À la suite de mon premier cas de test avec PL/SQL d’Oracle, j’ai souhaité mettre en place, un cas de test plus concret sur une fonction couplé avec la base de données, plus proche du quotidien d’un développeur SQL Oracle.

Ce document a pour but d’expliquer les différentes étapes de la mise en place de tests automatisés.

## Description du comportement de la fonction

La fonction que je vais tester a pour but de rechercher le numéro de cumul de lot d’un marché en fonction de l’identifiant du marché composé de quatre valeurs (identifiant société, exercice du marché, numéro de marché et numéro de lot du marché) et conditionné au fait que le cumul ait été validé.

Cette fonction peut retourner trois types résultats, composé de quatre cas d’usage :

* Cas 1 - La fonction trouve un numéro de cumul pour le marché répondant à l’intervalle de date du cumul et en retourne la valeur
* Cas 2 - La fonction ne trouve pas de numéro de cumul pour le marché répondant à l’intervalle de date du cumul mais trouve un numéro de cumul et en retourne la valeur
* Cas 3 - La fonction ne trouve pas de numéro de cumul pour le marché répondant à l’intervalle de date du cumul mais trouve plusieurs numéros de cumul et retourne 0
* Cas 4 - La fonction ne trouve pas de numéro de cumul pour le marché et retourne 0

Le retour de la valeur 0 indique que la fonction n’a pas réussi à retourner de résultat.

Je vais créer autant de tests que de sortie possible de la fonction.

J’ai d’abord écrit la fonction répondant au comportement attendu, mais je pense qu’il aurait été plus intéressant d’écrire les tests avant d’écrire la fonction.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

## Mise en place des tests sous SQL Developper

Affichage des tests sous SQL Developper, menu Affichage / Test d’unité

**Test 1 - Un cumul lot correspond à l’intervalle de date**

Afin que mes tests ne dépendent pas des modifications des données dans la base de données, j’ajoute un bout de code à exécuter avant le test qui va servir à l’exécution de mon test dans le bloc ‘Démarrer le processus’

Au niveau de l’implémentation du test, j’indique la valeur des paramètres entrants pour l’exécution du test (identifiant société, exercice du marché, numéro de marché et numéro de lot du marché) et le résultat attendu par l’exécution du test (Résultat = 4), ainsi que le type de résultat attendu (Succès).

**Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement**

L’exécution du test m’indique si le comportement de ma fonction ramène le résultat attendu.

**Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement**

Je répète cette action pour les trois autres tests.

**Test 2 - Aucun cumul lot ne correspond à l’intervalle de date mais il existe un résultat unique**

**Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement**

**Test 3 - Aucun cumul lot ne correspond à l’intervalle de date mais plusieurs résultats existent**

**Une image contenant texte, capture d’écran, logiciel, ligne

Description générée automatiquement**

**Test 4 - Aucun résultat ne correspond aux critères de recherche**

**Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement**

SQL developper propose de regrouper les tests dans un élément « Suites »

**Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement**

Et propose l’exécution de l’ensemble des tests d’une suite (clic droit, exécuter la suite)

Une image contenant texte, capture d’écran, nombre

Description générée automatiquement

## Petite conclusion

Au-delà de permettre de de tester le bon comportement de la fonction, les tests vont permettre de garantir le comportement de la fonction durant l’évolution de son cycle de vie, et au travers des refactoring qui pourraient être réalisés.

Les tests servent de filet de sécurité et garantissent que la modification de code ne modifie pas le comportement de la fonction.

Il serait idéalement intéressant d’écrire les tests avant d’écrire la fonction, ce qui permet sa validation lors de son écriture.

Le test peut également servir de documentation puisqu’il décrit le comportement attendu de la fonction.