



Dotnet France  
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

# Les déclencheurs

*Version 1.0*



Grégory CASANOVA

# Sommaire

---

1	Introduction.....	3
2	Pré-requis .....	4
3	Les déclencheurs du DML.....	5
3.1	Introduction.....	5
3.2	Création d'un déclencheur du DML .....	5
3.3	Suppression d'un déclencheur du DML.....	8
4	Les déclencheurs du DDL.....	10
4.1	Introduction.....	10
4.2	Création d'un déclencheur du DDL .....	10
4.3	Suppression d'un déclencheur du DDL.....	14
5	Conclusion .....	16

## 1 Introduction

Les déclencheurs sont des objets anciens de SQL Server, qui permettent d'effectuer **automatiquement** des tâches administratives de maintien de la base de données. En effet, les déclencheurs, ou triggers, vont nous permettre, dans leur généralité, d'effectuer des opérations automatiques, liées à une opération particulière sur un objet de la base, par l'utilisateur. Plus précisément, lorsque un utilisateur va appliquer une opération du DML par exemple, sur une table, si un déclencheur est défini sur cette table, une action automatique en rapport avec l'action de l'utilisateur s'exécutera. On commence alors de parler de la notion de programmation événementielle, en rapport au fait qu'un événement en déclenche un autre.

## 2 Pré-requis

Pour comprendre au mieux ce chapitre, il est important :

- De connaître les généralités sur les bases de données relationnelles (Chapitre 2).
- De connaître la plupart des instructions DDL de création d'objets dans la base (Chapitre 2, 3, 5 et 6).
- Connaître les généralités du T-SQL DML (Chapitre 4).

