Universidad de San Carlos de Guatemala
Facultad de ingeniería
Escuela de Ciencias y Sistemas
Lenguajes y Compiladores 2

MANUAL TÉCNICO

Tytus DB

Augusto German Mazariegos Salguero - 201114496
Glendy Marilucy Contreras González - 201025406
Brayan Ezequiel Santiago brito - 201114566
Luis Carlos valiente Salazar – 201122864

Grupo 4

Guatemala 23 de diciembre de 2020

CONTENIDO

DESCRIPCIÓN DEL LENGUAJE Tytus 2
SQL PARSER 2
HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO 3
VISUAL STUDIO CODE: 3
PYTHON (versión de cada integrante del equipo, 3.6-3.9) 3
GRAPHVIZ (versión 2.38) 3
PAQUETE PLY (versión 3.11) 3
PLATAFORMA DE USUARIO 4
REQUISITOS MÍNIMOS 4
SISTEMA OPERATIVO 4
RAM 4
ESPACIO DE ALMACENAMIENTO 4
ESPECIFICACIÓN DE CLASES Y MÉTODOS IMPORTANTES 5
Clase "Expresión" 5
Clase "instruccion" 7
CLASE SymbolTable 8
INTERFAZ GRAFICA 9
CLASE E ErrorTable 10
GRAMÁTICA ascendente 11

DESCRIPCIÓN DEL LENGUAJE TYTUS

Tytus es un proyecto open source de un administrador de bases de datos. Está compuesto por tres componentes interrelacionados:

1. El administrador de almacenamiento de la base de datos.
2. El administrador de la base de datos.
3. SQL parser. El cual es el que se especifica en este documento.

SQL PARSER

Este componente proporciona al servidor una función encargada de interpretar sentencias del subconjunto del lenguaje SQL.

Está compuesto por tres subcomponentes:

SQL Parser: Es el intérprete de sentencias de SQL, que proporciona una función para invocar al parser, al recibir una consulta el parser luego del proceso interno y de la planificación de la consulta invoca las diferentes funciones proporcionadas por el componente de administrador de almacenamiento.

Type Checker: Es un subcomponente que ayudará al parser a la comprobación de tipos. Al crear un objeto cualquiera se debe crear una estructura que almacenará los tipos de datos y cualquier información necesaria para este fin.

Query Tool: Es un subcomponente que consiste en una ventana gráfica similar al Query Tool de pgadmin de PostgreSQL, para ingresar consultas y mostrar los resultados, incluyendo el resalto de la sintaxis. La ejecución se realiza de todo el contenido del área de texto.

HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

VISUAL STUDIO CODE:

Como editor de código fuente, ya que incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Es de código abierto.

PYTHON (versión de cada integrante del equipo, 3.6-3.9)

Lenguaje de programación multiparadigma utilizado para la creación del parser.

GRAPHVIZ (versión 2.38)

Es un conjunto de herramientas de software para el diseño de diagramas definido en el lenguaje descriptivo DOT.

Utilizado para generar el árbol sintáctico de la gramática.

PAQUETE PLY (versión 3.11)

PLY es una implementación de Python puro de las populares herramientas de construcción de compiladores lex y yacc. El objetivo principal de PLY es mantenerse bastante fiel a la forma en que funcionan las herramientas tradicionales lex / yacc para la realización de analizadores léxico y sintáctico respectivamente.

PLATAFORMA DE USUARIO

REQUISITOS MÍNIMOS

SISTEMA OPERATIVO

- WINDOWS 8 EN ADELANTE
- UBUNTU 14 EN ADELANTE
- MAC OS 10.11 EN ADELANTE

RAM

2GB

ESPACIO DE ALMACENAMIENTO

40GB

ESPECIFICACIÓN DE CLASES Y MÉTODOS IMPORTANTES

CLASE "EXPRESIÓN"

CLASE	EXPRESIÓN	
DESCRIPCIÓN	Clase que lleva el manejo de las expresiones del lenguaje. Es una clase **abstracta** de la cual heredan sus características generales los diferentes tipos de expresiones. Esta clase es la base para todas las expresiones que se deben resolver. Es decir, los elementos que tienen alguna dependencia a la tabla de símbolos o a operaciones aritméticas.	
MÉTODOS	getValue(env)	Recibe el valor de una expresión.
	getGraph(graph, idParent)	Grafica el nodo correspondiente a una expresión
SUB CLASES	Arithmetic(Expression)	Administra todas las operaciones de tipo aritmético, contiene las subclases correspondientes a los distintos tipos de operaciones aritméticas: Addition(Arithmetic) Subtraction(Arithmetic) Multiplication(Arithmetic) Division(Arithmetic) Power(Arithmetic) Modulo(Arithmetic) UnaryMinus(Arithmetic) UnaryPlus(Arithmetic)
	BitWise(Expression)	Administra todas las operaciones de tipo bit a bit, contiene las subclases correspondientes a las distintas operaciones sobre números binarios.

