UNIVERSITÉ
DE MONTPELLIER

Rapport de projet:

Base de données répertoriant des recettes de cuisine

Réalisé par le Groupe 7:

Madani BENSIKHALED - 21801055
Sio BANDO - 21802005
Ozgur DOGAN - 21811290
Joffrey PUJADE - 22011040

Sommaire

I - Description du thème et sujet choisi

Le besoin initial était de développer un bot Discord francophone intégrant la possibilité de demander une recette de cuisine.

Vu que nous n'avons pas trouvé d'API francophone nous convenant pour notre projet, nous avons décidé de la créer nous-mêmes.

Notre projet consiste à créer un site où l'on remplit un formulaire pour ajouter une recette dans la base de données. L'API, quant à elle, ne contiendra que peu de requêtes. Nous allons implémenter la possibilité de chercher une recette ou de voir la totalité des recettes.

La base de données occupe une place centrale dans ce projet car elle regroupe toutes les informations du site et, par extension, de l'API. Elle sera légèrement modifiée à la fin du projet pour l'intégration dans le site.

Dans ce site, les **utilisateurs** sont identifiés par leur numéro d'identifiant. Un utilisateur est défini par un e-mail, un mot de passe, un nom, un prénom, un âge, un genre et un niveau de cuisine (3 niveaux : débutant, intermédiaire et avancé).

Une **recette** est identifiée par un numéro d'identifiant. Une recette est définie par un nom de recette, un nom dans l'API, un texte des instructions, pays d'origine, par sa difficulté de réalisation et une durée de réalisation. Une recette peut être un cocktail ou un plat.

Un **ingrédient** est identifié par un numéro d'identifiant. Il est défini par un nom, une unité de mesure, et un nombre de calories (par unité de mesure).

Un **type de diète** est identifié par un numéro d'identifiant. Il est défini par un nom de diète.

Un utilisateur peut consulter des recettes. Il peut aussi liker (aimer) une ou plusieurs recettes. Il peut aussi proposer une ou plusieurs recettes (La recette n'est proposée que par un unique utilisateur.). Il est possible pour un utilisateur de commenter une recette, le commentaire est posté à une date qui est identifiée par la date du commentaire. Le commentaire est défini par un texte de remarque. Un utilisateur peut être allergique à un ou plusieurs ingrédients.

Une recette contient des ingrédients, au minimum un ingrédient, en quantité déclarée. Une recette peut convenir à un type de diète ou plusieurs. Une recette peut en accompagner une autre, une recette peut en accompagner uniquement une autre et une recette peut être accompagnée que par une seule recette.

Ce projet a été réalisé avec le SGBD **PostgreSQL**.

II - Dictionnaire de données

Table : Utilisateur

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idUtilisateur	Identifiant unique de l'utilisateur	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire et unique	Donnée élémentaire
nom	Nom de famille de l'utilisateur	Chaîne de caractères	Inférieure ou égale à 25	Facultatif	Donnée élémentaire
prenom	Prénom de l'utilisateur	Chaîne de caractères	Inférieure ou égale à 25	Facultatif	Donnée élémentaire
age	Age de l'utilisateur	Numérique (Entier)	Inférieure ou égale à 3	Facultatif	Donnée élémentaire
genre	Genre de l'utilisateur	Chaîne de caractères	Egale à 1	Facultatif	Donnée élémentaire
nomUtilisateur	Pseudonyme de l'utilisateur	Chaîne de caractères	Inférieure ou égale à 50	Obligatoire	Donnée élémentaire
email	Adresse mail électronique de l'utilisateur	Chaîne de caractères	Inférieure ou égale à 150	Obligatoire	Donnée élémentaire
motDePasse	Mot de passe secret de l'utilisateur	Chaîne de caractères	Inférieure ou égale à 64	Obligatoire	Donnée élémentaire
niveauDeCuisine	Le niveau de maîtrise en cuisine de l'utilisateur	Numérique (Entier)	Egale à 1	Facultatif	Donnée élémentaire