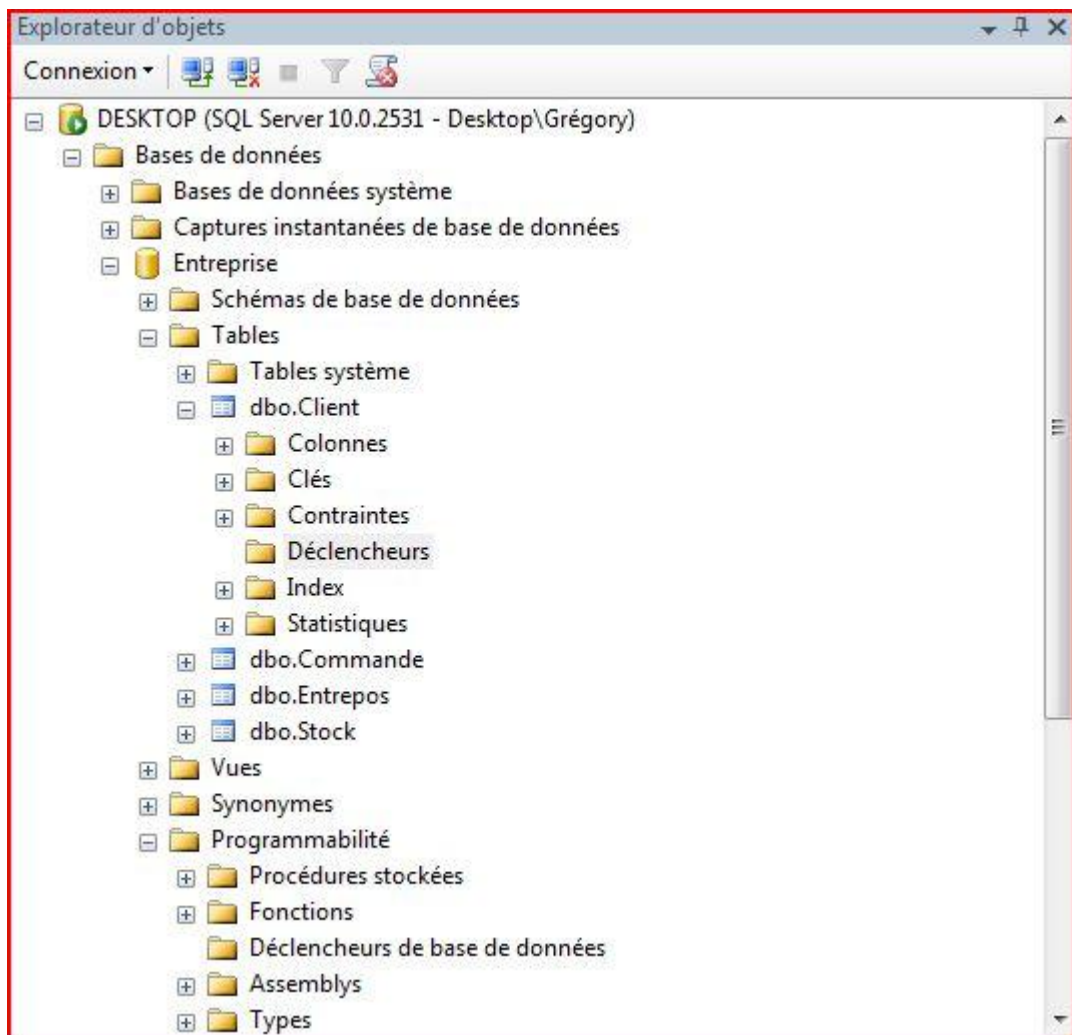
## 3 Les déclencheurs du DML

### 3.1 Introduction

Les déclencheurs sont donc des objets de la base qui nous permettent d'automatiser des actions. Quelle est la particularité des déclencheurs du DML ? Ils vont nous permettre d'effectuer une instruction, à chaque fois qu'une instruction du DML est effectuée pour une table donnée. Prenons un exemple pour mieux comprendre : Nous avons deux tables qui fonctionnent ensemble, table1 et table2. Nous définissons un déclencheur du DML sur table1, qui fait en sorte que, si la table table1 est mise à jour, les valeurs correspondantes dans table2 soit mise à jour de la même manière. C'est de cette manière que nous allons automatiser les actions sur les objets de la base. Pour une instruction donnée sur un objet, une action liée s'exécutera à la suite.

### 3.2 Création d'un déclencheur du DML

La création de ce genre d'objet de la base ne peut se faire que par du code T-SQL, en revanche, il est possible, comme pour tout autre objet de la base d'auto générer le modèle de création d'un déclencheur. Pour ce faire, étendez les nœuds de l'explorateur d'objet jusqu'à la table pour laquelle vous voulez créer le déclencheur, puis étendez la table choisie comme suit :



Une fois cette opération effectuée, faites un clic droit sur le nœud « Déclencheurs » et choisissez l'option « Nouveau déclencheur ... ». Un code auto généré apparaît alors, dans une nouvelle fenêtre de requêtes. Voici ce code :

```

-- =====
-- Template generated from Template Explorer using:
-- Create Trigger (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- See additional Create Trigger templates for more
-- examples of different Trigger statements.
--
-- This block of comments will not be included in
-- the definition of the function.
-- =====
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          <Author,,Name>
-- Create date:    <Create Date,,>
-- Description:    <Description,,>
-- =====
CREATE TRIGGER <Schema_Name, sysname, Schema_Name>.<Trigger_Name,
sysname, Trigger_Name>
    ON <Schema_Name, sysname, Schema_Name>.<Table_Name, sysname,
Table_Name>
    (AFTER | FOR | INSTEAD OF) <Data_Modification_Statements, ,
INSERT,DELETE,UPDATE>
[WITH APPEND][NOT FOR REPLICATION]
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here

```

Expliquons ce code. Tout d'abord, l'instruction `CREATE TRIGGER` est l'instruction DDL qui va nous permettre de créer un déclencheur. Il est nécessaire de préciser le nom du déclencheur à la suite de cette instruction, puisque comme tout objet de la base, un déclencheur peut être mentionné dans du code seulement grâce à son nom unique. La clause `ON` va nous permettre de définir sur quelle table nous allons définir le déclencheur que nous créons et les clauses `AFTER`, `INSTEAD OF` nous permettront par la suite, de donner les conditions d'action du déclencheur.

