TP N°05 (semaine 45-46) : PLSQL – Bloc Anonymes

Exercice 1 : Bloc anonyme – structure conditionnelle (IF) – Exceptions

Ecrire un bloc PL/SQL qui affiche le nombre de chambres entretenues par un agent d'entretien dont le nom est saisi par l'utilisateur (**ACCEPT – PROMPT**). Gérer l'erreur pré-définie ORACLE : **NO_DATA_FOUND**


```
ACCEPT s_nom PROMPT 'Entrer un nom d''un agent'
DECLARE
    v_nomAg     AGENT_ENTRETIEN.agt_nom%TYPE:= UPPER('&s_nom');
    …
BEGIN
    …
EXCEPTION
    WHEN NO_DATA_FOUND THEN …
END ;
/
```

Ex : Pour le nom DUSSE → Agent DUSSE inexistant
Pour le nom BEVIERE → Aucune chambre pour l'agent BEVIERE
Pour le nom FECHOL → L'agent FECHOL s'occupe de 7 chambres

Exercice 2 : Variables %TYPE – Variables de substitution – Exception Oracle prédéfinie

Ecrire un bloc PL/SQL qui, étant donné le nom d'un client saisi au clavier affiche son code, son identité (nom, prénom) et la ville de don adresse principale (adr_id = 1).

Gérer les erreurs (client inconnu et plusieurs clients de même nom) : **NO_DATA_FOUND, TOO_MANY_ROWS**

Remarque : Pour améliorer la lisibilité de la trace exécuter la commande **SET VERIFY OFF** avant le bloc.

Ex. : Pour le nom DIEUDONNE → *Le client 104 César DIEUDONNE habite COMPIEGNE*
Pour le nom DUSSE → *Désolé, pas de client nommé DUSSE.*
Pour le nom MARTIN → *Attention ! Plusieurs clients pour MARTIN.*

Exercice 3 : Gestion des EXCEPTIONS, erreurs prédéfinies et non prédéfinies ORACLE

a) Ecrire un bloc PLSQL, qui modifie la date de départ d'un agent d'entretien :

- Saisir le code de l'agent et la date de départ à lui affecter.
- Mettre à jour sa date de départ et afficher un message : *« L'agent a été modifié. »*


```
ACCEPT s_code PROMPT 'Entrer un code agent'
ACCEPT s_date PROMPT 'Entrer une date (dd/mm/yyyy)'

DECLARE
    v_code AGENT_ENTRETIEN.agt_id%TYPE := '&s_code';
    v_date AGENT_ENTRETIEN.agt_dpt%TYPE;

BEGIN
-- On vérifie si la date est correcte
v_date := TO_DATE ('&s_date','DD/MM/YYYY');
    …
EXCEPTION
    WHEN NO_DATA_FOUND THEN … ;
    WHEN OTHERS THEN … ;
END ;
```

Gérer les erreurs suivantes :

- Agent inexistant (**NO_DATA_FOUND**) : afficher « *L'agent A08 n'existe pas.* »
- Autre erreur (**OTHERS**) : afficher le message ORACLE correspondant (**SQLERRM**)

Tests :
- Tester avec un agent inexistant : A08
- Tester avec l'agent A06 auquel on affecte la date erronée (01/15/2021) puis 01/12/2022.
- Annuler la modification (Rollback)

b) Ecrire un autre bloc PLSQL similaire au précédent qui effectue la même mise à jour de la date de départ d'un agent d'entretien et qui affiche les mêmes messages mais sans utiliser l'erreur **NO_DATA_FOUND** : on utilisera cette fois l'attribut **ROWCOUNT** du curseur **implicite** utilisé par la requête UPDATE.
Vérifier en effectuant les mêmes tests.

SQL%ROWCOUNT : retourne le nombre de lignes impactées par la dernière instruction SQL

c) Ajouter à la table Agent d'entretien la contrainte qui vérifie que la date de départ est bien supérieure à la date d'embauche (**ALTER TABLE**).