Table : Recette

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idRecette	Identifiant unique de le recette	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire et unique	Donnée élémentaire
paysOriginaire	Pays d'origine de la recette	Chaîne de caractères	Inférieure ou égale à 20	Facultatif	Donnée élémentaire
nomRecette	Nom de la recette	Chaîne de caractères	Inférieure ou égale à 50	Facultatif	Donnée élémentaire
nomRecetteApi	Nom de la recette pour l'API	Chaîne de caractères	Inférieure ou égale à 55	Facultatif	Donnée élémentaire
diffculte	Difficulté de réalisation de la recette	Numérique (Entier)	Egale à 1	Facultatif	Donnée élémentaire
tempsPreparation	Temps de préparation	Numérique (Entier)	Inférieure ou égale à 3	Facultatif	Donnée élémentaire
preparation	Instructions de la préparation	Texte	Non définie	Facultatif	Donnée élémentaire
typeRecette	Type de la recette (Plat ou Cocktail)	Chaîne de caractères	Inférieure ou égale à 15	Facultatif	Donnée élémentaire
idUtilisateur	Identifiant de l'utilisateur qui propose la recette	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
recetteAccompagnant	Autre recette qui accompagne la recette	Numérique (Entier)	Inférieure ou égale à 4	Facultatif	Donnée élémentaire

Table : Ingredient

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idIngredient	Identifiant unique de l'ingrédient	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire et unique	Donnée élémentaire
nom	Nom de l'ingrédient	Chaîne de caractères	Inférieure ou égale à 25	Facultatif	Donnée élémentaire
calorie	Nombre de calories par unité de mesure	Flottant	Non définie	Facultatif	Donnée élémentaire
unite	Unité de mesure (Gramme/Unité)	Chaîne de caractères	Inférieure ou égale à 10	Facultatif	Donnée élémentaire

Table : Contient

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idRecette	Identifiant de la recette	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
idIngredient	Identifiant de l'ingredient que contient la recette	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
quantite	Quantité de l'ingrédient dans la recette	Numérique (Entier)	Inférieure ou égale à 4	Facultatif	Donnée élémentaire

Table : TypeDiete

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idDiete	Identifiant unique de la diète	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
nomDiete	Nom de la diète	Chaîne de caractères	Inférieure ou égale à 20	Facultatif	Donnée élémentaire

Table : Respecte

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idRecette	Identifiant de la recette	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
idDiete	Identifiant de la diète que respecte la recette	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire

Table : Commente

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idUtilisateur	Identifiant de l'utilisateur qui commente	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
idRecette	Identifiant de la recette qui est commentée	Numérique (Entier)	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
dateCommentaire	Date du commentaire	Date (Horodatage)	Non définie	Obligatoire	Donnée élémentaire
texteCommentaire	Texte que contient le commentaire	Texte	Non définie	Facultatif	Donnée élémentaire

Table : Aime

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idUtilisateur	Identifiant de l'utilisateur qui ajoute un "like"	Numérique	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
idRecette	Identifiant de la recette est qui est aimée	Numérique	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire

