❾ - UNIDEV - CORRECTION

Première partie – Gestion des contraintes sur les tables :

1) Mise à jour automatique d'un attribut calculé.

A. Procédure stockée :

```
CREATE OR REPLACE PROCEDURE AjouterJourneeTravail (
                              p_codeSalarie Travailler.codeSalarie%TYPE,
                                p_codeProjet Travailler.codeProjet%TYPE,
                               p_dateTravail Travailler.dateTravail%TYPE) IS
BEGIN
   INSERT INTO Travailler (codeSalarie, codeProjet, dateTravail)
                    VALUES (p_codeSalarie, p_codeProjet, p_dateTravail);
   UPDATE Salaries
   SET nbTotalJourneesTravail = nbTotalJOurneesTravail + 1
   WHERE codeSalarie = p_codeSalarie;
END;
```

B. Trigger :

```
CREATE OR REPLACE TRIGGER tr_aft_ins_Travailler
AFTER INSERT ON Travailler
FOR EACH ROW
BEGIN
   UPDATE Salaries
   SET nbTotalJourneesTravail = nbTotalJOurneesTravail + 1
   WHERE codeSalarie = :NEW.codeSalarie;
END;
```

2) Gestion des multiplicités maximales des associations.

A. Procédure stockée :

```
CREATE OR REPLACE PROCEDURE AffecterSalarieEquipe (
                             p_codeSalarie EtreAffecte.codeSalarie%TYPE,
                               p_codeEquipe EtreAffecte.codeEquipe%TYPE) IS
   v_nbAffectations NUMBER;
BEGIN
   SELECT COUNT(codeEquipe) INTO v_nbAffectations
   FROM EtreAffecte
   WHERE codeSalarie = p_codeSalarie;
   IF v_nbAffectations < 3 THEN
      INSERT INTO EtreAffecte (codeSalarie, codeEquipe)
                       VALUES (p_codeSalarie, p_codeEquipe);
   ELSE
      RAISE_APPLICATION_ERROR(-20001, 'Le salarié est déjà affecté à au
                                                              moins 3 équipes');
   END IF;
END;
```

B. Trigger :

```
CREATE OR REPLACE TRIGGER tr_bef_ins_EtreAffecte
BEFORE INSERT ON EtreAffecte
FOR EACH ROW
DECLARE
    v_nbAffectations NUMBER;
BEGIN
   SELECT COUNT(codeEquipe) INTO v_nbAffectations
   FROM EtreAffecte
   WHERE codeSalarie = :NEW.codeSalarie;
   IF v_nbAffectations >= 3 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Le salarié est déjà affecté à
                                                          au moins 3 projets');
   END IF;
END ;
```


3) Mise à jour des attributs calculés lors des modifications, insertions et suppressions.

```
CREATE OR REPLACE TRIGGER tr_aft_ins_Travailler
AFTER INSERT OR DELETE OR UPDATE OF codeSalarie ON Travailler
FOR EACH ROW
BEGIN
   IF (INSERTING OR UPDATING) THEN
      UPDATE Salaries
      SET nbTotalJourneesTravail = nbTotalJOurneesTravail + 1
      WHERE codeSalarie = :NEW.codeSalarie;
   END IF;
   IF (DELETING OR UPDATING) THEN
      UPDATE Salaries
      SET nbTotalJourneesTravail = nbTotalJOurneesTravail - 1
      WHERE codeSalarie = :OLD.codeSalarie;
   END IF;
END;
```

4) Contraintes entre plusieurs associations.

```
CREATE OR REPLACE PROCEDURE AjouterJourneeTravail (
                              p_codeSalarie Travailler.codeSalarie%TYPE,
                                p_codeProjet Travailler.codeProjet%TYPE,
                               p_dateTravail Travailler.dateTravail%TYPE) IS
   v_nb NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_nb
   FROM EtreAffecte ea
   JOIN Projets p ON ea.codeEquipe = p.codeEquipe
   WHERE codeSalarie = p_codeSalarie
   AND codeProjet = p_codeProjet;
   IF v_nb = 0 THEN
      RAISE_APPLICATION_ERROR(-20003, 'Un salarié ne peut pas travailler
                                sur un projet qui est réalisé par une équipe
                                     dans laquelle il n''est pas affecté');
   ELSE
      INSERT INTO Travailler (codeSalarie, codeProjet, dateTravail)
                       VALUES (p_codeSalarie, p_codeProjet, p_dateTravail);
      UPDATE Salaries
      SET nbTotalJourneesTravail = nbTotalJOurneesTravail + 1
      WHERE codeSalarie = p_codeSalarie;
   END IF;
END;

OU

CREATE OR REPLACE TRIGGER tr_bef_ins_Travailler
BEFORE INSERT ON Travailler
FOR EACH ROW
DECLARE
   v_nb NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_nb
   FROM EtreAffecte ea
   JOIN Projets p ON ea.codeEquipe = p.codeEquipe
   WHERE codeSalarie = :NEW.codeSalarie
   AND codeProjet = :NEW.codeProjet;
   IF v_nb = 0 THEN
      RAISE_APPLICATION_ERROR(-20003, 'Un salarié ne peut pas travailler
                                sur un projet qui est réalisé par une équipe
                                     dans laquelle il n''est pas affecté');
   END IF;
END;
```

Deuxième partie – Vues et Triggers :

5) Création d'un vue multitables.

```
CREATE OR REPLACE VIEW Affectations AS
SELECT s.codeSalarie, nomSalarie, prenomSalarie, ea.codeEquipe, nomEquipe
FROM Salaries s
JOIN EtreAffecte ea ON s.codeSalarie = ea.codeSalarie
JOIN Equipes e ON e.codeEquipe = ea.codeEquipe;
```

6) Trigger INSTEAD OF

Première version où on ne vérifie pas la cohérence entre le code de l'équipe et le nom de l'équipe. Ni celle entre le code du salarié et son nom et prénom.
On regarde donc uniquement le code salarié et le code équipe. Si ces codes n'existent pas dans les tables *Salaries* et *Equipes*, on crée des lignes dans ces tables. Par contre, si ces codes existent, on ne se soucie pas des valeurs de *nomEquipe*, *nomSalarie* et *prenomSalarie* qui sont indiquées.

```
CREATE OR REPLACE TRIGGER TriggerAuLieuInsererVue
INSTEAD OF INSERT ON Affectations
FOR EACH ROW
DECLARE
   v_nbSalaries NUMBER;
   v_nbEquipes NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_nbEquipes
   FROM Equipes
   WHERE codeEquipe = :NEW.codeEquipe;
   IF v_nbEquipes = 0 THEN
      INSERT INTO Equipes (codeEquipe, nomEquipe, codeSalarieChef)
                   VALUES (:NEW.codeEquipe, :NEW.nomEquipe, NULL);
   END IF;

   SELECT COUNT(*) INTO v_nbSalaries
   FROM Salaries
   WHERE codeSalarie = :NEW.codeSalarie;
   IF v_nbSalaries = 0 THEN
      INSERT INTO Salaries (codeSalarie, nomSalarie, prenomSalarie, nbTotalJourneesTravail)
                   VALUES (:NEW.codeSalarie, :NEW.nomSalarie, :NEW.prenomSalarie, 0);
   END IF;

   INSERT INTO EtreAffecte (codeSalarie, codeEquipe)
                    VALUES (:NEW.codeSalarie, :NEW.codeEquipe);
END;
```