REPORTE GRAMATICAL

A continuación se presentan un extracto de la granática ascendente utilizada para el proyecto SQL Parser, también se presenta la gramatica descendente convertida para poder trabajar con un analizador descendente.

Se tomó la decision de trabajar con la gramatica ascendente por las siguientes caracteristicas:

1. **Tytus-SQL Parser** está creado en Phyton utilizando PLY y YACC,PLY es una implementación de las herramientas de análisis lex y yacc para Python.
2. Ply utiliza el análisis sintáctico LR, también conocidos como Parser LR, son un tipo de analizadores para gramáticas libres de contexto. Pertenece a la familia de los **analizadores ascendentes**, ya que construyen el árbol sintáctico de las hojas hacia la raíz.
3. Despues de realizar y generar el AST para las dos versiones de gramáticas, la gramática ascendente funcionaba mejor y más rápido con las reglas asociadas a cada producción y con menos generacón de errores.
4. Con la gramática descendete, a varias producciones se les quitó la recursividad, al tomar esta acción se generaron producciones Epsilon(ε) con reglas **<empty>** asociadas, después de algunas corridas de prueba, se creaba un error **"vacío"**, donde al parecer se perdía en tratar de encontrar el camino a seguir en cada generación.
5. Con la instrucción de prueba (*INSERT INTO products VALUES (1, 'Cheese', 9.99);*), el AST generado utilizando la gramatica ascentende tiene 18 nodos y el AST generado con la gramatica descendente tiene 25 nodos, lo cual dificultaria al momento de recorrer el arbol para alguna interpretación. Nota: El tiempo para generar el AST, para el ascendente fué de 3.29 segundos y para el descendente fue de 3.42 segundos.

Por las razones presentadas anteriormente para implementar **Tytus-SQL Parser**, la opción elegida fué la gramática para analizador ascendente.

A continuación se muestran las gramáticas utilizadas en el proceso de comparación.

Gramática Ascendendente

<S> ::= <Init>
<Init> ::= <Statement_list>
<Stament_list> ::= <Statement_List> <Statement>
<Statement_list> ::= <Statement>
<Statement> ::= <Insert_statement>
| <Update_statement>
| <Delete_statement>
| <Enumtype>
<Enumtype> ::= CREATE TYPE <un_idx> AS ENUM PARIZQ <list_enum> PARDER

PTCOMA
| <Un_idx>
<list_enum> ::= <list_enum> COMA <otro_id>
| <otro_id>
<otro_id> ::= CADENACOMILLASIMPLE
<un_idx> ::= ID
<insert_statement> ::= INSERT INTO <table_name> <insert_columns_and_source>
PTCOMA
<table_name> ::= ID
<insert_columns_and_source> ::= <PARIZQ insert_column_list> PARDER VALUES
<query_expression_insert>
| VALUES <query_expression_insert>
| <insert_default_values>
<insert_default_values> ::= DEFAULT VALUES

<insert_column_list ::= <insert_column_list> COMA <column_name>
| <column_name>
<column_name> ::= ID
<query_expression_inset> ::= <insert_list>

<insert_list> ::= <insert_list> COMA <insert_value_list>
| <insert_value_list>
<insert_value_list> ::= PARIZQ <value_list> PARDER

<value_list> ::= <value_list> COMA <insert_value>
| <insert_value>
<insert_value> ::= ENTERO
| DECIMAL
| CADENACOMSIMPLE
| DEFAULT
| NOW PARIZQ PARDER

<update_statement -> UPDATE <table_name> SET <set_clause_list> WHERE
<search_condition> PTCOMA
<update_statement -> UPDATE <table_name> SET <set_clause_list> PTCOMA

<set_clause_list> ::= <set_clause_list> COMA <set_clause>
| <set_clause>
<set_clause> ::= <column_name> IGUAL <update_source>
<update_source> ::= <value_expression>
| NULL
<delete_statement> -> DELETE <table_name_decimal> FROM <table_name_d>
PTCOMA
<delete_statement> -> DELETE <table_name_d> PARDER <tname_ent> FROM
<table_name_d> WHERE <search_condition> PTCOMA
<tname_ent> ::= ENTERO

