



# Triggers et vues

Patricia Serrano Alvarado  
LINA

# Triggers, what for ?

---

- Les contraintes permettent d'assurer une certaine cohérence sémantique et structurelle d'un attribut ou d'une table
  - Unicité, clés, clés référentielles, check
- Comment assurer une cohérence sémantique plus complexe
  - Lorsqu'une table est modifiée, générer la modification d'une autre
  - Lorsqu'une valeur est insérée la modifier avant de la stocker dans la BD
  - Etc.

# Triggers

---

- Règle Événement-Condition-Action (ECA)
  - Événement
    - Insert, delete, update sur une table ou vue
  - Condition
    - Test ou prédicat logique
  - Action
    - Si la condition est satisfaite, code PL/SQL à exécuter

# Trigger

---

- Sont gérés au niveau du SGBD
- Se déclenchent quelque soit l'outil utilisée (sqlplus, formes, applications java, PHP, etc.)

# Utilité des triggers

- Générer automatiquement de valeurs
- Prévenir de manipulations non valides
- Renforcer la sécurité
- Renforcer l'intégrité référentielle des nœuds dans un base de données répartie
- Fournir une journalisation transparente d'événements
- Fournir auditing
- Maintenir de manière synchrone de tables dupliquées
- Obtenir de statistiques sur l'accès aux tables
- Modifier une table lorsqu'une opération DML est réalisée sur une vue
- Etc.

# Type de triggers (1)

Les événements courants (DML):

- INSERT

- S'exécute à chaque opération d'insertion lancée par l'utilisateur ou par un programme

- UPDATE

- S'exécute à chaque opération de mise à jour lancée par l'utilisateur ou par un programme

- DELETE

- S'exécute à chaque opération de suppression lancée par l'utilisateur ou un programme

## Type de triggers (2)

- On peut spécifier si le trigger se déclenche une fois pour tous les tuples d'une requête(**statement-level**) ou une fois par tuple (**row-level**)
- Un trigger peut être déclenché avant (**BEFORE**), après (**AFTER**) ou à la place (**INSTEAD OF**) d'une opération DML
- Les attributs des tables/vues sont accessibles à travers les variables **NEW** et **OLD**

# Syntaxe

```
create [or replace] trigger [schema .] trigger
{ before | after | instead of }
{ dml_event_clause
| { ddl_event [or ddl_event] ...
  | database_event [or database_event] ...
  }
on { [schema .] schema | database }
}
[when ( condition ) ]
{ pl/sql_block | call_procedure_statement }
```



# Les parties d'un trigger

---

## ■ EVENT

- Trigger statement (CREATE AFTER | BEFORE...)

## ■ CONDITION

- Trigger restriction (WHEN...)

## ■ ACTION

- Triggered action (FOR EACH.../)

# Action d'un trigger

---

- Comme dans les procédures stockées l'action d'un trigger peut
  - Contenir de sentences SQL, PL/SQL ou Java
  - Définir de constructeurs PL/SQL comme variables, constantes, curseurs, exceptions
  - Définir constructeurs Java
  - Faire appel à de procédures stockées

# Combinaison des triggers

---

- Les types de triggers peuvent être combinés dans un seul trigger
- Un « if » dans le block peut déterminer quelles commandes a déclenché le trigger
- Les colonnes concernées peuvent être spécifiées

# Maintenance de triggers (



- Création/suppression d'un trigger
  - CREATE TRIGGER nom\_déclencheur;
  - REPLACE TRIGGER nom\_déclencheur;
  - DROP TRIGGER nom\_déclencheur;
- Activation/désactivation d'un trigger
  - ALTER TRIGGER nom\_déclencheur DISABLE;
  - ALTER TRIGGER nom\_déclencheur ENABLE;
- Activer/désactiver tous les triggers définis sur une table
  - ALTER TABLE nom\_table DISABLE ALL TRIGGERS;
  - ALTER TABLE nom\_table ENABLE ALL TRIGGERS;

# Maintenance de triggers (2)

---

- Les informations sur les triggers sont visibles à travers les vues du dictionnaire de données
  - USER\_TRIGGERS pour les triggers appartenant au schéma
  - ALL\_TRIGGERS pour les triggers appartenant aux schémas accessibles
  - DBA\_TRIGGERS pour les triggers appartenant à tous les schémas

