

# MS SQL Server 2014 – partie 3

Bases de données : Avancé (420-465-CH)

Sana El Bahloul

# Les déclencheurs (trigger)

## Qu'est ce qu'un déclencheur ?

- ❑ Type particulier de procédure stockée qui est exécuté automatiquement lorsqu'une table est modifiée.
- ❑ Un déclencheur contient des instructions T-SQL, simples ou complexes.
- ❑ Il s'exécute automatiquement, il ne peut pas être appelé, on dit qu'il est alors activé.
- ❑ Il n'est pas possible de créer des déclencheurs sur tables temporaires ou tables systèmes, seulement sur des tables ou vues utilisateurs.
- ❑ SQL Server intègre trois types généraux de déclencheurs :
  - ❑ les déclencheurs DML;
  - ❑ les déclencheurs DDL;
  - ❑ les déclencheurs de connexion;

Nous discuterons plus particulièrement des déclencheurs DML.

# Les déclencheurs DML (trigger)

suite

## Qu'est ce qu'un déclencheur ?

- Un déclencheur DML est une action programmée pour s'exécuter lorsqu'un événement de langage de manipulation de données (Data Manipulation Language, DML) se produit dans un serveur de base de données.
- Les événements DML comprennent les instructions UPDATE, INSERT, DELETE exécutées sur une table ou sur une vue.




## Types de déclencheurs DML

- **Déclencheur AFTER** : est exécuté après l'action associée à une instruction INSERT, UPDATE ou DELETE.
  - Les déclencheurs AFTER peuvent être spécifiés uniquement sur des tables.
- **Déclencheur INSTEAD OF** : est exécuté à la place de l'action de déclenchement habituelle.
  - Les déclencheurs INSTEAD OF peuvent être définis sur une vue avec une ou plusieurs tables de base (permet d'étendre les types de mises à jour pouvant être prises en charge)

# Les déclencheurs (trigger)

suite

## Considérations sur les déclencheurs DML

- Si une instruction tente d'effectuer une opération qui viole une contrainte sur une table ou provoque une autre erreur  le déclencheur n'est pas activé.
- Un déclencheur et l'instruction qui l'active sont exécutés dans une même transaction.
  -  Par conséquent, toute instruction ROLLBACK provoque l'annulation du déclencheur et de l'événement de modification de données.
- Un déclencheur est activé une seule fois par instruction,  même si cette instruction affecte plusieurs lignes de données.
- Lorsqu'un déclencheur est activé ses résultats éventuels sont renvoyés à l'application appelante comme pour une procédure stockée.

# Les déclencheurs (trigger)

## Note importante :

- Normalement, un déclencheur ne devrait pas retourner de résultats à l'application appelante, car les opérations **INSERT**, **UPDATE** et **DELETE** ne génèrent pas de résultats, alors éviter les instructions qui retournent des valeurs, éviter **SELECT**, **SET**.

↳ Si vous voulez utiliser ces instructions, placer au début du déclencheur

SET NOCOUNT ON (permet d'empêcher le renvoi de lignes de résultat par exemple : *23 lignes affectées*) et à la fin SET NOCOUNT OFF.

Ce paramétrage n'affecte pas le renvoi de résultats réels d'une instruction SELECT mais seulement le message du nombre de lignes affectées. Ou encore prévoir, par le programme appelant, le traitement des valeurs retournées.

# Les déclencheurs (trigger)

suite

## Quand utiliser un déclencheur DML

- Utile pour maintenir l'intégrité des données et les règles d'entreprise
- Utile pour effectuer des vérifications de contraintes complexes (en particulier des validations de cohérence, soit les validations qui impliquent des champs d'autres tables).
- Utile pour effectuer des modifications en cascade dans des tables liées (ajout ou suppression).
- Inutile d'utiliser un déclencheur pour valider l'intégrité de l'entité lorsque qu'un index ou une contrainte de PRIMARY KEY peuvent être utilisés.
- Inutile d'utiliser un déclencheur pour valider l'intégrité de domaine lorsqu'on peut travailler avec une contrainte CHECK.
- Inutile d'utiliser un déclencheur pour valider l'intégrité référentielle lorsqu'une contrainte FOREIGN KEY peut faire ce travail.

# Les déclencheurs (trigger)

suite

## Quand utiliser un déclencheur DML

- On peut avoir plusieurs déclencheurs sur un même événement, il suffit de leur donner des noms différents.
- On peut avoir un seul déclencheur pour plusieurs événements.

