

# **SGBD - 2<sup>e</sup>**

## Chapitre 8 - Contraintes d'intégrités et déclencheurs

---

Daniel Schreurs

14 février 2022

Haute École de Province de Liège

# Table des matières du chapitre i

1. Introduction
2. Contraintes attachées aux tables
3. Particularités d'Oracle
4. Les déclencheurs

# Introduction

---

# Table des matières de la section : Introduction i

## 1. Introduction

### 1.1 Définition

### 1.2 Contraintes inhérentes au modèle relationnel

### 1.3 Contraintes liées à une application particulière

## 2. Contraintes attachées aux tables

## 3. Particularités d'Oracle

## 4. Les déclencheurs

Ce chapitre s'inscrit dans le cadre général de l'intégrité des données. Il a pour objectif d'en décrire la partie contrôle sémantique. Le contrôle sémantique des données : assure la cohérence des informations stockées par rapport à leur signification dans la réalité

Exemples :

- La cote d'un élève doit être comprise entre 0 et 20
- La valeur d'un stock ne peut être négative

Dans une base de données relationnelle :

- Contraintes inhérentes au modèle relationnel
- Contraintes liées à une application particulière

Ou contraintes structurelles<sup>1</sup> :

- Intégrité de domaine
- Intégrité de relation
- Intégrité de référence

---

1. (voir chapitre 2 : MRD et chapitre 3 : LDD)

# Introduction : Contraintes liées à une application particulière

Ou **contraintes applicatives** ou contraintes spécifiques :

Il s'agit d'une condition que doit vérifier un sous-ensemble de la base afin que l'on puisse affirmer que les informations sont cohérentes.



- Il faut pouvoir exprimer un ensemble de contraintes que doit satisfaire la base ;
- Il faut que le SGBD puisse, au cours des modifications des données, assurer que la base obéit toujours aux contraintes.
- Lors de la création d'une contrainte, le SGBD vérifie qu'elle n'est pas en désaccord avec les valeurs présentes dans la base.
  - En désaccord : rejetée
  - Pas en désaccord : enregistrée et immédiatement opérationnelle

- SQL 2 fait la distinction entre
  - contraintes générales qui peuvent faire intervenir plusieurs colonnes de plusieurs tables
  - contraintes attachées aux tables de base
- Remarque : une contrainte attachée à une table ne peut exister sans la table et est donc effacée en même temps que la table

## **Contraintes attachées aux tables**

---

# Contraintes attachées aux tables

Les contraintes attachées à une table se définissent lors de la création de la table (CREATE TABLE) ou après la création de la table par la commande ALTER TABLE. Voir le chapitre 3 : LDD

## **Particularités d'Oracle**

---

Oracle possède la clause `[NOT] DEFERRABLE`. Pour Oracle, une contrainte déclarée `DEFERRABLE` pourra être différée pendant la durée d'une transaction à condition de le préciser explicitement au moyen de la commande `SET CONSTRAINT`.

# Particularités d'Oracle : DEFERRABLE

## Exemple

```
1 CREATE TABLE t (  
2   c1 integer,  
3   c2 integer,  
4   CONSTRAINT c CHECK (c1 < c2) DEFERRABLE);
```

**La contrainte N'EST PAS différée!** Pour qu'elle le soit :

```
SET CONSTRAINT c DEFERRED;
```

# Particularités d'Oracle : IMMEDIATE

## Exemple

```
1 CREATE TABLE t
2 (
3     c1 integer,
4     c2 integer,
5     CONSTRAINT c CHECK (c1 < c2)
6         INITIALLY DEFERRED
7 );
```

**Pour inverser le comportement d'une contrainte INITIALLY DEFERRED, on utilise la commande** Pour qu'elle le soit :

**SET CONSTRAINT c IMMEDIATE;**



# Particularités d'Oracle : En résumé

- **INITIALLY IMMEDIATE** correspond à **NOT DEFERRABLE** de la norme et est le comportement par défaut
- **INITIALLY DEFERRED** est équivalent à **DEFERRABLE** de la norme

# **Les déclencheurs**

---

# Les déclencheurs : Introduction

- Un déclencheur (trigger) permet de définir un ensemble d'actions qui sont déclenchées automatiquement par le SGBD lorsque certains phénomènes se produisent.
- Les actions sont enregistrées dans la base et non plus dans les programmes d'application.
- Cette notion n'est pas spécifiée dans SQL2, elle le sera dans SQL3.