# Conditions d'erreurs

- Dans un trigger, différentes conditions d'erreur peuvent être définies à travers la procédure

`RAISE_APPLICATION_ERROR`

- Le numéro d'erreur peut varier de 20001 à 20999
- Si une erreur est levée, l'événement du trigger ne se réalise pas

# Example



```
create or replace trigger BOOKSHELF_BEF_DEL
before delete on BOOKSHELF
declare
    weekend_error EXCEPTION;
    not_library_user EXCEPTION;
begin
    if TO_CHAR(SysDate,'DY') = 'SAT' or
       TO_CHAR(SysDate,'DY') = 'SUN' THEN
        RAISE weekend_error;
    end if;
    if SUBSTR(User,1,3) <> 'LIB' THEN
        RAISE not_library_user;
    end if;
EXCEPTION
    WHEN weekend_error THEN
        RAISE_APPLICATION_ERROR (-20001,
            'Deletions not allowed on weekends');
    WHEN not_library_user THEN
        RAISE_APPLICATION_ERROR (-20002,
            'Deletions only allowed by Library users');
end;
/
```

# Appel à procédures



- A partir d'un trigger de procédures peuvent être appelées

```
create or replace trigger BOOKSHELF_AFT_INS_ROW
after insert on BOOKSHELF_AUDIT
for each row
begin
call INSERT_BOOKSHELF_AUDIT_DUP(:new.Title, :new.Publisher,
    :new.CategoryName, :new.Old_Rating, :new.New_Rating,
    :new.Audit_Date);
end;
/
```



# Triggers vs contraintes d'intégrité

- Possibilité d'utilisation dans les mêmes situations
- Utiliser les triggers lorsque l'utilisation de contraintes n'est pas possible
- Lorsqu'il est possible utiliser les contraintes
  - NOT NULL, UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
  - DELETE CASCADE
  - DELETE SET NULL

# Triggers en cascade

## SQL Statement

```
UPDATE t1 SET ...;
```

Fires the  
UPDATE\_T1  
Trigger

## UPDATE\_T1 Trigger

```
BEFORE UPDATE ON t1  
FOR EACH ROW  
BEGIN  
  .  
  .  
  INSERT INTO t2 VALUES (...);  
  .  
  .  
END;
```

Fires the  
INSERT\_T2  
Trigger

## INSERT\_T2 Trigger

```
BEFORE INSERT ON t2  
FOR EACH ROW  
BEGIN  
  .  
  .  
  INSERT INTO ... VALUES (...);  
  .  
  .  
END;
```

etc.

- Faire attention au déclenchement en cascade des triggers

# Misellaneous

- Les nouvelles valeurs peuvent être modifiées  
`:new.UpperName:=UPPER(:new.name);`
- Les fonctions UID, USER, USERENV, and SYSDATE peuvent être utilisées dans le block PL/SQL
- Il est conseillé que le nom d'un trigger indique la table sur laquelle il porte ainsi que le type de trigger

# Triggers utilisateur et système

- Depuis la version Oracle 8i, il est possible d'utiliser des déclencheurs pour suivre les changements d'état du système ainsi que les connexions/déconnexions utilisateur et la surveillance des ordres DDL et DML
- Lors de l'écriture de ces déclencheurs, il est possible **d'utiliser des attributs** pour identifier précisément l'origine des événements et adapter les traitements en conséquence

# Attributs (1)

- **ora\_client\_ip\_adress** Adresse IP du poste client qui se connecte
- **ora\_database\_name** Nom de la base de données
- **ora\_des\_encrypted\_password** Description codée du mot de passe de l'utilisateur créé ou modifié
- **ora\_dict\_obj\_name** Nom de l'objet visé par l'opération DDL
- **ora\_dict\_obj\_name\_list** Liste de tous les noms d'objets modifiés
- **ora\_dict\_obj\_owner** Propriétaire de l'objet visé par l'opération DDL
- **ora\_dict\_obj\_owner\_list** Liste de tous les propriétaires d'objets modifiés
- **ora\_dict\_obj\_type** Type de l'objet visé par l'opération DDL
- **ora\_grantee** Liste des utilisateurs disposant du privilège
- **ora\_instance\_num** Numéro de l'instance
- **ora\_is\_alter\_column** Vrai si la colonne en paramètre a été modifiée