- `AFTER` : le déclencheur s'exécutera après insertion, mise à jour ou suppression dans la table qui contient le déclencheur.
- `INSTEAD OF` : le déclencheur s'exécutera avant insertion, mise à jour ou suppression dans la table qui contient le déclencheur.

Des options sont disponibles pour les triggers du DML, et elles sont les suivantes :

- `WITH APPEND` : Cette option permet d'ajouter plusieurs déclencheurs sur un même objet et un même ordre SQL. Ceci est du à un comportement différent des déclencheurs au passage de la version compatible 65 et 70.
- `NOT FOR REPLICATION` : Le déclencheur défini avec cette option ne sera pas pris en compte dans un processus de modification des données par réplication.

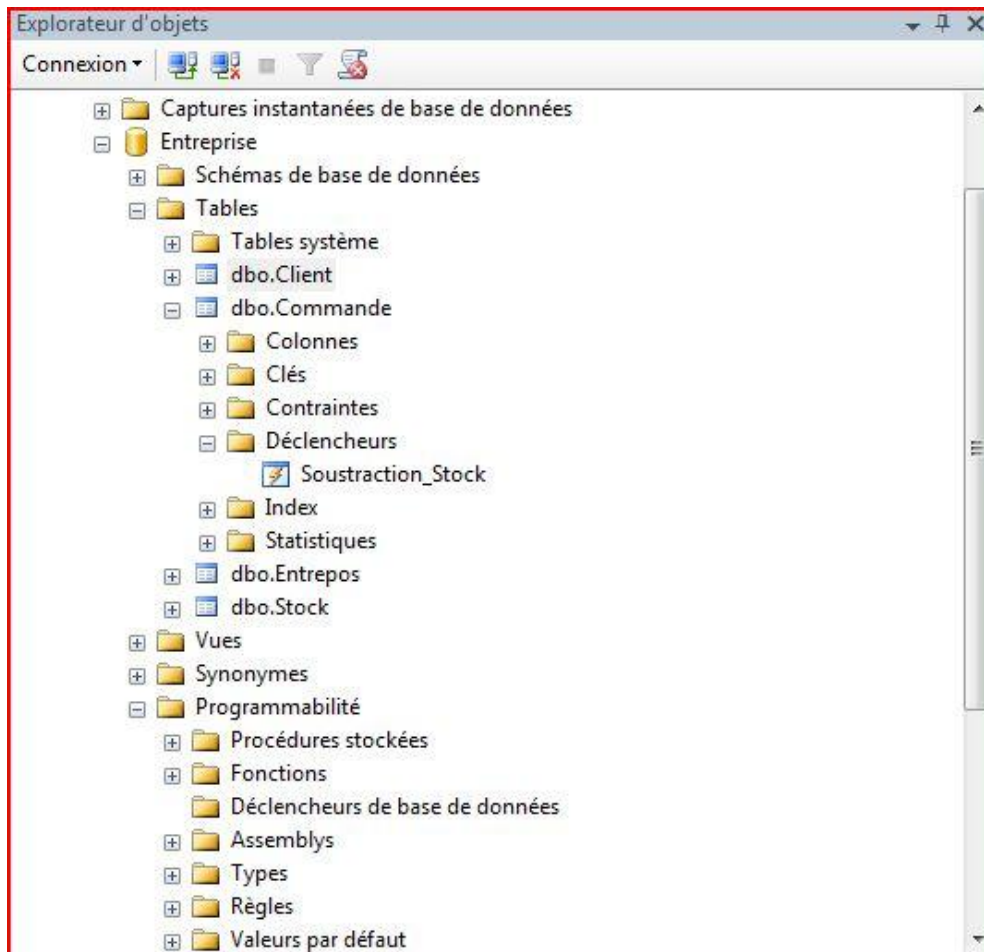
Par exemple, après insertion dans la table précisée, faire ceci. Comme pour une fonction, les délimiteurs des actions dans un déclencheur sont les clauses `AS BEGIN` et `END`. Tout le code présent entre ces deux délimiteurs sera exécuté, à la seule condition que l'action déclenchant le déclencheur soit faite auparavant. Prenons maintenant un exemple concret pour illustrer nos propos. Dans le code du script proposé en annexe de ce cour, il y a un déclencheur du DML nous allons l'utiliser ici comme exemple.