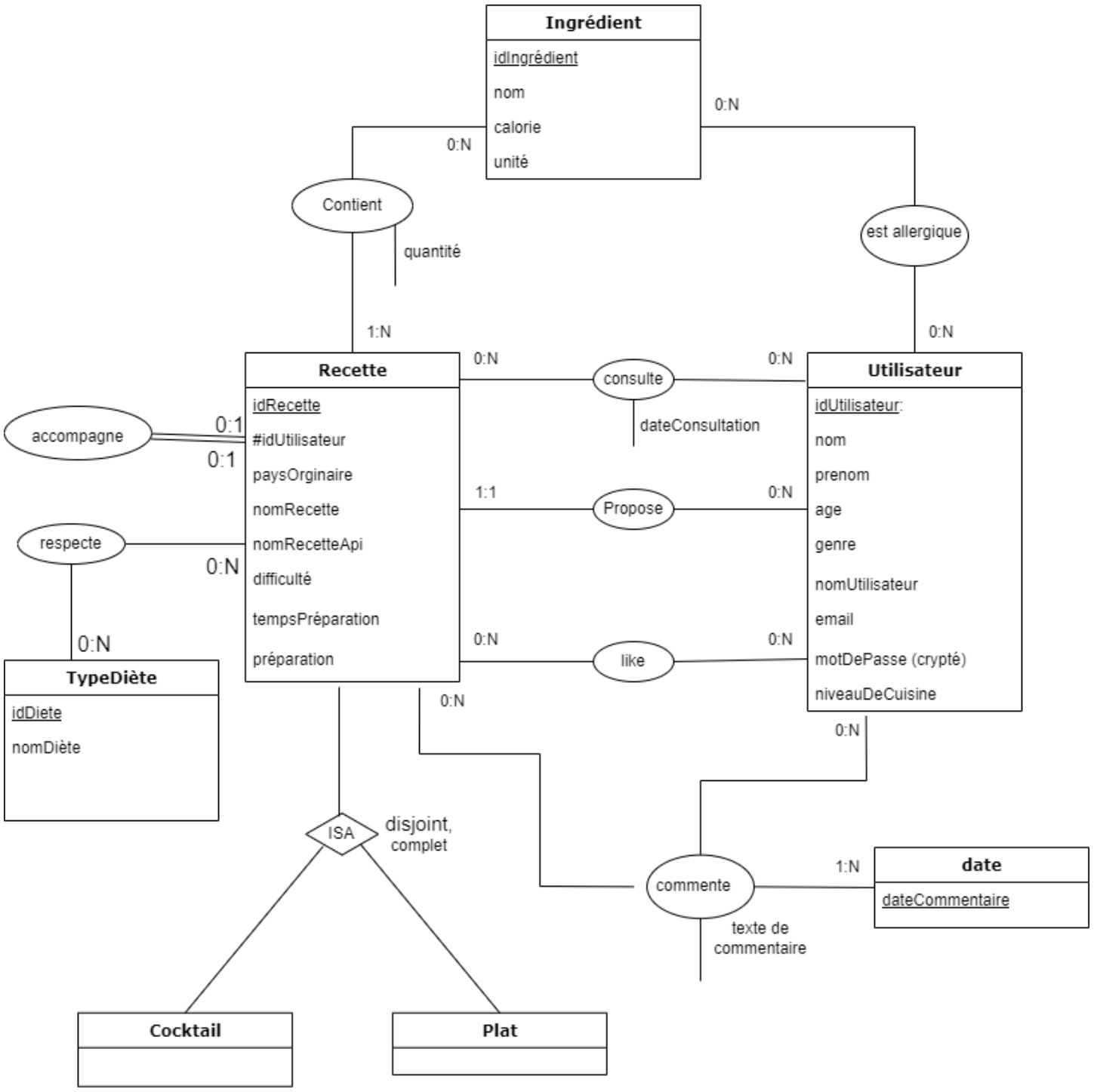
Table : Consulte

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
dateConsultation	Date de la consultation de la recette	Date (Horodatage)	Non définie	Obligatoire	Donnée élémentaire
idUtilisateur	Identifiant de l'utilisateur qui consulte	Numérique	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
idRecette	Identifiant de la recette qui est consultée	Numérique	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire

Table : Allergique

Code de l'attribut	Description	Type de données	Longueur	Nature	Règle de calcul
idUtilisateur	Identifiant de l'utilisateur qui est allergique	Numérique	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire
idIngredient	Identifiant de l'ingrédient allergène	Numérique	Inférieure ou égale à 4	Obligatoire	Donnée élémentaire

