Bases de Données Avancées

Rapport du Projet en Binôme

41LIDF01 / 41LIDF02
Professeur : Walid Bechkit

Eoin Brereton Hurley & Angela Pineda

December 1, 2025

1 Introduction

2 Partie Guidée

2.1 Fonctions

Petite descriptions des fonctions...

- Fonctions de base

  1. `ajouter_auteur(nom, pays)` Cette fonction permet d'insérer le nom de famille et le pays d'origine d'un auteur dans la table `auteur`. Les paramètres de la fonction commencent par '`p_`' pour rendre le code plus lisible.

     ```
     CREATE OR REPLACE FUNCTION ajouter_auteur(p_nom TEXT, p_pays TEXT)
     RETURNS VOID AS $$
     BEGIN
             INSERT INTO auteur(nom, pays) VALUES (p_nom, p_pays);
     END;
     $$ LANGUAGE plpgsql;
     ```

  2. `ajouter_livre(titre, id_auteur, id_categorie, annee, nb_exemplaires)` Cette foncntion permet d'insérer un livre dans la table `livre`. Pourtant, la conception de cette fonction présuppose que l'on sait déjà l'identifiant de l'auteur et de la catégorie du livre. Il serait peut-être utile d'avoir une fonction qui fait le même chose que celle-ci mais qui prend des chaînes de caractères en paramètres.

     ```
     CREATE OR REPLACE FUNCTION ajouter_livre(titre TEXT, id_auteur INT,
         id_categorie INT, annee INT, nb_exemplaires INT)
     RETURNS VOID AS $$
     BEGIN
             IF nb_exemplaires > 0 THEN
                     INSERT INTO livre(titre, id_auteur, id_categorie,
                     annee, nb_exemplaires)
             VALUES (titre, id_auteur, id_categorie, annee, nb_exemplaires);
             END IF;
     END;
     $$ LANGUAGE plpgsql;
     ```

     On a donc conçu une fonctiton `ajouter_livre_texte`. À l'aide de cette nouvelle fonction, lors de l'insertion des données, on peut fournir le nom de famille de l'auteur et la catégorie en texte. La fonction récupère ensuite les identifiants associés pour les stocker dans la table.

     ```
     CREATE OR REPLACE FUNCTION ajouter_livre_texte(titre TEXT, nom_auteur TEXT,
     nom_categorie TEXT, annee INT, nb_exemplaires INT)
     ```

```
RETURNS VOID AS $$
DECLARE
        id_aut INT;
        id_cat INT;
BEGIN
        SELECT id_auteur FROM auteur WHERE nom = nom_auteur
        INTO id_aut;
        SELECT id_categorie FROM categorie WHERE nom = nom_categorie
        INTO id_cat;
        INSERT INTO livre(titre, id_auteur, id_categorie, annee,
        nb_exemplaires)
        VALUES (titre, id_aut, id_cat, annee, nb_exemplaires);
END
$$ LANGUAGE plpgsql;
```

3. `nb_livres_categorie(id_categorie)`

- Fonctions avec boucle
  1. `maj_annee_livres()`
- Fonction avec curseur
  1. `liste_livres_auteur(nom_auteur)`
- Fonction avec SQL dynamique
  1. `compter_elements(table_name TEXT)`
- Trigger guidé
  1. `verif_disponibilite`