**Remarque :** Les instructions suivantes ne sont pas autorisés dans le corps du déclencheur, à savoir `CREATE`, `DROP`, `ALTER`, `TRUNCATE`, `GRANT`, `REVOKE`, `UPDATE STATISTICS`, `RECONFIGURE` et `LOAD`.

```
CREATE TRIGGER Soustraction_Stock
ON Commande
AFTER INSERT
AS
BEGIN
DECLARE @Id_Stock int, @Quantite int
SELECT @Id_Stock = Id_stock FROM INSERTED
SELECT @Quantite = Quantite FROM INSERTED
UPDATE [Stock]
SET [Quantite] = [Quantite]-@Quantite
WHERE Id_Stock = @Id_Stock
END
```

Au travers de ce code T-SQL, nous créons un déclencheur, dont le nom est `Soustraction_Stock`, sur la table `Commande`. Nous y associons l'événement `AFTER INSERT`, qui précise qu'après insertions de données dans la table `Commande`, le code contenu entre les délimiteurs `AS BEGIN` et `END` doit être exécuté. Le code contenu entre les délimiteurs, dans cet exemple, récupère certaines valeurs insérées dans la table `Commande`, en l'occurrence `Id_Stock` et `Quantité`, dans des variables déclarées en début de lot grâce aux mots clé `FROM INSERTED`, et soustrait la quantité de la commande, à la quantité du stock, en fonction d'un `Id_Stock` donné, grâce à un `UPDATE` de la table `Stock`. Les trois mots clé `FROM UPDATED`, `FROM INSERTED`, `FROM DELETED` nous permet de récupérer les valeurs insérées, mises à jour ou supprimées, avant l'activation du déclencheur.

Il est possible de voir les déclencheurs créés, en déployant les nœuds des tables prévus à cet effet, comme montré ci-dessous, ou encore en utilisant la procédure stockée `sp_helptrigger` :



### 3.3 Suppression d'un déclencheur du DML

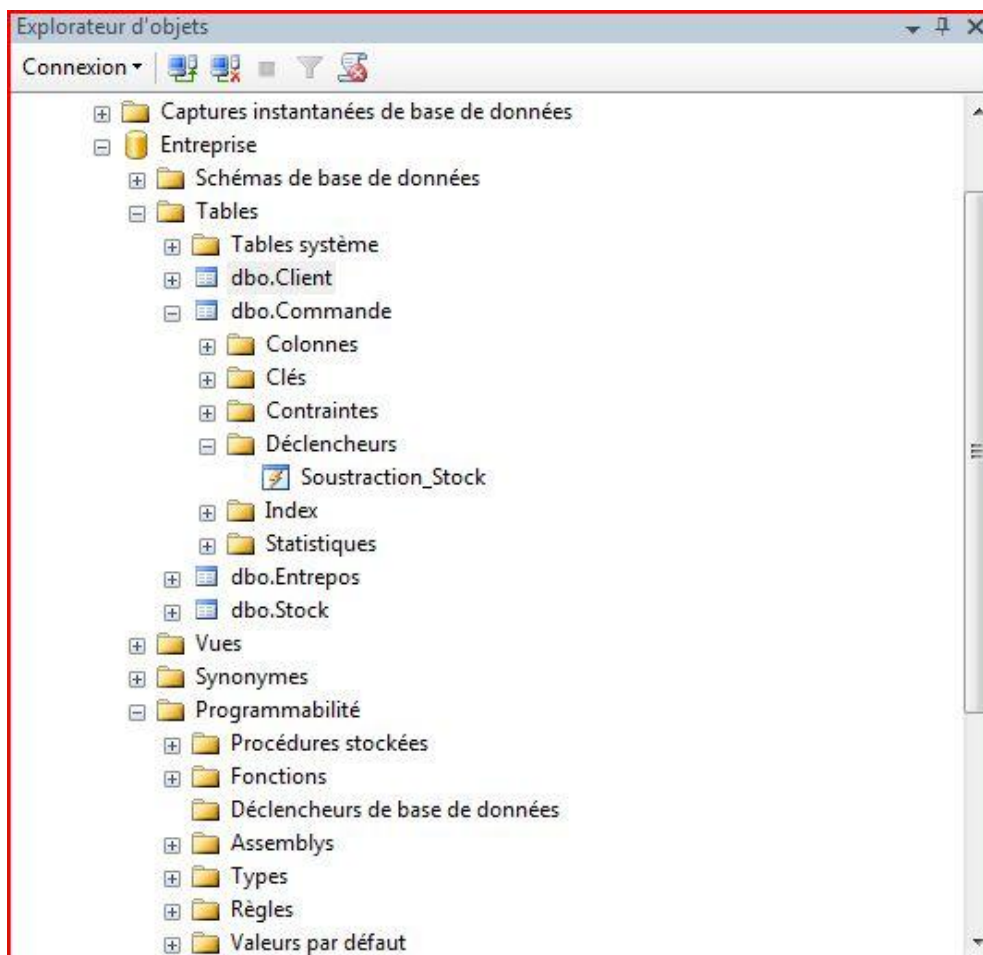
Comme pour tout objet de la base, le déclencheur peut être supprimé de deux manières, par code T-SQL ou encore par l'interface graphique. Voyons les deux manières.

