Fiches PSI

I. SQL de base

1. DROP

```
DROP TABLE Drh    CASCADE CONSTRAINTS;
```

2. CREATE

```
CREATE TABLE Serv (
    numServ     numBER(2)     NOT NULL,
    nomServ VARCHAR2(14),
    lieu        VARCHAR2(8),
    CONSTRAINT PK_Serv PRIMARY KEY (numServ)
);
```

3. INSERT

```
INSERT INTO Serv VALUES
    (1,'INFORMATIQUE','BAT 688');
```

4. ALTER TABLE

a. Ajout contrainte

```
ALTER TABLE Drh ADD (
    CONSTRAINT FK_Drh_Drh
        FOREIGN KEY (matChef)
        REFERENCES Drh (mat)
);
```

b. Ajout champ

```
ALTER TABLE Serv
ADD (NbPers NUMBER, NbForm NUMBER);
```

5. UPDATE

```
UPDATE Serv
SET Budget=MASSESALARIALE(numServ);
```

II. Fonctions

1. AFTER UPDATE OR DELETE OR INSER

```
CREATE OR REPLACE TRIGGER t_budget
AFTER UPDATE OR DELETE OR INSERT
ON Drh
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        UPDATE Serv
        SET Budget=Budget + :new.Sal + NVL(:new.Vac,0)
        WHERE numServ=:new.numServ;
    END IF;

    IF DELETING THEN
        UPDATE Serv
        SET Budget=Budget - :old.Sal - NVL(:old.Vac,0)
        WHERE numServ=:old.numServ;
    END IF;

    IF UPDATING THEN
        IF :new.numServ != :old.numServ THEN
          UPDATE Serv
          SET Budget=Budget - :old.Sal - NVL(:old.Vac,0)
          WHERE numServ=:old.numServ;

          UPDATE Serv
          SET Budget=Budget + :new.Sal + NVL(:new.Vac,0)
          WHERE numServ=:new.numServ;
        ELSE
          UPDATE Serv
          SET Budget=Budget + :new.Sal + NVL(:new.Vac,0)
          WHERE numServ=:new.numServ;
        END IF;
    END IF;
END;
```

```
CREATE OR REPLACE TRIGGER tr_up_nb
AFTER UPDATE OR DELETE OR INSERT
ON Drh
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    UPDATE Serv
    SET NbPers=NbPers+1;

    IF :new.emploi ='FORMATEUR' THEN
      UPDATE Serv
      SET NbForm=NbForm+1 WHERE numServ=:new.numServ;
    END IF;
  END IF;

  IF DELETING THEN
    UPDATE Serv
    SET NbPers=NbPers-1;

    IF :old.emploi ='FORMATEUR' THEN
      UPDATE Serv
      SET NbForm=NbForm-1
      WHERE numServ=:old.numServ;
    END IF;
  END IF;

  IF UPDATING THEN
    IF :new.emploi ='FORMATEUR' THEN
      UPDATE Serv
      SET NbForm=NbForm+1
      WHERE numServ=:new.numServ;
    END IF;

    IF :old.emploi ='FORMATEUR' THEN
      UPDATE Serv
      SET NbForm=NbForm-1
      WHERE numServ=:old.numServ;
    END IF;
  END IF;
END;
```

---

2. COUNT

```
CREATE OR REPLACE FUNCTION NbEmploye (numService IN NUMBER)
RETURN NUMBER
AS
    nbEmploye NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO nbEmploye
    FROM Drh
    WHERE numServ=numService;
    RETURN nbEmploye;
END;
```

III. Procédure

```
CREATE OR REPLACE
PROCEDURE up_nb
AS
    CURSOR c_serv IS SELECT DISTINCT numServ FROM Serv;
BEGIN
    FOR v_serv in c_serv
    LOOP
        UPDATE Serv
        SET NbPers=NbEmploye(v_serv.numServ),
        NbForm=NbFormateur(v_serv.numServ)
        WHERE numServ=v_serv.numServ;
    END LOOP;
END;
```

IV. TRIGGER

```
CREATE OR REPLACE
TRIGGER t_budget2
BEFORE UPDATE OR INSERT
ON Drh
FOR EACH ROW
BEGIN
    IF :old.emploi = 'FORMATEUR' THEN
        raise_application_error (-20001,'Un formateur a droit à une vacation !');
    END IF;
END;

CREATE OR REPLACE TRIGGER t_budget3
BEFORE UPDATE OR INSERT
ON Drh
FOR EACH ROW
BEGIN
    IF :new.matchef = :old.mat THEN
        raise_application_error (-20004,'Un salarié ne doit pas être son propre chef !');
    END IF;
END;

CREATE OR REPLACE TRIGGER t_budget4
BEFORE UPDATE OF dateEmb
ON Drh
FOR EACH ROW
BEGIN
    raise_application_error (-20005,'La date dembauche dun salarié est non modifiable.');
END;

CREATE OR REPLACE TRIGGER t_budget5
BEFORE UPDATE OF Sal
ON Drh
FOR EACH ROW
BEGIN
    IF (:new.Sal+NVL(:new.vac,0)) < (:old.Sal+NVL(:old.vac,0)) THEN
        raise_application_error (-20003,'Le salaire dun employé ne doit jamais diminuer !');
    END IF;
END;
```