Avancement du projet Base de donnée Semaine 01 G1B -5

Yann Guillevic, Léo Douville, Anthony Cabrillac, Victor Thompson, Shiyu Hu

1 - Procédure stockée
2 - Triggers
3 - Package + fonction

1 - Procédure stockée :

```
create or replace PROCEDURE CreerCompte(
        vNomc Client.nomc%TYPE,
        vPrenomC Client.prenomc%TYPE,
        vAdressemailC Client.adressemail%TYPE,
    Vmdp Client.motdepasse%TYPE

        )
IS

BEGIN

                INSERT INTO Client
(idclient,nomc,prenomc,adressemail,ptsfidelite,adressec,telportablec,motdepa
sse)
                VALUES (idClient.nextVal, vNomc, vPrenomC,
vAdressemailC,0,null,null,Vmdp);


                COMMIT;

END;
```

2 - Triggers :

Mise à jour des stocks de bonbon :

```
create or replace TRIGGER maj_stock_bonbons
AFTER INSERT OR DELETE OR UPDATE OF quantitekg, idb
ON CONTIENTBONBON
FOR EACH ROW
DECLARE
   catBonbon BONBONS.idcategorie%TYPE;
BEGIN
IF (INSERTING OR UPDATING) THEN
   SELECT B.idcategorie INTO catBonbon
   FROM BONBONS B, CATEGORIEB C
   WHERE  B.idcategorie = C.idcategorie
   AND B.idb = :NEW.idb;
   UPDATE CATEGORIEB
   SET stockb = stockb - :NEW.quantitekg
   WHERE idcategorie = catBonbon;
END IF;
IF (DELETING OR UPDATING) THEN
   SELECT B.idcategorie INTO catBonbon
   FROM BONBONS B, CATEGORIEB C
   WHERE B.idcategorie = C.idcategorie
   AND B.idb = :OLD.idb;
   UPDATE CATEGORIEB
   SET stockb = stockb + :OLD.quantitekg
   WHERE idcategorie = catBonbon;
END IF;
END;
```

Mise à jour des stocks de récipient :

```
create or replace TRIGGER maj_stock_recipient
BEFORE INSERT OR DELETE OR UPDATE OF quantiteunitaire, IDR
ON CONTIENTRECIPIENT
FOR EACH ROW
DECLARE
   idRecipient RECIPIENT.IDR%TYPE;
BEGIN
IF (INSERTING OR UPDATING) THEN
   SELECT IDR INTO idRecipient
   FROM RECIPIENT
   WHERE IDR = :NEW.IDR;
   UPDATE RECIPIENT
   SET stock = stock - :NEW.quantiteunitaire
   WHERE IDR = idRecipient;
END IF;
IF (DELETING OR UPDATING) THEN
   SELECT idr INTO idRecipient
   FROM RECIPIENT
   WHERE idr = :OLD.idr;
   UPDATE RECIPIENT
   SET stock = stock + :OLD.quantiteunitaire
   WHERE idr = idRecipient;
END IF;
END;
```

Vérification de la quantité choisie de bonbon lors d'une commande :

```
create or replace TRIGGER verif_qte_bonbon_commande
BEFORE INSERT OR UPDATE OF quantitekg, idb
ON CONTIENTBONBON
FOR EACH ROW
DECLARE
   qteBonbon CATEGORIEB.STOCKB%TYPE;
BEGIN
   IF (INSERTING OR UPDATING) THEN
      SELECT C.STOCKB INTO qteBonbon
      FROM CATEGORIEB C, BONBONS B
      WHERE B.idcategorie = C.idcategorie
      AND B.idb = :NEW.idb;
      IF (:NEW.quantitekg > qteBonbon) THEN
         RAISE_APPLICATION_ERROR(-20000, 'Erreur: Votre quantitée
choisie dépasse les stocks !');
      END IF;
   END IF;
END;
```

Vérification de la quantité choisie de récipient lors d'une commande :

```
create or replace TRIGGER verif_qte_recipient_commande
BEFORE INSERT OR UPDATE OF quantiteunitaire, idr
ON CONTIENTRECIPIENT
FOR EACH ROW
DECLARE
   qteRecipient RECIPIENT.STOCK%TYPE;
BEGIN
   IF (INSERTING OR UPDATING) THEN
      SELECT STOCK INTO qteRecipient
      FROM RECIPIENT
      WHERE IDR = :NEW.IDR;
      IF (:NEW.quantiteunitaire > qteRecipient) THEN
         RAISE_APPLICATION_ERROR(-20000, 'Erreur: Votre quantitée
choisie dépasse les stocks !');
      END IF;
```

```
   END IF;
END;
```

3 - Package + fonction :

```
create or replace PACKAGE PrixTotalCommande AS
        FUNCTION PrixR   (r Commande.idcommande%TYPE) RETURN
DECIMAL;
        FUNCTION PrixB (c Commande.idcommande%TYPE) RETURN
DECIMAL;
   PROCEDURE PrixTotal(p Commande.idcommande%TYPE) ;
END PrixTotalCommande;


create or replace PACKAGE BODY Prixtotalcommande AS
        FUNCTION PrixR(r Commande.idcommande%TYPE) RETURN
DECIMAL IS
   qr contientrecipient.quantiteunitaire%TYPE;
   pr recipient.prixunitaire%TYPE;
        prixR decimal;
   CURSOR c1 IS SELECT cr.quantiteunitaire, r.prixunitaire FROM
contientrecipient cr, recipient r where cr.idr = r.idr and cr.idcommande = r;
        BEGIN
   prixR:=0;
   OPEN c1;
   FETCH c1 INTO qr, pr;
   WHILE c1%FOUND LOOP
   prixR:= prixR + qr * pr;
   FETCH c1 INTO qr,pr;
   end loop;
   RETURN prixR;
        END PrixR;


   FUNCTION PrixB (c Commande.idcommande%TYPE) RETURN DECIMAL
IS
   qb contientbonbon.quantitekg%TYPE;
```

```
    pb bonbons.prixkg%TYPE;
        prixB decimal;
    CURSOR c1 IS SELECT cb.quantitekg, b.prixkg FROM contientbonbon cb,
bonbons b where cb.idb = b.idb and cb.idcommande = c;
        BEGIN
    prixB:=0;
    OPEN c1;
    FETCH c1 INTO qb, pb;
    WHILE c1%FOUND LOOP
    prixB:= prixB + qb * pb;
    FETCH c1 INTO qb, pb;
    end loop;
    RETURN prixB;
        END PrixB;


    PROCEDURE PrixTotal(p Commande.idcommande%TYPE) IS
    a1 decimal;
    a2 decimal;
    prixTotal decimal;
    BEGIN
    a1 := PrixB(p);
    a2 := PrixR(p);
    prixTotal := a1 + a2;
    DBMS_OUTPUT.PUT_LINE(prixTotal);
    END PrixTotal;


END ;
```