

Compte rendu TD2 & TD3

R3.07 SQL dans un langage de programmation

TD2 : TRIGGER

Liste des triggers

Nom Trigger	Type : before ou after	Insert, delete, update	Nom table	For each row : oui ou non
empêcher_diminution_salaire	after	update	employe	oui
empêcher_augmentation_hebdo	after	update	employe	oui
supprimer_employe	before	delete	employe	non
supprimer_projet	before	delete	projet	non
check_duree_insert	before	insert	travail	oui
check_duree_update	before & after	update	travail	oui
check_hebdo_update	before	update	employe	oui
check_responsable_projets	before	insert & update	projet	oui
check_service_projets	before	insert & update	concerne	oui
check_chef_salaire	before	insert & update	employe	oui
alerte_salaire	before	insert & update	employe	oui

Exercice 1 :

A. Jeux d'essai pour le trigger empêcher_diminution_salaire

Trigger :

```
CREATE OR REPLACE TRIGGER empecher_diminution_salaire
AFTER UPDATE OF salaire ON employe
FOR EACH ROW
BEGIN
  IF :NEW.salaire < :OLD.salaire THEN
    RAISE_APPLICATION_ERROR(-20001, 'Vous ne pouvez pas diminuer le salaire
    de l"employer");
  END IF;
END;
```

Cas de test pour déclancher le trigger :

```
UPDATE employe SET salaire = salaire - 100 WHERE NUEMPL = 20;
```

Résultat attendu : Une erreur est déclenchée car le salaire ne peut pas être diminué.

B. Jeux d'essai pour le trigger empecher_augmentation_hebdo

Trigger :

```
CREATE OR REPLACE TRIGGER empecher_augmentation_hebdo
AFTER UPDATE OF HEBDO ON employe
FOR EACH ROW
BEGIN
  IF :NEW.HEBDO > :OLD.HEBDO THEN
    RAISE_APPLICATION_ERROR(-20002, 'Vous ne pouvez pas augmenter la durée
    hebdomadaire de l"employé");
  END IF;
END;
```

Cas de test pour déclancher le trigger :

```
UPDATE employe SET HEBDO = HEBDO + 1 WHERE NUEMPL = 20;
```

Résultat attendu : Une erreur est déclenchée car la durée hebdomadaire ne peut pas être augmentée.

Exercice 2 :

A. Cas où la suppression de l'employé est interdite :

Trigger :

```
CREATE OR REPLACE TRIGGER supprimer_employe
BEFORE DELETE ON employe
DECLARE
  v_deleted_count NUMBER;
BEGIN
  -- Supprimer les lignes correspondantes dans la table travail
  -- seulement si l'employé n'est ni responsable de projet ni chef de service
  DELETE FROM travail
  WHERE NUEMPL IN (SELECT NUEMPL FROM employe) -- L'employé doit
  exister dans la table employe
  AND NUEMPL NOT IN (SELECT RESP FROM projet) -- Pas responsable de projet
  AND NUEMPL NOT IN (SELECT CHEF FROM service); -- Pas chef de service

  -- Vérifier combien de lignes ont été supprimées
```

```

v_deleted_count := SQL%ROWCOUNT;

-- Si aucune ligne n'a été supprimée, lever une exception
IF v_deleted_count = 0 THEN
    RAISE_APPLICATION_ERROR(-20005, 'Aucune ligne n''a été supprimée car
        l''employé est soit chef de projet soit chef de service, ou n''a pas de travail
        associé.');
```

END IF;
END;

Cas de test pour déclencher le trigger :

```
DELETE FROM employe WHERE NUEMPL = 41;
```

Résultat attendu : Une erreur est déclenchée car l'employé est chef de service.

Cas de test pour déclencher le trigger :

```
DELETE FROM employe WHERE NUEMPL = 30;
```

Résultat attendu : Une erreur est déclenchée car l'employé est responsable d'un projet.

B. Vérifier les données avant la suppression du projet :

Trigger :

```

CREATE OR REPLACE TRIGGER supprimer_projet
BEFORE DELETE ON projet
DECLARE
    v_deleted_count_travail NUMBER;
    v_deleted_count_concerne NUMBER;
BEGIN
    -- Supprimer les lignes dans la table travail où le projet n'existe plus dans la table
    -- projet
    DELETE FROM travail
    WHERE NUPROJ NOT IN (SELECT NUPROJ FROM projet);

    -- Enregistrer le nombre de lignes supprimées dans travail
    v_deleted_count_travail := SQL%ROWCOUNT;

    -- Supprimer les lignes dans la table concerne où le projet n'existe plus dans la
    -- table projet
    DELETE FROM concerne
    WHERE NUPROJ NOT IN (SELECT NUPROJ FROM projet);

    -- Enregistrer le nombre de lignes supprimées dans concerne
    v_deleted_count_concerne := SQL%ROWCOUNT;

    -- Optionnel: lever une erreur si aucune ligne n'a été supprimée
    IF v_deleted_count_travail = 0 AND v_deleted_count_concerne = 0 THEN
        RAISE_APPLICATION_ERROR(-20006, 'Aucune ligne supprimée dans travail ou
            concerne pour ce projet.');
```

