Exploitation d’une Base de Données

Cours 5 - PL/pgSQL

Anaïs Durand

28 Mars 2023

IUT CLERMONT AUVERGNE
Aurillac - Clermont-Ferrand - Le Puy-en-Velay
Montluçon - Moulins - Vichy

PL/pgSQL

Procedural Language/PostgreSQL

- Langage procédural supporté par le SGBD PostgreSQL
  - Fonctions
  - Conditions
  - Boucles
  - ...

  $\Rightarrow$ Permet de faire des opérations plus complexes qu'une simple requête SQL.

- Similaire à Oracle PL/SQL

- 1ère version : 1998 (PostgreSQL 6.4)

Bloc PL/pgSQL

`DECLARE` *(facultatif)*

Déclaration des variables locales au bloc, des constantes, des exceptions, des curseurs.

`BEGIN`

Instructions PL/pgSQL et SQL
Possibilités de blocs imbriqués

`EXCEPTION` *(facultatif)*

Traitement des erreurs, des cas particuliers

`END;`

Bloc PL/pgSQL

Fichier de commandes SQL

```
SELECT ...
INSERT ...
INSERT ...
```

$\xrightarrow{\texttt{SELECT ...}}$

Moteur SQL

Bloc PL/pgSQL

Fichier de commandes SQL

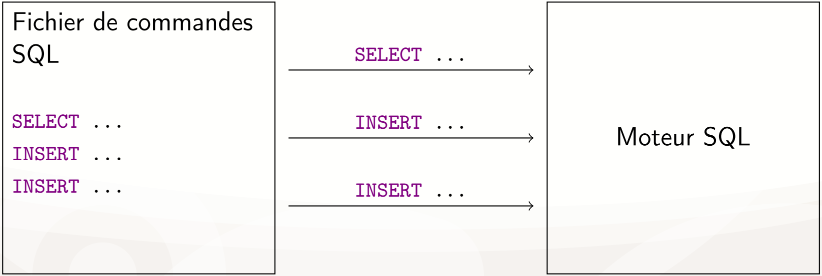
```
SELECT ...
INSERT ...
INSERT ...
```

$\xrightarrow{\texttt{SELECT ...}}$

$\xrightarrow{\texttt{INSERT ...}}$

Moteur SQL

Bloc PL/pgSQL



Bloc PL/pgSQL

Bloc PL/pgSQL

```
BEGIN ...
SELECT ...
IF ... THEN
INSERT ...
INSERT ...
END IF;
END;
```

```
BEGIN ...
SELECT ...
IF ... THEN
INSERT ...
INSERT ...
END IF;
END;
```

Moteur SQL

Bloc PL/pgSQL

- *Bloc anonyme :*

```
DO $$
    DECLARE
    ...
    BEGIN
    ...
    EXCEPTION
    ...
    END;
$$;
```

- *Fonction :*

```
CREATE FUNCTION ...
RETURNS ... AS $$
    DECLARE
    ...
    BEGIN
    ...
    EXCEPTION
    ...
    END;
$$ LANGUAGE plpgsql;
```

Déclaration de variable

```
DECLARE
    nom       varchar(50);
    nb        numeric := 12;
    id        Site.idSite%TYPE;
```

- `%TYPE`

Type de la variable = type de la colonne précisée

Ici le type de `Site.idSite`

Traitements - SELECT

`SELECT ... INTO ...` permet de récupérer le résultat d'une requête

Exemple :

```
DO $$
    DECLARE
        nbA numeric;
        nbB numeric;
    BEGIN
        SELECT count(*) INTO nbA
        FROM Etudiants
        WHERE annee = 1;

        SELECT count(*) INTO nbB
        FROM Etudiants
        WHERE annee = 2;

        RAISE NOTICE 'Il y a % étudiants de 1A et %
     étudiants de 2A.', nbA, nbB;
    END;
$$
```

Traitements - SELECT

- Si *plusieurs* lignes sont retournées : seule les valeurs de la *première* ligne sont mises dans les variables
- Si *aucune* ligne n'est retournée : les variables valent `NULL`

Traitements - SELECT

⚠

- Si *plusieurs* lignes sont retournées : seule les valeurs de la *première* ligne sont mises dans les variables
- Si *aucune* ligne n'est retournée : les variables valent `NULL`

Sinon, utiliser `STRICT` :

```
SELECT col1 , col2 INTO STRICT var1 , var2
....
```

Lève une exception s'il n'y a pas *exactement une* ligne retournée.

