```sql
DELIMITER $$

CREATE PROCEDURE sp_detecter_conflits_emploi_temps(
    IN p_edt_id INT,
    OUT p_nb_conflits INT
)
BEGIN
    DECLARE v_seance_id INT;
    DECLARE v_groupe_id INT;
    DECLARE v_jour VARCHAR(20);
    DECLARE v_heure_debut TIME;
    DECLARE v_heure_fin TIME;
    DECLARE v_salle_id INT;
    DECLARE v_formateur_id INT;
    DECLARE v_done INT DEFAULT FALSE;
    DECLARE v_charge_journaliere FLOAT;

    -- Curseur pour parcourir toutes les séances
    DECLARE cur_seances CURSOR FOR
        SELECT
            s.id,
            s.groupe_id,
            s.jour_semaine,
            s.heure_debut,
            s.heure_fin,
            s.salle_id,
            am.formateur_id
        FROM Seance s
        JOIN SeanceProgramme sp ON s.id = sp.seance_id
        JOIN AffectationModule am ON sp.affectation_module_id = am.id
        WHERE s.emploi_du_temps_id = p_edt_id;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_done = TRUE;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SET p_nb_conflits = -1;
```

```
    END;

    START TRANSACTION;

    -- Supprimer les anciens conflits non résolus de cet EDT
    DELETE c FROM Conflit c
    JOIN Seance s ON c.seance_id = s.id
    WHERE s.emploi_du_temps_id = p_edt_id
    AND c.est_resolu = FALSE;

    SET p_nb_conflits = 0;

    -- Parcourir chaque séance
    OPEN cur_seances;
    read_loop: LOOP
        FETCH cur_seances INTO
            v_seance_id,
            v_groupe_id,
            v_jour,
            v_heure_debut,
            v_heure_fin,
            v_salle_id,
            v_formateur_id;

        IF v_done THEN
            LEAVE read_loop;
        END IF;

        -- Vérifier conflit formateur
        IF fn_verifier_conflit_formateur(
            v_formateur_id, v_jour, v_heure_debut, v_heure_fin, v_seance_id,
p_edt_id
        ) THEN
            INSERT INTO Conflit (seance_id, type_conflit, description, gravite)
            VALUES (v_seance_id, 'formateur',
                    'Le formateur a plusieurs séances simultanées', 'haute');
            SET p_nb_conflits = p_nb_conflits + 1;
```

```
        END IF;

        -- Vérifier conflit salle
        IF v_salle_id IS NOT NULL AND fn_verifier_conflit_salle(
            v_salle_id, v_jour, v_heure_debut, v_heure_fin, v_seance_id, p_edt_id
        ) THEN
            INSERT INTO Conflit (seance_id, type_conflit, description, gravite)
            VALUES (v_seance_id, 'salle',
                    'La salle est utilisée par plusieurs séances', 'haute');
            SET p_nb_conflits = p_nb_conflits + 1;
        END IF;

        -- Vérifier conflit groupe
        IF fn_verifier_conflit_groupe(
            v_groupe_id, v_jour, v_heure_debut, v_heure_fin, v_seance_id,
p_edt_id
        ) THEN
            INSERT INTO Conflit (seance_id, type_conflit, description, gravite)
            VALUES (v_seance_id, 'groupe',
                    'Le groupe a plusieurs séances simultanées', 'haute');
            SET p_nb_conflits = p_nb_conflits + 1;
        END IF;

        SET v_charge_journaliere = fn_calculer_charge_journaliere_groupe(
            v_groupe_id, v_jour, p_edt_id
        );

        IF v_charge_journaliere > 7.5 THEN
            INSERT INTO Conflit (seance_id, type_conflit, description, gravite)
            VALUES (v_seance_id, 'charge_excessive',
                    CONCAT('Charge journalière excessive: ',
v_charge_journaliere, 'h'), 'moyenne');
            SET p_nb_conflits = p_nb_conflits + 1;
        END IF;
    END LOOP;
    CLOSE cur_seances;
```

```
    COMMIT;
END$$

DELIMITER ;
```