

SAE S3.01 - Développement d'une application

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

Semaine 2 :



Sommaire :

1- Création des 2 procédures stockées :	2
2- Triggers :	7

1- Création des 2 procédures stockées :

La 1^{ère} 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 rentrer 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.

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

Maintenant, si l'on met une valeur incohérente :

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

Erreur, vous devez rentrer 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	450000000000	Beluga
2	2	1	7	470000000000	BelugaXL
3	3	2	20	310000000000	Thon
4	4	2	15	320000000000	Thon VR
5	5	2	11	270000000000	Saumon
6	6	2	7	260000000000	Blue-J
7	7	3	4	470000000000	Orque
8	8	3	2	480000000000	Orque XL
9	9	3	7	250000000000	Blue-J
10	10	4	26	160000000000	Dauphin
11	11	4	3	320000000000	Otarie
12	12	5	14	670000000000	Calmar
13	13	5	8	650000000000	Cachalot
14	14	6	6	540000000000	Espadon
15	15	6	10	270000000000	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 rentrer 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- Triggers :

Pour ce qui est des triggers, nous n'avons pas eu besoin d'en créer, car tous nos tests se font directement en PHP (article en stock, mise à jour des infos du panier après update / insert / delete).

Pour ce qui est du stock des articles :

```
//variables

$nbarticle = $_POST['nbarticle'];

$idclient = $_SESSION['idclient'];

$idarticle = htmlentities($_GET['idarticle']);

$qteenstock = htmlentities($_GET['qtestock']);

$prixunitaire = htmlentities($_GET['prix']);

if($nbarticle > $qteenstock){
    echo '<script language="JavaScript" type="text/javascript">
        alert(" Pas assez de stock !");
        location.href = "detailArticle.php?idArticle='.$idarticle.'";
    </script>';
    exit();
}
```

Après avoir ajouté un article dans notre panier avec sa quantité, nous sommes redirigés vers la page traitPanier.php, qui s'occupe de réaliser tous les test, UPDATE et INSERT.

Ici, on récupère la quantité en stock de l'article en question et la quantité voulue par le client. On regarde si cela est possible, et si non on redirige le client vers la page detailProduit de l'article en question, avec un pop-up JS qui lui indique l'impossibilité.

Nous avons plus bas dans le code un test permettant de vérifier si l'article en question a assez de stock, en fonction de la quantité choisie par le client **et de la quantité de cet article déjà présent dans son panier.**

```
// requête vérifiant si l'article ajouté est déjà dans le panier
$reqdejapanier = "SELECT * FROM DETAILPANIER WHERE idPanier = :pID_PANIER AND idArticle = :pID_ARTICLE";

$dejapanier = oci_parse($connect, $reqdejapanier);

oci_bind_by_name($dejapanier, ":pID_PANIER", $idclient);

oci_bind_by_name($dejapanier, ":pID_ARTICLE", $idarticle);

// exécution de la requête
oci_execute($dejapanier);

// si l'article ajouté est déjà dans le panier, on modifie sa quantité
if (($test = oci_fetch_assoc($dejapanier)) != false) {

    if (($test['NBARTICLE'] + $nbarticle) > $qteenstock) {
        echo '<script language="JavaScript" type="text/javascript">
            alert(" Pas assez de stock ! (cette quantité plus celle d\' dans votre panier d\' passe le stock actuel pour cet article) ");
            location.href = "detailArticle.php?idArticle='.$idarticle.'";
        </script>';
        exit();
    }
}
```

En effet, la requête 'dejapanier' vérifie si le client n'a pas déjà l'article en question dans son panier. Cela permet de modifier la quantité de l'article au lieu d'insérer une nouvelle ligne dans la BD.

S'il a déjà l'article, on prend la quantité du panier et celle que le client veut commander, on additionne les 2 et on regarde ensuite si cela est possible, en fonction du stock de l'article.

Pour ce qui est du prix total du panier, nous faisons également la modification après chaque UPDATE / INSERT et DELETE dans un fichier PHP, cette fois ci le fichier Panier.php.

```
<?php

$detailpanier = "SELECT * FROM DETAILPANIER WHERE IDPANIER = '". $_GET['idclient']."'";

$lepanier = oci_parse($connect, $detailpanier);

oci_execute($lepanier);

$totalPanier = 0;

while (($spanierprixtotal = oci_fetch_assoc($lepanier)) != false) {
    $totalPanier = $totalPanier + ($spanierprixtotal['NBARTICLE'] * $spanierprixtotal['PRIXTTEARTICLE']);
}

//modifier le prix du panier dans la table panier

$reqmodifprixpanier = "UPDATE PANIER SET PRIXPANIER = :pPRIX_PANIER WHERE IDPANIER = '". $_GET['idclient']."'";

$modifprixpanier = oci_parse($connect, $reqmodifprixpanier);

oci_bind_by_name($modifprixpanier, ":pPRIX_PANIER", $totalPanier);

oci_execute($modifprixpanier);

//commit
oci_commit($connect);
```

Ici, on exécute une requête prenant tous les attributs de chaque article du panier du client. On crée une variable 'totalPanier', qui sera donc le prix total du panier. Ce prix est calculé en fonction de chaque article du panier, de sa quantité et de sa quantité. On modifie ensuite le prix du panier coté BD, avec la 2^{de} requête. On n'oublie pas de COMMIT.