La structure générale de suppression par code T-SQL est la même que pour n'importe quel objet de la base puisque nous allons utiliser DROP. Voyons un exemple :

```
DROP TRIGGER Soustraction_Stock
```

Il suffit simplement d'ajouter le mot clé `TRIGGER` pour spécifier que nous allons supprimer un déclencheur, puis le nom de l'objet à supprimer.

Par l'interface graphique, étendez les nœuds qui mènent à la table qui contient vos déclencheurs, comme ceci :



Effectuez alors un clic droit sur le déclencheur et sélectionnez l'option « supprimer ».

## 4 Les déclencheurs du DDL

### 4.1 Introduction

SQL Server propose des déclencheurs du DDL. Ce type de déclencheur fonctionne de la même manière que les déclencheurs du DML, simplement, les objets affectés par ces déclencheurs et les instructions auxquelles ils réagissent ne sont pas les mêmes. En effet, un trigger du DDL agit au niveau d'une base de données ou du serveur où il est défini, alors qu'un déclencheur du DML agit simplement pour la table ou la vue sur laquelle il est défini. Dans ce dernier cas, le déclencheur est stocké dans la base de données MASTER. De plus, ceux du type DML réagissent à des instructions du DML, alors ceux du DDL réagissent à des instructions du DDL, telles que CREATE, ALTER, DROP, GRANT, REVOKE, DENY et UPDATE STATISTICS. Comme pour ceux du DML, les déclencheurs du DDL sont exécutés après l'instruction DDL qui leur est attachée. Le but principal de ce type d'objet est de gérer l'ajout et la suppression ou encore la mise à jour d'objets dans la table, ou bien d'automatiser celle-ci. Enfin, et c'est la dernière différence entre les deux différents types de déclencheurs que sont les DML et les DDL, il n'est pas possible, contrairement à un DML, de créer un déclencheur INSTEAD OF de type DDL.

### 4.2 Création d'un déclencheur du DDL

Présentons maintenant la structure générale de création d'un déclencheur de type DDL :

```
USE Entreprise
GO

CREATE TRIGGER nom_trigger
ON DATABASE
FOR CREATE_TABLE
AS
BEGIN
    ROLLBACK TRAN
END
```

L'instruction `CREATE TRIGGER` nous permet d'annoncer que nous allons créer un déclencheur. Il est nécessaire de préciser son nom à la suite de cette instruction. Par la suite, on définit si ce déclencheur est défini pour une base de données en particulier ou pour tout le serveur grâce aux mots clé `ALL SERVER` et `DATABASE`. Nous définirons ensuite les options `WITH ENCRYPTION` et `EXECUTE AS`. La clause `AFTER` permet, comme pour un déclencheur du DML, de préciser après quelle instruction le déclencheur doit être activé. Ces instructions seront présentées ci-après, dans un tableau résumé. Les clauses `AS BEGIN` et `END` définissent le corps du déclencheur, c'est-à-dire les actions à effectuer dans le cas où celui-ci est activé.