# Attributs (2)

- **ora\_is\_creating\_nested\_table** Création ou non d'une table de fusion
- **ora\_is\_drop\_column** Modification ou non de la colonne en paramètre
- **ora\_is\_servererror** Vrai si le numéro erreur passé en paramètre se trouve dans la pile des erreurs
- **ora\_login\_user** Nom de la connexion
- **ora\_privileges** Liste des privilèges accordés ou retirés par un utilisateur
- **ora\_revokee** Liste des utilisateurs à qui le privilège a été retiré
- **ora\_server\_error** Numéro d'erreur dans la pile dont la position est passée en paramètre
- **ora\_sysevent** Nom de l'évènement système qui a activé le déclencheur
- **ora\_with\_grant\_option** Vrai si le privilège a été accordé avec option d'administration

# Triggers système

- CREATE TRIGGER nom\_déclencheur {BEFORE|AFTER} événement\_système ON{DATABASE|SCHEMA} bloc PL/SQL
- Événements système
  - STARTUP
    - Événement déclenché lors de l'ouverture de l'instance (AFTER seulement)
  - SHUTDOWN
    - Événement déclenché avant le processus d'arrêt de l'instance (non déclenché en cas d'arrêt brutal du serveur) (BEFORE seulement)
  - SERVERERROR
    - Événement déclenché lors d'une erreur Oracle (sauf ORA-1034, ORA-1403, ORA-1422, ORA-1423 et ORA-4030) (AFTER seulement)

# Trigger utilisateur (1)

- `CREATE TRIGGER nom_déclencheur {BEFORE|AFTER} événement_utilisateur ON{DATABASE|SCHEMA} bloc PL/SQL`
- Événements utilisateur
  - LOGON Après une connexion (AFTER seulement)
  - LOGOFF Avant une déconnexion (BEFORE seulement)
  - CREATE Lors de la création d'un objet
  - ALTER Lors de la modification d'un objet
  - DROP Lors de la suppression d'un objet
  - ANALYZE Lors de l'analyse d'un objet
  - ASSOCIATE STATISTICS Lors de l'association d'une statistique



# Trigger utilisateur (2)

- Événements utilisateur
  - AUDIT Lors de la mise en place d'un audit
  - NOAUDIT Lors de l'annulation d'un audit
  - COMMENT Lors de l'insertion d'un commentaire
  - DDL Lors de l'exécution des ordres DDL (sauf ALTER DATABASE, CREATE CONTROLFILE et CREATE DATABASE)
  - DISSOCIATE STATISTICS Lors de la dissociation d'une statistique
  - GRANT Lors de l'exécution d'une commande GRANT
  - RENAME Lors de l'exécution d'une commande RENAME
  - REVOKE Lors de l'exécution d'une commande REVOKE
  - TRUNCATE Lors d'une troncature de table

# Pour terminer qqs remarques

---

- Un déclencheur a un nom.
- Il est associé à une table et une seule.
- Il peut être actif ou inactif.
- Il est opérationnel jusqu'à sa suppression ou la suppression de la table à laquelle il est associé.
- Il n'est pas modifiable.

# Activer/désactiver/supprimer



- **Alter trigger** <nom\_decl> **enable/disable**;  
(désactiver un déclencheur donné)
- **Alter table** <nom\_table> **disable all triggers**;  
(désactiver tous les déclencheurs associés à une table)
- **Drop trigger** <nom\_decl>;

# Inconvénients

---

- L'utilisation de triggers peut ralentir l'exécution des requêtes notamment lors de la manipulation massive de données



# Les vues



Une introduction

# Les vues, what for?

---

- Indépendance entre la couche SGBD et la couche applicative
- Facilité d'usage de la BD
- Facilité et amélioration du contrôle de la confidentialité
- Amélioration des performances

# Vue

- Une vue est une requête stockée qui est interrogée comme une table
- Les contenu (tuples) de la vue ne sont pas stockées par défaut
- La vue facilite la vie du programmeur, pas du SGBD...
- Pendant l'exécution des requêtes, les vues seront remplacées par les tables dont la vue dépend

# Création de vues



```
CREATE VIEW <nom-vue> AS  
    <requête définissant le vue>
```

## ■ Exemple

Considérer la table :

Movies(title, year, length, genre, studioName,  
producerC)

```
CREATE VIEW ParamountMovies AS  
    SELECT title, year  
    FROM Movies  
    WHERE studioName='Paramount';
```



