Base de données

Cours 8 - Introduction à PL/SQL

Table des matières

Qu'est-ce que PL/SQL ?

PL/SQL (Procedural Language/Structured Query Language) est un langage de programmation procédural développé par Oracle Corporation. Il étend le langage SQL standard en y ajoutant des fonctionnalités de programmation impérative telles que :

- Variables et constantes
- Structures de contrôle (boucles, conditions)
- Gestion d'erreurs et exceptions
- Procédures et fonctions

Architecture PL/SQL

Quand on exécute du code PL/SQL :

1. Le code est envoyé au moteur PL/SQL dans la base de données Oracle
2. Le moteur PL/SQL compile et exécute le code
3. Le moteur PL/SQL interagit avec le moteur SQL pour exécuter les commandes SQL
4. Les résultats sont renvoyés à l'utilisateur ou à l'application

Structure d'un Bloc PL/SQL

Un programme PL/SQL est structuré en blocs :

- **Déclaration** (optionnelle) : où les variables, constantes, types et curseurs sont définis
- **Exécution** (obligatoire) : où le code procédural est écrit
- **Gestion des exceptions** (optionnelle) : où les erreurs sont gérées

Structure d'un Bloc PL/SQL

```
1  DO $$  -- Start of an anonymous code block
2  [DECLARE]
3      -- Declarations of variables, constants, types, cursors
4
5  BEGIN
6      -- Executable statements
7
8  [EXCEPTION]
9      -- Error handling statements
10
11 END;
12 END $$;
```

Listing: `https://www.postgresql.org/docs/current/plpgsql-structure.html`

Types de Données

Un programme PL/SQL est structuré en blocs :

- **Numériques** : pour stocker des nombres
- **Caractères** : pour stocker du texte
- **Date et Heure** : pour stocker des informations temporelles
- **Booléens** : pour stocker des valeurs vraies ou fausses
- **Types composites** : pour regrouper plusieurs valeurs sous un même type

Types de données - Numériques

- **NUMERIC(p, s)** : Nombre avec précision (p) et échelle (s)
- **INTEGER** : Entier binaire
- **REAL** : Nombre à virgule flottante

Types de données - Caractères

- **CHAR(n)** : Chaîne de caractères de longueur fixe (n)
- **VARCHAR(n)** : Chaîne de caractères de longueur variable (n)
- **TEXT** : Chaîne de caractères de longueur variable

Types de données - Dates et Heures

- **DATE** : Date et heure (jusqu'à la seconde)
- **TIMESTAMP** : Date et heure (avec fractions de seconde)
- **INTERVAL** : Durée entre deux dates ou heures

Types de données - Booléens

- **BOOLEAN** : Valeurs TRUE, FALSE, ou NULL

Types de données - Types Composites

- **Records** : Structure regroupant plusieurs champs de types différents
- **Collections** : Tableaux (ARRAY) ou listes associatives (associative arrays)

Types Composites - Records

```
1  DO $$
2  DECLARE
3      v_student student%ROWTYPE; -- Gets structure from student table
4  BEGIN
5      -- Assign values
6      v_student.student_id := 1;
7      v_student.first_name := 'John';
8      v_student.last_name := 'Doe';
9
10     RAISE NOTICE 'Student: % % (ID: %)',
11         v_student.first_name,
12         v_student.last_name,
13         v_student.student_id;
14 END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-structure.html

Types Composites - Collections

```
1  DO $$
2  DECLARE
3      v_names TEXT[] := ARRAY['Alice', 'Bob', 'Charlie']; -- Text array
4      v_name TEXT;
5  BEGIN
6      -- Operations on the collection
7  END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-structure.html

Structures de Contrôle

Il existe plusieurs structures de contrôle en PL/SQL pour gérer le flux d'exécution :

- **Conditions** : IF, ELSIF, ELSE, CASE
- **Boucles** : LOOP, WHILE, FOR

Structures de Contrôle - Conditionnelles

- **IF** : Exécute un bloc de code si une condition est vraie
- **ELSIF** : Exécute un bloc de code si une condition alternative est vraie
- **ELSE** : Exécute un bloc de code si aucune des conditions précédentes n'est vraie
- **CASE** : Permet de choisir entre plusieurs blocs de code en fonction de la valeur d'une expression

Structures de Contrôle - Conditionnelles

```
1  DO $$
2  DECLARE
3      v_score NUMERIC := 85;
4      v_grade VARCHAR(2);
5  BEGIN
6      IF v_score >= 90THEN
7          v_grade := 'A';
8      ELSIF v_score >= 80THEN  -- NOTE: ELSIF (not ELSEIF)
9          v_grade := 'B';
10     ELSIF v_score >= 70THEN
11         v_grade := 'C';
12     ELSE
13         v_grade := 'D';
14     END IF;
15 END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-control-structures.html

Structures de Contrôle - Conditionnelles