III - Schéma entité-association



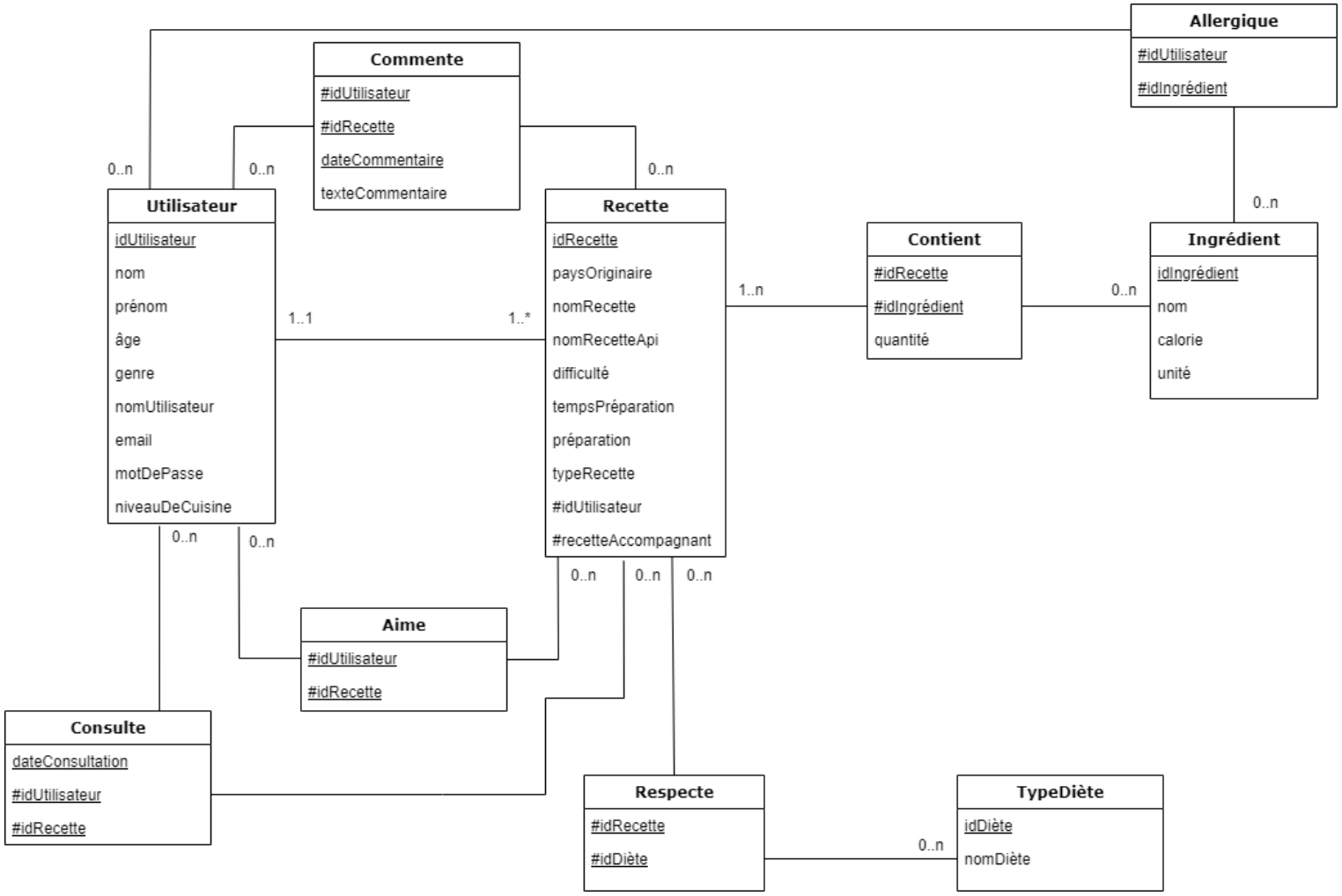
IV - Schéma relationnel associé

Schéma relationnel :

- Utilisateur(idUtilisateur, nom, prenom, age, genre, nomUtilisateur, email, motDePasse, niveauDeCuisine)
- Recette(idRecette, paysOriginaire, nomRecette, nomRecetteApi, difficulté, tempsPréparation, préparation, typeRecette, #idUtilisateur, #recetteAccompagnant)
- Ingrédient(idIngrédient, nom, calorie, unité)

- Contient(#idRecette, #idIngrédient, quantité)
- TypeDiète(idDiete, nomDiete)
- Respecte(#idRecette, #idDiete)
- Commente(#idUtilisateur, #idReccette, dateCommentaire, texteCommentaire)
- Aime(#idUtilisateur, #idRecette)
- Consulte(dateConsultation, #idUtilisateur, #idRecette)
- Allergique(#idUtilisateur, #idIngrédient)

