TP4 Déclencheur SQL

I - Déclencheurs BEFORE INSERT

1. Valider la note

```
DELIMITER //
CREATE TRIGGER valider_note
BEFORE INSERT ON evaluation
FOR EACH ROW
BEGIN
IF NEW.score < 1 OR NEW.score > 10 THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La note doit être entre 1 et 10.';
END IF;
END;//
DELIMITER ;
```

2. Remplir automatiquement la date

```
DELIMITER //
CREATE TRIGGER remplir_date
BEFORE INSERT ON evaluation
FOR EACH ROW
BEGIN
IF NEW.date_evaluation IS NULL THEN
SET NEW.date_evaluation = CURDATE();
END IF;
END;//
DELIMITER ;
```

3. Enregistrer l’historique des insertions

```
DELIMITER //
CREATE TRIGGER enregistrer_historique
BEFORE INSERT ON evaluation
FOR EACH ROW
BEGIN
INSERT INTO hist_eval (score, date_modif, commentaire)
VALUES (NEW.score, NEW.date_evaluation, NEW.commentaire);
END;//
DELIMITER ;
```

4. Vérification de la date

```
DELIMITER //
CREATE TRIGGER verifier_date_future
BEFORE INSERT ON evaluation
FOR EACH ROW
BEGIN
IF NEW.date_evaluation > CURDATE() THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La date ne peut pas être dans le futur.';
END IF;
END;//
DELIMITER ;
```

5. Vérification de la période decongé

```
DELIMITER //
CREATE TRIGGER verifier_periode_de_congé
BEFORE INSERT ON congé
FOR EACH ROW
BEGIN
IF NEW.date_debut > NEW.date_fin THEN
SIGNAL SQLSTATE'45000'SET MESSAGE_TEXT ='La date de début doit être avant la date de fin de congé.';
END IF;
END;//
DELIMITER /;
```

II - Déclencheurs AFTER INSERT

1. Ajouter un enregistrement dans la table evaluation à la création d'un employé

```
DELIMITER //
CREATE TRIGGER ajout_eval_employe
AFTER INSERT ON employe
FOR EACH ROW
BEGIN
INSERT INTO evaluation (id_employe, note, commentaire)
VALUES (NEW.id, 5, 'Évaluation initiale, bonne chance');
END;//
DELIMITER ;
```

2. Ajouter Des Congés Obligatoires

```
DELIMITER //
CREATE TRIGGER ajout_conges_obligatoires
AFTER INSERT ON employe
FOR EACH ROW
BEGIN
INSERT INTO conges (id_employe, date_debut, date_fin)
VALUES (NEW.id, '2024-08-01', '2024-08-21');
END;//
DELIMITER ;
```

III - Déclencheurs BEFORE UPDAT

1. Vérification de l’existence d’un département

```
DELIMITER //
CREATE TRIGGER verifier_departement_existant
BEFORE UPDATE ON employe
FOR EACH ROW
BEGIN
IF NOT EXISTS (SELECT 1 FROM departement WHERE id = NEW.id_departement) THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Département non existant.';
END IF;
END;//
DELIMITER ;
```

2. Empêcher D'allonger Les Congés Approuvés

```
DELIMITER //
CREATE TRIGGER verifier_prolongation_conge
BEFORE UPDATE ON congé
FOR EACH ROW
BEGIN
IF NEW.date_fin > OLD.date_fin THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Impossible d’allonger un congé
approuvé.';
END IF;
END;//
DELIMITER ;
```

3. Mettre Un Salaire Minimum

```
DELIMITER //
CREATE TRIGGER verifier_salaire_minimum
BEFORE UPDATE ON employe
FOR EACH ROW
BEGIN
IF NEW.salaire < 30000 THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Le salaire doit être supérieur à 30
000.';
END IF;
END;//
DELIMITER ;
```

IV - Déclencheurs AFTER UPDATE

1. Enregistrer la personne ayant modifié un salaire

```
CREATE TABLE hist_salaire (
   id INT PRIMARY KEY AUTO_INCREMENT,
   id_employe INT,
   ancien_salaire DECIMAL(10,2),
   nouveau_salaire DECIMAL(10,2),
   date_modif DATETIME,
   modifie_par VARCHAR(255)
);
```

```
DELIMITER //
CREATE TRIGGER log_modification_salaire
AFTER UPDATE ON employe
FOR EACH ROW
BEGIN
IF NEW.salaire != OLD.salaire THEN
INSERT INTO hist_salaire (id_employe, ancien_salaire, nouveau_salaire, date_modif,
modifie_par)
VALUES (NEW.id, OLD.salaire, NEW.salaire, NOW(), USER());
END IF;
END;//
DELIMITER ;
```

V - Déclencheurs BEFORE DELETE

1. Éviter de supprimer un département avec des employés

```
DELIMITER //
CREATE TRIGGER verifier_employes_dans_departement
BEFORE DELETE ON departement
FOR EACH ROW
BEGIN
IF EXISTS (SELECT 1 FROM employe WHERE id_departement = OLD.id) THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Impossible de supprimer un
département avec des employés.';
END IF;
END;//
DELIMITER ;
```

2. Éviter la suppression d'une évaluation d'un employé

```
DELIMITER //
CREATE TRIGGER verifier_evaluation_employe
BEFORE DELETE ON evaluation
FOR EACH ROW
BEGIN
IF EXISTS (SELECT 1 FROM employe WHERE id = OLD.id_employe) THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Impossible de supprimer l'évaluation
d'un employé actif.';
END IF;
END;//
DELIMITER ;
```

VI - DéclencheursAFTER DELETE

1. Supprimer Les données résultantes d'un employé

```
DELIMITER //
CREATE TRIGGER supprimer_donnees_employe
AFTER DELETE ON employe
FOR EACH ROW
BEGIN
DELETE FROM evaluation WHERE id_employe = OLD.id;
DELETE FROM conges WHERE id_employe = OLD.id;
DELETE FROM hist_salaire WHERE id_employe = OLD.id;
DELETE FROM projets WHERE id_employe = OLD.id;
DELETE FROM formation WHERE id_employe = OLD.id;
END;//
```

```
DELIMITER ;
```