PL/SQL
Structures de contrôle

Pr. Issam QAFFOU

Traitement conditionnel (IF)

```
IF condition-1 THEN
        Bloc d'instructions (déclarations)_1;
[ELSIF condition-2 THEN
        Bloc d'instructions (déclarations)_ 2;] ...
[ELSE
        Bloc d'instructions (déclarations)_ 3; ]
END IF;
```

Traitement conditionnel (IF)

- Les conditions peuvent être formulées avec:

>	Plus grand
>=	Plus grand ou égal
<	Plus petit
<=	Plus petit ou égal
=	égal
<>, !=, ~=, ^=	différent de
LIKE	Retourne vrai si le caractère recontré correspond à la valeur donnée.
BETWEEN..AND	Retourne vrai si la valeur est dans l'intervalle donné.
IN	Retourne vrai si la valeur est dans la liste.
IS NULL	Retourne vrai si la valeur est nulle.

Traitement conditionnel (IF)

- Pour combiner deux conditions on utilise AND ou OR.

  If amt > 5000 then

  If rate > 500 and qty < 10 then

  If rate between 100 and 200 then

- Exemple: écrire un programme qui augmente les frais du cours Oracle par 10% si plus de 100 étudiants sont engagés pour Oracle. Dans le cas contraire, il va décroitre de 10%.

Traitement conditionnel (IF)

```
declare
    v_ns number(5);
Begin

    -- Extraire le nbre d'étudiant pour le cours Oracle
    select count(*) into v_ns from étudiant where bcode in
                        ( select bcode from batches where ccode = 'ora');
    if v_ns > 100 then
            update cours set frais = frais * 1.1 where ccode = 'ora';
    else
            update cours set frais = frais * 0.9 where ccode = 'ora';
    end if;
    commit;
end;
/
```

- Refaire l'exemple précédent avec les données suivantes:

Nbre d'étudiants	Pourcentage de changement
>100	croissance par 20
>50	décrois par 15
>10	croissance par 5
<=10	décrois par 10

Traitement répétitif (LOOP)

```
LOOP
    déclarations - instructions;
END LOOP;
```

- Les déclarations au sein d'une boucle doivent vérifier une condition d'arrêt pour sortir (EXIT), sinon le traitement sera infini.
- "Exit" est utilisé pour sortir d'une boucle.

  EXIT [WHEN condition];

- Si EXIT est utilisé seul, il terminera la boucle courante.
- S'il est utilisé avec WHEN, la boucle courante est terminée seulement quand la condition donnée est satisfaite.

- Exemple1:

```
LOOP
        ...
        IF count > 10 THEN
                EXIT; -- termine la boucle
        END IF;
        ...
END LOOP;
```

- Example 2:

```
LOOP
    ...
    EXIT WHEN count >10; -- termine loop
    ...
END LOOP;
```

- Le programme suivant va afficher les nombres de 1 à 10 en utilisant LOOP.

```
declare
    i number(2) := 1;
begin
    loop
        dbms_output.put_line(i);
        i := i + 1;
        exit when i > 10;
    end loop;
end;
/
```

- Ecrire un programme PL/SQL qui affiche la table de multiplication jusqu'à 10 pour les numéros de 1 à 5.

- Sortie d’une boucle imbriquée

  Il est possible de sortir de la boucle courante en utilisant l’instruction EXIT. Il est également possible d'utiliser la déclaration EXIT pour quitter une boucle englobant. Ceci est réalisé en utilisant l'étiquette de boucle.
- Une étiquette de boucle est une étiquette qui est attribuée à une boucle. En utilisant cette étiquette, EXIT peut sortir d'une boucle spécifique au lieu de la boucle courante.

- Exemple:

```
<<outerloop>>
        LOOP
                ...
                LOOP
                ...
                        EXIT outerloop WHEN ... -- exit boucle externe
                END LOOP;
                ...
        END LOOP;
```

L'instruction EXIT utilise l'étiquette pour définir la boucle qui doit être terminée. EXIT utilise **<<outerloop>>,** qui est l'étiquette donnée à la boucle extérieure, pour mettre fin à la boucle externe.

