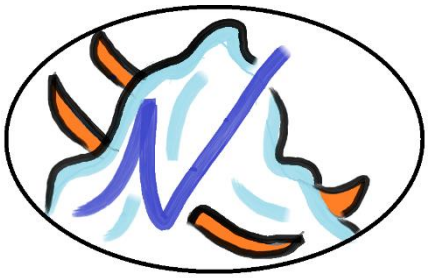
SAE S3.01 - Développement d'une application

Groupe 1B-6 Hai-Son DANG, Enzo MANCINI, Marwan YOUNMI, Esteban BIRET-TOSCANO

Création des triggers et procédures stockées :



Sommaire :

1- Création des 2 procédures stockées : 2

2- Création des 3 triggers : 7

1- Création des 2 procédures stockées :

La 1ère procédure stockée est AjoutArticle, qui permet d'augmenter la quantité en stock d'un article, en précisant en paramètre l'id de l'article et la quantité à ajouter. Le fonctionnement de cette procédure est simple, on met simplement à jour l'attribut qteStock de la table Article, en ajoutant la valeur fournie en paramètre. Ex, l'article 1 se trouve en 25 exemplaires dans la BD, après appel de la procédure en fournissant 1 et 5 comme paramètres, l'article 1 se trouvera en **30** exemplaires. Pour la gestion des erreurs & exceptions, j'utilise NO_DATA_FOUND quand l'idArticle fourni ne correspond à aucun article dans la BD, et je vérifie que la valeur de la quantité fournie par l'utilisateur est supérieure à 1. Sinon, un message dans la console informe que la quantité est incorrect, et donc la mise à jour ne se fait pas dans la BD. S'il n'y a aucune erreur, un message dans la console informe l'utilisateur que tout s'est bien passé. Plus tard, j'utiliserai **RAISE APPLICATION ERROR** pour faire remonter l'erreur au niveau PHP, l'affichage console est ici juste pour vérifier que tout fonctionne correctement.

```
CREATE OR REPLACE PROCEDURE AjoutArticle
(
p_id Article.idArticle%TYPE,
p_stock Article.qteStock%TYPE
)
IS

v_idArticle Article.idArticle%TYPE;
erreur_stock EXCEPTION;

BEGIN

IF (p_stock < 1) THEN
    RAISE erreur_stock;
END IF;

SELECT idArticle INTO v_idArticle FROM Article
WHERE idArticle = p_id;

UPDATE ARTICLE
SET qteStock = qteStock + p_stock
WHERE idArticle = p_id;

COMMIT;

DBMS_OUTPUT.PUT_LINE ('Article.s inséré.s !');

EXCEPTION
WHEN erreur_stock THEN
    DBMS_OUTPUT.PUT_LINE ('Erreur, vous devez rentrez une valeur entière > 1 pour la quantité !');


WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('Article inconnu !');
END;
/
```

Test de la procédure :

	IDARTICLE	IDCATEGORIE	QTESTOCK	PRIXARTICLE	NOMARTICLE
1	1	1	104	45000000000	Beluga
2	2	1	7	47000000000	BelugaXL
3	3	2	20	31000000000	Thon
4	4	2	15	32000000000	Thon VR
5	5	2	11	27000000000	Saumon
6	6	2	7	26000000000	Blue-J
7	7	3	4	47000000000	Orque
8	8	3	2	48000000000	Orque XL
9	9	3	7	25000000000	Blue-J
10	10	4	26	16000000000	Dauphin
11	11	4	3	32000000000	Otarie
12	12	5	14	67000000000	Calmar
13	13	5	8	65000000000	Cachalot
14	14	6	6	54000000000	Espadon
15	15	6	10	27000000000	Saumon

On voit ici que l'article 'Beluga' (idArticle = 1), est en 104 exemplaires. Utilisons la procédure stockée pour rajouter 20 exemplaires :

```
SET SERVEROUTPUT ON;
EXEC AjoutArticle(1, 20);

Article.s inséré.s !

Procédure PL/SQL terminée.
```

On voit donc que l'article est désormais présent en 124 exemplaires.

	IDARTICLE	IDCATEGORIE	QTESTOCK	PRIXARTICLE	NOMARTICLE
1	1	1	124	45000000000	Beluga
2	2	1	7	47000000000	BelugaXL
3	3	2	20	31000000000	Thon
4	4	2	15	32000000000	Thon VR
5	5	2	11	27000000000	Saumon
6	6	2	7	26000000000	Blue-J
7	7	3	4	47000000000	Orque
8	8	3	2	48000000000	Orque XL
9	9	3	7	25000000000	Blue-J
10	10	4	26	16000000000	Dauphin
11	11	4	3	32000000000	Otarie
12	12	5	14	67000000000	Calmar
13	13	5	8	65000000000	Cachalot
14	14	6	6	54000000000	Espadon
15	15	6	10	27000000000	Saumon

Maintenant, si l'on met une valeur incohérente :

```
SET SERVEROUTPUT ON;
EXEC AjoutArticle(1, -20);

Erreur, vous devez rentrez une valeur entière > 1 pour la quantité !


Procédure PL/SQL terminée.
```

Enfin, si l'on renseigne l'id d'un article inexistant :

```
SET SERVEROUTPUT ON;
EXEC AjoutArticle(71, 20);

Article inconnu !


Procédure PL/SQL terminée.
```