Voici les instructions disponibles pour un déclencheur du DDL (Ces instructions ont deux cibles distinctes, les objets du serveur ou les objets de la base. Nous détaillerons les deux.):

#### Les instructions sur une base de données :

- **Tables :**
  - DDL\_TABLE\_EVENTS
  - CREATE TABLE
  - ALTER\_TABLE
  - DROP\_TABLE
- **Vues :**

CREATE\_VIEW  
ALTER\_VIEW  
DROP\_VIEW

- **Synonymes :**

CREATE\_SYNONYM  
DROP\_SYNONYM

- **Fonctions :**

CREATE\_FUNCTION  
ALTER\_FUNCTION  
DROP\_FUNCTION

- **Procédures :**

CREATE\_PROCEDURE  
ALTER\_PROCEDURE  
DROP\_PROCEDURE

- **Déclencheurs :**

CREATE\_TRIGGER  
ALTER\_TRIGGER  
DROP\_TRIGGER

- **Notifications :**

CREATE\_EVENT\_NOTIFICATION  
DROP\_EVENT\_NOTIFICATION

- **Index :**

CREATE\_INDEX  
ALTER\_INDEX  
DROP\_INDEX

- **Statistiques :**

CREATE\_STATISTICS  
UPDATE\_STATISTICS  
DROP\_STATISTICS

- **Assemblies :**

CREATE\_ASSEMBLY  
ALTER\_ASSEMBLY  
DROP\_ASSEMBLY

- **Types :**

CREATE\_TYPE  
DROP\_TYPE

- **Users :**

CREATE\_USER  
ALTER\_USER  
DROP\_USER

CREATE\_ROLE  
ALTER\_ROLE  
DROP\_ROLE

- **Rôles :**

CREATE\_APPLICATION\_ROLE  
ALTER\_APPLICATION\_ROLE  
DROP\_APPLICATION\_ROLE

- **Schémas :**

CREATE\_SCHEMA

ALTER\_SCHEMA  
DROP\_SCHEMA

- **Messages Types :**

CREATE\_MESSAGE\_TYPE  
ALTER\_MESSAGE\_TYPE  
DROP\_MESSAGE\_TYPE

- **Contrats :**

CREATE\_CONTRACT  
ALTER\_CONTRACT  
DROP\_CONTRACT

- **Queues :**

CREATE\_QUEUE  
ALTER\_QUEUE  
DROP\_QUEUE

- **Services :**

CREATE\_SERVICE  
ALTER\_SERVICE  
DROP\_SERVICE

- **Route :**

CREATE\_ROUTE  
ALTER\_ROUTE  
DROP\_ROUTE

- **Service Binding :**

CREATE\_REMOTE\_SERVICE\_BINDING  
ALTER\_REMOTE\_SERVICE\_BINDING  
DROP\_REMOTE\_SERVICE\_BINDING

- **Droits sur la base de données :**

GRANT\_DATABASE  
DENY\_DATABASE  
REVOKE\_DATABASE

- **Secexpr :**

CREATE\_SECEXP  
DROP\_SECEXP

- **XML :**

CREATE\_XML\_SCHEMA  
ALTER\_XML\_SCHEMA  
DROP\_XML\_SCHEMA

- **Fonction de partition :**

CREATE\_PARTITION\_FUNCTION  
ALTER\_PARTITION\_FUNCTION  
DROP\_PARTITION\_FUNCTION

- **Schémas de partition :**

CREATE\_PARTITION\_SCHEME  
ALTER\_PARTITION\_SCHEME  
DROP\_PARTITION\_SCHEME

**Les instructions sur le serveur :**

- **Logins :**

CREATE\_LOGIN  
ALTER\_LOGIN  
DROP\_LOGIN

- **Endpoints :**  
CREATE\_HTTP\_ENDPOINT  
DROP\_HTTP\_ENDPOINT
- **Droits d'accès au serveur :**  
GRANT\_SERVER\_ACCESS  
DENY\_SERVER\_ACCESS  
REVOKE\_SERVER\_ACCESS
- **Certificats :**  
CREATE\_CERT  
ALTER\_CERT  
DROP\_CERT

