ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

HONORIS UNITED UNIVERSITIES

Rapport de projet

3ème année

Ingénierie Informatique et Réseaux

Devoir SQL Server: Les Triggers

REALISE PAR :

CHAKOUR Imad

ENCADRE PAR :

TUTEUR DE L'ECOLE : Dr. Nouhaila bensalah

ANNEE UNIVERSITAIRE : 2023-2024

ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

HONORIS UNITED UNIVERSITIES

PLAN

I. INTRODUCTION

II. Part I : Les Triggers

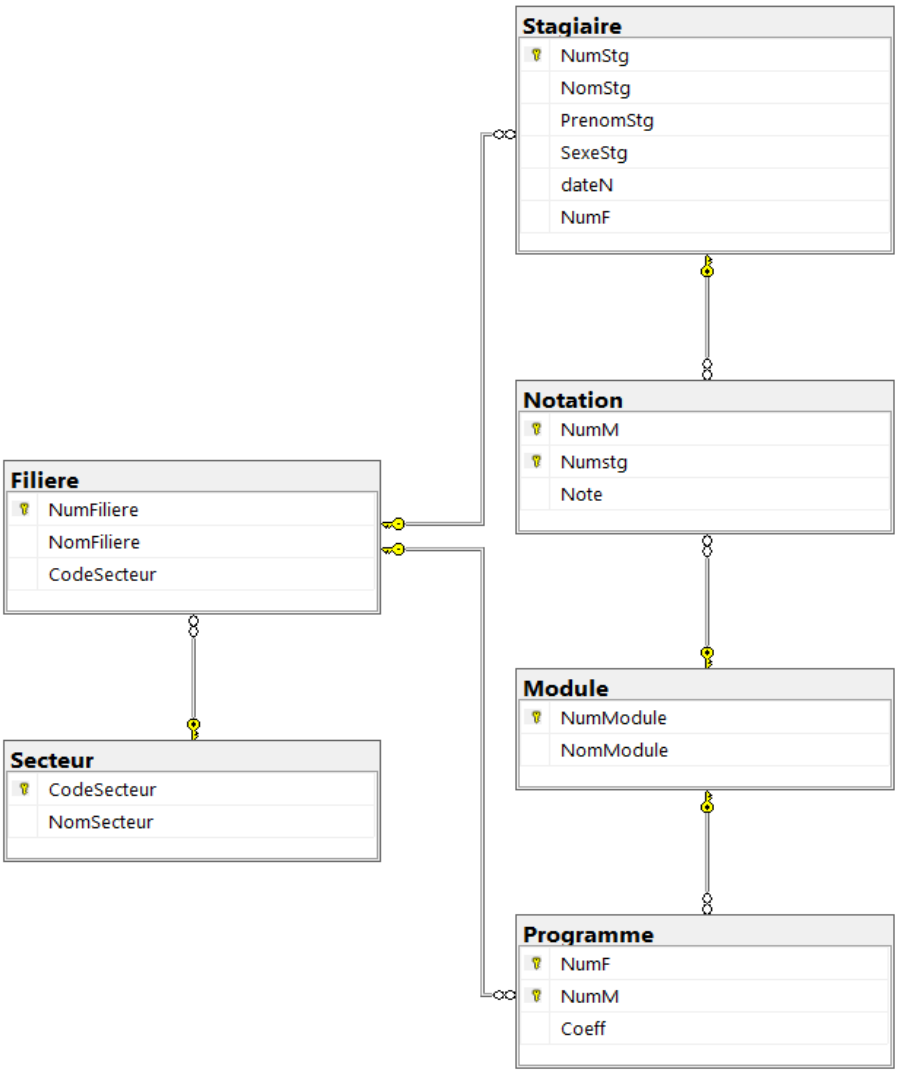
III. Devoir II : Part I : Full Backup, Part II : Differential Backup, Partie III : Backup « transaction Log »

IV. CONCLUSION

I. INTRODUCTION

Un trigger dans SQL Server est un type spécial de procédure stockée qui s'exécute automatiquement en réponse à certains événements sur une table ou une vue particulière. Les triggers sont utilisés pour appliquer des règles métier, maintenir l'intégrité des données et auditer les modifications dans la base de données. Ils peuvent être définis pour s'exécuter avant ou après les opérations d'insertion, de mise à jour et de suppression.

