université
PARIS-SACLAY

Bases de données
PL/SQL – les bases

Silviu Maniu

2021/2022

Polytech App3

Contenu

PL/SQL

PL/SQL : Programming Language for SQL = SQL + langage procédural (impératif)

Rappel :

- **declaratif** (SQL) : dire **quoi**
- **procedural** : dire **comment**

PL/SQL : l'union entre SQL et un langage de programmation

PL/SQL – Pourquoi

Factorisation : faire fonction/procedure pour ne pas repeter des ordres SQL

Expressivité :

- tout n'est pas exprimable en SQL
- PL/SQL langage programmation = tout ce qui est calculable

Performance : PL/SQL envoyé au serveur une seule fois, compilé puis exécuté

PL/SQL – Pourquoi

Architecture en couches : sépare BD du reste de l'application

- traitements BD stockées dans la base (*procédure/fonction stockée*)
- **encapsulation** : chaque traitement est transparent pour l'utilisateur

Portabilité : toute la partie BD peut être changée sans conséquence sur l'interface (Web, programme)

PL/SQL – Comment

Etapes :

1. programmeur écrit des procédures et fonctions PL/SQL pour les traitements dans le cahier de charges
2. les procédures sont compilées et stockées dans la base
3. elles sont appelées quand besoin par les couches hautes (interface)

Depuis :

- mode interactif (SQL*Plus)
- mode programme (Java, PHP, etc.)

Contenu

Le langage PL/SQL

PL/SQL contient :

- partie **programmation** : les blocs, types, variables, operateurs, tests, boucles
- partie **BD** : ordres SQL, types

Exemple rapide :

```
create or replace function nombrelignes
is
  nblig integer;
begin
  select count(*) from table into nblig;
  dbms_output.put_line(nblig);
end;
/ -- seulement en SQL*Plus
```

Le langage PL/SQL

Déroulement :

- chargement en SQL*Plus = compilation + stockage
- message résultat : ok ou erreur de compilation (debug avec `show errors`)
- `set serveroutput on` – nécessaire pour affichage
- appel : `execute nombrelignes`

Bloc PL/SQL

Possibilité d'exécuter directement du code PL/SQL en SQL*Plus

Structure :

```
declare
  --variables, constantes, curseurs, etc.
begin
  -- ordres sql, instructions PL/SQL, structures controle
exceptions
  -- traitement des exceptions
end;
```

Bloc PL/SQL

Possibilité d'exécuter directement du code PL/SQL en SQL*Plus (**bloc ephemere**)

Exemple :

```
declare
  x varchar2(10);
begin
  x:= 'Hello World';
  dbms_output.put_line(x);
end;
```

PL/SQL – Variables

Déclarées après `is` ou `declare` (si bloc PL/SQL) et avant `begin`

Variable = nom + type (+ contrainte BD)

Types de variable :

- tous types BD (`integer`, `varchar2`, etc.)
- programmation : non-BD (booléen), dérivé

```
--programmation
adresse varchar2(20);
x integer:=1;
--BD
nom varchar2(10) not null;
```

PL/SQL – Variables

Types dérivés, référence à une entité existante

- programmation : variable
- BD : colonne, table, curseur

```
--programmation
x integer;
y x%type;
--BD
vnom client.nom%type;
cli client%rowtype;
```

PL/SQL – Variables

Affectation a une variable :

- programmation : résultat d'une **expression**
- BD : résultat **requête** (`select into` – seule ligne résultat, `fetch into` – curseur)

Exemple programmation :

```
x := 0;
vnom := 'Monsieur' || vnom;
y := (x + 5) * y;
```

PL/SQL – Variables

Exemple BD :

```
declare
  vref varchar2(10);
  vprix articles.prixht%type;
  cli client%rowtype;
begin
  select refart, prixht
    into vref, vprix
    from articles
    where nom = 'cravate';
  select *
    into cli
    from client
    where nom = 'Toto';
end;
```

PL/SQL – Opérateurs, tests

Opérateurs :

- programmation : $=$, $<$, $>$, !, $>=$, $<=$, `between`, `like`, `and`, `or`, etc.
- BD : comme en SQL

Tests (`if-then-else`) :

```
if vnocli = 10
then
  update ... ;
  insert ... ;
else delete ... ;
end if;
```

PL/SQL – Boucles

```
for n in 100 . . 110 loop
  insert into client(nocli) values (n);
end loop;

n := 100;
while n <= 110 loop
  insert into client(nocli) values (n);
  n := n + 1;
end loop;
```

PL/SQL – Procédures stockées (syntaxe)

```
create or replace procedure p
   (x1 in | out | inout t1, x2 ... , ...)
is
  ... -- déclarations (variables)
begin
  ... -- corps (ordres SQL, tests, boucles, etc.)
[exception ... ] -- gestion des exceptions
end;
```

Paramètres :

- `in` (par défaut) – variable entrée seulement
- `out` – variable sortie (référence)
- `inout` – variable entrée-sortie)

PL/SQL – Fonctions stockées (syntaxe)

```
create or replace function f (...)
    return integer /* ou autre */
is
  ...
begin
  ...
  return x;
end;
```

Supprimer procedure/fonction (ordre SQL) :

- `drop procedure p`
- `drop function f`

PL/SQL – Appels

Depuis bloc PL/SQL :

```
declare
        x char;
begin
  suppr(x);
  ...
end;
```

SQL*Plus :

```
execute suppr('W');
```

Note : pour appeler une fonction sous SQL*Plus il faut utiliser un bloc PL/SQL (voir feuilles exemples)

Compétences à acquérir

Analyse/implementation : savoir écrire des procédures et fonctions stockées utilisant des ordres SQL et des éléments de programmation (test, boucles, etc.)

Étapes :

1. écrire les ordres BD nécessaires
2. choisir procédure ou fonction
3. choix variables et types (paramètres, passage information entre ordres)
4. appel, debug, etc.

Remerciements

Le contenu de ce cours est grandement inspiré des cours d'Emmanuel Waller

- `https://www.lri.fr/~waller/`