Certaines des actions DDL contenues dans la liste ci-dessus ne nous sont pas encore connues, tout simplement parce qu'elles interviendront dans le cadre des cours sur l'administration de SQL Server, ou bien parce que nous n'avons pas encore vu le cours correspondant dans le programme de la partie développement de base de données de SQL Server. La liste ci-dessus n'est pas exhaustive.

Voyons maintenant ce que signifient les options disponibles :

- **WITH ENCRYPTION** : Permet de crypter le code du déclencheur dans la base de données MASTER.

Prenons un exemple concret de création d'un déclencheur du DDL.

```
USE Entreprise
GO

CREATE TABLE log_database
(date_log datetime2, nom_user nvarchar(50), Instruction nvarchar(100))
GO

CREATE TRIGGER trigger_log
ON DATABASE
AFTER DDL_TABLE_EVENTS
AS
BEGIN
    INSERT INTO log_database
    (date_log, nom_user, Instruction)
    VALUES
    (GETDATE(),
    CURRENT_USER,
    EVENTDATA().value('/EVENT_INSTANCE/EventType)
    [1]', 'nvarchar(50)'))
END
```

Dans ce cas là, pour chaque instruction du DDL faite pas un utilisateur quelconque, nous allons insérer dans la table `log_database` la date, l'utilisateur et le type d'évènements qu'il a utilisé, pour constituer au final, une sorte de registre des actions DDL faites sur les tables. La dernière ligne de l'instruction `INSERT (EVENTDATA().value('/EVENT_INSTANCE/EventType) [1]', 'nvarchar(50)'))` n'est pas à connaître pour le moment, puisque il s'agit d'un travail sur un fichier XML, ce qui fera l'objet du chapitre suivant. Retenez simplement qu'il s'agit d'une méthode d'extraction d'information d'un fichier XML.

Voilà ce qu'il se passe lorsque nous créons une table, et que nous exécutons le code suivant :

```
SELECT * FROM dbo.log_database
```

	date_log	nom_user	Instruction
1	2009-07-16 09:47:54.8230000	dbo	CREATE_TABLE

Un autre type de déclencheur existe, même si on ne peut pas parler directement de type de déclencheur puisqu'il fait partie des déclencheurs du DDL. C'est le type des triggers de connexion. Ils se présentent de la même manière qu'un déclencheur DDL simple, seule une différence est à noter ; la présence du mot clé FOR LOGON, comme dans la structure suivante :

```
CREATE TRIGGER trigger_log
ON ALL SERVER
FOR LOGON
AS
BEGIN
    --instructions
END
```

Il faut faire très attention avec les déclencheurs de connexions, car dans certains cas, ils peuvent vous empêcher une connexion au serveur.

### 4.3 Suppression d'un déclencheur du DDL

Comme toujours, il est possible de supprimer un objet de la base de données de deux manières, nous allons montrer les deux.

