Manual Tecnico – Analizador SQL

Realizado por: Javier Alejandro Mérida Gómez – 202230368

Descripción:

Bienvenido al "Analizador SQL"!
El sistema consiste en la creación de un analizador léxico usando la herramient JFlex, el lenguaje que debe de reconocer es el de SQL, pero para esta ocasión también se le solicita que implemente una analizador sintáctico para que que reconozca la estructura del lenguaje y una vez que la entrada es analizada, debe extraer la información analizada y generar diagramas de las tablas.

Requisitos del Sistema

- Plataforma: El analizador Léxico fue desarrollado para ejecutarse en sistemas operativos Windows y Linux.
- Entorno de Desarrollo: Se utilizó Apache NetBeans IDE 20 para el desarrollo del código y dependencias de Graphviz.
- Lenguaje de Programación: El analizador Léxico se implementó en Java (Programación Orientada a Objetos), versión 17.
- Sistema de Desarrollo: El desarrollo se llevó a cabo en un entorno Ubuntu Linux.
- Link del repositorio: https://github.com/JavierMeridaLk/ProyectoFinalLFP

Componentes del analizador SQL

Backen:

- AnalizadorSintactico: El AnalizadorSintáctico es una clase en Java diseñada para procesar y verificar la estructura de sentencias SQL, categorizándolas según su tipo y reportando errores o identificando sentencias válidas. Cuenta con listas para almacenar las sentencias procesadas, errores, reportes de tablas y modificaciones, así como gráficos relacionados con las estructuras analizadas. Además, incluye contadores específicos para las operaciones SQL más comunes: CREATE, DELETE, UPDATE, SELECT y ALTER. El método principal, analizar(String textoAceptado), se encarga de dividir el texto en sentencias

individuales, validarlas según reglas predefinidas, y clasificar las sentencias válidas en gráficos o reportes mientras registra las incorrectas en la lista de errores. La clase utiliza métodos auxiliares para separar palabras y validar la estructura de sentencias como la creación de bases de datos y tablas, la inserción, la actualización y la eliminación, asegurando que cada sentencia cumpla con los formatos SQL esperados. También implementa métodos para validar declaraciones de datos y claves, verificando que las restricciones y definiciones de columnas sean correctas y adecuadas para las operaciones SQL.

- Archivos: La clase Archivos se encarga de manejar operaciones básicas relacionadas con la lectura y escritura de archivos de texto, facilitando la interacción con un componente JTextPane. La clase ofrece funcionalidades para abrir archivos y mostrar su contenido, guardar textos en archivos existentes o en una nueva ubicación, y gestionar la creación de carpetas específicas para almacenar los archivos. Se emplean componentes gráficos como JFileChooser y JOptionPane para la selección de archivos y la presentación de mensajes al usuario.
- Graficos: La clase Graficos se encarga de generar representaciones visuales de sentencias DDL (Data Definition Language) en formato Graphviz y de convertirlas en gráficos PNG. Está diseñada para procesar líneas de comandos SQL y transformarlas en nodos de un gráfico que representan las operaciones de creación, alteración y eliminación de bases de datos y tablas.
- Reportes: La clase Reportes se encarga de generar distintos reportes en forma de tablas (JTable) para mostrar información sobre errores léxicos y sintácticos, nombres de tablas, tipos de modificaciones y operaciones en una base de datos. Esta clase utiliza DefaultTableModel para manejar los datos de las tablas.
- Token: La clase Token representa un elemento léxico que se encuentra en el análisis de código o texto. Esta clase es fundamental en aplicaciones que requieren el análisis y manejo de tokens, como analizadores léxicos y compiladores.

- AnalizadorLexico: Analiza el lexico de un texto de entrada utilizando la herramienta Jflex, utilizando expresiones regulares, comparaciones, nos devulve listas d elso tokens aceptados y su categoria y nos da una lista de errores encontrados dentro del analisis.