	BitwiseAnd(BitWise) BitwiseOr(BitWise) BitwiseNot(BitWise) BitwiseXOR(BitWise) BitwiseRightShift(BitWise) BitwiseLeftShift(BitWise)
Call(Expression)	Maneja las llamadas a funciones.
Concat(Expression)	Concatena dos instrucciones
Literal(Expression)	Maneja las entradas de los literales ID.
Logical(Expression)	Administra todas las operaciones de tipo lógico, contiene las subclases correspondientes a las distintas operaciones de tipo booleano. And_class(Logical) Or_class(Logical) Not_class(Logical)
Relational(Expression)	Administra todas las operaciones relacionales, contiene las subclases correspondientes a las distintas operaciones de este tipo. MayorQue(Relational) MenorQue(Relational) MayorIgual(Relational) MenorIgual(Relational) IgualQue(Relational) Distinto(Relational)
native	Archivo que contiene las funciones nativas y los métodos correspondientes a ellas.

CLASE "INSTRUCCION"

CLASE	INSTRUCCIÓN	
DESCRIPCIÓN	Esta clase es la base para todas las instrucciones del lenguaje. Es una clase ***abstracta*** de la cual heredan sus características generales los diferentes tipos de expresiones.	
MÉTODOS	getValue(env)	Recibe el valor de una instrucción.
	getGraph(graph, idParent)	Grafica el nodo correspondiente a una instrucción.
SUB- CLASES	Alter_db(Instruction)	Método que maneja las actualizaciones a la base de datos.
	Alter_tb(Instruction)	Método que maneja las actualizaciones una tabla.
	Create_db(Instruction)	Método que crea la base de datos.
	Create_tb(Instruction)	Método para crear una tabla.
	Delete_Reg(Instruction)	Método para eliminar registros de la tabla seleccionada.
	Drop_db(Instruction)	Elimina la base de datos.
	DropTable(Instruction)	Elimina una tabla específica de la base de datos.
	Insertinto(Instruction)	Inserta registros dentro de una tabla específica.
	Select(Instruction)	Método que maneja los querys correspondientes a la instrucción 'select'
	Show_db(Instruction)	Muestra la base de datos.
	UseDataBase(Instruction)	Método que especifica la base de datos actual.

CLASE SYMBOLTABLE

DESCRIPCIÓN

Su objetivo es generar administración para los símbolos.

Su atributo principal es el diccionario que administra los símbolos.

```
ATRIBUTOS:
        simbols = []
        treeview = None
```

METODOS IMPORTANTES

create(parent)	Método que crea la tabla.
	`tree = Treeview(parent)`
	`tree['columns'] = ('tipo', 'nombreTabla')`
	`tree.heading('#0', text='Nombre')`
	`tree.column('#0', anchor='center', width=25)`
	`tree.heading('tipo', text='TIPO')`
	`tree.column('tipo', anchor='center', width=200)`
	`tree.heading('nombreTabla', text='Nombre Tabla')`
	`tree.column('nombreTabla', anchor='center', width=25)`

load()	Actualiza el símbolo que este asignado al diccionario.
clear()	Elimina todos los símbolos de la tabla. Solo se usa antes de iniciar una ejecución para prevenir que la ejecución utilice símbolos de una anterior.
add(list)	Agrega un nuevo registro a la tabla.

INTERFAZ GRAFICA		
DESCRIPCIÓN	Su objetivo es manejar la interfaz gráfica del programa. Utiliza la librería ***tkinter*** principalmente para cumplir su cometido.	
METODOS IMPORTANTES	EntradaEditor(tk.Frame)	Ventana en la que se escribe y maneja el reporte gramátical.
	Window()	Maneja la ventana principal del IDE. Contiene los menús, barras de herramientas, área de texto para el editor y consola.
	CustomText(tk.Text)	Muestra un texto automático que es modificable.
	TextLineNumbers(tk.Canvas)	Muestra el número de líneas es especialmente útil para el reporte gramatical.

<table>
<tr><th colspan="3">CLASE E ERRORTABLE</th></tr>
<tr><td>DESCRIPCIÓN</td><td colspan="2">Se utiliza para el manejo de errores.

Atributos:
<pre>errors = []
treeview = None</pre></td></tr>
<tr><td rowspan="4">SUB CLASES</td><td>create(parent)</td><td>Estructura con los atributos de cada error:

Atributos:
<pre>tree = Treeview(parent)
tree['columns'] = ('description', 'line')
tree.heading('#0', text='Tipo')
tree.column('#0', anchor='center', width=25)
tree.heading('description', text='Descripción')
tree.column('description', anchor='center', width=200)
tree.heading('line', text='Línea')
tree.column('line', anchor='center', width=25)
tree.grid(sticky=(N, S, W, E))

ErrorTable.treeview = tree

parent.grid_rowconfigure(0, weight=1)
parent.grid_columnconfigure(0, weight=1)</pre></td></tr>
<tr><td>load()</td><td>Actualiza el registro de errores.</td></tr>
<tr><td>clear()</td><td>Elimina todos los errores registrados</td></tr>
<tr><td>add(list)</td><td>Agrega un nuevo registro de error a la lista.</td></tr>
</table>

