REQUÊTES SQL
LES TRIGGERS

Avant la lecture de ce document, je vous invite à télécharger les fichiers en relation avec ce document :

- Requêtes pour créer la structure de la base de données
- Requêtes pour insérer des données dans l'ensemble des tables de la base de données

Les requêtes sont à utiliser dans une base de données nommée "etudiantFormation" dans SQL SERVER

En espérant que ce document vous aidera !

Bon courage à toutes et à tous !

TABLE DES MATIÈRES

Qu'est ce qu'un trigger (déclencheur) ?

Pour gérer des contraintes complexes au niveau du SGBD, il est souvent nécessaire de recourir aux déclencheurs (ou triggers). Un trigger (déclencheur) est un programme qui se lance automatiquement lorsqu'un événement se produit. Autrement dit, c'est une procédure programmée au niveau du SGBD qui s'exécute automatiquement lorsqu'un événement précis survient. Par événement, on entend l'ajout (**INSERT**), la suppression (**DELETE**) et/ou la mise à jour (**UPDATE**) d'un enregistrement dans la table ou dans une colonne particulière. Les événements déclencheurs sont les suivants :

Tableau récapitulatif des événements déclencheurs		**Déclencheurs**			
		BEFORE	**INSTEAD OF**	**FOR**	**AFTER**
Instructions	**INSERT**	exécution du code **AVANT** une insertion de donnée(s)	exécution du code **AU LIEU D'**une insertion de donnée(s)	exécution du code **POUR** une insertion de donnée(s)	exécution du code **APRES** une insertion de donnée(s)
	UPDATE	exécution du code **AVANT** une modification de donnée(s)	exécution du code **AU LIEU D'**une modification de donnée(s)	exécution du code **POUR** une modification de donnée(s)	exécution du code **APRÈS** une modification de donnée(s)
	DELETE	exécution du code **AVANT** une suppression de donnée(s)	exécution du code **AU LIEU D'**une suppression de donnée(s)	exécution du code **POUR** une suppression de donnée(s)	exécution du code **APRES** une suppression de donnée(s)

/ ! \ **POINTS IMPORTANTS** / ! \ :

Sous **SQL SERVER**, les triggers reposant sur l'utilisation du déclencheur **BEFORE** **ne sont pas disponibles**.

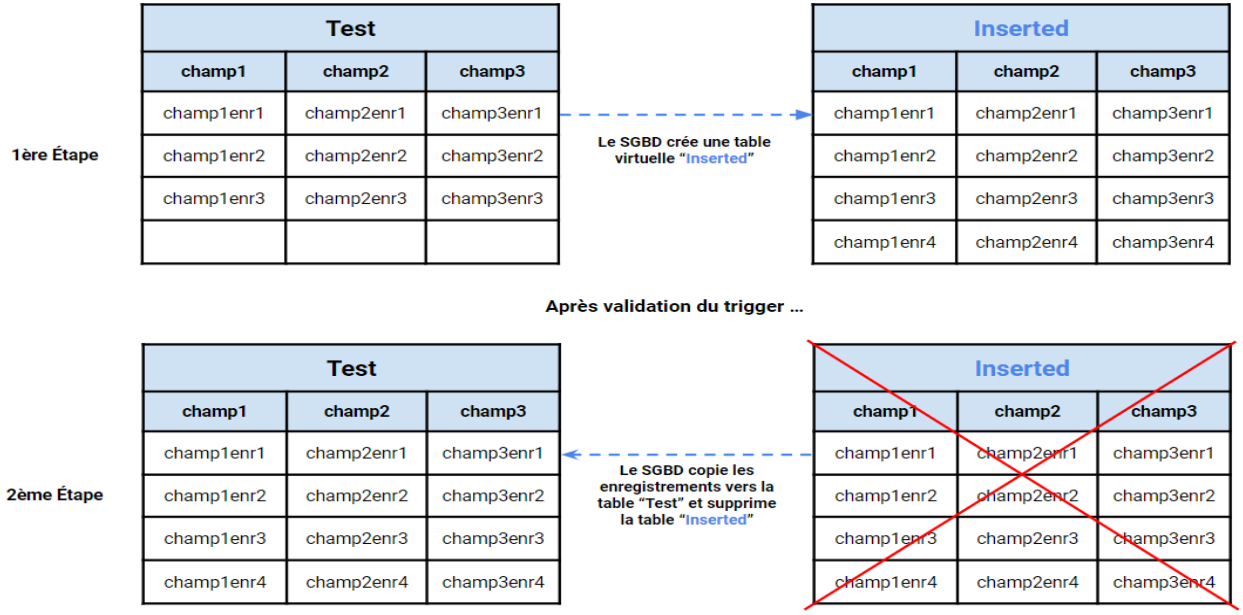
Utilisation d'un trigger avec l'instruction INSERT