# Interrogation de vues

---

- On interroge une vue comme on interroge une table
- Pour l'exécution, le SGBD enrichi la requête de l'utilisateur avec la définition de la vue
- 2 techniques
  - Ré-écriture de requêtes
  - Concaténation d'arbres relationnels

# Mise à jour de vues



- Est-il envisageable de « mettre à jour » une vue ? i.e., requêtes INSERT|DELETE|UPDATE
  - Pour certaines vues oui
    - Il est possible de « traduire » la mise à jour sur la vue vers les tables dont la vue dépend
    - WITH CHECK OPTION: les mises à jour ou les insertions faites à travers la vue ne produisent que des lignes qui font partie de la sélection de la vue
  - Pour la majorité des vues non
    - Exemple

# Règles SQL pour faire des vues modifiables



- En gros, dans la création des vues
  1. Utiliser SELECT et **pas** SELECT DISTINCT
  2. La clause FROM doit consister en une seule occurrence d'une relation (R) et aucune autre relation
  3. La clause WHERE ne doit pas utiliser R dans une sous requête
  4. La liste d'attributs dans le SELECT doit contenir tous les attributs NOT NULL (qui n'ont pas de valeurs par défaut) pour que les autres soient remplis avec de valeurs NULL

# INSTEAD OF trigger

---

- On peut prendre en main le problème de cohérence avec le trigger INSTEAD OF
- Avec ce trigger on peut re-envoyer les opérations DML vers les tables de base de la vue
- Exemple

# Dictionnaire des données Oracle

---

- Le dictionnaire des données est constitué d'un ensemble de tables et de vues qui mémorisent toutes les informations nécessaires au SGBD pour gérer la base de données
- Les tables/vues du dictionnaire de données sont interrogeables selon les privilèges système accordés à chaque utilisateur
- L'accès est uniquement en lecture

# Classe de vues « user »

---

- Les vues de cette classe sont préfixées par *user\_*
- Permettent une vérification de tous les objets (tables, vues triggers, ...) dont l'utilisateur est propriétaire.
- *User\_views, user\_triggers, user\_tables, user\_constraints, user\_errors...*

## Mais aussi :

|                     |   |
|---------------------|---|
| ROLE_SYS_PRIVS      | This view contains information about system privileges granted to roles. Information is provided only about roles to which the user has access. |
| ROLE_TAB_PRIVS      | This view contains information about object privileges granted to roles. Information is provided only about roles to which the user has access. |
| USER_COL_PRIVS      | Describes column object grants for which the current user is the object owner, grantor, or grantee  |
| USER_COL_PRIVS_MADE | Describes column object grants for which the current user is the grantor  |
| USER_COL_PRIVS_RECD | Describes column object grants for which the current user is the grantee  |
| USER_ROLE_PRIVS     | Lists roles granted to the current user   |
| USER_TAB_PRIVS      | Lists grants on all objects where the current user is the grantee   |
| USER_SYS_PRIVS      | Lists system privileges granted to the current user   |
| USER_TAB_PRIVS_MADE | Lists grants on all objects owned by the current user   |
| USER_TAB_PRIVS_RECD | Lists object grants for which the current user is the grantee   |
| SESSION_PRIVS       | Lists the privileges that are currently enabled for the user  |
| SESSION_ROLES       | Lists the roles that are currently enabled to the user  |

Etc.

# D' autres dictionnaires

- SESSION\_ROLES
  - Tous les rôles actifs
- USER\_SOURCE
  - Le code des procédures appartenant à l'utilisateur
- ALL\_SOURCE
  - Le code des procédures appartenant à l'utilisateur ou à ceux auxquels il a accès
- DBA\_SOURCE
  - Toutes les procédures de la BD
- USER\_CATALOG
  - Information sur les tables, vues, séquences et synonymes de l'utilisateur
- USER\_OBJECTS
  - Tout type d'objet Oracle (*clusters, database links, directories, functions, indexes, libraries, packages, java classes, abstract datatypes, resource plans, sequences, synonyms, tables, triggers, materialized views, LOBs, and views*)
- Etc



# Bibliographie

- Hector Garcia Molina, Jeffrey D. Ulman and Jennifer Widom. Database Systems. Second Edition, Pearson Prentice Hall, International Edition. 2009.
- Georges Gardarin. Bases de Données. Eyrolles, sixième tirage, 2005.

