420-C42

Langages d'exploitation des bases de données

Partie 18

PL/pgSQL II

Variables, structures de contrôle et curseur

PL/pgSQL II

DECLARE - variables

- La déclaration de variable se fait obligatoirement dans la section DECLARE.
- Les types possibles sont :
  - tous les types du SQL
  - tous les types personnalisés : ENUM, DOMAIN, ...
  - le type d'un élément existant (en utilisant *xyz*%TYPE) :
    - type d'une variable existante : ma_variable%TYPE
    - Type d'une colonne de table existante : ma_table.ma_colonne%TYPE
  - les types d'un enregistrement spécifique (en utilisant *xyz*%ROWTYPE) :
    - d'une table existante : ma_table%ROWTYPE
    - types composés (pas vu dans le cadre du cours)
  - un enregistrement générique (record)

PL/pgSQL II

DECLARE - variables

- Synopsis de la déclaration d'une variable :

nom_var [CONSTANT] type [NOT NULL] [{ DEFAULT | := } expression];

- L'assignation se fait par l'opérateur (attention à =)
  :=

```
DO $$
DECLARE
    nombre1 INTEGER NOT NULL := 10;
    nombre2 INTEGER;
BEGIN
    nombre2 := nombre1 * 2 + 5;
    RAISE NOTICE 'Nombre 1: %', nombre1;
    RAISE NOTICE 'Nombre 2: %', nombre2;
END
$$;
```

PL/pgSQL II

SQL -> PL/pgSQL

- Le passage de SQL vers PL/pgSQL peut se faire de deux façons avec la clause **INTO** :
  - avec le DQL directement dans le SELECT
  - avec le DML en utilisant la clause RETURNING

```
id_employe employe.id%TYPE;
nombre_employe INT;
employe_du_mois employe%ROWTYPE;

--DQL
SELECT COUNT(*) INTO nombre_employe FROM employe;
SELECT * INTO employe_du_mois FROM employe WHERE nas = 123;

-- DML
INSERT INTO employe (nom) VALUES ('Lancelot') RETURNING id INTO id_employe;
```

PL/pgSQL II

RETURN

- Le mot réservé RETURN permet de terminer immédiatement :
  - le bloc anonyme sans valeur de retour
  - la procédure sans valeur de retour
  - la fonction avec une valeur de retour.

RETURN;

RETURN *expression*;

PL/pgSQL II

IF

- Il est possible de créer une structure de contrôle conditionnelle :

IF *expression-booleenne* THEN
  expression
ELSIF *expression-booleenne* THEN
  expression
ELSE
  expression
END IF;

PL/pgSQL II

CASE

- Le CASE s'utilise de deux façons en PL/pgSQL :

- CASE direct

```
-- la première expression correspondant à
-- l'expression de recherche est réalisée
-- si aucun match, else -> si aucun else,
-- exception CASE_NOT_FOUND
CASE expression-recherche
  WHEN expression, expression, … THEN
    expression
  WHEN expression, expression, … THEN
    expression
  ELSE
    expression
END CASE;
```

- CASE par recherche

```
-- l'expression de la première condition vraie
-- est réalisée
-- si aucun match, else -> si aucun else,
-- exception CASE_NOT_FOUND
CASE
  WHEN expression-booleenne THEN
    expression
  WHEN expression-booleenne THEN
    expression
  ELSE
    expression
END CASE;
```

PL/pgSQL II

LOOP

- Il est possible de faire une boucle contrôlée de l'intérieur :

```
LOOP
  expression

  IF expression-booleenne THEN -- première façon de contrôler la boucle
    EXIT; -- termine la boucle
  ELSIF expression-booleenne THEN
    CONTINUE; -- passe à la prochaine itération sans terminer l'itération courante
  ELSIF expression-booleenne THEN
    RETURN; -- termine le bloc en cours – nécessite la valeur de retour pour une fonction
  END IF

  EXIT WHEN expression-booleenne; -- deuxième façon de contrôler la boucle

END LOOP;
```

PL/pgSQL II

WHILE

- Il est possible de faire une boucle WHILE :

WHILE *expression-booleenne* LOOP
  expression
END LOOP;

PL/pgSQL II

FOR

- Il est possible de faire une boucle FOR pour une suite d'entiers :

FOR *nom_variable* IN [REVERSE] *de*..*a* [BY *increment*] LOOP

expression

END LOOP;