Explications

Pour cet exemple, on va prendre une table fictive nommée "Test" avec des champs et des enregistrements. On veut exécuter un trigger avec l'instruction **INSERT**. Le SGBD va créer une table virtuelle nommée "**INSERTED**". Cette table n'est accessible qu'au niveau des triggers et son contenu est perdu dès que les triggers sont validés.

Lors de l'insertion, les enregistrements ajoutés sont placés dans cette table temporaire "**INSERTED**". Après la validation du trigger, on copie les enregistrements vers notre table "Test" et supprime la table "**INSERTED**". Avec cette méthode, on peut limiter ou empêcher l'ajout ou encore effectuer des vérifications sur les valeurs avant l'ajout.

Illustration



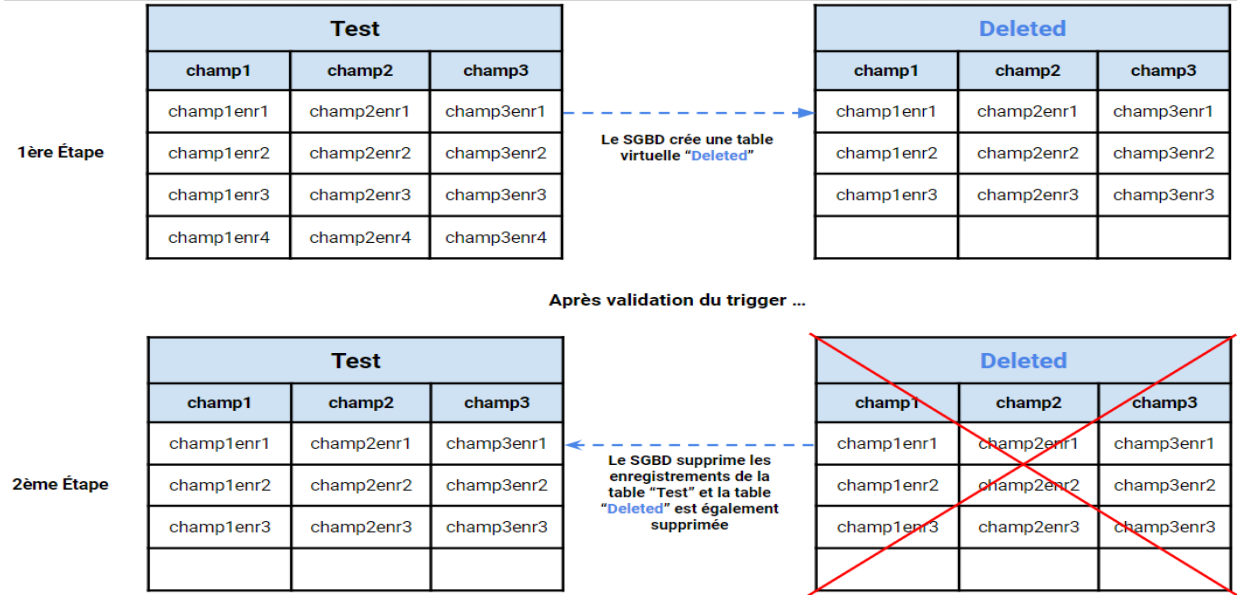
Utilisation d'un trigger avec l'instruction DELETE

Explications

Pour cet exemple, on utilise à nouveau la table fictive "Test". On veut exécuter un trigger avec l'instruction **DELETE**. Le SGBD va créer une table virtuelle nommée cette fois "**DELETED**". Cette table n'est accessible qu'au niveau des triggers et son contenu est perdu dès que les triggers sont validés.

