# Compte Rendu - TP3 PSQL

## Exercice 1 : Création des Triggers

### Objectif

L'objectif de cet exercice était de créer des triggers pour automatiser certaines règles métier dans la base de données.

### Triggers Créés

1. **`trg_product_prodid`** : Génère automatiquement un `PRODID` si celui-ci n'est pas spécifié lors de l'insertion dans la table `PRODUCT`.
2. **`trg_item_itemtot`** : Calcule automatiquement la valeur totale (`ITEMTOT`) d'un article dans la table `ITEM` en fonction du prix unitaire (`ACTUALPRICE`) et de la quantité (`QTY`).
3. **`trg_customer_repid`** : Vérifie que le représentant (`REPID`) associé à un client est un employé ayant le rôle de `SALESMAN`. Sinon, une erreur est levée.
4. **`trg_price_dates`** : Vérifie que la date de début (`STARTDATE`) est antérieure à la date de fin (`ENDDATE`) dans la table `PRICE`. Sinon, une erreur est levée.
5. **`trg_item_price_check`** : Vérifie que le prix appliqué (`ACTUALPRICE`) d'un article est supérieur ou égal au prix minimum (`MINPRICE`) valide à la date de la commande. Sinon, une erreur est levée.

---

## Exercice 2 : Tests des Triggers

### Objectif

L'objectif de cet exercice était de tester les triggers créés dans l'exercice 1 pour s'assurer qu'ils fonctionnent correctement.

### Résultats des Tests

1. **Test du Trigger `trg_product_prodid`** :
   - **Test 1.1** : Insertion sans spécifier `PRODID` → Un `PRODID` est généré automatiquement.
   - **Test 1.2** : Insertion avec un `PRODID` spécifié → Le `PRODID` fourni est utilisé.
2. **Test du Trigger `trg_item_itemtot`** :
   - **Test 2.1** : Insertion d'un nouvel article → Le total (`ITEMTOT`) est calculé correctement.
   - **Test 2.2** : Mise à jour de la quantité (`QTY`) → Le total est recalculé.
   - **Test 2.3** : Mise à jour du prix unitaire (`ACTUALPRICE`) → Le total est recalculé.
3. **Test du Trigger `trg_customer_repid`** :
   - **Test 3.1** : Insertion avec un `REPID` valide (`SALESMAN`) → L'insertion réussit.
   - **Test 3.2** : Insertion avec un `REPID` non valide → Une erreur est levée.
   - **Test 3.3** : Mise à jour avec un `REPID` non valide → Une erreur est levée.

4. **Test du Trigger** `trg_price_dates` :
    - **Test 4.1** : Insertion avec des dates valides → L'insertion réussit.
    - **Test 4.2** : Insertion avec une date de fin (`ENDDATE`) nulle → L'insertion réussit.
    - **Test 4.3** : Insertion avec des dates invalides (`STARTDATE > ENDDATE`) → Une erreur est levée.
    - **Test 4.4** : Mise à jour rendant les dates invalides → Une erreur est levée.
5. **Test du Trigger** `trg_item_price_check` :
    - **Test 5.1** : Insertion avec un prix valide (≥ `MINPRICE`) → L'insertion réussit.
    - **Test 5.2** : Insertion avec un prix trop bas (< `MINPRICE`) → Une erreur est levée.
    - **Test 5.3** : Mise à jour rendant le prix invalide → Une erreur est levée.
    - **Test 5.4** : Gestion de plusieurs prix selon la période → Les règles sont respectées en fonction des périodes définies.

---

## Détails des Tests

### **Trigger** `trg_product_prodid`

- **Test 1.1** :

```
INSERT INTO PRODUCT (DESCRIP) VALUES ('NOUVELLE RAQUETTE DE TENNIS');
SELECT * FROM PRODUCT WHERE DESCRIP = 'NOUVELLE RAQUETTE DE TENNIS';
```

- **Test 1.2** :

```
INSERT INTO PRODUCT (PRODID, DESCRIP) VALUES (999999, 'RAQUETTE TEST');
SELECT * FROM PRODUCT WHERE PRODID = 999999;
```

### **Trigger** `trg_item_itemtot`

- **Test 2.1** :

```
INSERT INTO ITEM (ORDID, ITEMID, PRODID, ACTUALPRICE, QTY) VALUES (610, 4,
'100890', 58, 2);
SELECT ITEMTOT FROM ITEM WHERE ORDID = 610 AND ITEMID = 4;
```