V - Schéma physique associé



VI - Requêtes accompagnées d'une explication

Nous avons fait 5 requêtes :

1) La première requête utilise le GROUP BY :

```
SELECT nomRecette, COUNT(nomRecette) AS nombre_de_consultation FROM Recette
  INNER JOIN Consulte ON Recette.idRecette = Consulte.idRecette
  INNER JOIN Utilisateur On Utilisateur.idUtilisateur = Consulte.idUtilisateur
    WHERE Utilisateur.age BETWEEN 30 AND 40 GROUP BY nomRecette
      ORDER BY nombre_de_consultation DESC LIMIT 3;
```

Elle permet de trouver les 3 premières recettes les plus consultées pour les utilisateurs qui ont entre 30 et 40 ans.

2) La deuxième requête utilise une division :

```
SELECT nomRecette FROM Recette
   WHERE NOT EXISTS (SELECT * FROM Utilisateur
      WHERE NOT EXISTS (
         SELECT * FROM Consulte WHERE Recette.idRecette = Consulte.idRecette AND
                              Consulte.idUtilisateur = Utilisateur.idUtilisateur));
```

Elle permet de chercher les recettes qui ont été consultées par tous les utilisateurs.

3) La troisième requête utilise une sous-requête :

```
SELECT nomRecette FROM Recette
   INNER JOIN Consulte ON Recette.idRecette = Consulte.idRecette
      WHERE Consulte.idRecette NOT IN (SELECT idRecette FROM AIME)
      GROUP BY nomRecette;
```

Elle permet de trouver les recettes qui ont été consultées mais qui n'ont pas reçu de "like".

4) La quatrième requête utilise plusieurs sous-requêtes et sous sous-requêtes :

```
SELECT nomRecette FROM Recette
   WHERE idRecette IN (SELECT idRecette FROM Consulte)
      AND idRecette IN (SELECT idRecette FROM Respecte
                        WHERE idDiete IN (SELECT idDiete FROM Respecte WHERE  idRecette = (SELECT
idRecette FROM RECETTE WHERE nomRecette='Virgin Cuba Libre'))
                           GROUP BY idRecette HAVING COUNT(*)=(SELECT count(*) FROM Respecte WHERE
idRecette = (SELECT idRecette FROM RECETTE WHERE nomRecette='Virgin Cuba Libre')));
```

Elle permet de trouver les recettes qui ont été consultées et qui respectent exactement les mêmes types de diète que la recette "Virgin Cuba Libre".

5) La cinquième requête utilise une sous-requête corrélée :

```
SELECT U.nom, U.prenom, nomRecette FROM Utilisateur U
   INNER JOIN Recette ON U.idUtilisateur = Recette.idUtilisateur
      WHERE Recette.difficulte = (SELECT MAX(difficulte) FROM Recette WHERE idUtilisateur =
U.idUtilisateur);
```

Elle permet de trouver les recettes de plus grande difficulté de réalisation pour chaque utilisateur.

VII - Explication des procédures, fonctions et triggers associés

VII - 1) Explication des procédures

obtenirUneRecette

Procédure obtenirUneRecette(-> idR : Entier, <-> resRec : Indéfini, <-> resCont : Indéfini, <-> resIngr : Indéfini)

Entrée :

idR : Identifiant d'une recette.

Sortie :

resRec : Toutes les données d'une recette.

resCont : Toutes les relation entre la recette et tous ses ingrédients.

resIngr : Tous les ingrédients de la recette.

Traitement : Recherche une recette et en renvoie toutes ses données ainsi que la totalité de ses ingrédients.