<tname_ent> ::= VARCHAR
<table_name_decimal> ::= DECIMAL
<table_name_d> ::= ID

<search_condition> ::= <search_condition> OR <boolean_term>
| <boolean_term>
<boolean_term> ::= <boolean_term> AND <boolean_factor>
| <boolean_factor>
<boolean_factor> ::= NOT <boolean_test>
| <boolean_test>
<boolean_test> ::= <boolean_primary>
<boolean_primary> ::= PARIZQ <search_condition> PARDER

<value_expression> ::= ENTERO
| DECIMAL
| CADENACOMSIMPLE
| DEFAULT
| ID
| NOW PARIZQ PARDER

Gramática Descendente

A continuación se presenta la gramática convertida para un analizador descenente, que se utilizó para realizar pruebas y generar el AST para realizar comparaciones con la gramática ascendente.

La gramática está escrita en formato Backus–Naur Form conocido como **BNF**

<S> ::= <Init>
<Init> ::= <Statement_list>
<Stament_list> ::= <Statement> <Statement_List_P'>
<Statement_list_P'> ::= <Statement> <Statement_List_P'>
| **ε**
<Statement> ::= <Insert_statement>
| <Update_statement>
| <Delete_statement>
| <Enumtype>
<Enumtype> ::= CREATE TYPE <un_idx> AS ENUM PARIZQ <list_enum> PARDER PTCOMA
| <Un_idx>
<list_enum> ::= <otro_id> <list_enum_P'>
| COMA <otro_id> <list_enum_P'>
| **ε**
<otro_id> ::= CADENACOMILLASIMPLE
<un_idx> ::= ID
<insert_statement> ::= INSERT INTO <table_name> <insert_columns_and_source>

PTCOMA
<table_name> ::= ID
<insert_columns_and_source> ::= <PARIZQ insert_column_list> PARDER VALUES
<query_expression_insert>
| VALUES <query_expression_insert>
| <insert_default_values>
<insert_default_values> ::= DEFAULT VALUES

<insert_column_list> ::= <column_name> <insert_column_list_P'>
<insert_column_list_P'> ::= <COMA column_name> <insert_column_list_p'>
| **ε**

<column_name> ::= ID
<query_expression_inset> ::= <insert_list>

<insert_list> ::= <insert_value_list> <insert_list_P'>
<insert_list_P'> ::= COMA <insert_value_list> <insert_list_P'>
| **ε**
<insert_value_list> ::= PARIZQ <value_list> PARDER

<value_list> ::= <insert_value> <value_list_P'>
<value_list_P'> ::= COMA <insert_value> <value_list_P'>
| **ε**
<insert_value> ::= ENTERO
| DECIMAL
| CADENACOMSIMPLE
| DEFAULT
| NOW PARIZQ PARDER

<update_statement -> UPDATE <table_name> SET <set_clause_list> WHERE
<search_condition> PTCOMA
<update_statement -> UPDATE <table_name> SET <set_clause_list> PTCOMA

<set_clause_list> ::= <set_clause> <set_clause_list_P'>
<set_clause_list_P'> ::= COMA <set_clause> <set_clause_list_p'>
| **ε**
<set_clause> ::= <column_name> IGUAL <update_source>
<update_source> ::= <value_expression>
| NULL
<delete_statement> -> DELETE <table_name_decimal> FROM <table_name_d>
PTCOMA
<delete_statement> -> DELETE <table_name_d> PARDER <tname_ent> FROM
<table_name_d> WHERE <search_condition> PTCOMA
<tname_ent> ::= ENTERO
<tname_ent> ::= VARCHAR

<table_name_decimal> ::= DECIMAL
<table_name_d> ::= ID

<search_condition> ::= <boolean_term> <search_condition_P'>
<search_condition_p'> ::= OR <boolean_term> <search_condition_P'>
|**ε**

<boolean_term> ::= <boolean_factor> <boolean_term_P'>
<boolean_term_P'> ::= AND <boolean_factor> <boolean_term_P'>
| **ε**

<boolean_factor> ::= NOT <boolean_test>
| <boolean_test>
<boolean_test> ::= <boolean_primary>
<boolean_primary> ::= PARIZQ <search_condition> PARDER

<value_expression ::= ENTERO
| DECIMAL
| CADENACOMSIMPLE
| DEFAULT
| ID
| NOW PARIZQ PARDER