RAZE Nicolas

RAPPORT
→ Langage & Compilation: ←

LIEN GITHUB REPOSITORY=
https://github.com/Sw0ocHCorp/BiLang_Project

DOMAIN MODEL=

Voir programme abstractsyntax

SYNTAXE CONCRÈTE=

RÈGLE TASK (SYNTAXE DU PROGRAMME GLOBAL)=

```
Task returns Task:
    'Task'name=ID":"
        "load" fileextractor+=FileExtractor (datafiltering+=DataFiltering)?
        (fileextractor+= FileExtractor (datafiltering+= DataFiltering)?)*
        '[dashboard:' dashboard=DashBoard']';
```

RÈGLE FILEEXTRACTOR=

```
FileExtractor returns FileExtractor:
    name=EString "=>"
    path=EString ("null replacement->" nullreplacement+= NullReplacement
    ("," nullreplacement+=NullReplacement)*)?;
```

RÈGLE FILTERINGSTEP=

```
FilteringStep returns FilteringStep:
    QuantitativeFiltering | QualitativeFiltering;
```

RÈGLE PREPROCESSINGSTEP=

```
PreprocessingStep returns PreprocessingStep:
    MathOperation | ColReference | StatisticalOperation;
```

RÈGLE PREPROCESSINGSTEP=

```
Plot returns Plot:
    BarPlot | LinePlot | DonutPlot | ScatterPlot | PolarPlot | RadarPlot
    | PiePlot;
```

RÈGLE DATAFILTERING=

```
DataFiltering returns DataFiltering:
    "=>" fileextractor=[FileExtractor|EString]
        ('processing=' processingstep+=PreprocessingStep
            ("," processingstep+=PreprocessingStep)* )?
        ('filtering=' filteringstep+=FilteringStep
            ( "," filteringstep+=FilteringStep)*)? ;
```

RÈGLE DASHBOARD=

```
DashBoard returns DashBoard:
        name=EString "=>" (fileextractor+=[FileExtractor|EString]
                'plots' '{'
                        plot+=Plot
                        (plot+=Plot)*
                        '}')+ ;
```

RÈGLE NULLREPLACEMENT=

```
NullReplacement returns NullReplacement:
        colName= EString "=" (label= EString
                                        | statisticaloperation= StatisticalOperation) ;
```

RÈGLE QUANTITATIVEFILTERING=

```
QuantitativeFiltering returns QuantitativeFiltering:
        {QuantitativeFiltering}
        axis= EString operator= QuantitativeOperator values= EFloat;
```

RÈGLE QUALITATIVEFILTERING=

```
QualitativeFiltering returns QualitativeFiltering:
        {QualitativeFiltering}
        axis= EString operator= QualitativeOperator labels= EString;
```

RÈGLE MATHOPERATION=

```
MathOperation returns MathOperation:
        (newColName= EString "=")? ("(")? 'l' lside=PreprocessingStep
        operator=MathOperator 'r' rside=PreprocessingStep (")")?;
```

RÈGLE COLREFERENCE=

```
ColReference returns ColReference:
        {ColReference}
        target= EString;
```

RÈGLE STATISTICALOPERATION=

```
StatisticalOperation returns StatisticalOperation:
        operator=StatisticalOperator"("colreference=ColReference")";
```

RÈGLE POUR CHACUNS DES TYPES DE PLOTS (Exemple LINEPLOT)=

```
LinePlot returns LinePlot:
        {LinePlot}
        'line' name=ID "->" 'xAx' xAxis=EString 'yA' yAxis=EString
        ('loc' location=EInt)? ('colors' colors=EString)?
        ('thickness' thickness=EFloat)?;
```

RÈGLE ENUM MATHOPERATOR=

```
enum MathOperator returns MathOperator:
        PLUS = '+' | MINUS = '-' | MULTIPLY = '*' | DIVIDING = '/';
```

RÈGLE ENUM STATISTICALOPERATOR=

```
enum StatisticalOperator returns StatisticalOperator:
    MEAN = 'mean' | MEDIAN = 'median' | STD = 'std';
```

RÈGLE ENUM QUALITATIVEOPERATOR=

```
enum QualitativeOperator returns QualitativeOperator:
    EQUALS= "= " | NOT= "!= "
```

RÈGLE ENUM QUANTITATIVEOPERATOR=

```
enum QuantitativeOperator returns QuantitativeOperator:
    EQUAL= "=" | INFERIOR= "<" | SUPERIOR= ">"
```

DESCRIPTION DU LANGAGE=

Pour développer mon DSL, BiLang, je me suis inspiré des logiciels de Business Intelligence comme PowerBI par exemple. J'ai essayé de développer un langage permettant de réaliser les tâches de bases afin de pouvoir produire des dashboard simples mais efficaces.

Le programme de génération de code se passe en plusieurs étapes.