GRAMÁTICA ASCENDENTE

```
'S : INSTS'

'''INSTS : INSTS INST PCOMA
         | INST PCOMA'''

'''INST : CREATE_TYPE
        | SELECT
        | CREATE_DB
        | SHOW_DB
        | DROP_DB
        | ALTER_DB
        | USE_DB
        | CREATE_TB
        | UNION
        | UNIONALL
        | INTERSECT
        | INTERSECTALL
        | EXCEPT
        | EXCEPTALL
        | DROPTABLE
        | DELETE
        | INSERT
        '''

'''OPDB : OWNER
         | MODE
         '''

'''CREATE_DB : CREATE DATABASE IF NOT EXISTS E OPDB IGUAL E OPDB IGUAL E
             | CREATE DATABASE IF NOT EXISTS E OPDB IGUAL E
             | CREATE DATABASE E OPDB IGUAL E
             | CREATE OR REPLACE DATABASE E
             | CREATE DATABASE E
             '''

'''DROP_DB : RDROP DATABASE E
           | RDROP DATABASE IF EXISTS E'''

'''ALTER_DB : ALTER DATABASE E RENAME TO E
            | ALTER DATABASE E OWNER  TO E'''
```

```
    '''SHOW_DB : SHOW DATABASE '''

    '''CREATE_TB : CREATE TABLE IDENTIFIER PARI ATRIBUTOS PARD'''

    '''ATRIBUTOS : ATRIBUTOS COMA ATRIBUTO
                 | ATRIBUTO'''

    '''ATRIBUTO : ATRIBUTO PARAMS_ATRIBUTO
                | PARAMS_ATRIBUTO'''


    '''PARAMS_ATRIBUTO : PARM_ID_ATRIBUTO
                       | TIPO
                       | PARM_PK_ATRIBUTO
                       | PARM_CONSTRAINT_ATRIBUTO
                       | PARM_CHECK_ATRIBUTO
                       | PARM_NOTNULL_ATRIBUTO
                       | PARM_UNIQUE_ATRIBUTO'''

    '''PARM_ID_ATRIBUTO : ID'''

    '''PARM_PK_ATRIBUTO : PRIMARY KEY'''

    '''PARM_CONSTRAINT_ATRIBUTO : CONSTRAINT IDENTIFIER'''

    '''PARM_CHECK_ATRIBUTO : CHECK PARI E PARD'''

    '''PARM_NOTNULL_ATRIBUTO : NOT NULL'''

    '''PARM_UNIQUE_ATRIBUTO : UNIQUE'''

    '''TIPO : INTEGER
            | VARCHAR PARI E PARD
            | REAL
            | DATE
            | BOOLEAN '''

    'CREATE_TYPE : CREATE TYPE ID AS ENUM PARI LE PARD'

    '''SELECT : RSELECT E'''

    '''UNION : SELECT RUNION UNION
             | SELECT RUNION SELECT'''
```

```
'''UNIONALL : SELECT RUNION ALL UNIONALL
            | SELECT RUNION ALL SELECT'''

'''INTERSECT : SELECT RINTERSECT INTERSECT
             | SELECT RINTERSECT SELECT'''

'''INTERSECTALL : SELECT RINTERSECT ALL INTERSECTALL
                | SELECT RINTERSECT ALL SELECT'''

'''EXCEPT : SELECT REXCEPT EXCEPT
          | SELECT REXCEPT SELECT'''

'''EXCEPTALL : SELECT REXCEPT ALL EXCEPT
             | SELECT REXCEPT ALL SELECT'''

'''USE_DB : USE DATABASE ID
          | USE ID'''

'''DROPTABLE : RDROP TABLE ID '''

'''DELETE : RDELETE FROM ID'''

'''INSERT : RINSERT INTO ID VALUES PARI LE PARD
          | RINSERT INTO ID PARI LE PARD VALUES PARI LE PARD'''

'''LE : LE COMA E
      | E'''

'''CALL : ID PARI LE PARD
        | ID PARI PARD'''

'''LITERAL : INT
           | DECIMAL
           | CADENA
           | BOOL_TRUE
           | BOOL_FALSE
           | IDENTIFIER'''

'BOOL_TRUE : TRUE'

'BOOL_FALSE : FALSE'

'IDENTIFIER : ID'
```

```
'''E : E AND E
     | E OR E
     | NOT E %prec UNOT
     | E MENORQ E
     | E MAYORQ E
     | E MAYORIGUAL E
     | E MENORIGUAL E
     | E IGUALQ E
     | E DISTINTO E
     | E SUMA E
     | E RESTA E
     | E MULT E
     | E DIVISION E
     | E POTENCIA E
     | E MODULO E
     | E CONCAT E
     | E BAND E
     | E BOR E
     | E MOVD E
     | E MOVI E
     | E NUMERAL E
     | E AS E
     | E PUNTO E
     | E IGUAL E
     | VIRGULILLA E %prec UBNOT
     | RESTA E %prec UMINUS
     | SUMA E %prec UPLUS
     | PARI E PARD
     | CALL
     | LITERAL'''
```