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) ?
Utilisation d'un trigger avec l'instruction INSERT
Utilisation d'un trigger avec l'instruction DELETE
Utilisation d'un trigger avec l'instruction UPDATE
Syntaxe générale de la création d'un trigger
Comportements des triggers
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.
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".
Cas n°3 : Supprimer une formation dans la table "Formation" engendrant la suppression des étudiants suivant cette formation dans la table "Etudiant".

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 Déclencheurs						
		BEFORE	INSTEAD OF	FOR	AFTER			
	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)			
Instructions	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 /!\:

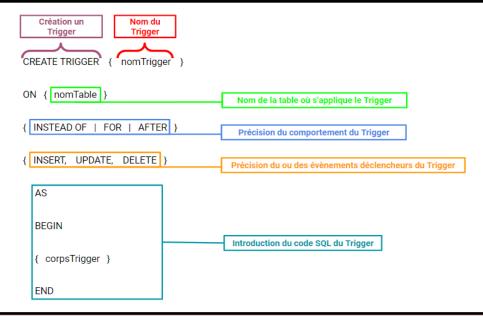
Sous **SQL SERVER**, les triggers reposant sur l'utilisation du déclencheur **BEFORE** ne sont pas disponibles.

Utilisation d'un trigger avec l'instruction INSERT **Explications** Illustration Test Inserted Pour cet exemple, on va prendre une table fictive nommée "Test" avec des champs et champ1 champ2 champ3 champ1 champ2 champ3 des enregistrements. On veut exécuter un champ1enr1 champ2enr1 champ3enr1 champ1enr1 champ2enr1 champ3enr1 trigger avec l'instruction INSERT. Le SGBD Le SGBD crée une table va créer une table virtuelle nommée 1ère Étape champ1enr2 champ2enr2 champ3enr2 champ1enr2 champ2enr2 champ3enr2 virtuelle "Inserted" "INSERTED". Cette table n'est accessible champ1enr3 champ2enr3 champ3enr3 champ1enr3 champ2enr3 champ3enr3 qu'au niveau des triggers et son contenu est perdu dès que les triggers sont validés. champ3enr4 champ1enr4 champ2enr4 Lors de l'insertion, les enregistrements Après validation du trigger ... ajoutés sont placés dans cette table temporaire "INSERTED". Après la validation Test Inserted du trigger, on copie les enregistrements champ1 champ2 champ3 champ' champ2 champ3 vers notre table "Test" et supprime la table "INSERTED". Avec cette méthode, on peut champ1enr1 champ2enr1 champ3enr1 champ1enr1 champ2enr1 champ3enr1 limiter ou empêcher l'ajout ou encore Le SGBD conie les enregistrements vers la 2ème Étape champ1enr2 champ2enr2 phamp2enr2 champ3enr2 effectuer des vérifications sur les valeurs champ3enr2 champ1enr2 table "Test" et supprime la table "Inserted" avant l'ajout. champ1enr3 champ2enr3 champ3enr3 champ1em champ2enr3 champ3enr3 champ1enr4 champ2enr4 champ3en champ1enr4 champ2enr4 champ3enr4 **Utilisation d'un trigger avec l'instruction DELETE Explications** Illustration Pour cet exemple, on utilise à nouveau la **Deleted** Test table fictive "Test". On veut exécuter un champ1 champ2 champ3 champ1 champ2 champ3 trigger avec l'instruction DELETE. Le SGBD va créer une table virtuelle nommée cette champ1enr1 champ3enr1 champ2enr1 champ3enr1 champ1enr1 champ2enr1 fois "DELETED". Cette table n'est accessible Le SGBD crée une table champ3enr2 1ère Étape champ1enr2 champ2enr2 champ1enr2 champ2enr2 champ3enr2 qu'au niveau des triggers et son contenu est perdu dès que les triggers sont validés. champ3enr3 champ1enr3 champ2enr3 champ1enr3 champ2enr3 champ3enr3 champ1enr4 champ2enr4 champ3enr4 Lors de la suppression, le ou les enregistrements supprimés sont placés Après validation du trigger ... dans cette table temporaire "DELETED". Après la validation du trigger, on supprime Test **Deleted** définitivement le ou les enregistrements voulus de notre table "Test" et la table champ3 champ2 champ3 champi champ2 champ1 "DELETED" est supprimée. Avec cette champ1enr1 champ2enr1 champ3enr1 champ1enr1 champ2enr1 champ3enr1 méthode, on peut limiter ou empêcher la Le SGBD supprime les suppression ou encore effectuer des enregistrements de la 2ème Étape champ1enr2 champ2enr2 champ3enr2 champ2enr2 champ3enr2 champ1enr2 table "Test" et la table contrôles sur les valeurs avant l'opération "Deleted" est également champ1enr3 champ2enr3 champ3enr3 supprimée champ1em3 champ2enr3 champ3enr3 de suppression.