Fronted:

- FramePrincipal:La clase FramePrincipal se encarga de la configuración y funcionalidad principal de una interfaz gráfica para un analizador SQL. Al inicializar, establece dimensiones, título y características de los componentes de texto, asegurando que sean de solo lectura. Implementa un DocumentListener que actualiza el conteo de líneas y habilita o deshabilita el botón de análisis según si hay texto en el área de entrada. Además, gestiona el posicionamiento del cursor mediante un CaretListener, mostrando la fila y columna actuales. El método analizar() se utiliza para procesar el texto ingresado, invocando un analizador léxico para identificar y clasificar diferentes tipos de tokens (como identificadores y comentarios) a través de atributos de estilo. Posteriormente, se realiza un análisis sintáctico para verificar la estructura del SQL, generando reportes de errores y métricas de los comandos detectados. La función pintarTokens() se encarga de aplicar el estilo adecuado a cada token en el documento, mientras que calcularPosicionInicial() determina la ubicación exacta de un token dentro del texto para facilitar la visualización. En conjunto, estos métodos permiten al usuario interactuar de manera efectiva con el analizador SQL, visualizando la sintaxis y errores de manera clara y estructurada.
- ReportesErrorSintacticos: La clase reportesErrorSintacticos es un diálogo modal que presenta un informe de errores sintácticos en una tabla. Configura su tamaño y título en el constructor, y utiliza GroupLayout para organizar visualmente sus componentes. El método subirTabla(JTable tabla) permite añadir un JTable dentro de un JScrollPane para facilitar el desplazamiento del contenido, actualizando el panel y ajustando el tamaño del diálogo. Esta estructura simplifica la gestión de la interfaz al incluir variables para los componentes principales.

- ReportesErrorToken: La clase reportesErrorToken es un diálogo modal que presenta un informe de errores sintácticos en una tabla. Configura su tamaño y título en el constructor, y utiliza GroupLayout para organizar visualmente sus componentes. El método subirTabla(JTable tabla) permite añadir un JTable dentro de un JScrollPane para facilitar el desplazamiento del contenido, actualizando el panel y ajustando el tamaño del diálogo. Esta estructura simplifica la gestión de la interfaz al incluir variables para los componentes principales.
- ReportesModificadores:La clase ReportesModificadores es un diálogo modal que presenta un informe de errores sintácticos en una tabla. Configura su tamaño y título en el constructor, y utiliza GroupLayout para organizar visualmente sus componentes. El método subirTabla(JTable tabla) permite añadir un JTable dentro de un JScrollPane para facilitar el desplazamiento del contenido, actualizando el panel y ajustando el tamaño del diálogo. Esta estructura simplifica la gestión de la interfaz al incluir variables para los componentes principales.
- ReportesTablas:La clase ReportesTablas es un diálogo modal que presenta un informe de errores sintácticos en una tabla. Configura su tamaño y título en el constructor, y utiliza GroupLayout para organizar visualmente sus componentes. El método subirTabla(JTable tabla) permite añadir un JTable dentro de un JScrollPane para facilitar el desplazamiento del contenido, actualizando el panel y ajustando el tamaño del diálogo. Esta estructura simplifica la gestión de la interfaz al incluir variables para los componentes principales.
- ReportesOperaciones:La clase ReportesOperaciones es un diálogo modal que presenta un informe de errores sintácticos en una tabla. Configura su tamaño y título en el constructor, y utiliza GroupLayout para organizar visualmente sus componentes. El método subirTabla(JTable tabla) permite añadir un JTable dentro de un JScrollPane para facilitar el desplazamiento del contenido, actualizando el panel y ajustando el tamaño del diálogo. Esta estructura simplifica la gestión de la interfaz al incluir variables para los componentes principales.

Gramáticas y Autónomas:

Gramatica Formales.

