PL/SQL: Gestion des exceptions

Issam QAFFOU

C’est quoi une exception?

- Dans PL/SQL, les erreurs et les warnings sont appelés "exceptions".
- Quand une erreur prédéfinie est produite dans un programme, PL/SQL déclenche une exception.

  Par exemple: diviser un nombre par zero, l’exception est appelée ZERO_DIVIDE. Si SELECT ne trouve pas d’enregistrements alors l’exception est: NO_DATA_FOUND.

- PL/SQL possède une collection d'exceptions prédéfinies. Chacune a un nom. Ces exceptions sont automatiquement déclenchées par PL/SQL lorsque l'erreur correspondante est produite.
- En plus de ces exceptions, le programmeur peut aussi créer ses propres exceptions pour traiter les erreurs dans l'application.
- Comprendre comment gérer une exception est très important.

Comment gérer les exceptions?

- Quand PL/SQL déclenche une exception prédéfinie, le programme est interrompu par l'affichage du message d'erreur.
- Si le programme doit traiter l'exception soulevée par PL/SQL, alors nous devons utiliser la partie du bloc: Gestion des Exceptions.
- La partie Gestion des Exceptions est utilisée pour spécifier les instructions à exécuter si une exception survient.

- La syntaxe de la partie: gestion des exceptions

```
WHEN exception-1 [or exception -2] ... THEN
        {instructions};
[WHEN exception-3 [or exception-4] ... THEN
        {instructions}; ] ...
[WHEN OTHERS THEN
        {instructions}; ]
```

- exception-1 et exception-2 sont les exceptions à gérer. Elles sont soit des exceptions prédéfinies ou des exceptions définies par le programmeur.
- L'exemple suivant traite l'exception: NO_DATA_FOUND déclenchée quand SELECT ne récupère aucune ligne.

```
declare
        …
begin
        select …
exception
        when no_data_found then
                {instructions};
end;
```

- Lorsque deux ou plusieurs exceptions sont données avec un seul WHEN alors les instructions sont exécutées quand l'une des exceptions prévues se produit.

```
declare
    ...
begin
    select ...
exception
    when no_data_found or too_many_rows then
        instructions;
end;
```

- L'extrait suivant gère ces deux exceptions de différentes manières.

```
declare
        ...
begin
        select ...
exception
        when no_data_found then
                instructions_1;
        when too_many_rows then
                 instructions_2;
end;
```

- "**WHEN OTHERS**" est utilisé pour exécuter des instructions quand une exception autre que celles mentionnées dans le gestionnaire d'exception s'est produite.
- **Remarque:** Si une exception est soulevée, mais pas prise en charge par la partie gestion des exceptions alors le bloc PL/SQL se termine par l'affichage d'un message d'erreur lié à l'exception.

- **Exemple:**

Ce programme attribue les frais du cours "C++" au cours "C". Si le cours "C++" n'existe pas, alors il définit les frais du cours "C" par la moyenne des frais de tous les cours.

```
declare
    v_frais cours.frais%type;
begin
    select frais into v_frais
    from cours
    where ccode = 'c++';
    update cours
          set frais= v_frais
          where ccode=‘c’;
    exception
          when no_data_found then
          update cours
                    set frais= ( select avg( frais) from cours)
                                                  where ccode ='c';
end;
/
```

Obtenir des informations sur l'erreur - SQLCODE et SQLERRM

1. Dans la section WHEN OTHERS du gestionnaire d'exception, vous pouvez utiliser les fonctions SQLCODE et SQLERRM pour obtenir le numéro d'erreur et le message d'erreur respectivement.
2. L'autre méthode consiste à associer une exception avec une erreur Oracle.

L'exemple suivant montre comment utiliser SQLCODE et SQLERRM.

```
declare
    newccode varchar2(5) := null;
begin
    update cours set ccode = newccode where ccode = 'c';
exception
    when dup_val_on_index then
          dbms_output.put_line('Duplication du code du cours ');
    when others then
           dbms_output.put_line( sqlcode);
          dbms_output.put_line( sqlerrm);
end;
```

- En exécutant ce programme, on obtient le message:

ORA-01407: impossible de mettre à jour ("SYSTEM"."COURS"."CCODE") avec NULL

Procédure PL/SQL terminée avec succès

- Ce message est généré par " WHEN OTHERS " de la partie "gestion d'exception".
- Puisque (-01407) n'est associé à aucune exception prédéfinie, la partie WHEN OTHERS du gestionnaire d'exception est exécuté.

