**TRAVAUX PRATIQUES - NUMERO 9**

**Langage Transact SQL (T-SQL)**

**Les Déclencheurs (Triggers) - Partie 2**

Tableaux récapitulatifs des syntaxes :

**Syntaxe générale pour créer/modifier un Trigger:**

|  |  |
| --- | --- |
| **Élément Syntaxique** | **Description** |
| **CREATE [ OR ALTER ] TRIGGER** nom\_du\_trigger | Crée un nouveau trigger ou modifie un trigger existant. Remplacez "**nom\_du\_trigger**" par le nom que vous souhaitez donner à votre trigger. |
| **ON** nom\_de\_la\_table/vue | Spécifie la table ou la vue sur laquelle le trigger doit agir. Remplacez "**nom\_de\_la\_table/vue**" par le nom de la table ou de la vue que vous voulez utiliser. |
| **AFTER / FOR / INSTEAD OF** | Spécifie quand le trigger doit être activé. "**AFTER**" signifie que le trigger doit être activé après l'événement spécifié **(INSERT, UPDATE, DELETE)**. "**FOR**" est synonyme de "**AFTER**". "**INSTEAD OF**" signifie que le trigger doit être activé à la place de l'événement spécifié. |
| **{INSERT, UPDATE, DELETE}** | Spécifie l'événement qui active le trigger. Vous pouvez choisir parmi les options suivantes : **INSERT, UPDATE, DELETE**. Vous pouvez spécifier plus d'une option en les séparant par des virgules. |
| **AS** | Indique le début du bloc de code que le trigger doit exécuter. |
| **-- Les instructions SQL à exécuter lorsque le déclencheur est activé** | Ici, vous devez écrire **le code SQL** que le trigger doit exécuter lorsque l'événement spécifié se produit. C'est ce que le trigger "fait". |

**Commandes utiles:**

|  |  |
| --- | --- |
| **Élément** | **Description** |
| La fonction : **UPDATE(column)** | Dans le contexte d'un trigger, la fonction **UPDATE(column)** est utilisée pour déterminer si une colonne spécifique **a été mise à jour**. Elle retourne **TRUE** si la colonne a été mise à jour lors de l'opération qui a déclenché le trigger ; sinon, elle retourne **FALSE**. |

**Gestion des Triggers :**

|  |  |
| --- | --- |
| **Commande** | **Description** |
| **Afficher la liste des triggers** | Pour afficher la liste des triggers dans une base de données, vous pouvez exécuter la requête suivante :  **SELECT name FROM sys.triggers;** |
| **Suppression d'un trigger** | Pour supprimer un trigger, vous pouvez utiliser la commande DROP TRIGGER. La syntaxe générale est :  **DROP TRIGGER trigger\_name;** |

Schéma de la Base de données :

Schéma de la base de données **EMSIBD** :

**Eleve(**id\_eleve, nom, prenom, date\_naissance, **#**id\_classe**)**

**Enseignant(**id\_enseignant, nom, prenom, email**)**

**Cours(**id\_cours, nom, **#**id\_enseignant**)**

**Classe(**id\_classe, nom**)**

**Note(#**id\_eleve, **#**id\_cours, note**)**

Manipulation :

**Question 1:** Créez un trigger qui s’assure la règle métier suivante: lors de la mise à jour de la note d’un élève (la note d’un élève est définie par l’élève “ID\_eleve” et le cours “ID\_cours” , cette note doit être supérieure ou égale à l’ancienne note.

**Étapes :**

1. Créer un trigger qui s'exécute **au lieu de la mise à jour** dans la table "**Note**".
2. Dans ce trigger, si la règle n’est pas respecté, déclencher une exception. Le programme principale devra intercepter cette exception et afficher un message d’erreur.
3. Sinon, procéder à la mise à jour de la note.

**Question 2:** tester le trigger de la Question 1 : faire la mise à jour de la note de l’élève ID\_eleve = 1 pour le cours ID\_cours = 2 tel que:

1. La nouvelle note est : 10
2. La nouvelle note est : 14

**Question 3:** Créer un trigger qui, à chaque suppression d’une note, enregistre celle-ci dans une table **NotesArchive**.

**Étapes :**

1. Créer une table “**NotesArchive**” avec les champs suivants: IDnoteArchive, ID\_eleve, ID\_cours, note. (IDnoteArchive est la clé primaire et est de type IDENTITY, c’est à dire qu’elle est incrémentée automatiquement - utiliser l’option IDENTITY(1,1).
2. Récupérer la ligne supprimée.
3. L’insérer dans la table “**NotesArchive**”.

**Question 4:** tester le trigger de la Question 3 en supprimant une note

**Question 5:** Désactiver le trigger de la question 3.

**Question 6:** Ecrire un trigger qui archive les élèves supprimées avec leurs détails (leurs notes). Pour cela, on considère 2 tables similaires **ElevesArchive** et **NotesArchive** respectivement de même schéma que les tables **Eleve** et **Note** en plus d’un attributs de type IDENTITY(1,1) pour identifier les lignes dans les tables d’archive.

**Étapes :**

1. Définir un trigger INSTEAD OF sur la table Eleves
2. Dans le trigger, supprimer les notes
3. Dans le trigger, supprimer l’élève,
4. Ensuite, faire l’archivage des notes
5. Enfin, faire l’archivage de l’élève

**Question 7:** tester le trigger de la question 6 avec un élève.