La 2nde procédure stockée est RetraitArticle, qui permet de réduire la quantité en stock d'un article, en précisant en paramètre l'id de l'article et la quantité à retirer. Cette procédure fonctionne comme AjoutArticle, sauf que lors de l'UPDATE, on met **'–'** au lieu de **'+'**. La gestion d'erreurs et d'exceptions est également la même. Néanmoins, une erreur n'est pas traitée ici (lors du retrait d'article, la quantité en stock ne peut pas descendre en dessous de 0). Cette erreur est traitée grâce à un déclencheur, que nous allons voir juste après.

```
CREATE OR REPLACE PROCEDURE RetraitArticle
(
p_id Article.idArticle%TYPE,
p_stock Article.qteStock%TYPE
)
IS

v_idArticle Article.idArticle%TYPE;
erreur_stock EXCEPTION;

BEGIN

IF (p_stock < 1) THEN
    RAISE erreur_stock;
END IF;

SELECT idArticle INTO v_idArticle FROM Article
WHERE idArticle = p_id;

UPDATE ARTICLE
SET qteStock = qteStock - p_stock
WHERE idArticle = p_id;

COMMIT;

DBMS_OUTPUT.PUT_LINE ('Article.s retiré.s !');

EXCEPTION
WHEN erreur_stock THEN
    DBMS_OUTPUT.PUT_LINE ('Erreur, vous devez rentrez une valeur entière > 1 pour la quantité !');

WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('Article inconnu !');
END;
/
```

Test de la procédure :

	IDARTICLE	IDCATEGORIE	QTESTOCK	PRIXARTICLE	NOMARTICLE
1	1	1	124	45000000000	Beluga
2	2	1	7	47000000000	BelugaXL
3	3	2	20	31000000000	Thon
4	4	2	15	32000000000	Thon VR
5	5	2	11	27000000000	Saumon
6	6	2	7	26000000000	Blue-J
7	7	3	4	47000000000	Orque
8	8	3	2	48000000000	Orque XL
9	9	3	7	25000000000	Blue-J
10	10	4	26	16000000000	Dauphin
11	11	4	3	32000000000	Otarie
12	12	5	14	67000000000	Calmar
13	13	5	8	65000000000	Cachalot
14	14	6	6	54000000000	Espadon
15	15	6	10	27000000000	Saumon

On voit ici que l'article 'Beluga' (idArticle = 1), est en 124 exemplaires. Utilisons la procédure stockée pour retirer 3 exemplaires :

```
SET SERVEROUTPUT ON;
EXEC RetraitArticle(1, 3);
```

```
Article.s retiré.s !


Procédure PL/SQL terminée.
```

	IDARTICLE	IDCATEGORIE	QTESTOCK	PRIXARTICLE	NOMARTICLE
1	1	1	121	45000000000	Beluga
2	2	1	7	47000000000	BelugaXL
3	3	2	20	31000000000	Thon
4	4	2	15	32000000000	Thon VR
5	5	2	11	27000000000	Saumon
6	6	2	7	26000000000	Blue-J
7	7	3	4	47000000000	Orque
8	8	3	2	48000000000	Orque XL
9	9	3	7	25000000000	Blue-J
10	10	4	26	16000000000	Dauphin
11	11	4	3	32000000000	Otarie
12	12	5	14	67000000000	Calmar
13	13	5	8	65000000000	Cachalot
14	14	6	6	54000000000	Espadon
15	15	6	10	27000000000	Saumon

On voit donc que l'article est désormais présent en 121 exemplaires.

Maintenant, si l'on met une valeur incohérente :

```
SET SERVEROUTPUT ON;
EXEC RetraitArticle(1, -3);
```

```
Erreur, vous devez rentrez une valeur entière > 1 pour la quantité !


Procédure PL/SQL terminée.
```

Enfin, si l'on renseigne l'id d'un article inexistant :

```
SET SERVEROUTPUT ON;
EXEC RetraitArticle(145, 3);
```

```
Article inconnu !


Procédure PL/SQL terminée.
```

2- Création des 3 triggers :

Le 1er trigger est un trigger permettant de vérifier que l'utilisateur puisse bien ajouter un article dans son panier, en fonction de la quantité voulue et du stock de l'article. C'est donc un trigger de type BEFORE INSERT, portant sur la table Panier.

```
CREATE OR REPLACE TRIGGER t_b_i_Panier
BEFORE INSERT ON Panier
FOR EACH ROW
DECLARE
    v_qteStock Article.qteStock%TYPE;

BEGIN

SELECT qtestock INTO v_qteStock FROM Article
WHERE idArticle = :NEW.idArticle;

IF v_qtestock < :NEW.qtcom THEN
    RAISE_APPLICATION_ERROR(-20001,'Erreur, pas assez de stock !');
END IF;

END;
/
```

Le 2nd trigger permet de mettre à jour la quantité des articles (ou de l'article) après qu'un utilisateur ai passé une commande. C'est donc un trigger de type AFTER INSERT, portant sur la table Commande.

```
CREATE OR REPLACE TRIGGER t_a_i_Commande
AFTER INSERT ON Commande
FOR EACH ROW
DECLARE
    v_qteCommande Panier.qteCom%TYPE;

BEGIN

SELECT qtcom INTO v_qteCommande FROM Panier p

UPDATE Article a
SET a.qteStock = a.qteStock - p.qteArticle
WHERE p.idPanier = :NEW.idPanier
AND a.idArticle = p.idArticle;
END;
/
```

Enfin, le dernier trigger s'occupe de gérer l'erreur expliquée dans la 2nde procédure stockée. C'est donc un trigger de type BEFORE UPDATE, portant sur la table Article.

```
CREATE OR REPLACE TRIGGER t_a_u Retrait
BEFORE UPDATE OF qteStock
ON Article
FOR EACH ROW
DECLARE
    v_qteStock Article.qteStock%TYPE;Q

BEGIN

    IF v_qtestock < 0
        RAISE_APPLICATION_ERROR(-20001,'Erreur, le stock serait en négatif après ce retrait !');
    END IF;
END;
/
```