Traitements - Conditions

```
IF ... THEN ... ELSIF ... THEN ... ELSE ... END IF;
```

Exemple :

```
DO $$
    DECLARE
        n numeric;
    BEGIN
        SELECT note INTO n
        FROM Resultats
        WHERE examen='BDS201' AND etudiant='E0001';

        IF n < 10 THEN
            RAISE NOTICE 'Il faut retravailler !';
        ELSIF n < 15 THEN
            RAISE NOTICE 'Bien !';
        ELSE
            RAISE NOTICE 'Excellent !';
        END IF;
    END;
$$;
```

Traitements - Boucle WHILE

```
WHILE ... LOOP ... END LOOP;
```

Exemple :

```
DO $$
    DECLARE
       n   numeric := 2;
       p   numeric := 6;
       i   numeric := 1;
       res numeric := 1;
    BEGIN
       WHILE i <= p LOOP
           res := res*n;
           i := i+1;
       END LOOP;
       RAISE NOTICE '% puissance % = %', n, p, res;
    END;
$$;
```

Exceptions

- Instruction qui se passe mal ⇒ l'exécution du bloc cesse et une exception est levée

- Exemples d'exceptions PostgreSQL :

SQLSTATE	Nom	Détails
22012	division_by_zero	Division par zéro
23503	foreign_key_violation	Contrainte REFERENCES violée
23505	unique_violation	Contrainte UNIQUE violée
23514	check_violation	Contrainte CHECK violée
24000	invalid_cursor_definition	Problème de curseur
P0002	no_data_found	Pas de lignes retournées
P0003	too_many_rows	Plusieurs lignes retournées alors qu'on en attendait qu'une

Exceptions

Il est possible de capturer les exceptions pour les gérer dans la partie `EXCEPTION` (comme les `try ... catch` en C++).

Exemple :

```
DO $$
    DECLARE
        n   numeric := 0;
        res numeric;
    BEGIN
        res := 12 / n;
        RAISE NOTICE 'res = %', res;
    EXCEPTION
        WHEN division_by_zero THEN
            RAISE NOTICE 'Division par zero';
        WHEN OTHERS THEN
            RAISE NOTICE '% - %', SQLSTATE, SQLERRM;
    END;
$$;
```

Fonctions

- Permet de réutiliser un bloc
- Peut prendre des paramètres
- Peut retourner des valeurs (simples ou tables)

```
CREATE FUNCTION nom(param1, param2, ...)
RETURNS type_retour AS $$
    ....
$$ LANGUAGE plpgsql;
```

Surcharge possible : plusieurs fonctions de même nom, mais avec des paramètres différents

Fonctions

Exemple :

```
CREATE FUNCTION moyenne(etu Etudiant.code%TYPE)
RETURNS numeric AS $$
    DECLARE
        res numeric;
    BEGIN
        SELECT avg(note) INTO res
        FROM Resultat r
        WHERE e.code = etu;

        RETURN res;
    END;
$$ LANGUAGE plpgsql;

SELECT moyenne('E0001');

SELECT *, moyenne(code)
FROM Etudiant
WHERE moyenne(code) >= 12;
```

Fonctions

Exemple :

```sql
CREATE FUNCTION anniversaire(d date)
RETURNS TABLE(nom Etudiant.nom%TYPE,
              prenom Etudiant.prenom%TYPE) AS $$
    BEGIN
        RETURN QUERY SELECT nom, prenom
        FROM Etudiant
        WHERE dateNaiss = d;
    END;
$$ LANGUAGE plpgsql;

SELECT *
FROM anniversaire(current_date);
```