Exceptions prédéfinies

- PL / SQL a défini certaines erreurs courantes et donné des noms à ces erreurs, qui sont appelées: exceptions prédéfinies.
- Chaque exception a un code d'erreur Oracle correspondant. Ce qui suit est la liste des exceptions prédéfinies et le code d'erreur Oracle correspondant.

Exception	Oracle Error	SQLCODE Value
ACCESS_INTO_NULL	ORA-06530	-6530
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
SELF_IS_NULL	ORA-30625	-30625
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
SYS_INVALID_ROWID	ORA-01410	-1410
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

Exception	Description
CURSOR_ALREADY_OPEN	Déclenchée si vous essayez d'ouvrir un curseur déjà ouvert.
DUP_VAL_ON_INDEX	Déclenchée si vous essayez de stocker des valeurs en double dans une colonne qui est contrainte par un index unique.
INVALID_CURSOR	Déclenchée si vous essayez une opération de curseur illégale.
INVALID_NUMBER	Soulevée dans une instruction SQL si la conversion d'une chaîne de caractères à un certain nombre échoue parce que la chaîne ne représente pas un nombre valide.
NO_DATA_FOUND	Déclenchée si une instruction SELECT INTO ne retourne aucune ligne ou si vous faites référence à une ligne non initialisée dans une table PL / SQL.
SUBSCRIPT_BEYOND_COUNT	Déclenché lorsque le programme fait référence à une table imbriquée ou à un élément tableau à l'aide d'un numéro d'index plus grand que le nombre d'éléments dans la collection.

Quand l’exception " NO_DATA_FOUND" est-elle déclenchée?

- Quand l’exception NO_DATA_FOUND n'est pas soulevée par PL/SQL, même si aucune ligne n'est récupérée ou effectuée
  - Quand une fonction de groupe est utilisée dans l'instruction SELECT.
  - Lorsque UPDATE et DELETE sont utilisés.
- Lorsque la commande SELECT utilise une fonction de groupe alors l’exception NO_DATA_FOUND ne sera pas soulevée par PL/SQL même si aucune ligne n'est récupérée.

- Exemple:

  L'exemple suivant permet d'afficher la durée moyenne des lots (batches) du C . Si aucun lot C n’a été achevé il affiche un message. Puisque la fonction AVG retourne NULL lorsqu’aucun ligne n’est récupérée par SELECT, on vérifie la valeur retournée de AVG et on affiche un message d'erreur si elle est NULL.

```
declare
    v_avgdur number(3);
begin
    -- La durée moyenne des lots C
    select avg( enddate - stdate) into v_avgdur
    from batches
    where enddate is not null and ccode='c';
    /* afficher erreur si AVG retourne null */
    if v_avgdur is null then
            dbms_output.put_line ('aucun lot C n'a pas été achevé');
    else
            dbms_output.put_line ('La durée moyenne des lots C: '   ||
    v_avgdur);
    end if;
end;
/
```

Exception définie par le programmeur

- PL/SQL permet à l'utilisateur de créer ses propres exceptions.
- Elles ne sont valables que dans des blocs où elles sont créées.
- Étapes de création de ces exceptions:
  1. Déclarer l'exception
  2. Soulever l'exception en utilisant la commande RAISE
  3. Re-soulever une exception

Exception définie par le programmeur

1. Déclarer l'exception

L'exception doit être déclarée dans la section "declare" du bloc PL/SQL. La syntaxe est la suivante:

nom_exception exception;

Exemple:

```
declare
        pas_dans _stock   exception;
begin
        Instructions;
end;
```

Exception définie par le programmeur

2. Soulever l'exception en utilisant la commande RAISE

- L'exception définie par l'utilisateur doit être soulevée par la commande RAISE, sinon elle ne sera pas soulevée.

RAISE nom_exception;

- L'utilisateur doit décider quand son exception doit être soulevée.

Par exemple, si on veut soulever l'exception pas_dans_stock quand la valeur de la variable qté est plus petite que 10, on écrit:

```
if qté < 10 then
        raise   pas_dans _stock ;
end if;
```

Quand l'exception est soulevée, elle doit être traitée.

```
declare
        pas_dans _stock  exception; -- déclarer exception
begin
        if condition then
                raise pas_dans _stock ; -- soulever exception
        end if;
exception
        when pas_dans _stock then -- traiter exception
        . . .
end;
```