END IF;
END;

Cas de test pour déclencher le trigger :

Suppression d'un projet (par exemple, Projet 1)

DELETE FROM projet **WHERE** NUPROJ = 1;

Vérification des suppressions dans les autres tables Devrait retourner 0 ligne

SELECT * FROM travail **WHERE** NUPROJ = 1;

SELECT * FROM concerne **WHERE** NUPROJ = 1;

Résultat attendu : Les lignes associées au projet supprimé sont également supprimées dans les tables travail et concerne.

Exercice 3 :

A - La somme des durées de travail d'un employé ne doit pas excéder son temps de travail hebdomadaire

Pour vérifier la contrainte **SUM(duree) <= hebdo**, nous devons mettre en place des triggers qui se déclenchent lors des opérations INSERT et UPDATE sur les tables employe et travail.

Opérations sur la table travail :

- INSERT : Lorsqu'une nouvelle ligne est insérée dans la table travail, nous devons vérifier que la somme des durées de travail de l'employé ne dépasse pas son temps de travail hebdomadaire.
- UPDATE : Lorsqu'une ligne existante est mise à jour dans la table travail, nous devons vérifier que la somme des durées de travail de l'employé ne dépasse pas son temps de travail hebdomadaire. Opérations sur la table employe :
- UPDATE : Lorsqu'un employé met à jour son temps de travail hebdomadaire (hebdo), nous devons vérifier que la somme des durées de travail de l'employé ne dépasse pas son nouveau temps de travail hebdomadaire.

Triggers :

```
CREATE OR REPLACE TRIGGER check_duree_insert
BEFORE INSERT ON travail
FOR EACH ROW
DECLARE
    v_sum_duree NUMBER;
    v_hebdo employe.hebdo%TYPE;
BEGIN
    -- Calculer la somme des durées pour l'employé
    SELECT SUM(duree) INTO v_sum_duree
    FROM travail
    WHERE nuempl = :NEW.nuempl;

    -- Ajouter la nouvelle durée
    v_sum_duree := v_sum_duree + :NEW.duree;

    -- Obtenir le temps de travail hebdomadaire de l'employé
    SELECT hebdo INTO v_hebdo
    FROM employe
    WHERE nuempl = :NEW.nuempl;

    -- Vérifier si la somme des durées dépasse le temps de travail hebdomadaire
    IF v_sum_duree > v_hebdo THEN
        RAISE_APPLICATION_ERROR(-20005, 'La somme des durées de travail dépasse
            le temps de travail hebdomadaire.');
```

END IF;
END;

```
CREATE OR REPLACE TRIGGER check_duree_update
FOR UPDATE ON travail
COMPOUND TRIGGER
```

```
-- Variable to hold the sum of durations for each employee
```

```
TYPE emp_duree_type IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
emp_duree emp_duree_type;
```

```
-- Before each row is updated
```

```
BEFORE EACH ROW IS
```

```
BEGIN
```

```
-- Initialize or reset the sum for the employee if it's the first row being processed
```

```
IF emp_duree.EXISTS(:NEW.nuempl) THEN
```

```
    emp_duree(:NEW.nuempl) := emp_duree(:NEW.nuempl) + :NEW.duree -
        NVL(:OLD.duree, 0);
```

```
ELSE
```

```
    emp_duree(:NEW.nuempl) := :NEW.duree;
```

```
END IF;
```

```
END BEFORE EACH ROW;
```

```
-- After the entire statement (to avoid mutating table error)
```

```
AFTER STATEMENT IS
```

```
    v_hebdo employe.hebdo%TYPE;
```

```
    v_sum_duree NUMBER;
```

```
BEGIN
```

```
-- Loop through each employee whose records were modified
```

```
FOR indx IN emp_duree.FIRST..emp_duree.LAST LOOP
```

```
-- Get the weekly limit for the employee
```

```
    SELECT hebdo INTO v_hebdo
```

```
    FROM employe
```

```
    WHERE nuempl = indx;
```

```
-- Get the total duration for the employee from the table (now allowed after the mutation phase)
```

```
    SELECT SUM(duree) INTO v_sum_duree
```

```
    FROM travail
```

```
    WHERE nuempl = indx;
```

```
-- Check if the sum of durations exceeds the weekly limit
```

```
IF v_sum_duree > v_hebdo THEN
```

```
    RAISE_APPLICATION_ERROR(-20006, 'La somme des durées de travail
    dépasse le temps de travail hebdomadaire pour l''employé ' || indx || '.');
```

```
END IF;
```

```
END LOOP;
```

```
END AFTER STATEMENT;
```

```
END check_duree_update;
```

```
CREATE OR REPLACE TRIGGER check_hebdo_update
```

```
BEFORE UPDATE OF hebdo ON employe
```

```
FOR EACH ROW
```

```
DECLARE
```

```
    v_sum_duree NUMBER;
```

```
BEGIN
```

```
-- Calculer la somme des durées pour l'employé
```

```
    SELECT SUM(duree) INTO v_sum_duree
```

```

FROM travail
WHERE nuempl = :NEW.nuempl;

-- Vérifier si la somme des durées dépasse le nouveau temps de travail
  hebdomadaire
IF v_sum_duree > :NEW.hebdo THEN
  RAISE_APPLICATION_ERROR(-20007, 'La somme des durées de travail dépasse
    le nouveau temps de travail hebdomadaire.');
```