Lors de la suppression, le ou les enregistrements supprimés sont placés dans cette table temporaire "**DELETED**". Après la validation du trigger, on supprime définitivement le ou les enregistrements voulus de notre table "Test" et la table "**DELETED**" est supprimée. Avec cette méthode, on peut limiter ou empêcher la suppression ou encore effectuer des contrôles sur les valeurs avant l'opération de suppression.

Illustration



Utilisation d'un trigger avec l'instruction UPDATE

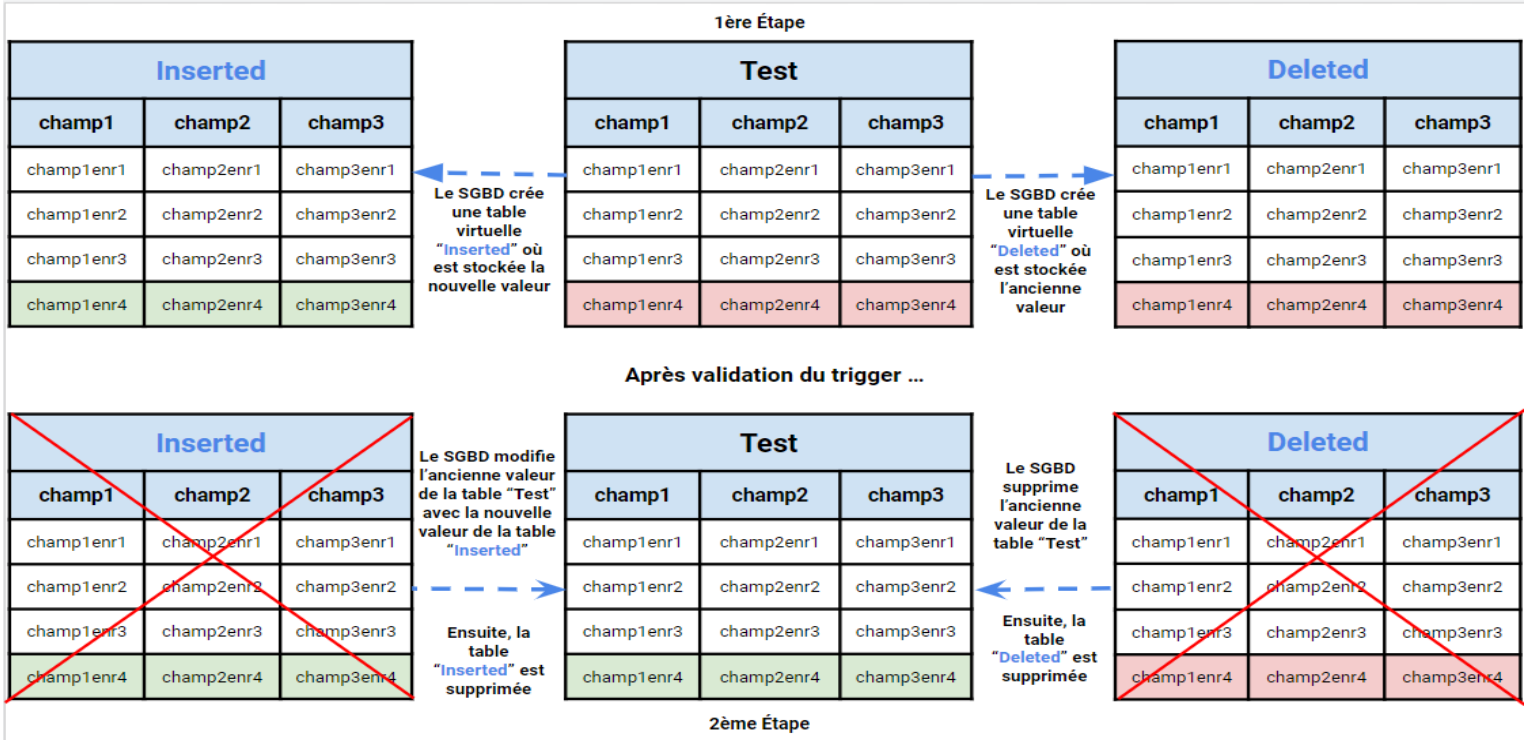
Explications

Toujours à l'aide de la table fictive "Test", on veut exécuter un trigger avec l'instruction **UPDATE**. Le SGBD va créer cette fois deux tables virtuelles "**DELETED**" et "**INSERTED**". Ces tables ne sont accessibles qu'au niveau des triggers et leur contenu est perdu dès que les triggers sont validés.

