramètres texte, généralement par copie d'un ancien.

- Positionner la variable d'environnement `ORACLE_SID`.
- Créer le service associé à l'instance (plate-forme Windows) ou créer le fichier de mots de passe pour l'identification `SYSDBA` (plate-forme Unix ou Linux).
- Lancer SQL*Plus et se connecter `AS SYSDBA`.
- Créer un fichier de paramètres serveur (pas obligatoire, mais conseillé).
- Démarrer l'instance en état `NOMOUNT`.
- Créer la base de données (ordre SQL `CREATE DATABASE`).
- Finaliser la création du dictionnaire (quelques scripts à exécuter).
- Configurer Oracle Net pour la nouvelle base de données.
- Enregistrer la nouvelle instance dans le fichier `oratab` (plate-forme Unix ou Linux).
- Configurer EM Express.

La création d'une nouvelle base de données suppose l'installation préalable d'Oracle (chapitre Installation).

Remarque

Si le serveur abrite déjà des bases de données Oracle, il est vivement conseillé d'effectuer une sauvegarde de ces bases de données avant de démarrer le processus de création.

Après ces étapes, la nouvelle base de données est ouverte et contient :

- les tablespaces `SYSTEM` et `SYSAUX` avec leur(s) fichier(s) de données associé(s) ;
- éventuellement un tablespace d'annulation et un tablespace temporaire selon les options utilisées ;
- les fichiers de contrôle et de journalisation ;
- les deux comptes DBA standard (`SYS` et `SYSTEM`) ;
- le segment d'annulation `SYSTEM` ;
- le dictionnaire de données.

À ce stade, la base de données est prête pour accueillir des structures complémentaires qui vont constituer l'application.

1.3 Méthodes disponibles

La nouvelle base de données peut être créée à la main avec les outils du système d'exploitation et SQL*Plus ; dans ce cas, il est très simple d'écrire ou de récupérer des scripts et de les réutiliser à chaque fois. Les étapes de création de la base de données proprement dite sont toujours les mêmes et dépendent (relativement) peu des caractéristiques de l'application (et en tout état de cause, des paramètres peuvent être ajustés ultérieurement en fonction des caractéristiques de l'application) ; utiliser des scripts "génériques" de création de bases est donc envisageable.

La nouvelle base de données peut aussi être créée à l'aide d'un assistant graphique, l'assistant **Configuration de base de données**. Cet assistant facilite la création de la base de données en offrant la possibilité d'utiliser des modèles de base de données prêts à l'emploi et/ou en permettant de définir très précisément les caractéristiques de la nouvelle base de données à l'aide de plusieurs écrans. Par ailleurs, il est possible de définir ses propres modèles de base de données, comprenant ou non des fichiers de données prêts à l'emploi puis de les utiliser lors de la création ultérieure d'une nouvelle base de données. L'assistant graphique offre aussi la possibilité de générer les scripts de création de la base de données sans créer la base de données ; c'est un bon moyen pour constituer nos scripts "génériques".

Remarque

Utiliser l'assistant graphique est la méthode recommandée par Oracle pour créer une nouvelle base de données.

2. Création de la base de données manuellement

2.1 Créer les répertoires sur les disques

Pour respecter les recommandations du standard OFA (voir le chapitre Installation), vous devez créer :

– un répertoire d'administration portant le nom de la base de données, situé dans le répertoire `%ORACLE_BASE%\admin` (Windows) ou `$ORACLE_BASE/admin` (Linux/Unix),

– un répertoire de données, portant le nom de la base de données, situé dans un répertoire `oradata` lui-même situé dans `ORACLE_BASE` ou sur un autre volume.

Remarque

Depuis la version 11 et l'apparition du Référentiel de diagnostic automatique, le répertoire d'administration contient moins de répertoires et de fichiers.

Le répertoire d'administration contient généralement les répertoires suivants :

`adump`	Répertoire pour des fichiers d'audit.
`create` ou `scripts`	Répertoire des scripts de création de la base de données.
`dpdump`	Répertoire pour les fichiers d'export.
`pfile`	Répertoire pour les fichiers de paramètres texte.

Si le serveur comporte plusieurs disques, il est judicieux de répartir les différents fichiers de la base de données sur ces disques afin d'optimiser les entrées/sorties et d'éviter les contentions ; dans ce cas, il faut créer d'autres répertoires de données sur les disques concernés.

Un répertoire supplémentaire peut être créé pour la zone de récupération rapide (voir le chapitre Sauvegarde et récupération).

Remarque

Généralement, la base de données et l'instance portent le même nom.

2.2 Préparer un nouveau fichier de paramètres texte

2.2.1 Principes

Comme indiqué dans la section La base de données du chapitre Les bases de l'architecture Oracle, il est conseillé d'utiliser un fichier de paramètres serveur, celui-ci étant initialement créé à partir d'un fichier de paramètres texte.

Pour respecter le standard OFA, ce fichier de paramètres texte doit s'appeler `init.ora` et se trouver dans le sous-répertoire `pfile` du répertoire d'administration. Généralement, ce fichier de paramètres texte est créé par duplication d'un fichier existant ou d'un fichier modèle que vous aurez défini.

Nous ne créerons pas de fichier `init<SID>.ora` (avec une inclusion du fichier `init.ora`) à l'emplacement par défaut de la plate-forme (`dbs` sous Unix/Linux, `database` sous Windows) ; ainsi, nous ne risquons pas de démarrer l'instance par mégarde avec un fichier de paramètres texte.

Il y a plus de 250 paramètres documentés par Oracle ! Il n'est évidemment pas question de les spécifier tous ! Sur la totalité des paramètres, une trentaine de paramètres qu'il convient de connaître, sont suffisants pour la plupart des bases de données.

Certains paramètres seront décrits brièvement dans cette partie puis présentés de manière plus détaillée dans des chapitres ultérieurs.

2.2.2 Les principaux paramètres

Les paramètres ne sont pas listés dans un ordre alphabétique mais dans un ordre thématique.

DB_NAME

Nom de la base (jusqu'à 8 caractères). Généralement DB_NAME est égal au nom de l'instance (ORACLE_SID).

Exemple :

```
DB_NAME = hermes
```

DB_DOMAIN

Localisation logique de la base sur le réseau (jusqu'à 128 caractères). Ce paramètre, associé au paramètre DB_NAME, permet à Oracle de construire le nom global de la base de données, sous la forme DB_NAME.DB_DOMAIN. Ce paramètre est important si la base de données appartient à un système distribué (ou susceptible de l'être) ; sinon, il peut être ignoré.

Exemple :

```
DB_DOMAIN = olivier-heurtel.fr
```

DB_UNIQUE_NAME

Nom unique de base de données (jusqu'à 30 caractères). Des bases de données ayant le même DB_NAME au sein du même DB_DOMAIN (par exemple une base de production et une base de test) doivent avoir un DB_UNIQUE_NAME différent. Ce paramètre est apparu en version 10. Il est, par défaut, égal à DB_NAME.

Ce paramètre doit être spécifié si vous souhaitez ouvrir simultanément sur un serveur deux bases portant le même nom (le même DB_NAME) ; il permet de les différencier.

Exemple :

```
DB_UNIQUE_NAME = hermes_demo
```

COMPATIBLE

Indique un numéro de version d'Oracle avec laquelle la base de données doit être compatible. Valeurs possibles : 11.0.0 jusqu'au numéro de la version actuelle (12.1.0.1). Valeur par défaut : 12.0.0.