- **Test 2.2** :

```
UPDATE ITEM SET QTY = 3 WHERE ORDID = 610 AND ITEMID = 4;
SELECT ITEMTOT FROM ITEM WHERE ORDID = 610 AND ITEMID = 4;
```

- **Test 2.3** :

```
UPDATE ITEM SET ACTUALPRICE = 60 WHERE ORDID = 610 AND ITEMID = 4;
SELECT ITEMTOT FROM ITEM WHERE ORDID = 610 AND ITEMID = 4;
```

**Trigger** `trg_customer_repid`

- **Test 3.1** :

```
INSERT INTO CUSTOMER (CUSTID, NAME, REPID) VALUES (999, 'TEST CLIENT',
7499);
```

- **Test 3.2** :

```
BEGIN
    INSERT INTO CUSTOMER (CUSTID, NAME, REPID) VALUES (998, 'TEST CLIENT
INVALIDE', 7369);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur attendue: ' || SQLERRM);
END;
```

- **Test 3.3** :

```
BEGIN
    UPDATE CUSTOMER SET REPID = 7902 WHERE CUSTID = 999;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur attendue: ' || SQLERRM);
END;
```

**Trigger** `trg_price_dates`

- **Test 4.1** :

```
INSERT INTO PRICE (PRODID, STDPRICE, MINPRICE, STARTDATE, ENDDATE)
VALUES ('100860', 40, 32, TO_DATE('01-JAN-2023', 'DD-MON-YYYY'),
TO_DATE('31-DEC-2023', 'DD-MON-YYYY'));
```

- **Test 4.2** :

```
INSERT INTO PRICE (PRODID, STDPRICE, MINPRICE, STARTDATE)
VALUES ('100860', 42, 34, TO_DATE('01-JAN-2024', 'DD-MON-YYYY'));
```

- **Test 4.3** :

```
BEGIN
    INSERT INTO PRICE (PRODID, STDPRICE, MINPRICE, STARTDATE, ENDDATE)
    VALUES ('100860', 38, 30, TO_DATE('01-JAN-2025', 'DD-MON-YYYY'),
TO_DATE('31-DEC-2024', 'DD-MON-YYYY'));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur attendue: ' || SQLERRM);
END;
```

- **Test 4.4** :

```
BEGIN
    UPDATE PRICE
    SET ENDDATE = TO_DATE('31-DEC-2022', 'DD-MON-YYYY')
    WHERE PRODID = '100860' AND STARTDATE = TO_DATE('01-JAN-2023', 'DD-MON-
YYYY');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur attendue: ' || SQLERRM);
END;
```

**Trigger** `trg_item_price_check`

- **Test 5.1** :

```
INSERT INTO ITEM (ORDID, ITEMID, PRODID, ACTUALPRICE, QTY)
VALUES (999, 1, '200381', 45, 1);
```

- **Test 5.2** :

```
BEGIN
    INSERT INTO ITEM (ORDID, ITEMID, PRODID, ACTUALPRICE, QTY)
    VALUES (999, 2, '200381', 35, 1);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur attendue: ' || SQLERRM);
END;
```

- **Test 5.3** :

```
BEGIN
    UPDATE ITEM SET ACTUALPRICE = 38 WHERE ORDID = 999 AND ITEMID = 1;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur attendue: ' || SQLERRM);
END;
```

- **Test 5.4** :

```
INSERT INTO PRICE (PRODID, STDPRICE, MINPRICE, STARTDATE, ENDDATE)
VALUES ('200381', 60, 50, TO_DATE('01-FEB-2023', 'DD-MON-YYYY'),
TO_DATE('28-FEB-2023', 'DD-MON-YYYY'));
INSERT INTO ORD (ORDID, ORDERDATE, CUSTID)
VALUES (998, TO_DATE('15-FEB-2023', 'DD-MON-YYYY'), 100);
BEGIN
    INSERT INTO ITEM (ORDID, ITEMID, PRODID, ACTUALPRICE, QTY)
    VALUES (998, 1, '200381', 45, 1);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur attendue: ' || SQLERRM);
END;
INSERT INTO ITEM (ORDID, ITEMID, PRODID, ACTUALPRICE, QTY)
VALUES (998, 1, '200381', 55, 1);
```

---

## Conclusion

Les triggers créés dans l'exercice 1 fonctionnent comme prévu et respectent les règles métier définies. Les tests réalisés dans l'exercice 2 ont permis de valider leur bon fonctionnement dans différents scénarios.