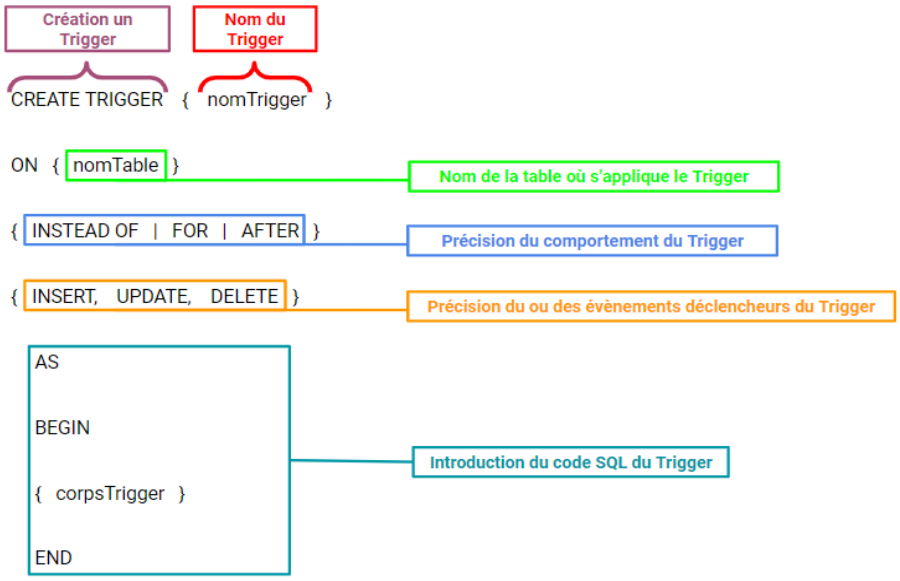
Pourquoi se retrouve-t-on avec deux tables ? Avec l'utilisation de l'instruction **UPDATE**, on se retrouve avec deux valeurs : une ancienne valeur qui va être stockée dans la table "**DELETED**" et une nouvelle valeur qui va être stockée dans la table "**INSERTED**".

Après la validation du trigger, la donnée va être mise à jour au niveau de la table "Test" et les deux tables "**DELETED**" et "**INSERTED**" seront supprimées. Avec cette méthode, on peut facilement réaliser une comparaison entre l'ancienne valeur et la nouvelle valeur.

Illustration



Syntaxe générale de la création d'un trigger



/ ! \ **POINTS IMPORTANTS** / ! \ :

Le corps du trigger créé peut inclure n'importe quelles instructions excepté **ALTER DATABASE**, **CREATE DATABASE**, **DROP DATABASE**, **RESTORE DATABASE**, **RESTORE LOG**, **RECONFIGURE**.

Comportements des triggers

Comportements	Définitions	Représentations	Utilisations
INSTEAD OF	Un comportement **INSTEAD OF** indique que le trigger est déclenché **A LA PLACE** de l'action déclenchante	Au lieu de faire EVENEMENT Faire INSTRUCTION Au lieu de faire INSERT, UPDATE, DELETE Faire INSTRUCTION	• Exécution d'autre opération avant l'opération de mise à jour (**INSERT, UPDATE, DELETE**) • Empêcher l'exécution de l'opération de mise à jour (**INSERT, UPDATE, DELETE**) • Effectuer des vérifications avant l'exécution de l'opération de mise à jour (**INSERT, UPDATE, DELETE**)
FOR	Un comportement **FOR** indique que le corps du trigger est exécuté **EN MÊME TEMPS** que l'action déclenchante	Avec EVENEMENT Faire INSTRUCTION Avec INSERT, UPDATE, DELETE Faire INSTRUCTION	• Effectuer une mise à jour instantanée dans un autre champs calculé soit dans la même table soit dans une autre table
AFTER	Un comportement **AFTER** indique que le trigger est déclenché **APRÈS** l'action déclenchante	Après EVENEMENT Faire INSTRUCTION Après INSERT, UPDATE, DELETE Faire INSTRUCTION	• Effectuer une mise à jour ultérieure dans un autre champs calculé soit de la même table soit dans une autre table

/ ! \ **POINTS IMPORTANTS** / ! \ :

Les comportements **AFTER** sont exécutés après la validation des contraintes associées à la table. **Si une contrainte n'est pas vérifiée, ce comportement ne se déclenche pas**.