Les déclencheurs permettent de définir des contraintes dynamiques, ils peuvent être utilisés pour :

- Générer automatiquement une valeur de clé primaire,
- Résoudre le problème des mises à jour en cascade
- Enregistrer les accès à une table
- Gérer automatiquement la redondance
- Empêcher la modification par des personnes non autorisées (dans le domaine de la confidentialité)
- Mettre en œuvre des règles de fonctionnement plus complexes

# Les déclencheurs : Déclencheurs dans Oracle

- Les déclencheurs permettent de réaliser des opérations sophistiquées, car ils constituent un bloc **PL/SQL**
- Depuis Oracle8i, il est possible de définir des déclencheurs s'activant suite à des **commandes du LDD** ou à certains événements systèmes
- Nous nous concentrerons sur les déclencheurs réagissant à l'exécution de commandes du LMD
- Il existe 12 types de déclencheurs sensibles aux **commandes LMD** en fonction de l'instruction déclenchante (ajout, suppression, modification), du niveau du déclencheur (ligne ou table) et du moment du déclenchement (avant ou après)

# Les déclencheurs : Syntaxe

## Syntaxe

```
1 CREATE TRIGGER nom_déclencheur
2   BEFORE | AFTER
3     DELETE | INSERT | UPDATE | OF liste_colonne
4     ON nom_table
5   [FOR EACH ROW]
6   [WHEN condition]
7   [bloc PL/SQL];
```

# Les déclencheurs : FOR EACH ROW

**FOR EACH ROW** : déclencheur du niveau tuple : sera exécuté pour toutes les lignes provoquant l'activation du déclencheur. En son absence, le déclencheur est du niveau table et le bloc PL/SQL n'est exécuté qu'une seule fois.

**WHEN** : permet de préciser une condition supplémentaire. Le bloc PL/SQL ne sera exécuté que si la condition de la clause WHEN est évaluée à vrai. Ex : vérifier, lors d'une mise à jour d'une colonne que la nouvelle valeur dépasse l'ancienne valeur.



Deux restrictions sur les commandes du bloc PL/SQL d'un déclencheur :

- Un déclencheur ne peut contenir de **COMMIT** ni de **ROLLBACK**
- Il est impossible d'exécuter une commande du LDD dans un déclencheur
- Un déclencheur de niveau de ligne ne peut pas :
  - Lire ou modifier le contenu d'une table mutante
  - Lire ou modifier les colonnes d'une clé primaire, unique ou étrangère d'une table contraignante!

Table Mutante : table en cours de modification. Pour un déclencheur, il s'agit de la table sur laquelle il est défini

Table contraignante : table qui peut éventuellement être accédée en lecture afin de vérifier une contrainte de référence

# Les déclencheurs : Exemple

Exemple : contrainte dynamique

```
1 CREATE TRIGGER upd_salaire_personnel
2 BEFORE UPDATE OF salaire ON personnel
3 FOR EACH ROW
4 WHEN (OLD.salaire > NEW.salaire)
5 DECLARE
6     salaire_diminue EXCEPTION;
7 BEGIN
8     RAISE salaire_diminue;
9 EXCEPTION
10    WHEN salaire_diminue THEN
11        raise_application_error (-20001,
12                                'Le salaire ne peut diminuer');
13 END;
```

Exemple qui ne fait que lancer une exception

`Raise_application_error` : permet de transmettre un code d'erreur

Les déclencheurs

et un message au bloc appelant `OLD` et `NEW` sont prédéfinis, accèdent

# Les déclencheurs : Exemple i

Dans la table Service : colonne `Nombre_Emp` contient le nombre d'employés de chaque service  $\Rightarrow$  redondance

Exemple : gestion automatique de la redondance

```
1 SELECT num_service, COUNT(*)  
2 FROM personnel  
3 GROUP BY num_service;
```

Pour assurer la cohérence des données, on crée la table Service avec 0 comme valeur par défaut pour la colonne `Nombre_Emp` et on gère la redondance au moyen d'un déclencheur.

# Les déclencheurs : Exemple

Exemple : gestion automatique de la redondance

```
1 CREATE TRIGGER maj_nb_emp
2   BEFORE INSERT OR DELETE OR UPDATE OF num_service
3   ON personnel
4   FOR EACH ROW
5 BEGIN
6   IF inserting THEN
7     UPDATE service
8     SET nombre_emp = nombre_emp + 1
9     WHERE num_service = :NEW.num_service;-- NEW
10  ELSIF deleting THEN
11    UPDATE service
12    SET nombre_emp = nombre_emp - 1
13    WHERE num_service = :OLD.num_service;-- OLD
14  ELSIF updating THEN
15    UPDATE service
16    SET nombre_emp = nombre_emp + 1
17    WHERE num_service = :NEW.num_service;-- NEW
```

Il s'agit d'assurer qu'une mise à jour s'effectue uniformément dans plusieurs tables. Ce problème se pose

- Lorsqu'une clé primaire est modifiée
- Lorsqu'on supprime d'une table un tuple qui est référencé par d'autres tuples dans d'autres tables

# Les déclencheurs : Mise à jour en cascade

Exemple : mise à jour en cascade

```
1 CREATE TRIGGER maj_cascade
2 BEFORE DELETE OR UPDATE OF num_auteur
3   ON auteurs
4   FOR EACH ROW
5   BEGIN
6     IF updating THEN
7       UPDATE a_ecrit SET num_auteur = :NEW.num_auteur
8         WHERE num_auteur = :OLD.num_auteur;
9     ELSIF deleting THEN
10      DELETE FROM a_ecrit
11        WHERE num_auteur = :OLD.num_auteur;
12    END IF;
13  END;
```