Exception définie par le programmeur

3- Re-soulever une exception

- La commande RAISE peut également être utilisée pour re-soulever une exception de sorte que l'exception en cours soit propagée au bloc externe.
- Si un sous bloc exécute la déclaration RAISE sans donner le nom de l'exception dans le gestionnaire d'exception alors l'exception courante est soulevée à nouveau.

```
declare
      pas_dans _stock  exception;
begin
      ...
      begin  -- début du sous block (inner block)
      ...
      if ... then
                raise  pas_dans _stock ; -- soulever  l'exception
      end if;
      …
exception
      when  pas_dans _stock   then   -- traiter l'erreur dans le sous bloc
                raise; -- redéclencher l'exception courante, qui est  pas_dans _stock
      ...
      end;    -- fin du sous bloc
exception
      when  pas_dans _stock   then  -- traiter l'exception redéclenchée dans le bloc externe
      ...
end;
```

Associer une exception à une Erreur Oracle

- Une exception définie par l'utilisateur peut lui être accordé un nombre d'Erreur Oracle. Ainsi, quand l'Erreur Oracle arrive alors l'exception sera déclenchée par PL/SQL automatiquement.
- L'exemple suivant associe à l'exception NULL_VALUE_ERROR le nombre d'erreur -1407 en utilisant l'instruction PRAGMA EXCEPTION_INIT.

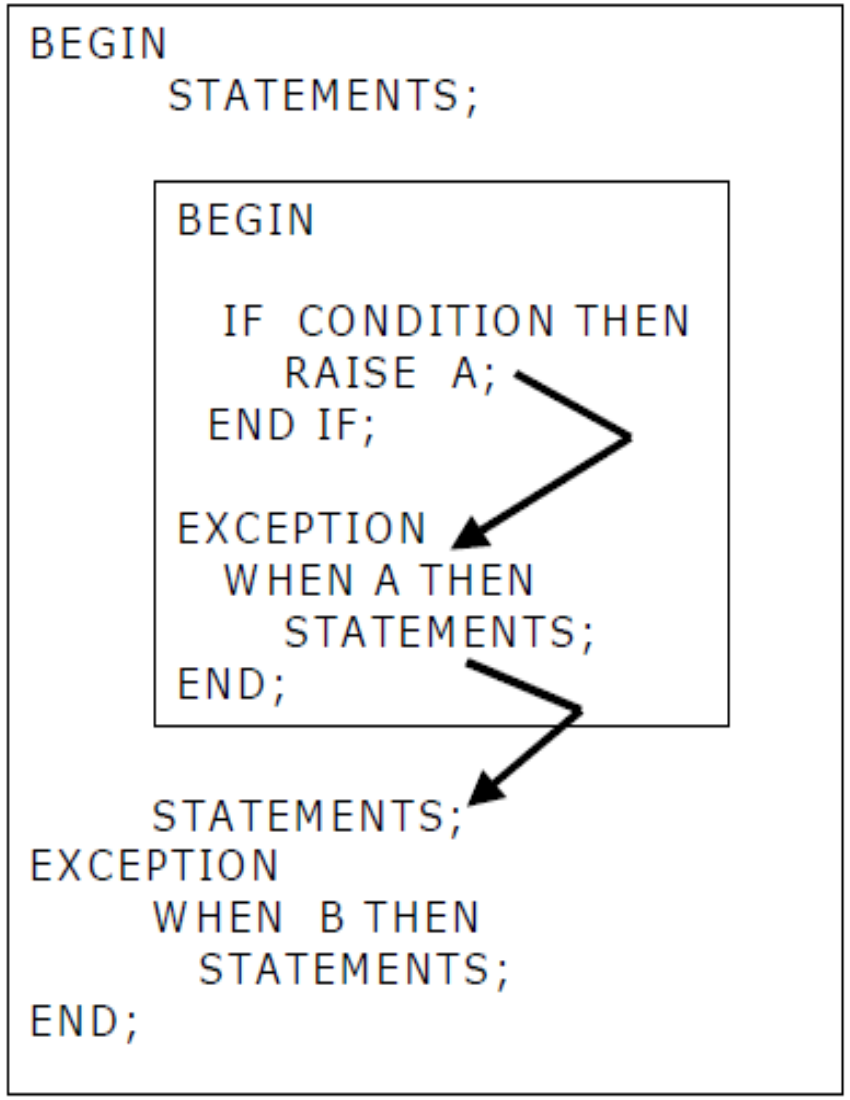
```
declare
    null_value_error exception;
    pragma exception_init(null_value_error, -1407);
    newccode varchar2(5) := null;
begin
    update cours
    set ccode = newccode
    where ccode = 'c';
exception
    when null_value_error then
    dbms_output.put_line(‘Attention! Vous essayez d’affecter
    une valeur Null à un champ Not Null');
end;
/
```

Propagation d’exception

- Quand une exception est déclenchée par PL/SQL et elle n’est pas traitée dans le bloc courant, l’exception est alors propagée.
- Ainsi, l’exception est envoyée aux blocs recouvrant un après l’autre à partir de l’interne à l’externe jusqu’à ce qu’un gestionnaire d’erreur soit trouvé ou qu’il n’y ait plus de blocs pour chercher des gestionnaires.

