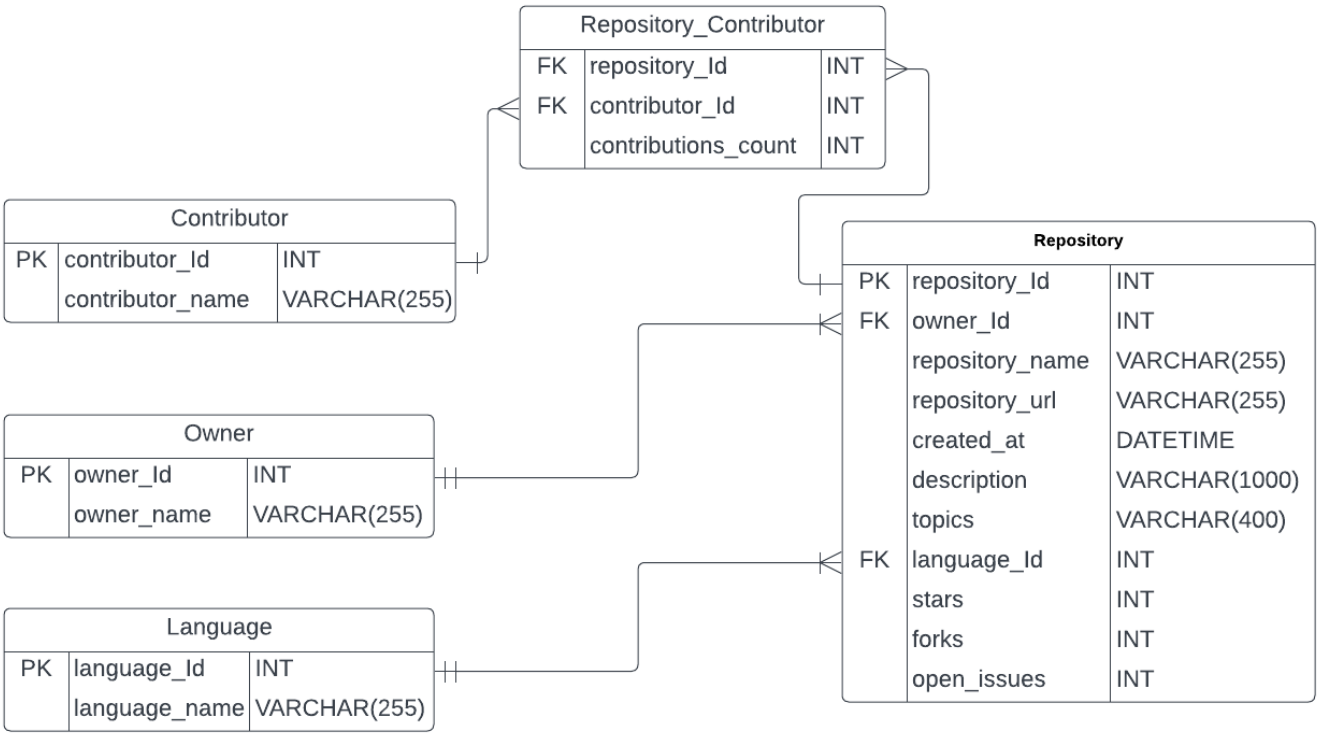
Modélisation et stockage des données GitHub dans une base de données relationnelle :

Introduction :

Ce rapport décrit les étapes réalisées pour importer les données d'un fichier CSV dans une base de données SQL Server. En outre, nous avons effectué des opérations d'indexation pour améliorer les performances des requêtes.

Conception :

Avant d'importer les données dans la base de données SQL Server, la première étape consiste à concevoir la modélisation de la base de données. Cela comprend la création de classes, leurs attributs et les relations entre elles. Voici la modélisation conceptuelle de la base de données :



Après la conception :

1-Import des données depuis le fichier CSV dans Jupyter :
Dans cette première partie, nous avons importé les données du fichier CSV 'cleaaaaaan.csv' dans un DataFrame pandas. Le DataFrame contient les colonnes suivantes : 'repository', 'owner', 'repository_url', 'created_at', 'description', 'topics', 'language', 'stars', 'forks', 'open_issues', 'contributor', 'contributions'.

2- Nettoyage des données :
Nous avons créé des DataFrames pour chaque classe dont nous aurons besoin plus tard, notamment 'contributor', 'owner', 'language', et 'repository'.

3- Création d'identifiants uniques pour les entités :
Afin de mieux gérer les relations entre les différentes entités, nous avons créé des identifiants uniques pour les colonnes 'contributor', 'owner', 'language', et 'repository'. Ces identifiants permettront de faire référence à ces entités de manière plus efficace dans la base de données.

4- Préparation des tables pour l'insertion dans la base de données :
Nous avons organisé les DataFrames en vue de leur insertion dans les tables de la base de données SQL Server. Nous avons sélectionné les colonnes nécessaires pour chaque table et nous les avons organisées dans le bon ordre.

5- Création des tables dans la base de données :
À l'aide de la bibliothèque pyodbc, nous avons créé les tables 'Contributor', 'Owner', 'Language', 'Repository', et 'Repository_Contributor' dans la base de données SQL Server. Chaque table possède ses propres colonnes avec des contraintes et des clés primaires.

6- Insertion des données dans les tables de la base de données :
Nous avons itéré à travers chaque DataFrame pour insérer les données dans les tables correspondantes de la base de données. Avant chaque insertion, nous avons vérifié si la ligne existe déjà dans la table pour éviter les doublons.

7- Création d'index pour améliorer les performances des requêtes :
Afin d'accélérer les requêtes de recherche dans la base de données, nous avons créé des index sur les colonnes 'repository_name', 'contributor_name', 'language_name', et 'owner_name' des tables 'Repository', 'Contributor', 'Language', et 'Owner', respectivement.

8- Mesure des performances avant et après l'indexation :
Nous avons effectué une requête pour chercher un nom de référentiel spécifique avant et après la création de l'index sur la colonne 'repository_name'. Nous avons mesuré le temps d'exécution des deux requêtes pour comparer les performances avant et après l'indexation.

Conclusion :

Le rapport présente les principales étapes pour importer des données d'un fichier CSV dans une base de données SQL Server et optimiser les performances avec des index.