Memo PL/SQL

Par Jean-Baptiste Cochinard

Table des matières

[PL/SQL 2](#_Toc69899590)

[Structure d’un bloc PL/SQL 2](#_Toc69899591)

[DECLARE : déclaration des variables 2](#_Toc69899592)

[Sous-types 2](#_Toc69899593)

[Les types collections 3](#_Toc69899594)

[BEGIN : instructions principales 3](#_Toc69899595)

[Affectation et affichage des message de debug 3](#_Toc69899596)

[Conditionnel 4](#_Toc69899597)

[Boucle 4](#_Toc69899598)

[CURSEUR 5](#_Toc69899599)

[EXCEPTION 5](#_Toc69899600)

[Erreur Utilisateur 5](#_Toc69899601)

[Procédures stockées 6](#_Toc69899602)

[Les fonctions 6](#_Toc69899603)

[TRIGGER : Les déclencheurs de base de données 7](#_Toc69899604)

[Les packages 8](#_Toc69899605)

[SQL dynamique 8](#_Toc69899606)

[Les Transactions 9](#_Toc69899607)

*Quand on code au plus vite et de manière non optimale, on contracte une dette technique que l’on rembourse tout au long du projet […]*

[Ward Cunningham](https://fr.wikipedia.org/wiki/Ward_Cunningham)

# PL/SQL

Le PL/SQL est un langage procédural qui permet d’optimiser l’accès aux données. Les procédures sont stockées et exécuté sur le serveur de base de données et sont plus rapide que les requête envoyer depuis notre application.

Bloc PL/SQL peuvent être interne au serveur de base de données ou externe (dit anonyme) qui sera envoyé et exécuter sur le serveur de base données (surtout utiliser maintenance, test et développement).

## Structure d’un bloc PL/SQL

DECLARE – Déclaration des variables (facultative)

BEGIN – le bloc d’instruction commence

EXCEPTION – gestion des erreurs (facultative)

END – Fin du bloc d’instruction

## DECLARE : déclaration des variables

Les types possibles des variables du **PL/SQL** sont :

* Tous les type **SQL** (avec différence sur la taille de stockage)
* **BOOLEAN** (true/false et NULL)
* **PLS\_INTEGER** (stock des entiers, utiliser ce type au lieu de number)
* Sous-type

|  |  |
| --- | --- |
| **Déclaration** |  |

### Sous-types

*Les sous types sont des type de variable définis par l’utilisateur*

|  |  |
| --- | --- |
| **SUBTYPE** |  |
| **%TYPE** |  |
| **RECORD** |  |
| **%ROWTYPE** |  |

### Les types collections

*Les collections permettent de regrouper des variables de même type (équivalent au tableau). Les tableaux disposent de méthodes*

|  |  |
| --- | --- |
| **INDEX BY TABLE** |  |
| **NESTED TABLE** |  |
| **VARRAY** |  |

## BEGIN : instructions principales

### Affectation et affichage des message de debug

|  |  |
| --- | --- |
| **Affectation** | **Message d’erreur** |
|  |  |
|  |  |

### Conditionnel

|  |  |
| --- | --- |
| **IF…ELSE** |  |
| **CASE**  *Avec valeur* |  |
| **CASE**  *Avec condition* |  |

### Boucle

|  |  |
| --- | --- |
| **LOOP** |  |
| **FOR** |  |
| **WHILE** |  |

### CURSEUR

*Les curseurs permettent de stocker le résultat d’une requête.*

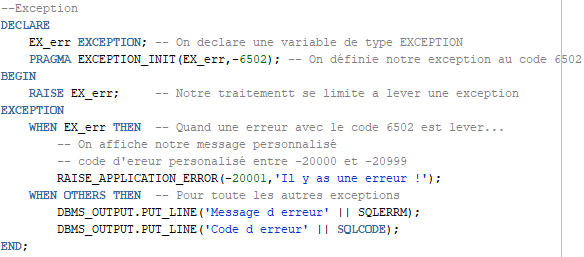
|  |  |
| --- | --- |
| **FOR…IN** |  |
| **FOR…IN** |  |
| **CURSOR**  **FOR UPDATE**  **FOR…IN**  **WHERE CURRENT OF** |  |
| *Curseur implicite* |  |