VII - 2) Explication des fonctions

nombreLikes

Fonction nombreLikes(idR : Entier) : Entier
Entrée : Identifiant d'une recette.
Sortie : Nombre de likes de la recette.
Traitement : Calcule le nombre de likes d'une recette.

nombreCalories

Fonction nombreCalories(idR : Entier) : Réel
Entrée : Identifiant d'une recette.
Sortie : Nombre de calories de la recette.
Traitement : Calcule le nombre de likes d'une recette.

fctInsertionModificationRecette

Fonction fctInsertionModificationRecette() : Trigger
Entrée : Rien / Aucune.
Sortie : Un trigger.
Traitement : Vérifie que le nouvel identifiant de la recette est bel et bien différent l'identifiant de la recette accompagnante, que l'auteur n'est pas allergique à l'un des ingrédients de sa propre recette, ainsi que la difficulté de la recette ne dépasse pas le niveau de cuisine de son auteur.

fctInsertionModificationContient

Fonction fctInsertionModificationContient() : Trigger
Entrée : Rien / Aucune.
Sortie : Un trigger.
Traitement : Vérifie que l'auteur n'est pas allergique à l'un des ingrédients de sa propre recette, sinon ladite recette est supprimée et une erreur est lancée.

fctInsertionModificationAllergique

Fonction fctInsertionModificationAllergique() : Trigger
Entrée : Rien / Aucune.
Sortie : Un trigger.
Traitement : Vérifie que si l'utilisateur a déjà écrit des recettes lorsqu'il ajoute des ingrédients auquel il est allergique, il n'a pas posté de recettes possédant des ingrédients auxquels il est allergique.

VII - 3) Explication des triggers

insertionModificationRecette

Trigger insertionModificationRecette

Ce trigger vérifie, lors de l'insertion ou de la modification dans la table recette,
pour chaque colonne, si le nouvel identifiant de la recette est bel et bien différent
l'identifiant de la recette accompagnante, que l'auteur n'est pas allergique à l'un
des ingrédients de sa propre recette, ainsi que la difficulté de la recette ne dépasse pas le niveau de cuisine de son auteur.

insertionModificationContient

Trigger insertionModificationContient
Ce trigger vérifie, lors de l'insertion ou de la modification dans la table contient,
pour chaque colonne, si l'auteur n'est pas allergique à l'un des ingrédients de sa
propre recette, sinon ladite recette est supprimée et une erreur est lancée.

insertionModificationAllergique

Trigger insertionModificationAllergique
Ce trigger vérifie, lors de l'insertion ou de la modification dans la table allergique,
pour chaque colonne, si l'utilisateur a déjà écrit des recettes lorsqu'il ajoute des
ingrédients auxquels il est allergique, il n'a pas posté de recettes possédant des ingrédients auxquels il est allergique.

VIII - Tests effectués

1) Les premiers tests effectués vérifient que le trigger insertionModificationRecette() fonctionne bien :

```
INSERT INTO Recette VALUES (100, 'Italie', 'Pâtes', 'Pâtes', 1, 10, '1) Mettre les pattes dans l'eau bouillante pendant 10 minutes.' , 'PLAT', 000, 100);
```

Ici, on insère une recette qui s'accompagne elle-même. Le trigger renvoie alors une erreur pour empêcher ce type d'insertion. Le trigger fonctionne donc bien.

```
INSERT INTO Allergique VALUES (011, 016);
INSERT INTO Recette VALUES (100, 'Italie', 'Pâtes', 'Pâtes', 1, 10, '1) Mettre les pâtes dans l'eau bouillante pendant 10 minutes.' , 'PLAT', 11, null);
INSERT INTO Contient VALUES (100, 016, 3);
```

Ici, on fait en sorte qu'un utilisateur propose un plat qui contient un ingrédient dont il est allergique. Le trigger renvoie alors une erreur pour empêcher ce type d'insertion. Le trigger fonctionne donc bien.

```
INSERT INTO Utilisateur VALUES(012, 'utilisateur12', 'uti12', 38, 'N', 'use1r', 'sudosu@bel.com', 'azerty123', 0);
```

Ici, on fait en sorte d'insérer un utilisateur avec un niveau de cuisine égal à 0. Le trigger renvoie alors une erreur pour empêcher ce type d'insertion. Le trigger fonctionne donc bien.

```
INSERT INTO Recette VALUES (50, 'Italie', 'Pâtes2', 'Pâtes2', 2, 10, '1) Mettre les pâtes dans l"eau bouillante pendant 10 minutes.' , 'PLAT', 9, NULL);
```

Ici, on vérifie que le niveau de l'utilisateur ne peut pas être inférieur au niveau requis pour la recette qu'il vient de proposer. Le trigger renvoie alors une erreur pour empêcher ce type d'insertion. Le trigger fonctionne donc bien.