Exemples de triggers : le cas “etudiantFormation”

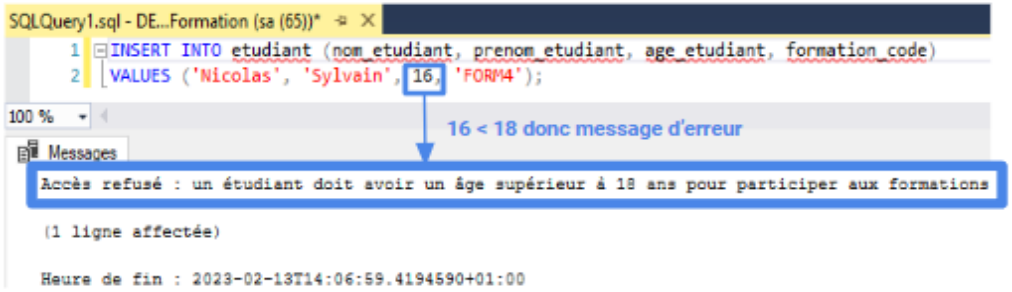
Cas n°1 :
Vérifier l’âge d’un étudiant lors de l’insertion de ses données dans la table “Etudiant”. Si l’âge de l’étudiant est inférieur à 18 ans, il ne peut pas participer à l’une des formations proposées.

Création du Trigger

```
CREATE TRIGGER TR_Verif_Age_Etudiant
ON etudiant
INSTEAD OF INSERT
AS BEGIN
        IF (SELECT age_etudiant FROM inserted) < 18
        BEGIN
                print 'Accès refusé : un étudiant doit avoir un âge supérieur à 18 ans pour participer aux formations';
        END
        ELSE
        BEGIN
                INSERT INTO etudiant SELECT nom_etudiant, prenom_etudiant, age_etudiant, formation_code FROM inserted
        END
END
```

Exemple n°1

```
INSERT INTO etudiant (nom_etudiant, prenom_etudiant, age_etudiant, formation_code)
VALUES ('Nicolas', 'Sylvain', 16, 'FORM4');
```



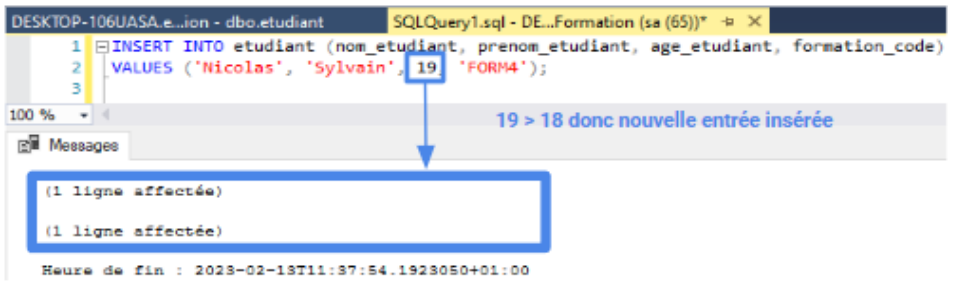

```
SELECT * FROM etudiant;
```

Résultats | Messages

	matricule_etudiant	nom_etudiant	prenom_etudiant	age_etudiant	formation_code
1	1	Badi	Lila	20	FORM1
2	2	Douis	Jonathan	21	FORM2
3	3	Rougé	Mickaël	20	FORM1
4	4	Campan	Laurent	22	FORM3
5	5	Mendes	Cindy	25	FORM2
6	6	Luciano	Romain	25	FORM2
7	7	Raoux	Nicolas	20	FORM1
8	8	Faggioli	Yann	19	FORM1
9	9	Bouisson	Mallaury	20	FORM3
10	10	Ben Kalaia	Mohamed	24	FORM3
11	11	Leempoels	Sébastien	28	FORM1
12	12	Hamadou	Qaïs	27	FORM2
13	13	Lapraz	Isabelle	20	FORM4
14	14	Luciano	Florian	23	FORM5

