420-C42

Langages d'exploitation des bases de données

Partie 12

Objets supplémentaires II
Procédures et fonctions

Procédure SQL et fonction SQL

Sommaire

- Le langage SQL rend possible la création de procédures et fonctions.
- Avant de les présenter, il est important de savoir :
  - qu'il existe plusieurs types de procédures et fonctions :
    - SQL : c'est le sujet de ces diapositives
    - PLPGSQL : que nous verrons plus loin
  - avec PostgreSQL, il est possible de substituer les apostrophes pour délimiter une chaîne de caractères - on utilise à la place des « *dollar-quoted* » ainsi : $*xyz*$ une chaîne de caractère $*xyz*$ -- où *xyz* est défini par l'utilisateur
  - avec PostgreSQL, la surcharge est possible – attention, si vous désirez modifier une procédure ou une fonction, vous devez d'abord la détruire sinon vous risquez de créer une surcharge.
- Ces objets sont semblables mais présentent toutefois quelques nuances dans leur structure et leurs intentions.

Procédure SQL et fonction SQL

Sommaire

- Similarités :
  - Contrairement aux requêtes préparées, les procédures et les fonctions sont persistantes dans la base de données.
  - Comme pour les requêtes préparées, les optimisations mentionnées sont réalisées lorsqu'elles s'appliquent (accroissement de la performance).
  - Dans les deux cas, elles permettent de créer un regroupement d'opérations paramétrables comme le ferait une fonction dans n'importe quel autre langage de programmation.
- Différences :
  - Une procédure ne retourne rien directement et doit être appelé explicitement.
  - Une fonction SQL doit retourner une valeur et peut être appelée dans n'importe quelle requête comme n'importe quelle fonction. Elle peut aussi être appelée explicitement.

Procédure SQL DDL

- Les instructions simplifiées du DDL sont :

CREATE [OR REPLACE] PROCEDURE *nom_procedure* ([*liste_arguments*])
  { LANGUAGE *nom_langage* }
  AS 'definition_de_la_procedure';
-- la liste d'arguments est une suite : [*mode*] [*nom*] *type* [{ DEFAULT | = } *valeur*]
-- où *mode* est IN, INOUT ou VARIADIC (IN par défaut)
-- *nom* est le nom de la variable du paramètre
-- *type* est le type de la variable - explicitement ou relatif : *table*.*colonne*%TYPE
-- *valeur* est la valeur par défaut
-- *nom_langage* est SQL ou PLPGSQL

DROP PROCEDURE [IF EXISTS] *nom_procedure* [CASCADE | RESTRICT];

- On utilise la clause CALL pour exécuter une procédure.

Procédure SQL

Exemple

- Un exemple :

```
CREATE OR REPLACE PROCEDURE ajout_employe_nouveau_patron(
        nas_emp employee.nas%TYPE, -- de loin préférable!
        nom_emp VARCHAR(32),
        prenom_emp VARCHAR(32),
        nom_departement department.name%TYPE)
  LANGUAGE SQL
  AS $$
    INSERT INTO employe(nas, nom, prenom, id_dep)
        VALUES(nas_emp, nom_emp, prenom_emp,
                    (SELECT id FROM departement WHERE nom = nom_departement));
    UPDATE departement
         SET responsable = (SELECT id FROM employe WHERE nas = nas_emp);
         WHERE (SELECT id FROM departement WHERE nom = nom_departement);
  $$;

CALL ajout_employe_nouveau_patron(123, 'Dupuis', 'Lancelot', 'Ventes');
```

Fonction SQL DDL

- Les instructions simplifiées du DDL sont :

CREATE [OR REPLACE] FUNCTION *nom_fonction* ([*liste_arguments*])
RETURNS *type_retour*
{ LANGUAGE *nom_langage* }
{ CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT }
AS 'definition_de_la_procedure';
-- la liste d'arguments est une suite : [*mode*] [*nom*] *type* [{ DEFAULT | = } *valeur*]
-- où *mode* est IN, INOUT ou VARIADIC (IN par défaut)
-- *nom* est le nom de la variable du paramètre
-- *type* est le type de la variable
-- *valeur* est la valeur par défaut
-- *nom_langage* est SQL ou PLPGSQL
-- *type_retour* est le type de retour de la fonction

DROP FUNCTION [IF EXISTS] *nom_fonction* [CASCADE | RESTRICT];

- On exécute la fonction comme n'importe quelle autre. Il est aussi possible d'appeler la fonction sans SELECT grâce à la clause PERFORM.

Fonction SQL

Exemple

- Un exemple :

```
CREATE OR REPLACE FUNCTION id_departement(
            nom_departement departement.nom%TYPE)
            RETURNS INT
  LANGUAGE SQL
  AS $$
    SELECT id FROM departement WHERE nom = nom_departement;
  $$;

SELECT id_departement('Ventes');
```

Fonction SQL

Exemple

- Un autre exemple :

```
CREATE OR REPLACE FUNCTION dernier_id_employe()
            RETURNS INT
  LANGUAGE SQL
  AS $$
    SELECT id FROM employe ORDER BY id DESC LIMIT 1;
  $$;

SELECT dernier_id_employe();
```

Autres objets

Nous verrons dans la section PL/SQL les :

- extensions des fonctions et des procédures
- déclencheurs