2) Le deuxième test vérifie que le trigger fctInsertionModificationAllergique() fonctionne bien :

```
INSERT INTO Allergique VALUES (3, 1);
```

Ici, on ajoute une allergie à un utilisateur. Cette utilisateur a déjà proposé autrefois une recette contenant l'ingrédient dont il est maintenant allergique. Le trigger renvoie alors une erreur pour empêcher ce type d'insertion. Le trigger fonctionne donc bien.

3) Le troisième test vérifie que la fonction nombreCalories() fonctionne :

```
SELECT nomRecette, typeRecette, nombreCalories(idRecette) AS Calories FROM Recette ORDER BY Calories DESC;
```

Ici, on insère une requête qui affiche toutes les recettes et leur nombre de calories total par ordre décroissant.

4) Le quatrième test vérifie que la fonction nombreLikes() fonctionne:

```
SELECT nomRecette, typeRecette, nombreLikes(idRecette) AS nombre_de_likes FROM Recette ORDER BY nombre_de_likes DESC;
```

Ici, on insère une requête qui affiche toutes les recettes avec leur nombre de likes par ordre décroissant.

5) Le cinquième test vérifie que la procédure obtenirUneRecette() fonctionne bien :

```
DROP FUNCTION IF EXISTS appel1() CASCADE;
DROP FUNCTION IF EXISTS appel2() CASCADE;
DROP FUNCTION IF EXISTS appel3() CASCADE;
DROP FUNCTION IF EXISTS appelProcedure() CASCADE;

CREATE OR REPLACE FUNCTION appel1() RETURNS TEXT AS $$
DECLARE
  recetteChoisie RECORD;
  contientRecetteChoisie RECORD;
  ingredientsRecetteChoisie RECORD;
BEGIN
  CALL obtenirUneRecette(1, recetteChoisie, contientRecetteChoisie, ingredientsRecetteChoisie);

  RETURN recetteChoisie;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION appel2() RETURNS TEXT AS $$
DECLARE
  recetteChoisie RECORD;
  contientRecetteChoisie RECORD;
  ingredientsRecetteChoisie RECORD;
BEGIN
  CALL obtenirUneRecette(1, recetteChoisie, contientRecetteChoisie, ingredientsRecetteChoisie);

  RETURN contientRecetteChoisie;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION appel3() RETURNS TEXT AS $$
DECLARE
  recetteChoisie RECORD;
  contientRecetteChoisie RECORD;
  ingredientsRecetteChoisie RECORD;
BEGIN
  CALL obtenirUneRecette(1, recetteChoisie, contientRecetteChoisie, ingredientsRecetteChoisie);

  RETURN ingredientsRecetteChoisie;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION appelProcedure() RETURNS TEXT AS $$
BEGIN
  RETURN appel1() || appel2() || appel3();
END;
$$ LANGUAGE plpgsql;

SELECT nomRecette, appelProcedure() AS appel FROM Recette WHERE idRecette = 1;

DROP FUNCTION IF EXISTS appel1() CASCADE;
DROP FUNCTION IF EXISTS appel2() CASCADE;
DROP FUNCTION IF EXISTS appel3() CASCADE;
DROP FUNCTION IF EXISTS appelProcedure() CASCADE;
```

Ici, on teste la procédure avec une recette choisie aléatoirement. 4 fonctions sont créées pour l'appel de la procédure, elles sont présentes pour convertir en chaîne de caractères les résultats de la procédure. L'affichage du résultat dans la console de commandes est un peu dur à lire, mais on y voit bel et bien tous les résultats escomptés. La procédure fonctionne donc bien.