Utilisation d'un trigger avec l'instruction UPDATE Explications Illustration 1ère Étape Toujours à l'aide de la table fictive "Test", on veut exécuter un trigger avec **Deleted** Inserted Test l'instruction UPDATE. Le SGBD va créer cette fois deux tables virtuelles "DELETED" champ1 champ2 champ3 champ1 champ2 champ3 champ1 champ2 champ3 et "INSERTED". Ces tables ne sont champ1enr1 champ2enr1 champ3enr1 champ1enr1 champ2enr1 champ3enr1 champ1enr1 champ2enr1 champ3enr1 accessibles qu'au niveau des triggers et Le SGBD crée Le SGBD crée leur contenu est perdu dès que les triggers une table champ1enr2 champ2enr2 champ3enr2 champ1enr2 champ2enr2 champ3enr2 une table champ2enr2 champ3enr2 champ1enr2 virtuelle virtuelle sont validés. "Inserted" où "Deleted" où champ1enr3 champ2enr3 champ3enr3 champ1enr3 champ2enr3 champ3enr3 champ1enr3 champ2enr3 champ3enr3 est stockée la est stockée nouvelle valeur l'ancienne Pourquoi se retrouve-t-on avec deux tables champ2enr4 champ3enr4 champ2enr4 champ2enr4 champ1enr4 champ1enr4 champ3enr4 valeur champ1enr4 champ3enr4 ? Avec l'utilisation de l'instruction UPDATE, on se retrouve avec deux valeurs : une Après validation du trigger ... ancienne valeur qui va être stockée dans la table "DELETED" et une nouvelle valeur qui va être stockée dans la table "INSERTED". **Deleted** Inserted Test Le SGBD modifie Le SGBD l'ancienne valeur supprime champ1 champ2 champ3 de la table "Test" champ1 champ2 champ3 champ1 champ2 champ3 Après la validation du trigger, la donnée va l'ancienne avec la nouvelle être mise à jour au niveau de la table "Test" valeur de la valeur de la table champ1enr1 champ2enr1 champ3enr1 champ1enr1 champ2enr1 champ3enr1 table "Test" champ1enr1 champ2enr1 champ3enr1 "Inserted" et les deux tables "DELETED" et "INSERTED" seront supprimées. Avec cette namp2enn amp2eni champ1enr2 champ3enr2 champ1enr2 champ2enr2 champ3enr2 champ1enr2 champ3enr2 méthode, on peut facilement réaliser une Ensuite, la champ1enf3 champ2enr3 mp3enr3 Ensuite, la champ1enr3 champ2enr3 champ3enr3 champ1enf3 champ2enr3 chemp3enr3 table comparaison entre l'ancienne valeur et la table 'Deleted" est nouvelle valeur. "Inserted" est hamp1enr4 champ2enr4 champ3en champ1enr4 champ2enr4 champ3enr4 supprimée champ2enr4 champ3enx4 amp1enr4 supprimée

2ème Étape

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 fair Au lieu de fair		Faire 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 Avec	EVENEMENT INSERT, UPDATE, DELETE	Faire 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 Après	EVENEMENT INSERT, UPDATE, DELETE	Faire 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.

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

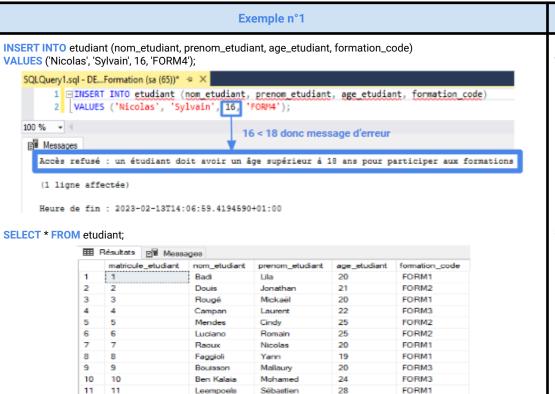
IF (SELECT age_etudiant FROM inserted) < 18
BEGIN

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



Qais

Isabelle

Florian

27

20

23

FORM2

FORM4

FORM5

Hamadou

Lapraz

Luciano

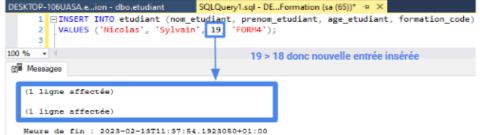
12 12

13 13

14

END

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



Exemple n°2

SELECT * FROM etudiant;