- Les valeurs *de*, *a* et *increment* doivent être des entiers et sont évalués une seule fois avant l'exécution de la boucle.
- La variable *nom_variable* est automatiquement un entier et vaudra, pour chaque itération, la valeur courante.

```
FOR i IN 1..5 LOOP                    RAISE NOTICE '%', i;    END LOOP;
FOR i IN REVERSE 5..1 BY 2 LOOP       RAISE NOTICE '%', i;    END LOOP;
```

PL/pgSQL II

FOR

- Il est possible de faire une boucle FOR parcourant le résultat d'une requête:

```
FOR nom_variable IN requête LOOP
  expression
END LOOP;
```

- La variable *nom_variable* doit préalablement être déclarée selon un type RECORD ou au type d'un enregistrement approprié.

```
DO LANGUAGE plpgsql $bloc$
DECLARE
  dep_rec departement%ROWTYPE;
BEGIN
  FOR dep_rec IN SELECT * FROM departement ORDER BY nom LOOP
    RAISE NOTICE 'Le département % se situe à %', dep_rec.nom, dep_rec.ville;
  END LOOP;
END $bloc$;
```

PL/pgSQL II

Curseur

- Un curseur est un objet encapsulant une requête.
- Le curseur offre plusieurs caractéristiques intéressantes :
  - les données peuvent être extraites une partie à la fois (très avantageux pour des requêtes retournant beaucoup d'information),
  - il est possible de positionner le curseur à n'importe quelle ligne de la requête,
  - il est possible de modifier ou supprimer les données de la ligne courante,
  - il peut être retourné par une fonction ou une procédure,
  - Il peut être passé en paramètre à une fonction ou à une procédure,
  - peut être ouvert ou fermé à volonté.
- Les curseurs sont très importants. PostgreSQL les utilise implicitement dans plusieurs situations (FOR *x* IN *requête* LOOP ... par exemple).

PL/pgSQL II

Curseur

- Exemple 1 – Curseur lié :

```
DO LANGUAGE plpgsql $bloc$
DECLARE
  dep_cur CURSOR FOR -- déclaration du curseur
       SELECT * FROM departement;
  dep_rec departement%ROWTYPE; -- déclaration de la variable de suivi, ligne par ligne
BEGIN
  OPEN dep_cur; -- ouverture du curseur
  LOOP -- parcours des données
    FETCH dep_cur INTO dep_rec; -- avance le curseur et accède au tuple courant
    EXIT WHEN NOT FOUND; -- quitte la boucle de parcours
    RAISE NOTICE 'Le département % se situe à %', dep_rec.nom, dep_rec.ville;
  END LOOP;

  CLOSE dep_cur; -- fermeture du curseur
END $bloc$;
```

PL/pgSQL II

Curseur

- Exemple 2 – Curseur lié paramétré :

```
DO LANGUAGE plpgsql $bloc$
DECLARE
  dep_cur CURSOR(v departement.ville%TYPE) FOR -- déclaration du curseur
       SELECT * FROM departement WHERE ville = v; -- le curseur est paramétré
  dep_rec departement%ROWTYPE; -- déclaration de la variable de suivi, ligne par ligne
BEGIN
  OPEN dep_cur('Montréal'); -- ouverture du curseur avec paramètre

  LOOP -- parcours des données
    FETCH dep_cur INTO dep_rec; -- avance le curseur et accède au tuple courant
    EXIT WHEN NOT FOUND; -- quitte la boucle de parcours
    RAISE NOTICE 'Le département % se situe à %', dep_rec.nom, dep_rec.ville;
  END LOOP;

  CLOSE dep_cur; -- fermeture du curseur
END $bloc$;
```

PL/pgSQL II

Curseur

- Exemple 3 – Curseur non lié :

```
DO LANGUAGE plpgsql $bloc$
DECLARE
  dep_cur REFCURSOR; -- déclaration du curseur
  dep_rec departement%ROWTYPE; -- déclaration de la variable de suivi, ligne par ligne
BEGIN
  OPEN dep_cur FOR -- ouverture du curseur
      SELECT * FROM departement;

  LOOP -- parcours des données
    FETCH dep_cur INTO dep_rec;
    EXIT WHEN NOT FOUND; -- quitte la boucle de parcours
    RAISE NOTICE 'Le département % se situe à %', dep_rec.nom, dep_rec.ville;
  END LOOP;

  CLOSE dep_cur; -- fermeture du curseur
END $bloc$;
```