END IF;

END;

Cas de test pour déclencher le trigger :

```
INSERT INTO travail VALUES (20, 237, 90);
```

Résultat attendu : Une erreur est déclenchée car la somme des durées de travail dépasse le temps de travail hebdomadaire.

```
UPDATE travail SET DUREE = 99 WHERE NUEMPL = 20 AND NUPROJ = 492;
```

Résultat attendu : Une erreur est déclenchée car la somme des durées de travail dépasse le temps de travail hebdomadaire.

```
UPDATE employe SET HEBDO = 5 WHERE NUEMPL = 20;
```

Résultat attendu : Une erreur est déclenchée car la somme des durées de travail dépasse le nouveau temps de travail hebdomadaire.

B. Un employé est responsable au plus sur 3 projets

Trigger :

```

CREATE OR REPLACE TRIGGER check_responsable_projets
BEFORE INSERT OR UPDATE OF resp ON projet
FOR EACH ROW
DECLARE
  v_count NUMBER;
BEGIN
  -- Calculer le nombre de projets pour lesquels l'employé est responsable
  SELECT COUNT(*)
  INTO v_count
  FROM projet
  WHERE resp = :NEW.resp;

  -- Ajouter le nouveau projet si c'est une insertion
  IF INSERTING THEN
    v_count := v_count + 1;
  END IF;

  -- Vérifier si le nombre de projets dépasse 3
  IF v_count > 3 THEN
    RAISE_APPLICATION_ERROR(-20008, 'Un employé ne peut pas être responsable
      de plus de 3 projets.');
```

END IF;

END;

Insertion d'un projet :

```
INSERT INTO PROJET VALUES (103, 'Projet 103', 30);
```

C - Un service ne peut être concerné par plus de 3 projets

Trigger :

```
CREATE OR REPLACE TRIGGER check_service_projets
BEFORE INSERT OR UPDATE OF NUSERV ON concerne
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    -- Calculer le nombre de projets pour lesquels le service est concerné
    SELECT COUNT(*)
    INTO v_count
    FROM concerne
    WHERE NUSERV = :NEW.NUSERV;

    -- Ajouter le nouveau projet si c'est une insertion
    IF INSERTING THEN
        v_count := v_count + 1;
    END IF;

    -- Vérifier si le nombre de projets dépasse 3
    IF v_count > 3 THEN
        RAISE_APPLICATION_ERROR(-20009, 'Un service ne peut être concerné par
        plus de 3 projets. ');
    END IF;
END;
```

Insertion d'une association de projet avec un service :

```
INSERT INTO CONCERNE VALUES (1, 492);
INSERT INTO CONCERNE VALUES (1, 160);
```

Résultat attendu : Le projet est bien associé au service.

D - un chef de service gagne plus que les employés de son service

E - un chef de service gagne plus que les employés responsables de projets

F - Est-il possible de regrouper les deux derniers trigger

Exercice 5 :

A. Trigger alerte_salaire

Trigger :

```
CREATE OR REPLACE TRIGGER alerte_salaire
BEFORE INSERT OR UPDATE OF salaire ON employe
FOR EACH ROW
BEGIN
    IF :NEW.SALAIRE > 5000 THEN
        INSERT INTO employe_alerte
        VALUES (:NEW.NUEMPL, :NEW.NOMEMPL, :NEW.HEBDO, :NEW.AFFECT,
        :NEW.SALAIRE);

    END IF;
END;
```

Ce trigger insère une ligne dans la table employe_alerte lorsque le salaire d'un employé dépasse 5000.

```
INSERT INTO employe (NUEMPL, NOMEMPL, HEBDO, AFFECT, SALAIRE)  
VALUES (50, 'Dupont', 35, 1, 6000);
```

Résultat attendu : Une nouvelle ligne est ajoutée dans la table employe_alerte pour cet employé.