	matricule_etudiant	nom_etudiant	prenom_etudiant	age_etudiant	formation_code
1	1	Badi	Lila	20	FORM1
2	2	Douts	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 Après Insertion et Modification INSERT INTO etudiant (nom_etudiant, prenom_etudiant, age_etudiant, formation_code) Table "Etudiant" VALUES ('Kennedy', 'Leon', 25, 'FORM1'); matricule_etudiant nom_etudiant prenom_etudiant age_etudiant formation_code 20 FORM1 Badi Lila Table "Etudiant" 21 2 2 FORM2 Douis Jonathan matricule_etudiant nom_etudiant prenom_etudiant age_etudiant formation_code 20 FORM1 3 3 Rougé Mickaël 20 FORM1 Badi Lila 2 21 FORM2 22 FORM3 Douis Jonathan 4 Campan Laurent 20 FORM1 3 3 Rougé Mickaël 5 Mendes 25 FORM2 Cindy Campan Laurent 22 FORM3 25 6 Luciano Romain FORM2 Mendes Cindy 25 FORM2 Nicolas 20 FORM1 Raoux 25 FORM2 Luciano Romain 8 Faggioli Yann 19 FORM1 20 FORM1 Raoux Nicolas 20 19 9 Bouisson Mallaury FORM3 Faggioli FORM1 FORM3 9 Rouisson Mallaury 20 10 10 Mohamed 24 FORM3 Ben Kalaia 10 24 FORM3 28 11 11 Leempoels Sébastien FORM1 11 11 Sébastien 28 FORM1 Leempoels 12 12 Hamadou Qaïs 27 FORM2 12 12 27 FORM2 Hamadou Qaïs 13 13 20 FORM4 Lapraz Isabelle FORM4 13 13 Isabelle 20 23 14 14 Florian FORM5 14 23 FORM5 Luciano 14 Luciano Florian FORM1 15 15 Leon Table "Formation" Table "Formation" code_formation nom_formation nombre_total_places_formation nombre restant places formation date_debut_formation date_fin_formation nom_formation nombre_total_places_formation | nombre_restant_places_formation | date_debut_formation date_fin_formation FORM1 10 2023-06-10 2023-12-10 Formation Un FORM1 2023-06-10 Formation Un 10 2023-12-10 FORM2 10 2023-04-05 2023-10-06 Formation Deux FORM2 10 6 2023-04-05 2023-10-06 Formation Deux FORM3 Formation Trois 12 2023-02-09 2023-06-10 FORM3 2023-02-09 2023-06-10 Formation Trois 12 9 FORM4 Formation Quatre 10 2023-03-06 2023-09-10 FORM4 8 2023-03-06 2023-09-10 Formation Quatre FORM5 Formation Cinq 2023-07-20 2023-12-25 FORM5 Formation Cinq 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

CODELETE FROM code_formation 1 FORM1 2 FORM3 3 FORM4 4 FORM5		nombre_total_p	mation = 'FC	DRM2'; Formation" nombre_restant_p	places_formation	date_debut_formation	date_fin_formation
code_formation FORM1 FORM3 FORM4	ion nom_formation Formation Un Formation Trois	nombre_total_p	Table "	Formation"	olaces_formation	date debut formation	date fin formation
3 FORM4				9		2023-06-10 2023-02-09	2023-12-10 2023-06-10
-		tre 10		8		2023-03-06	2023-09-10
	Formation Cing			4		2023-07-20	2023-12-25
	1 2 3 4 5 6 7 8 9 10	1 3 4 7 8 9 10 11 13 14	Table	"Etudiant"	20 20 22 20 19 20 24 28 20 23	FORM1 FORM3 FORM1 FORM1 FORM3 FORM3 FORM3 FORM3 FORM4 FORM4 FORM5	
		1 2 3 4 5 6 7 8 9 10	3 4 4 7 5 8 6 9 7 10 8 11 9 13 10 14 11 15	Table matricule_etudiant nom_etudiant 1 1 Badi 2 3 Rougé 3 4 Campan 4 7 Raoux 5 8 Faggioli 6 9 Bouisson 7 10 Ben Kalaia 8 11 Leempoels 9 13 Lapraz 10 14 Luciano 11 15 Kennedy	1 1 Badi Lila 2 3 Rougé Mickaël 3 4 Campan Laurent 4 7 Raoux Nicolas 5 8 Faggioli Yann 6 9 Bouisson Mallaury 7 10 Ben Kalaia Mohamed 8 11 Leempoels Sébastien 9 13 Lapraz Isabelle 10 14 Luciano Florian 11 15 Kennedy Leon	Table "Etudiant" matricule_etudiant nom_etudiant prenom_etudiant age_etudiant 1 1 Badi Lila 20 2 3 Rougé Mickaël 20 3 4 Campan Laurent 22 4 7 Raoux Nicolas 20 5 8 Faggioli Yann 19 6 9 Bouisson Mallaury 20 7 10 Ben Kalaia Mohamed 24 8 11 Leempoels Sébastien 28 9 13 Lapraz Isabelle 20 10 14 Luciano Florian 23 11 15 Kennedy Leon 25	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