Exploitation d'une base de données

Département Informatique

IUT2 de Grenoble

BUT1 - Ressource 2.06

Introduction à l'exploration et l'administration des BD

1. Révisions : BD relationnelle et SQL simple
2. SQL pour explorer et analyser
3. SQL augmenté : *triggers*
4. Administration de 1er niveau

3. SQL augmenté : *triggers*

3.1. Triggers et fonctions simples

- Introduction
- Fonction en PL/pgSQL
- Trigger sur table
- Pptés : déclenchement, var. prédéf., validation/annulation

3.2. Triggers sur vues

3.3. Fonctions avancées en PL/pgSQL

- Déclaration de variables
- Instructions

PL/pgSQL

Syntaxe plus complète pour une fonction déclenchée par un trigger

CREATE FUNCTION *nom_fonc()* RETURNS trigger
AS $$
DECLARE
    déclarations
BEGIN
    suite de requêtes SQL de modifications de données
    et d'instructions
    RETURN *rec*;
END; $$ LANGUAGE 'plpgsql';

Commentaires

- - sur une ligne

/* ou sur plusieurs
lignes */

PL/pgSQL : déclarations

Exemples

***nom type*;**
***nom type* := *expression*;**
***nom* CONSTANT *type* := *expression*;**
...

Types utilisables

- les mêmes que pour les attributs de tables
- pour des n-uplets :
  - `RECORD`
  - `nom_table`
- pour une variable simple :
  - `nom_table.nom_att%TYPE`

PL/pgSQL : instructions (1/3)

Langage de programmation

- **var := val;**
- **IF . . . THEN . . . ELSIF . . . THEN . . . ELSE . . . END IF;**
- **FOR . . . IN [REVERSE] . . . LOOP . . . END LOOP;**
- **WHILE . . . LOOP . . . END LOOP;**

SQL

- **INSERT, UPDATE, CREATE TABLE . . .**
- **SELECT *attribut[,. . .]* INTO var[,. . .] FROM . . .**

PL/pgSQL : instructions (2/3)

Mélange

- **IF EXISTS(SELECT . . .)**
  THEN . . . END IF;
- **FOR var1,var2 in SELECT . . .**
  LOOP . . . END LOOP;

Variable pré-définie `FOUND`

- variable booléenne = statut du résultat d'une requête
- positionnée par la dernière exécution de
  - **SELECT . . . INTO**
  - **UPDATE, INSERT, DELETE**
  - **FOR**

PL/pgSQL : instructions (3/3)

Pour annuler : **RAISE EXCEPTION**

- annule requête + ensemble des actions incluses dans la fonction
- que la fonction soit déclenchée **`BEFORE`** ou **`AFTER`**

⚠ Attention

- pas de **`SELECT`** sans **`INTO`**
- toujours préférer une requête à une instr. de programmation :
  - **IF** $\leftrightarrow$ **WHERE**
  - **FOR** $\leftrightarrow$ **SUM, AVG,...**

Exemple d'application

Vérifier une contrainte *complexe* : entre différents tuples

- **2 versions d'un tuple** : elle porte sur **l'évolution** de la valeur d'un attribut.
- **2 tuples d'une même table** : comparaison entre **différentes occurrences** d'un attribut .
- **2 tuples de tables différentes** : comparaison entre les valeurs d'attributs **de deux tables**

$\Rightarrow$ exception levée lorsque la contrainte n'est pas respectée.