PROJET BD I : SPJRUD to SQL

Implémentation en Python d'un outil de compilation de requêtes SPJRUD vers des requêtes SQL

Problèmes rencontrés :

Le principal problème rencontré a été le choix de la structure : comment structurer le code ?

Plusieurs manières ont été envisagé, il a été décidé de diviser de la manière suivante :

- Un fichier general.py, qui contient toutes les classes de base : *Database, Relation, Operation* et ses classes filles
- Un fichier expression.py, contenant toutes les expressions SPJRUD et le code permettant de le compiler en SQL

Un autre problème majeur était lié au timing, enchainer les autres projets, suivre les cours tout en étudiant pour les examens était une tâche ardue. À part cela, aucun réel problème n'a été rencontré.

Choix d'implémentation :

Toutes les expressions sont une classe fille de la classe *Expression*. Elle contient tout ce qu'il y a de commun entre les expressions. Seul l'attribut old_tables semble un peu plus particulier : Il permet de garder en mémoire la table d'origine, utile pour les sous expressions, permettant d'accéder facilement aux colonnes d'une table.

Le select ainsi que toutes les autres expressions ont été conçus sur a même structure :
-verify()
-compile()
-to_SQL(database)

La première méthode vérifie que le programme fonctionne sans erreurs, la méthode compile retourne un string correspondant à la requête SQL, et la dernière méthode exécute cette commande.

Le select a été conçu de la manière suivante : si on fait une opération entre une constante et une colonne, la colonne sera inscrite à gauche et la constante à droite. On peut faire une opération entre deux colonnes mais pas entre deux constantes.

Aucune remarque sur les autres expressions à part qu'elles permettent toutes l'utilisation de sous expressions

Pour faciliter la manipulation des bases de données et des tables, deux classes ont été créées, elles permettent de simplifier les opérations tout en rendant le tout beaucoup plus pratique, avec l'utilisation intuitive des noms.

Les fonctions utilitaires demandées sont implémentées :

- les méthodes __str__ de chaque expressions
- to_table(), dans *Expr*
- il faut print la méthode to_SQL()

Il est à noter que même si les erreurs sont implémentées, elles ne sont pas aussi détaillées que prévues, manque de temps.

Exemple d'utilisation :

database = Database('path of your database')

Ensuite, il est conseillée de créer manuellement la ou les tables de la base de données pour faciliter l'utilisation, mais nous verrons plus bas que ce n'est pas obligatoire

TABLE = Relation(database, 'name of your relation')

Ensuite, voici comment utiliser les sous expressions :

```
select = Select(Operation(("column"), Cst(value)), Relation or Expr)
print(select.to_SQL(database))
```

Attention, il faut indiquer le nom de l'opération voulue (Eq, St, …) et non pas Operation

```
project = Proj(["column1", "column2"], Relation or Expr)
print(project.to_SQL(database))

join = Join(Relation or Expr, Relation or Expr)
print(join.to_SQL(database))

rename = Rename('old name', 'new name', Relation)

union = Union(Relation or Expr, Relation or Expr)
print(union.to_SQL(database))

difference = Difference(Relation or Expr, Relation or Expr)
print(diff.to_SQL(database))
```