Rapport IT Projet

**BDD :**

Notre BDD est la suivante :

Diagram

Description automatically generated

De ce fait, notre script SQL pour implémenter la BDD est le suivant (avec les implémentations demandés) :

drop database projetjava;  
  
create database if not exists projetjava;  
  
use projetjava;  
  
CREATE TABLE Category(  
 CategoryID INT AUTO\_INCREMENT,  
 CategoryName VARCHAR(50),  
 PRIMARY KEY(CategoryID)  
);  
  
INSERT INTO Category VALUES (1, 'policy');  
INSERT INTO Category VALUES (2, 'plan');  
INSERT INTO Category VALUES (3, 'report');  
INSERT INTO Category VALUES (4, 'receipt');  
INSERT INTO Category VALUES (5, 'order');  
  
CREATE TABLE Topic(  
 TopicID INT AUTO\_INCREMENT,  
 TopicName VARCHAR(50),  
 PRIMARY KEY(TopicID)  
);  
  
INSERT INTO Topic VALUES (1, 'CS243 Course Files in Fall 2021');  
INSERT INTO Topic VALUES (2, 'Cluster Graduation Projet en 2022');  
  
CREATE TABLE Tag(  
 TagID INT AUTO\_INCREMENT,  
 TagName VARCHAR(50),  
 PRIMARY KEY(TagID)  
);  
  
INSERT INTO Tag VALUES (1, 'legal');  
INSERT INTO Tag VALUES (2, 'medical');  
INSERT INTO Tag VALUES (3, 'administrative');  
INSERT INTO Tag VALUES (4, 'technical');  
INSERT INTO Tag VALUES (5, 'annee2022');  
INSERT INTO Tag VALUES (6, 'reporting');  
  
CREATE TABLE Document(  
 DocumentID INT AUTO\_INCREMENT,  
 DocumentName VARCHAR(50),  
 DocumentDate DATE,  
 StorageAddress VARCHAR(50),  
 TopicID INT NOT NULL,  
 CategoryID INT NOT NULL,  
 PRIMARY KEY(DocumentID),  
 FOREIGN KEY(TopicID) REFERENCES Topic(TopicID),  
 FOREIGN KEY(CategoryID) REFERENCES Category(CategoryID)  
);  
  
CREATE TABLE contenir(  
 DocumentID INT,  
 TagID INT,  
 PRIMARY KEY(DocumentID, TagID),  
 FOREIGN KEY(DocumentID) REFERENCES Document(DocumentID),  
 FOREIGN KEY(TagID) REFERENCES Tag(TagID)  
);

Comme un document est composé de Topic et de Catégorie des tables précédentes alors ceci explique pourquoi on a des Foreign clef sur ces mêmes clefs primaires.

De plus pour la liaison contenir comme elle lie Document et Tag avec n, n on est obligé d’avoir des Foreign Clef sur cette table dans le MLD, pour ne pas lier des Objets qui ne sont pas présent dans la BDD.

Dans notre code java, nos class ont les mêmes paramètres que les tables dans la BDD.

**Document :**

Pour créer un document on utilise un constructeur comme suit :

public Document(String DocumentName, java.sql.Date DocumentDate, String StorageAdress, Topic topic, Category category, List<Tag> tagList){  
 System.*out*.println("\nCreating a document in Process...");  
 this.DocumentName = DocumentName;  
 this.DocumentDate = DocumentDate;  
 this.StorageAdress = StorageAdress;  
 this.topic = new Topic(topic);  
 this.category = new Category(category);  
 for (Tag tag : tagList){  
 Tag newTag = new Tag(tag);  
 this.tagList.add(newTag);  
 }  
 System.*out*.println("...Done Creating a document");  
}

Le fait de créer nos attributs à partir des attributs eux même exemple :

Tag newTag = new Tag(tag);

Est u au fait que on doit créer cet objet pour qu’il s’instancie dans la BDD et qu’il n’y est pas d’erreurs de Foreign clef comme évoqué précédemment.

Pour mieux comprendre, on a décidé de créer une classe abstraite Table dans le constructeur est le suivant :

public Table(String Name){  
 this.Name = Name;  
 findId();  
}

Ainsi à chaque fois qu’un objet est créé on va chercher sa clef primaire et si elle n’existe pas, alors on ajoute cet élément à la BDD et à la Table correspondante (nous avons bien fait attention à mettre la fonction findId() en privée car on ne l’utilise que dans cette classe abstraite).

L’utilité de cette classe abstraite est que pour les 3 tables, Category, Topic et Tag on a besoin de trouver les ID et d’ajouter ces éléments si l’ID n’est pas déjà répertorié. De ce fait pour savoir dans quelle Table nous nous situons on a utilisé dans Table la méthode suivante :

private String NameSubClass = String.*valueOf*(Table.super.getClass()).substring(6);

Et donc pour effectuer ce dont on a besoin on procède de la façon suivante :