- While

  exécute une série de déclarations tant que la condition est vraie.

  WHILE condition LOOP

  Statements;

  END LOOP;

  On sort de la boucle quand la condition n'est plus vérifiée.

- Exemple: affichage de 1 jusqu'à 10

```
declare
    i number(2) := 1;
begin
    while i <= 10
    loop
        dbms_output.put_line(i);
        i := i + 1;
    end loop;
end;
/
```

- FOR

  Cette structure de boucle est la mieux adaptée pour les cas où nous avons à exécuter, à plusieurs reprises, un ensemble de déclarations en faisant varier une variable d'une valeur à l'autre.

```
FOR counter IN [REVERSE] lowerrange .. upperrange LOOP
        Statements;
END LOOP;
```

- Exemple: affichage de 1 jusqu'à 10

```
begin
    for i in 1..10
    loop
            dbms_output.put_line(i);
    end loop;
end;
/
```

- Exemple: avec REVERSE de 10 à 1.

```
begin
   for i in REVERSE 1..10
    loop
            dbms_output.put_line(i);
    end loop;
end;
/
```

- Exemple de programme utilisant la boucle FOR:

  L'exemple suivant permet d'afficher les "rollno" des étudiants manquants. Le programme commence du "rollno" le plus bas et monte au plus grand. Il permet d'afficher les "rollno" qui sont dans l'intervalle et pas dans la table des étudiants.

```
declare
    v_minrollno étudiant.rollno%type;
    v_maxrollno étudiant.rollno%type;
    v_count number(2);
    i étudiant.rollno%type;
begin
    -- prendre le min et le max des rollno
    select min(rollno) , max(rollno) into v_minrollno, v_maxrollno
    from étudiant;
    for i in v_minrollno .. v_maxrollno
        loop
            select count(*) into v_count from étudiant where rollno = i;
            -- afficher rollno si count est 0
            if v_count = 0 then
                dbms_output.put_line(i);
            end if;
        end loop;
end;
/
```

- Ce programme prend les "rollno" minimum et maximum en utilisant les fonctions MIN et MAX. ensuite il établit une boucle commençant du "rollno" minimal et monte au maximal. Dans chaque itération, il vérifie s'il existe un étudiant avec le "rollno" courant (représenté par i). Si aucune ligne n'est trouvée, alors COUNT(*) sera 0.

- GOTO

  Il transfère le contrôle à l'étiquette nommée. L'étiquette doit être unique et doit précéder une déclaration PL / SQL exécutable ou un bloc PL / SQL.

  L'exemple suivant montre comment créer une étiquette et la façon de transférer le contrôle de l'étiquette avec GOTO.

```
BEGIN
        ...
        GOTO change_details;
        ...
        <<change_details>>
        update étudiant... ;
END;
```

- Limites du GOTO:

  Ne peut pas brancher
  - À un bloc IF,
  - À un LOOP
  - À un sous-bloc
  - Sur une procédure ou fonction
  - D’un gestionnaire d’exceptions au block courant

- Exemple:

```
BEGIN
    ...
    /* ce n'est pas valide parce que GOTO ne peut pas brancher
    à un IF */
    GOTO change_details;
    ...
    IF condition THEN
            ...
            <<change_details>>
            update étudiant... ;
            ...
    END IF;
END;
```

Exercices:

- Écrire un bloc PL / SQL pour diminuer la durée du cours C si plus de 2 Batches ont commencé dans les deux derniers mois.
- Écrire un bloc PL / SQL pour insérer une nouvelle ligne dans la table des paiements avec les données suivantes:
  - “Rollno” est le nombre d’étudiants ayant le nom George Micheal.
  - Date de paiement est lundi précédent.
  - Le montant est le montant du solde à payer par l'étudiant.
- Afficher le nombre d'étudiants se sont joints à chaque mois de l'année en cours.