# Les déclencheurs AFTER et INSTEAD OF

## Comparaison des fonctionnalités des déclencheurs AFTER et INSTEAD OF

Fonction	Déclencheur AFTER	Déclencheur INSTEAD OF
Applicabilité	Tables	Tables et vues
Quantité par table ou vue	Plusieurs par action de déclenchement (UPDATE, DELETE et INSERT)	Un par action de déclenchement (UPDATE, DELETE et INSERT)
Références en cascade	Aucune restriction	Les déclencheurs INSTEAD OF UPDATE et DELETE ne sont pas autorisés sur des tables qui sont des cibles de contraintes d'intégrité référentielle en cascade.
Exécution	Après : <ul style="list-style-type: none"><li>• Traitement des contraintes</li><li>• Actions référentielles déclaratives</li><li>• Création de tables <b>inserted</b> et <b>deleted</b></li><li>• L'action de déclenchement</li></ul>	Avant : <ul style="list-style-type: none"><li>• Traitement des contraintes</li></ul> Au lieu de : <ul style="list-style-type: none"><li>• L'action de déclenchement</li></ul> Après : <ul style="list-style-type: none"><li>• Création de tables <b>inserted</b> et <b>deleted</b></li></ul>
Ordre d'exécution	La première et la dernière exécution peuvent être spécifiées	Non applicable
Références de colonnes <b>varchar(max)</b> , <b>nvarchar(max)</b> et <b>varbinary(max)</b> dans des tables <b>inserted</b> et <b>deleted</b>	Autorisées	Autorisées
Références de colonnes <b>text</b> , <b>ntext</b> et <b>image</b> dans des tables <b>inserted</b> et <b>deleted</b>	Non autorisées	Autorisées

- ❑ Dans le SGBD Oracle, il existe également un déclencheur BEFORE
- ❑ On peut imbriquer des déclencheurs, SQL Server permet un maximum de 32 niveaux.



# Tables des déclencheurs

## Les tables temporaires créées par le déclencheur

- Deux tables temporaires sont créées au moment de la réalisation de l'événement et accessibles uniquement par le déclencheur. Ces tables sont le canal de communication entre l'événement et le déclencheur : les tables ***inserted*** et ***deleted***.
- Ces tables ont la même structure que la table sur laquelle le déclencheur est défini.
- La table ***deleted*** contient une copie des lignes affectées par l'exécution de l'instruction DELETE, c'est à dire que les lignes supprimées sont transférées dans la table ***deleted***. Cette table est vide si c'est l'événement INSERT qui est appelé.
- La table ***inserted*** contient une copie des lignes insérées dans la table. Cette table est vide si c'est l'événement DELETE qui est exécuté.
- L'opération UPDATE n'existe pas en soi ; c'est la conséquence d'un DELETE suivi d'un INSERT. Dans le cas d'une opération UPDATE, les deux tables ***deleted*** et ***inserted*** contiennent respectivement les anciennes valeurs et les nouvelles valeurs d'un enregistrement.

# Instruction de gestion des déclencheurs

## CRÉATION DE DÉCLENCHEURS

**CREATE TRIGGER**  
nomDeclencheur  
on nomTable/nomVue  
AFTER (FOR) / INSTEAD OF  
INSERT / DELETE / UPDATE  
AS  
Instruction(s)

**ACTIVATION**  
**D'UN DÉCLENCHEUR**  
ENABLE TRIGGER nomDecl  
ON nomTable/nomVue

## MODIFICATION DE DÉCLENCHEURS

ALTER TRIGGER nomDecl.  
on nomTable/nomVue  
AFTER (FOR) / INSTEAD OF  
INSERT / DELETE / UPDATE  
AS  
Instruction(s)

**DÉSACTIVATION**  
**D'UN DÉCLENCHEUR**  
DISABLE TRIGGER nomDecl  
ON nomTable/nomVue

## SUPPRESSION DE DÉCLENCHEURS

DROP TRIGGER nomDecl

```
CREATE TRIGGER departementEmployee ON Employee
INSTEAD OF DELETE AS
BEGIN
    DECLARE @Count int;
    SET @Count = @@ROWCOUNT;
    IF @Count = 0
        RETURN;
    SET NOCOUNT ON;
    BEGIN
        IF @@TRANCOUNT > 0
            BEGIN
                ROLLBACK TRANSACTION;
            END
    END;
END;
```

```

CREATE TRIGGER ins_cde_taux ON COMMANDES AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON
    DECLARE @numero_cde int
    DECLARE @numero_cli int
    DECLARE @nombre_commandes int
    SELECT @numero_cde =numero_cde,
           @numero_cli =numero_cli FROM inserted;
    SELECT @nombre_commandes=COUNT(*) FROM COMMANDES
           WHERE numero_cli=@numero_cli;
    IF (@nombre_commandes>10 )
        UPDATE COMMANDES SET taux_remise=5 WHERE
           numero_cde=@numero_cde;
END;
END;

```