CONSIDERONS LA BASE DE DONNEES GESTIONSTAGE AVEC LE 'DATAGRAM' :



II. Part I : Les Triggers

- Ecrire un trigger qui, à la suppression d'un stagiaire, vérifie s'il a été évalué et si des notes lui ont été attribuées. Si c'est le cas Empêcher la suppression.

SQLQuery2.sql - YGB...tionStage (sa (69))* | YGblade\DATACAMP...tage - Diagram_1*

```
create or alter trigger tr_Etudiants_forDelete
ON Stagiaire
INSTEAD OF DELETE
AS
BEGIN
DECLARE @id VARCHAR(30);
SELECT @id = NumStg FROM deleted;
IF EXISTS (SELECT NumM FROM Notation WHERE Numstg = @id)
BEGIN
RAISERROR ('Impossible de supprimer NumStg = %s. Les Notes existent.', 16, 1, @id);
ROLLBACK TRANSACTION;
END
ELSE
BEGIN
DELETE FROM Stagiaire WHERE NumStg = @id;
INSERT INTO StudentsAudit
VALUES ('Un étudiant ayant l"Id ' + CONVERT(VARCHAR(20), @id) + ' est supprimé le ' + CONVERT(VARCHAR(30), GETDATE()));
END
END;
```

Défini comme un trigger INSTEAD OF DELETE sur la table Stagiaire. Cela signifie que le trigger s'exécutera à la place de l'opération de suppression.

Capture le NumStg (numéro d'étudiant) à partir de la table deleted, qui contient temporairement les lignes qui doivent être supprimées.

Vérifie s'il existe des entrées dans la table Notation pour l'étudiant en utilisant une clause IF EXISTS.

Lève une erreur en utilisant RAISERROR et annule la transaction, empêchant ainsi la suppression.

Pvrocède à la suppression de la table Stagiaire et insère éventuellement un enregistrement dans une table d'audit (StudentsAudit) pour garder une trace des suppressions.

- Ecrire un trigger qui, à l'insertion dans la table Notation, vérifie les contraintes d'intégrité référentielle et que la note saisie soit entre 0 et 20, Sinon Empêcher l'insertion.

SQLQuery2.sql - YGB...tionStage (sa (69))* | YGblade\DATACAMP...tage - Diagram_1*

```
CREATE TRIGGER tr_Valider_Notation
ON Notation
INSTEAD OF INSERT
AS
BEGIN
DECLARE @Numstg INT, @NumM INT, @Note DECIMAL(4,2);
SELECT @Numstg = Numstg, @NumM = NumM, @Note = Note FROM inserted;
IF NOT EXISTS (SELECT NumStg FROM Stagiaire WHERE NumStg = @Numstg)
BEGIN
RAISERROR ('le stagiaire avec NumStg = %d n"existe pas.', 16, 1, @Numstg);
ROLLBACK TRANSACTION;
RETURN;
END
IF NOT EXISTS (SELECT 1 FROM Module WHERE NumModule = @NumM)
BEGIN
RAISERROR ('le module avec NumM = %d n"existe pas.', 16, 1, @NumM);
ROLLBACK TRANSACTION;
RETURN;
END
IF @Note < 0 OR @Note > 20
BEGIN
RAISERROR ('La note n"est pas valide.', 16, 1);
ROLLBACK TRANSACTION;
RETURN;
END
INSERT INTO Notation (Numstg, NumM, Note)
SELECT Numstg, NumM, Note FROM inserted;
END;
```

Ce trigger garantit que seules des données valides sont insérées dans la table Notation, maintenant ainsi l'intégrité référentielle et la validité des données.