Propagation d’exception

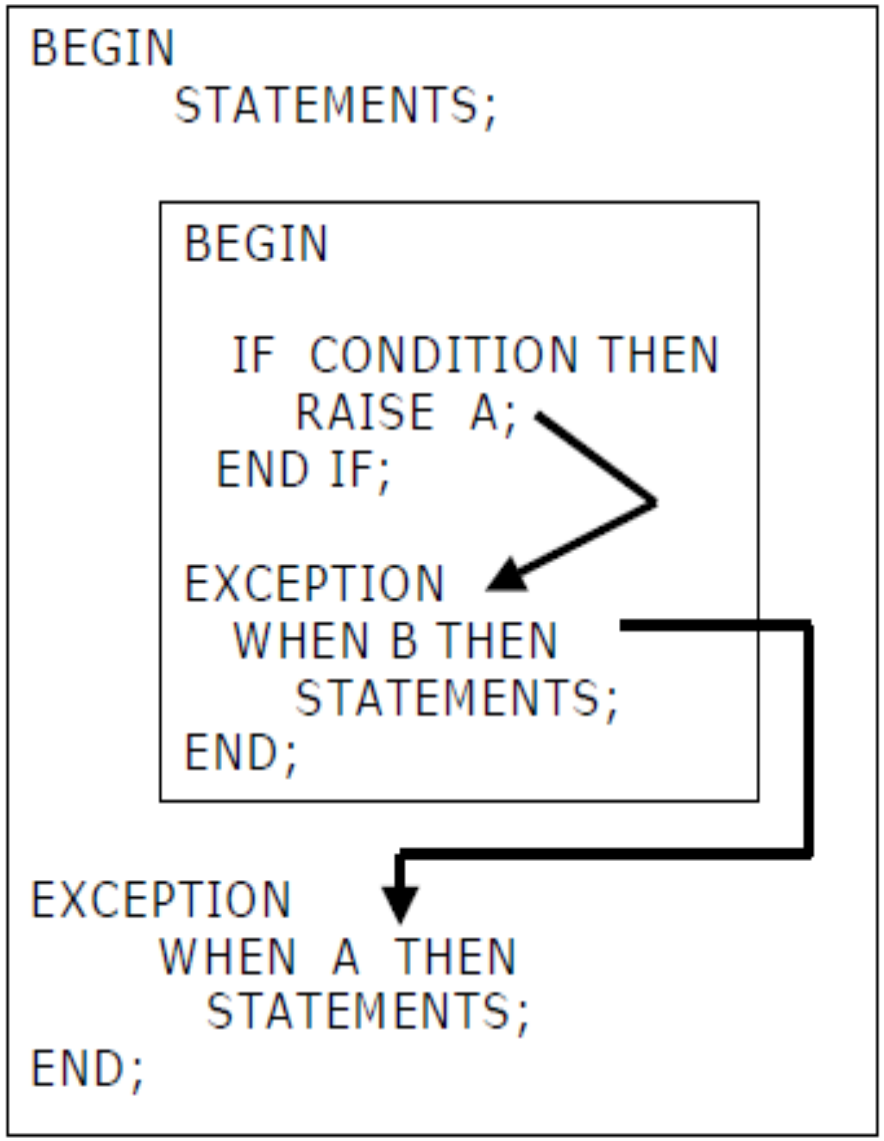
- Dans la figure suivante, l’exception est déclenchée par le bloc interne. Comme il y a un gestionnaire pour l’exception A, alors elle est y traitée elle-même. Après le traitement de l’exception, le contrôle reprend les déclarations après le bloc interne en bloc externe.
- Comme l’exception est traitée dans le bloc où elle est déclenchée, l’exception n’est pas propagée et le contrôle reprend avec le bloc le plus proche.



Après le traitement de l'exception A dans le bloc interne le contrôle reprend avec le bloc le plus proche.

Propagation au bloc externe

- Dans la figure suivante, le bloc interne déclenche l'exception A mais comme elle n'est pas traitée dans le bloc courant elle est propagée au premier bloc externe. Comme il y a un gestionnaire pour l'exception A dans le bloc externe, le contrôle est passé à ce gestionnaire et l'exception est traitée dans le bloc externe.



comme l'exception n'est pas traitée dans le bloc interne elle est propagée au bloc externe.