- **Par code T-SQL :**

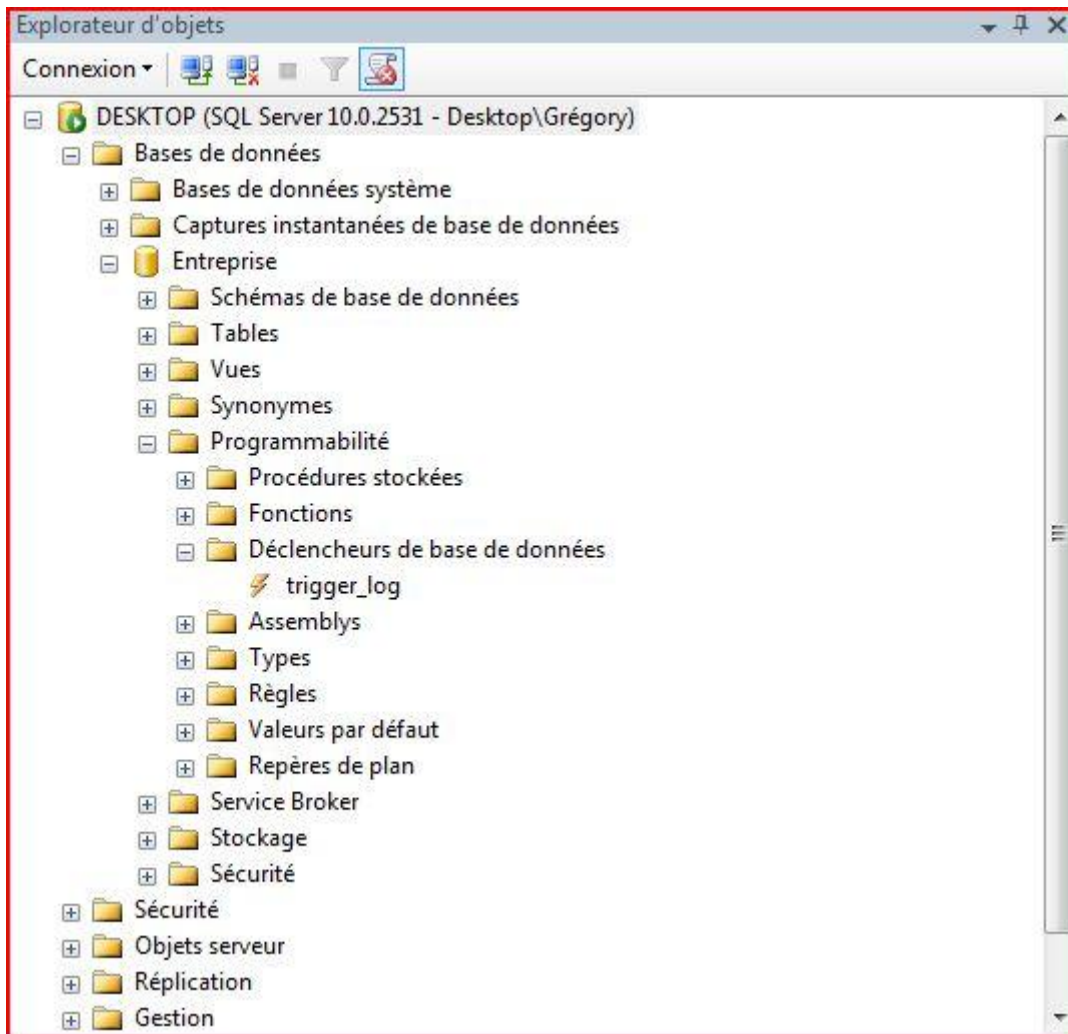
La syntaxe de suppression d'un déclencheur est simple. Elle est la suivante :

```
DROP TRIGGER trigger_log ON DATABASE
```

On utilise donc l'instruction suivie du nom unique du déclencheur. Il est par la suite nécessaire de préciser si le déclencheur DDL est défini sur une base de données ou sur le serveur entier.

- **Par l'interface graphique :**

Il vous suffit dans un premier temps de déployer les nœuds de l'explorateur d'objet de SSMS, jusqu'à votre déclencheur de cette manière :



Il vous suffit alors simplement de faire un clic droit sur le déclencheur à supprimer, par exemple `trigger_log`, et de choisir l'option « Supprimer ».

## 5 Conclusion

Dans ce chapitre, nous avons appris une notion très importante : les déclencheurs. Nous avons donc vu qu'il en existe deux types principaux, ceux du DML et ceux du DDL, qui comprennent aussi les déclencheurs de connexion. Ils nous permettent d'utiliser le principe de programmation événementielle, puisque nous utilisons un évènement pour activer notre déclencheur. Dans le chapitre suivant, nous allons apprendre à travailler avec des données de type XML, les extraire aussi bien d'un fichier plat que d'une colonne d'une table dans SQL Server.