Devoir II :

Part I : Full Backup en utilisant le script

GestionStage.sql -...SQL.master (sa (89))*

```
USE GestionStage;
BACKUP DATABASE GestionStage TO DISK='C:\Users\chako\OneDrive\Desktop\backup\GestionStage.bak';

drop table Notation;
select * from notation;

USE master;
RESTORE DATABASE GestionStage FROM DISK='C:\Users\chako\OneDrive\Desktop\backup\GestionStage.bak' WITH REPLACE;

USE GestionStage;
select * from notation;
```

111 %

Messages

```
Processed 720 pages for database 'GestionStage', file 'GestionStage' on file 1.
Processed 2 pages for database 'GestionStage', file 'GestionStage_log' on file 1.
RESTORE DATABASE successfully processed 722 pages in 0.018 seconds (313.151 MB/sec).

Completion time: 2024-07-07T02:53:20.0070527+01:00
```

Base de données a sauvegarder (soit la base de données GestionStage)

Utilisation de l'instruction BACKUP DATABASE pour restaurer complétement (Full Backup) la base de données GestionStage dans le fichier 'GestionStage.bak' localisé à l'emplacement « `C:\Users\chako\OneDrive\Desktop\backup` »

Pour la restauration de la base de données à partir d'un Full Backup, on doit

a. Se placer sur la base de données système « Master »

b. Utiliser l'instruction RESTORE DATABASE de la manière suivante : Utilisation l'option « with replace » ; pour remplacer la version actuelle avec la dernière version sauvegardée

Je remarque que après la restauration de la base donnés, la table `Notation` a été restaurée, cela signifie qu'elle était présente dans la sauvegarde.

Part II : Differential Backup en utilisant le script

GestionStage.sql -...SQL.master (sa (89))*

```
insert into Stagiaire values (1006,'Amine','Chakour','M','2012-11-08',101);
insert into Stagiaire values (1007,'Vanny','Chakour','F','2023-06-10',102);
select * from Stagiaire;

use GestionStage
BACKUP DATABASE GestionStage to DISK='C:\Users\chako\OneDrive\Desktop\backup\GestionStage1.bak'
WITH DIFFERENTIAL;

select * from Stagiaire
-- supprimer la table acheter
drop table Stagiaire

use master;
RESTORE DATABASE GestionStage FROM DISK='C:\Users\chako\OneDrive\Desktop\backup\GestionStage.bak'
with NORECOVERY

use master;
RESTORE DATABASE GestionStage FROM DISK='C:\Users\chako\OneDrive\Desktop\backup\GestionStage1.bak'
WITH RECOVERY;
```

111 %

Messages

```
Processed 240 pages for database 'GestionStage', file 'GestionStage' on file 1.
Processed 2 pages for database 'GestionStage', file 'GestionStage_log' on file 1.
RESTORE DATABASE successfully processed 242 pages in 0.010 seconds (188.671 MB/sec).

Completion time: 2024-07-07T02:30:31.3545239+01:00
```

- Insertion de Données :

  Deux nouveaux stagiaires ont été ajoutés à la table Stagiaire.

- Backup Différentiel :

  Une sauvegarde différentielle de la base de données GestionStage a été créée.