Exemple n°2

```
INSERT INTO etudiant (nom_etudiant, prenom_etudiant, age_etudiant, formation_code)
VALUES ('Nicolas', 'Sylvain', 19, 'FORM4');
```




```
SELECT * FROM etudiant;
```

Résultats | Messages

	matricule_etudiant	nom_etudiant	prenom_etudiant	age_etudiant	formation_code
1	1	Badi	Lila	20	FORM1
2	2	Douis	Jonathan	21	FORM2
3	3	Rougé	Mickaël	20	FORM1
4	4	Campan	Laurent	22	FORM3
5	5	Mendes	Cindy	25	FORM2
6	6	Luciano	Romain	25	FORM2
7	7	Raoux	Nicolas	20	FORM1
8	8	Faggioli	Yann	19	FORM1
9	9	Bouisson	Mallaury	20	FORM3
10	10	Ben Kalaia	Mohamed	24	FORM3
11	11	Leempoels	Sébastien	28	FORM1
12	12	Hamadou	Qaïs	27	FORM2
13	13	Lapraz	Isabelle	20	FORM4
14	14	Luciano	Florian	23	FORM5
15	15	Nicolas	Sylvain	19	FORM4

Cas n°2 :
Modifier le nombre de places d'une formation lors de l'insertion des données d'un nouvel étudiant dans la table "Etudiant".

Création du Trigger

```
CREATE TRIGGER TR_Modif_Nombre_Place
ON etudiant
AFTER INSERT
AS BEGIN
UPDATE formation SET nombre_restant_places_formation = nombre_restant_places_formation - 1
WHERE code_formation = (SELECT formation_code FROM inserted)
END
```

Avant Insertion

Table "Etudiant"

	matricule_etudiant	nom_etudiant	prenom_etudiant	age_etudiant	formation_code
1	1	Badi	Lila	20	FORM1
2	2	Douis	Jonathan	21	FORM2
3	3	Rougé	Mickaël	20	FORM1
4	4	Campan	Laurent	22	FORM3
5	5	Mendes	Cindy	25	FORM2
6	6	Luciano	Romain	25	FORM2
7	7	Raoux	Nicolas	20	FORM1
8	8	Faggioli	Yann	19	FORM1
9	9	Bouisson	Mallaury	20	FORM3
10	10	Ben Kalaia	Mohamed	24	FORM3
11	11	Leempoels	Sébastien	28	FORM1
12	12	Hamadou	Qaïs	27	FORM2
13	13	Lapraz	Isabelle	20	FORM4
14	14	Luciano	Florian	23	FORM5

Table "Formation"

	code_formation	nom_formation	nombre_total_places_formation	nombre_restant_places_formation	date_debut_formation	date_fin_formation
1	FORM1	Formation Un	10	5	2023-06-10	2023-12-10
2	FORM2	Formation Deux	10	6	2023-04-05	2023-10-06
3	FORM3	Formation Trois	12	9	2023-02-09	2023-06-10
4	FORM4	Formation Quatre	10	8	2023-03-06	2023-09-10
5	FORM5	Formation Cinq	5	4	2023-07-20	2023-12-25

Après Insertion et Modification

```
INSERT INTO etudiant (nom_etudiant, prenom_etudiant, age_etudiant, formation_code)
VALUES ('Kennedy', 'Leon', 25, 'FORM1');
```

Table "Etudiant"

	matricule_etudiant	nom_etudiant	prenom_etudiant	age_etudiant	formation_code
1	1	Badi	Lila	20	FORM1
2	2	Douis	Jonathan	21	FORM2
3	3	Rougé	Mickaël	20	FORM1
4	4	Campan	Laurent	22	FORM3
5	5	Mendes	Cindy	25	FORM2
6	6	Luciano	Romain	25	FORM2
7	7	Raoux	Nicolas	20	FORM1
8	8	Faggioli	Yann	19	FORM1
9	9	Bouisson	Mallaury	20	FORM3
10	10	Ben Kalaia	Mohamed	24	FORM3
11	11	Leempoels	Sébastien	28	FORM1
12	12	Hamadou	Qaïs	27	FORM2
13	13	Lapraz	Isabelle	20	FORM4
14	14	Luciano	Florian	23	FORM5
15	15	Kennedy	Leon	25	FORM1