Tester maintenant le bloc précédemment écrit avec l'agent A06 auquel on tente d'affecter le 20/10/1998 comme date de départ. Que se passe-t-il ? (Repérer le code de l'erreur)

Modifier le bloc PLSQL afin de gérer spécifiquement l'erreur due à la contrainte mise sur les dates de la table AGENT_ENTRETIEN (**PRAGMA EXCEPTION_INIT**). On affichera le message :
« La date de départ de l'agent ne peut pas être inférieure à sa date d'embauche . »

Il est possible d'associer un code erreur Oracle à vos propres variables exception à l'aide du mot clé **PRAGMA EXCEPTION_INIT**, dans le cadre de la section déclarative de la façon suivante :

```
DECLARE
    nom_exception EXCEPTION ;
    PRAGMA EXCEPTION_INIT(nom_exception, -code_error_oracle);
BEGIN
…
EXCEPTION
      WHEN nom_exception THEN …
END ;
/
```

Exercice 4 : Gestion des EXCEPTIONS, erreurs utilisateur, prédéfinies et non prédéfinies ORACLE

Avant de commencer cet exercice, désactiver les déclencheurs précédemment créés : TR_CHAMBRE et TR_AGENT

a) Parfois les chambres sont amenées à changer d'agent d'entretien. Ecrire un bloc PLSQL qui, modifie (remplace) l'agent associé à une chambre dans la table CHAMBRE :

- Le code de la chambre à modifier et celui du nouvel agent seront saisis par l'utilisateur et stockés dans les variables `v_chb_id` et `v_agt_id`.
- Lorsque la modification est réalisée, on affichera un message du type :
  « *Modification effectuée* : *L'agent A03 est affecté à la chambre 30* »

Gérer les erreurs suivantes :

- Chambre existante (**NO_DATA_FOUND**) : Afficher *« La chambre 35 n'existe pas »*.
- Agent inexistant (utiliser **PRAGMA EXCEPTION_INIT**) : Afficher « *L'agent A08 n'existe pas* »
- Autre erreur (**OTHERS**) : afficher le message ORACLE correspondant.

Tests :
- Tester avec la chambre 20 à laquelle on affecte l'agent A03 (modif effectuée)
- Tester avec la chambre 35 à laquelle on affecte l'agent A03 (chambre inexistante)
- Tester avec la chambre 20 à laquelle on affecte l'agent A08 (agent inexistante)

b) Un agent ne peut être en charge de plus de 8 chambres.

Avant la modification vérifier que l'agent peut être affecté à cette chambre sans dépasser son quota sinon on utilisera l'instruction RAISE pour déclencher une erreur utilisateur et afficher un message du type :

« Trop de chambres pour l'agent A01. Modification annulée.»

Test : - Tester avec l'agent A01 auquel on tente d'affecter la chambre 20 (trop de chambres)

En plus des erreurs Oracle, il est possible d'intercepter ses propres erreurs en déclarant des variables dont le type est **exception** et en provoquant le saut dans la section de gestion des erreurs à l'aide de l'instruction **RAISE**


```
DECLARE
     E_quota Exception ;
     …
BEGIN
     …
     Raise E_quota;
EXCEPTION
     WHEN E_quota Then …
END ;
```

c) Ecrire une nouvelle version du bloc PLSQL précédemment écrit mais en gérant, cette fois, de la même façon l'existence de l'agent et celle de la chambre c'est-à-dire en utilisant uniquement l'erreur prédéfinie Oracle **NO_DATA_FOUND** (au lieu de **PRAGMA EXCEPTION_INIT**) : pour cela utiliser une imbrication de 2 blocs permettant ainsi 2 sections **EXCEPTION** et donc 2 **WHEN NO_DATA_FOUND**.


```
DECLARE
   …
BEGIN
   SELECT …
   BEGIN
        SELECT …
        …
   EXCEPTION
        WHEN NO_DATA_FOUND THEN …
   END ;
EXCEPTION
   WHEN NO_DATA_FOUND THEN …
END ;
```