3 Partie Ouverte

4 Figures

5 Annexes

5.1 Code pertinent

```
-- Partie 1: Fonctions de base

--Fonction qui insère un nouvel auteur dans la table auteur :
CREATE OR REPLACE FUNCTION ajouter_auteur(p_nom TEXT, p_pays TEXT)
RETURNS VOID AS $$
BEGIN
    IF p_nom IS NOT NULL THEN
```

```
        INSERT INTO auteur(nom, pays) VALUES (p_nom, p_pays);
                END IF;
END;
$$ LANGUAGE plpgsql;

--Tests :
SELECT ajouter_auteur('HUGO', 'France');
SELECT ajouter_auteur(NULL, 'France');
SELECT ajouter_auteur('JOYCE', NULL);
SELECT ajouter_auteur('WERBER', 'France');

--Pour voir les données :
SELECT * FROM auteur;
--Supprimer les données si besoin :
DELETE FROM auteur;
ALTER SEQUENCE auteur_id_auteur_seq RESTART WITH 1;

--Fonction qui ajoute un livre et vérifie que le nombre d'exemplaires est positif :
CREATE OR REPLACE FUNCTION ajouter_livre(titre TEXT, id_auteur INT, id_categorie INT,
RETURNS VOID AS $$
BEGIN
                IF nb_exemplaires > 0 THEN
                        INSERT INTO livre(titre, id_auteur, id_categorie, annee, nb_
                END IF;
END;
$$ LANGUAGE plpgsql;

--propuesta angela
CREATE OR REPLACE FUNCTION ajouter_livre(p_titre TEXT, p_id_auteur INT, p_id_categori
RETURNS VOID AS $$
BEGIN
    IF p_titre IS NOT NULL AND p_nb_exemplaires >= 0 THEN
        INSERT INTO livre(titre, id_auteur, id_categorie, annee, nb_exemplaires)
        VALUES (p_titre, p_id_auteur, p_id_categorie, p_annee, p_nb_exemplaires);
    END IF;
END;
$$ LANGUAGE plpgsql;


--Tests :
--id_auteur = 1 : "WERBER"

SELECT ajouter_livre('Les Fourmis', 4, 2, 1991, 10);
SELECT ajouter_livre('Le Dernier Jour d''un Condamnné', 7, 1, 1991, 10);
```

```
SELECT ajouter_livre('Exemple', 6, 2, 1991, 10);


/*
--Fonction qui ajoute un livre et vérifie que le nombre d'exemplaires est positif :
CREATE OR REPLACE FUNCTION ajouter_livre_test(titre TEXT, nom_auteur TEXT, id_categor
        DECLARE
                id_aut INT;
        BEGIN
                SELECT id_auteur FROM auteur WHERE nom = nom_auteur INTO id_aut;
                INSERT INTO livre(titre, id_auteur, id_categorie, annee, nb_exemplair
        END
$$ LANGUAGE plpgsql;

--Tests :
SELECT ajouter_livre_test('Les Fourmis', 'WERBER', NULL, 1991, 10);
SELECT ajouter_livre_test('Le Dernier Jour d''un Condamné', 'HUGO', NULL, 1829, 10);
*/

--Pour voir les données :
SELECT * FROM  livre;
--Supprimer les données si besoin :
DELETE FROM livre;
ALTER SEQUENCE livre_id_livre_seq RESTART WITH 1;


--Fonction qui insère une nouvelle catégorie dans la table catégorie :
CREATE OR REPLACE FUNCTION ajouter_categorie(p_nom TEXT) RETURNS VOID AS $$
        BEGIN
                IF p_nom IS NOT NULL THEN
                        INSERT INTO categorie(nom) VALUES(p_nom);
                END IF;
        END
$$ LANGUAGE plpgsql;


--Test categorie:
SELECT ajouter_categorie('Biographie');
SELECT ajouter_categorie('Drame');
SELECT ajouter_categorie('Roman');
SELECT ajouter_categorie('Science');
SELECT ajouter_categorie('Essai');

--Pour voir les données :
```

```sql
SELECT * FROM categorie;

--Fonction qui retourne le nombre total de livres appartenant à une catégorie donnée.
CREATE OR REPLACE FUNCTION nb_livres_categorie(p_id_categorie INT)
RETURNS INT AS $$
    DECLARE
                    nom_livres INT;
    BEGIN
                    SELECT COUNT (*) INTO nom_livres FROM livre WHERE id_categorie =
                RETURN nom_livres;
END
$$ LANGUAGE plpgsql;

SELECT nb_livres_categorie(3);

-- Partie 2: Fonctions avec boucle
--test

-- Partie 3: Fonctions avec curseur

-- Partie 4: Fonctions avec SQL dynamique

-- Partie 5: Trigger guidé
```