Table "Formation"

	code_formation	nom_formation	nombre_total_places_formation	nombre_restant_places_formation	date_debut_formation	date_fin_formation
1	FORM1	Formation Un	10	4	2023-06-10	2023-12-10
2	FORM2	Formation Deux	10	6	2023-04-05	2023-10-06
3	FORM3	Formation Trois	12	9	2023-02-09	2023-06-10
4	FORM4	Formation Quatre	10	8	2023-03-06	2023-09-10
5	FORM5	Formation Cinq	5	4	2023-07-20	2023-12-25

Cas n°3 :
Supprimer une formation dans la table “Formation” engendrant la suppression des étudiants suivant cette formation dans la table “Etudiant”.

Création du Trigger

```
CREATE TRIGGER TR_Suppr_Formation
ON formation
INSTEAD OF DELETE
AS BEGIN
        DELETE FROM etudiant WHERE formation_code = (SELECT code_formation FROM deleted);
        DELETE FROM formation WHERE code_formation = (SELECT code_formation FROM deleted);
END
```

Avant Suppression

Table “Formation”

	code_formation	nom_formation	nombre_total_places_formation	nombre_restant_places_formation	date_debut_formation	date_fin_formation
1	FORM1	Formation Un	10	4	2023-06-10	2023-12-10
2	FORM2	Formation Deux	10	6	2023-04-05	2023-10-06
3	FORM3	Formation Trois	12	9	2023-02-09	2023-06-10
4	FORM4	Formation Quatre	10	8	2023-03-06	2023-09-10
5	FORM5	Formation Cinq	5	4	2023-07-20	2023-12-25

Table “Etudiant”

	matricule_etudiant	nom_etudiant	prenom_etudiant	age_etudiant	formation_code
1	1	Badi	Lila	20	FORM1
2	2	Douis	Jonathan	21	FORM2
3	3	Rougé	Mickaël	20	FORM1
4	4	Campan	Laurent	22	FORM3
5	5	Mendes	Cindy	25	FORM2
6	6	Luciano	Romain	25	FORM2
7	7	Raoux	Nicolas	20	FORM1
8	8	Faggioli	Yann	19	FORM1
9	9	Bouisson	Mallaury	20	FORM3
10	10	Ben Kalaia	Mohamed	24	FORM3
11	11	Leempoels	Sébastien	28	FORM1
12	12	Hamadou	Qaïs	27	FORM2
13	13	Lapraz	Isabelle	20	FORM4
14	14	Luciano	Florian	23	FORM5
15	15	Kennedy	Leon	25	FORM1

Après Suppression

```
DELETE FROM formation WHERE code_formation = 'FORM2';
```

Table “Formation”

	code_formation	nom_formation	nombre_total_places_formation	nombre_restant_places_formation	date_debut_formation	date_fin_formation
1	FORM1	Formation Un	10	4	2023-06-10	2023-12-10
2	FORM3	Formation Trois	12	9	2023-02-09	2023-06-10
3	FORM4	Formation Quatre	10	8	2023-03-06	2023-09-10
4	FORM5	Formation Cinq	5	4	2023-07-20	2023-12-25

La formation ‘FORM2’ est supprimée …

Table “Etudiant”

	matricule_etudiant	nom_etudiant	prenom_etudiant	age_etudiant	formation_code
1	1	Badi	Lila	20	FORM1
2	3	Rougé	Mickaël	20	FORM1
3	4	Campan	Laurent	22	FORM3
4	7	Raoux	Nicolas	20	FORM1
5	8	Faggioli	Yann	19	FORM1
6	9	Bouisson	Mallaury	20	FORM3
7	10	Ben Kalaia	Mohamed	24	FORM3
8	11	Leempoels	Sébastien	28	FORM1
9	13	Lapraz	Isabelle	20	FORM4
10	14	Luciano	Florian	23	FORM5
11	15	Kennedy	Leon	25	FORM1

… et les informations des étudiants en rapport avec la formation ‘FORM2’ sont également supprimées.