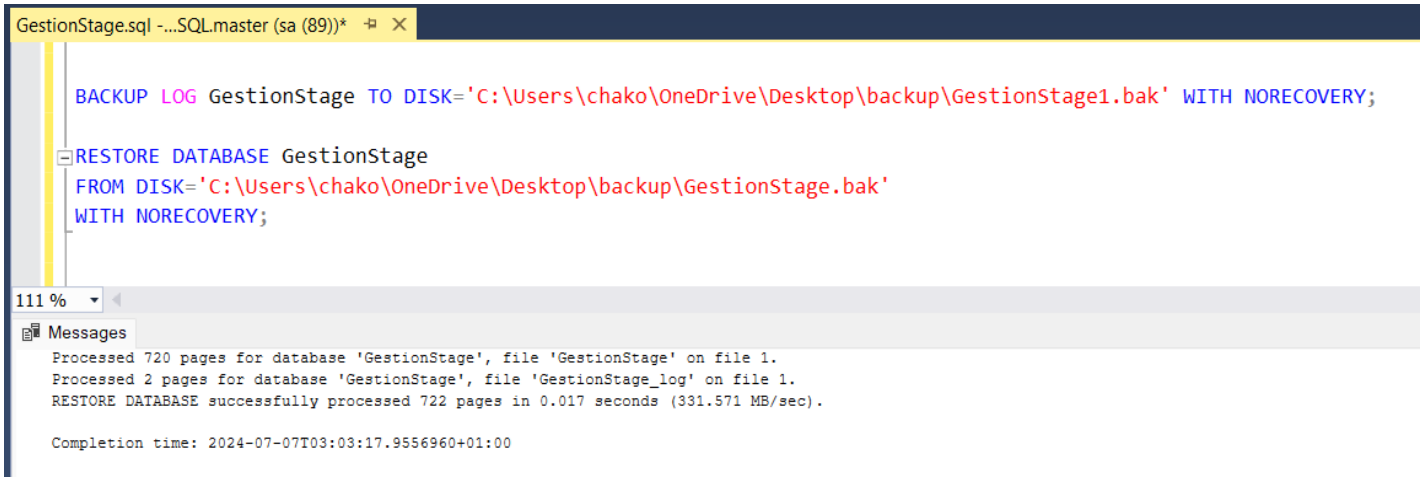
- Suppression de la Table Stagiaire :

  La table Stagiaire a été affichée puis supprimée.

- Restauration de la Base de Données :

  La base de données GestionStage a été restaurée d'abord à partir d'une sauvegarde complète, suivie d'une restauration à partir de la sauvegarde différentielle pour récupérer les données ajoutées.

Partie III : Backup « transaction Log »



Ces étapes vous permettront de sauvegarder et restaurer le log des transactions de votre base de données, en assurant la continuité et l'intégrité des données même après des transactions importantes ou des modifications fréquentes.

Création et Gestion des Utilisateurs

Utilisateurs SQL Server : Identité au sein de la base de données, liée à un login SQL Server ou créée indépendamment.

```
CREATE USER YG_chak WITHOUT LOGIN;

ALTER USER YG_chak WITH DEFAULT_SCHEMA = reey;

DROP USER YG_chak;
```

user master.YG_chak

Les rôles SQL Server

Rôles de Serveur : Prédéfinis comme sysadmin, serveradmin, securityadmin, etc.

Rôles de Base de Données : Prédéfinis comme db_owner, db_securityadmin, db_datareader, db_datawriter, etc.

```
CREATE SERVER ROLE sysadmin;
ALTER SERVER ROLE sysadmin ADD MEMBER YG_chak;

CREATE ROLE igl;
CREATE ROLE HRRole;
ALTER ROLE igl ADD MEMBER YG_chak;

GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo TO igl;
GRANT SELECT ON SCHEMA::dbo TO HRRole;
```

111 %

Messages

```
Commands completed successfully.

Completion time: 2024-07-07T03:46:03.1449202+01:00
```

CONCLUSION

La gestion des utilisateurs et des rôles dans SQL Server est cruciale pour assurer la sécurité et la gestion efficace des bases de données. En utilisant des logins et des rôles, on peut contrôler l'accès aux données et aux fonctionnalités spécifiques, en limitant les privilèges aux utilisateurs autorisés selon les besoins de l'organisation.

Les étapes clés comprennent la création de logins pour authentifier les utilisateurs, la création de rôles prédéfinis ou personnalisés pour organiser les permissions, et l'attribution d'autorisations précises à ces rôles. Il est également possible de refuser des autorisations spécifiques lorsque nécessaire, assurant ainsi une granularité dans la gestion des accès.

En résumé, une gestion proactive des utilisateurs et des rôles dans SQL Server contribue à maintenir l'intégrité des données, à renforcer la sécurité et à optimiser la gestion des bases de données au sein de l'entreprise.