*/\*\*  
 \* trouver l'id de la sous classe, si il n'existe pas, on l'ajoute  
 \** ***@return*** *void  
 \** ***@params*** *none  
 \*/*private void findId(){  
 System.*out*.println("\nFind ID of " + this.NameSubClass + " in Process...");  
 int id = 0;  
 //Rechercher en fonction des paramètre de la sous classe  
 final String sql = "SELECT " + this.NameSubClass + "ID FROM " + this.NameSubClass + " WHERE " + this.NameSubClass + "Name = ?";  
 try (Connection con = DriverManager.*getConnection*(Main.*dbUrl*, Main.*user*, Main.*password*);  
 PreparedStatement stmt = con.prepareStatement(sql)) {  
 //rechercher le ID à partir du Name  
 stmt.setString(1, this.Name);  
 ResultSet rs = stmt.executeQuery();  
 //lister toutes les category par Name  
 while (rs.next()) {  
 id = rs.getInt(this.NameSubClass + "ID");  
 }  
 //si existe deja alors fixe l'id  
 if (id != 0){  
 this.ID = id;  
 }  
 //sinon ajoute à la BDD puis fixe l'id  
 else {  
 this.ajouterBdd();  
 while (rs.next()) {  
 this.ID = rs.getInt(this.NameSubClass + "ID");  
 }  
 }  
 }  
 //Exception mauvaise requete SQL  
 catch (SQLException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("...Done Find ID of " + this.NameSubClass);  
}  
  
*/\*\*  
 \* ajoute a la sous table un nouvel element  
 \** ***@return*** *void  
 \** ***@params*** *none  
 \*/*private void ajouterBdd(){  
 System.*out*.println("\nAdding a new " + this.NameSubClass + " in Process...");  
 final String sql = "INSERT INTO " + this.NameSubClass + " VALUES (0, ?)";  
 //Rechercher en fonction des paramètre de la sous classe  
 try (Connection con = DriverManager.*getConnection*(Main.*dbUrl*, Main.*user*, Main.*password*);  
 PreparedStatement stmt = con.prepareStatement(sql)) {  
 //set les paramètres de category  
 stmt.setString(1, Name);  
 stmt.executeUpdate();  
 //Exception mauvaise requete SQL  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("...Done Adding a new " + this.NameSubClass);  
}

Ainsi lorsqu’on créera un Document avec des nouvelles données pas répertorié dans la BDD, alors elles s’ajoueront.

Enfin on va créer un document de la même manière que précédemment pour les autres tables. Et à la fin de l’ajout on va trouver son ID et grâce à cet ID on va faire les liens avec les Tags dans Contenir de la manière suivante :

Dans Document :

*/\*\*  
 \* faire le lien entre Document et Tag, en ajoutant les combinaisons dans Contenir  
 \** ***@return*** *void  
 \** ***@params*** *none  
 \*/*private void lienContenir(){  
 this.findId();  
 //ajouter tous les liens entre Document et Tag  
 for (Tag tag : tagList){  
 //Si nouveau Tag detectee alors creation  
 tag.findCombinaisonContenir(this.DocumentID);  
 }  
}

Dans Tag :

*/\*\*  
 \* trouver la combinaison de DocumentID et TagID si il n'existe pas, on l'ajoute  
 \** ***@return*** *void  
 \** ***@params*** *none  
 \*/*public void findCombinaisonContenir(int IdDocument){  
 System.*out*.println("\nFind ID of Contenir in Process...");  
 Map<Integer, Integer> map = new HashMap<>();  
 //Rechercher en fonction des paramètre de la sous classe  
 final String sql = "SELECT DocumentID, TagID FROM Contenir";  
 try (Connection con = DriverManager.*getConnection*(Main.*dbUrl*, Main.*user*, Main.*password*);  
 PreparedStatement stmt = con.prepareStatement(sql)) {  
 ResultSet rs = stmt.executeQuery();  
 //lister toutes les relations TagID / docID par ID  
 while (rs.next()) {  
 int DocumentID = rs.getInt("DocumentID");  
 int TagID = rs.getInt("TagID");  
 map.put(DocumentID, TagID);  
 }  
 //si il n'y a pas la combinaison voulu, alors on l'ajoute  
 if (!(map.containsKey(IdDocument) && map.containsValue(super.getID()) && map.get(IdDocument).equals(super.getID()))) {  
 this.ajouterBddContenir(IdDocument);  
 }  
 }  
 //Exception mauvaise requete SQL  
 catch (SQLException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("...Done Find ID of Contenir");  
}  
  
*/\*\*  
 \* ajoute a la Contenir la combinaion voulu  
 \*  
 \** ***@return*** *void  
 \** ***@params*** *int  
 \*/*private void ajouterBddContenir(int IdDocument){  
 System.*out*.println("\nAdding a new Contenir in Process...");  
 final String sql = "INSERT INTO Contenir VALUES (?, ?)";  
 //Rechercher en fonction des paramètre de la sous classe  
 try (Connection con = DriverManager.*getConnection*(Main.*dbUrl*, Main.*user*, Main.*password*);  
 PreparedStatement stmt = con.prepareStatement(sql)) {  
 //set les paramètres de category  
 stmt.setInt(1, IdDocument);  
 stmt.setInt(2, this.getID());  
 stmt.executeUpdate();  
 //Exception mauvaise requete SQL  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("...Done Adding a new Contenir");  
}

Pour les combinaisons d’ID entre Document et Tag on utilise une Map pour les lier est vérifier si les liens existe, si non alors on les ajoutes.