## EXCEPTION

Une exception est lever quand une erreur a été rencontré. Il existe différent type d’erreur qui les déclenchent :

* Les erreurs de compilation : c’est une erreur du développeur
* Les erreurs d’exécution : erreurs prédéfinies numéroté dans la console (voir SQLEERM et SQLCODE)
* Les erreurs utilisateur : ce sont des erreurs définies par l’utilisateur

### Erreur Utilisateur



## Procédures stockées

Permet de nommer et stocker nos blocks de **PL/SQL** sur le server pour l’appeler au besoin. Ces blocks deviennent internes.

|  |  |
| --- | --- |
| **CREATE PROCEDURE**  **IN et OUT**  **IN OUT**  **EXECUTE** |  |

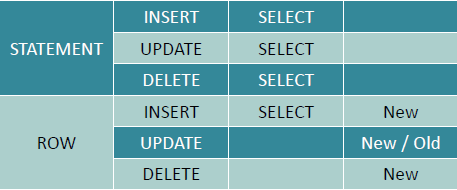
## Les fonctions

Une fonction est un block de code nommé et stocké sur le serveur et elle retourne toujours une et une seule valeur.

* Les paramètres d’entrées sont exclusivement de type **IN**
* Les paramètres d’entrée et de retour doivent être de type **SQL** et non **PL/SQL**
* Les fonctions ne doivent pas faire de **DML** (**INSERT**, **UPDATE**, **DELETE**)

|  |  |
| --- | --- |
| **CREATE FUNCTION**  *Utilisation* |  |

## TRIGGER : Les déclencheurs de base de données

C’est un traitement qui est déclencher automatiquement lorsqu’un évènement (**INSERT**, **DELETE**, **UPDATE**) s’exécute sur une table ou une vue. Il peut être déclenché pour chaque ligne (**FOR EACH ROW**) ou toute la requête, avant (**BEFORE**) ou après (**AFTER**) les contraintes d’intégrité.

* Ce sont des blocks associés à un nom
* Peuvent appeler des sous-programmes
* Ne sont pas paramétrable
* Commit et rollback interdit

|  |  |
| --- | --- |
| **CREATE TRIGGER**  **INSERTING**  **DELETING**  **UPDATING**  **:OLD**  **:NEW**  **WHEN**  **INSTEAD OF** |  |

## Les packages

Un package SQL permet de stockées plusieurs instruction SQL. Un package est un objet du schéma, il permet d’améliorer la gestion des objet PL/SQL et permet de regrouper un ensemble d’objet homogènes.

|  |  |
| --- | --- |
| **PACKAGE**  *Spécification*  **PACKAGE**  *Corps*  *Utilisation*  **DROP** |  |

## SQL dynamique

Le SQL dynamique permet de créer des traitements génériques et de construire dynamiquement des requêtes.

|  |  |
| --- | --- |
| **RETURNING**  **EXECUTE IMMEDIATE**  *Syntaxe alternative* |  |

## Les Transactions

Il y a toujours une seule transaction principale active et en cours, celle-ci est automatique. Une transaction peut être validée (**COMMIT**) ou invalidé (**ROLLBACK**). Les transactions permettent d’assurer la cohérence des données (tant que la transaction n’est pas validée, les utilisateurs ne voient pas les modification).

Tant qu’il n’y a pas de **COMMIT** de fait, seule la connexion courante voit les mises à jour.

Les transactions autonomes, sont des transactions dans la transaction principale. Cela peut être utilisé par exemple dans les procédures stockées.