```
1  CASE variable
2      WHEN value1 THEN instructions1
3      WHEN value2 THEN instructions2
4      ELSE default_instructions
5  END CASE;
6
7  -- OR --
8
9  CASE
10     WHEN condition1 THEN instructions1
11     WHEN condition2 THEN instructions2
12     ELSE default_instructions
13 END CASE;
```

Listing: `https://www.postgresql.org/docs/current/plpgsql-control-structures.html`

Structures de Contrôle - Boucles

- **LOOP** : Exécute un bloc de code de manière répétée jusqu'à ce qu'une condition de sortie soit rencontrée
- **WHILE** : Exécute un bloc de code tant qu'une condition est vraie
- **FOR** : Itère sur une plage de valeurs ou une collection

Structures de Contrôle - Boucles

```
1  DO $$
2  BEGIN
3      FOR i IN 1..5 LOOP
4          RAISE NOTICE 'Iteration %', i;
5      END LOOP;
6  END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-control-structures.html

Structures de Contrôle - Boucles

```
1  DO $$
2  DECLARE
3      v_counter INTEGER := 1;
4  BEGIN
5      WHILE v_counter <= 5LOOP
6          RAISE NOTICE 'Iteration %', v_counter;
7          v_counter := v_counter + 1;
8      END LOOP;
9  END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-control-structures.html

Structures de Contrôle - Boucles

```
1  DO $$
2  BEGIN
3      -- Ascending loop
4      FOR i IN 1..5 LOOP
5          RAISE NOTICE 'Ascending: %', i;
6      END LOOP;
7
8      -- Descending loop
9      FOR i IN REVERSE 1..5 LOOP
10         RAISE NOTICE 'Descending: %', i;
11     END LOOP;
12 END $$;
```

Listing: `https://www.postgresql.org/docs/current/plpgsql-control-structures.html`

Gestion des Exceptions

Il existe plusiers types d'exceptions en PL/SQL pour gérer les erreurs :

- **Exceptions prédéfinies** : Déjà définies
- **Exceptions utilisateur** : Définies par le programmeur
- **Exceptions non prédénies** : Attrapées avec OTHERS

Gestion des Exceptions

- **RAISE** : Lève une exception
- **EXCEPTION** : Bloc pour gérer les exceptions
- **WHEN** : Spécifie quelle exception gérer

Gestion des Exceptions

```
1  DO $$
2  DECLARE
3      v_student student%ROWTYPE;
4      e_grade_too_low EXCEPTION;
5  BEGIN
6      -- Some code that might raise error -20001
7      SELECT * INTO v_student FROM student WHERE id = 1;
8
9      IF v_student.grade < 60THEN
10         RAISE EXCEPTION 'Grade is too low!';
11     END IF;
12
13 EXCEPTION
14     WHEN NO_DATA_FOUND THEN
15         RAISE NOTICE 'Student with ID 1 not found';
16     WHEN OTHERS THEN
17         RAISE NOTICE 'An error occurred: %', SQLERRM;
18 END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-errors-and-messages.html

Curseurs et Gestion des Données

Un curseur est un pointeur vers le résultat d'une requête SQL. Il permet de traiter les lignes une par une.

- **Curseurs implicites** : Gérés automatiquement par PL/SQL pour les opérations DML (INSERT, UPDATE, DELETE)
- **Curseurs explicites** : Déclarés et contrôlés par le programmeur

Curseurs et Gestion des Données

```
1  -- Implicit cursor: Loop through students with ID < 100
2  DO $$
3  BEGIN
4      -- Implicit cursor: Loop through students with ID < 100
5      FOR student_rec IN
6          SELECT * FROM student WHERE student_id < 100
7      LOOP
8          RAISE NOTICE 'Student: %', student_rec.last_name;
9      END LOOP;
10 END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-cursors.html

Curseurs et Gestion des Données

```
1  -- Explicit cursor: Professors in a specific department
2  DO $$
3  DECLARE
4      -- Declare a REFCURSOR instead of CURSOR
5      c_professors_by_department REFCURSOR;
6      prof_record professor%ROWTYPE;
7  BEGIN
8      -- Explicit cursor: Professors in a specific department
9      OPEN c_professors_by_department FOR
10         SELECT * FROM professor
11         WHERE department_id = 10;
12
13     LOOP
14         FETCH c_professors_by_department INTO prof_record;
15         EXIT WHEN NOT FOUND;
16         RAISE NOTICE 'Professor: %', prof_record.last_name;
17     END LOOP;
18
19     CLOSE c_professors_by_department;
20 END $$;
```

Listing: https://www.postgresql.org/docs/current/plpgsql-cursors.html

Bibliographie

- https://www.postgresql.org/docs/current/plpgsql-overview.html
- https://www.postgresql.org/docs/current/plpgsql-structure.html
- https://www.postgresql.org/docs/current/plpgsql-declarations.html
- https://www.postgresql.org/docs/current/plpgsql-statements.html
- https://www.postgresql.org/docs/current/plpgsql-control-structures.html
- https://www.postgresql.org/docs/current/plpgsql-cursors.html
- https://www.postgresql.org/docs/current/plpgsql-errors-and-messages.html
- https://www.geeksforgeeks.org/plsql/pl-sql-data-types/