- Tout d'abord, nous avons l'étape de chargement des données. Durant le chargement, l'utilisateur peut choisir de combler les valeurs manquantes. Lorsqu'une opération mathématique / statistique pour effectuer le remplacement est choisie, l'algorithme d'opérations est exécuté. Cet algorithme stocke en arborescence les différents composants de l'opération à effectuer, si plusieurs termes sont spécifiés.
- La phase de filtrage s'effectue en se déplaçant dans le fichier cible et supprimer les données dont la condition de filtrage n'est pas vérifiée
- La phase de preprocessing permet à l'utilisateur d'exécuter l'algorithme d'opération afin de créer de nouvelles colonnes(caractéristiques) à partir des autres colonnes.
- La phase de comptage permet à l'utilisateur de créer une nouvelle structure de données utilisable avec certains types de graphiques

(pie, doughnut), dans laquelle se trouve les occurence de valeur dans des colonnes précises du fichier source.

- Enfin, durant la phase de création du Dashboard, on stocke les informations des graphiques dans une structure de données avant de générer le code correspondant pour pouvoir l'organiser à la convenance de l'utilisateur (notamment la position des graphiques dans le dashboard). Pour afficher le Dashboard, on va écrire les données à afficher en but, allouer une position à chaques graphiques pour les tracer.

EXEMPLE DE SCÉNARIOS=

Les fichiers CSV sources utilisés pour ces exemples se trouvent en pièces jointe de mon mail de rendu

DASHBOARD SCENARIO#1 & SCENARIO#2(Uniquement les pays sans valeurs inconnues dans CHACUNES des colonnes)=

```
Task t1:
        load f1 =>
"C:/Users/nclsr/OneDrive/Bureau/Cours_L3IA/Compilation_DSL/Backup_Projets/cars.
csv"
                        -> f1
                                filtering=
                                        "Horsepower" > 100.0, "Origin" = "US"

                f2 =>
"C:/Users/nclsr/OneDrive/Bureau/Cours_L3IA/Compilation_DSL/Backup_Projets/fact
book.csv"
                        null replacement-> "ALL"="NULL"
                        -> f2
                                filtering="ALL" != "NULL"


        dashboard:
                d1=> f1 plots {
                                bar l1-> xAx "Car" yA "Horsepower"
                        }
                        f2 plots {
                                line l2-> xAx "Country" yA "Population,
Telephones_mobile_cellular" colors "#EC401D, #49F60D"
                        }
```

MISE EN RELATION DES NOTES DES CÉRÉALES AVEC LEURS COMPOSITION=

```
Task t2:
        load f2 =>
"C:/Users/nclsr/OneDrive/Bureau/Cours_L3IA/Compilation_DSL/Backup_Projets/cere
al.csv"
                -> f2
```

```
                    filtering= "calories" < 110.0
                    count= "fat" {
                                < 2.0 -> "Good Cereal"
                                2.0 [] 3.0 -> "Average Cereal"
                                > 3.0 -> "Bad Cereal"
                    }= "countCereal"

        dashboard:
                d2 => f2 plots{
                    radar r2-> xAx "name" yA "rating"
                    line l2-> xAx "name" yA "calories, fiber, carbo, sugars"
                    pie p2 -> countID "countCereal"
                }
```

EXEMPLE “PROOF OF CONCEPT” PERMETTANT DE MONTRER LA SYNTAXE CONCRÈTES DES “DERNIÈRES FONCTIONNALITÉS=

```
Task t1:
        load f1 =>
"C:/Users/nclsr/OneDrive/Bureau/Cours_L3IA/Compilation_DSL/Backup_Projets/cars.
csv"
                    -> f1
                            filtering=  "Horsepower" > 90.0, "Origin" = "Japan"
                            processing= "debug"=l (l "Displacement" + r "Weight") + r (l
(l "MPG" + r median("Acceleration")) - r "Cylinders")
                            count= "Weight" {
                                    < 1500.0 -> "Light Car"
                                    1500.0 [] 2500.0 -> "Medium Car"
                                    > 2500.0 -> "Heavy Car"
                            }= "countWeight"

        dashboard:
                d1 => f1 plots{

                    bar b1-> xAx "Car" yA "debug, Displacement, Horsepower"
                                                loc 1 colors "#EC401D, #49F60D, #B604EF"
thickness 2.5
                    line l1-> xAx "Car" yA "Horsepower" loc 2
                    line l2 -> xAx "Car" yA "Weight" loc 0
                    pie p1 -> countID "countWeight"

                }
```

CAS D'USAGE DU LANGAGE=

Tout le code est présent dans le repo github ou dans le dossier zip en pièce jointe de mon mail de rendu

Malgré qu'il y ai quelques fonctionnalités à ajouter, ce langage a pour but d'être utilisé pour réaliser des petits projets de BI ou des croquis de dashboard à retravailler en détails avec des outils de BI plus puissants.

L'utilisateur peut, grâce à ce langage, créer un dashboard "dynamique", les sources de ce dashboard peuvent être multiples. Pour afficher des données précises, l'utilisateur peut filtrer les données en fonction des différentes composantes (colonnes) des fichiers sources(CSV), il peut aussi créer de nouvelles composantes résultant d'opérations mathématiques / statistiques de colonnes du fichier source, enfin, si le fichier présente des données manquantes, l'utilisateur peut les remplacer, de manière générale ou spécifiques en fonction des différentes colonnes du fichier.