Enteros: $Gr = (N, T, P, S)$

$N = \{E, O\}$

$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$P = \{E \to OE,\ E \to \lambda,\ E \to 0|1|2|3|4|5|6|7|8|9\}$

$S = \{E\}$

Fecha: $Gr = (N, T, P, S)$

$N = \{F, Y, M, D\}$

$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -\}$

$P = \{F \to 'Y-M-D',\ Y \to \backslash d\backslash d\backslash d\backslash d,\ M \to 0[1-9] \mid 1[0-2],$
$D \to 0[1-9] \mid [12][0-9] \mid 3[01]\}$

$S = \{F\}$

Identificador: $Gr = (N, T, P, S)$

$N = \{I, I', A\}$

$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \_, a, b, c, d, e, f, g, h, i, j, k, l, m, n, ñ, o, p, q, r, s, t, u, v, z\}$

$P = \{I \to AI',\ I' \to \_AI' \mid \lambda,\ A \to 0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|g|h|i|j|k|l|m|n|ñ|o|p|q|r|s|t|u|v|z\}$

$S = \{I\}$

Cadena: $Gr = (N, T, P, S)$

$N = \{C, T\}$

$T = \{', 1\}$

$P = \{C \to 'T',\ T \to 1T \mid 1\}$

$S = \{C\}$

Autómatas de pila.

Entero $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

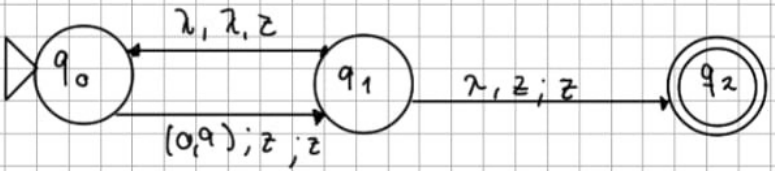
$\Gamma = \{Z\}$

$\delta = \{(q_0, \lambda, \lambda) \rightarrow (q_1, Z), (q_1, d, Z) \rightarrow (q_1, Z), (q_1, \lambda, Z) \rightarrow (q_2,$

$q_0 = \{q_0\}$

$F = \{q_2\}$

Diagrama:



Identificador. $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{(a-z), (0-9), \_\}$

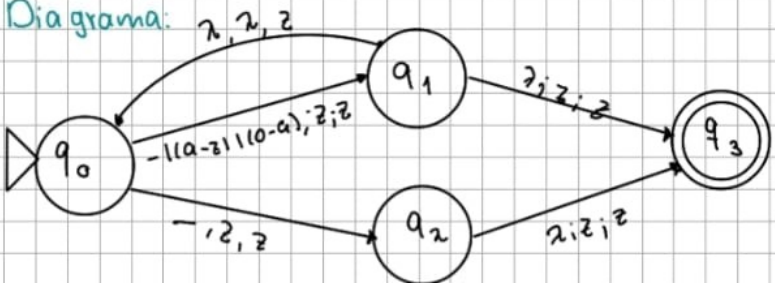
$\Gamma = \{Z\}$

$\delta = \{(q_0, \lambda, \lambda) \rightarrow (q_1, Z), (q_1, \_, Z) \rightarrow (q_1, Z), (q_1, \lambda, Z) \rightarrow$
$(q_2, Z), (q_2, y, Z) \rightarrow (q_2, Z), (q_2, \_, Z) \rightarrow (q_2, Z), (q_1, \lambda, Z)$
$\rightarrow (q_3, Z), (q_2, \lambda, Z) \rightarrow (q_3, Z)$

$q_0 = \{q_0\}$

$F = \{q_3\}$

Diagrama:



Cadena $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{'.'\}$

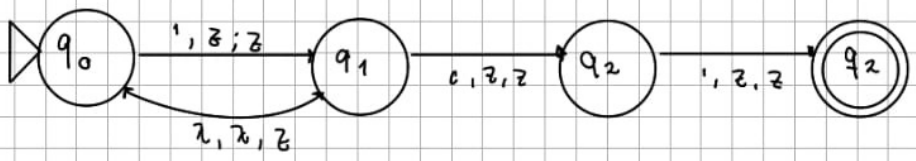
$\Gamma = \{Z\}$

$\delta = \{(q_0, \lambda, \lambda) \to (q_1, Z), (q_1, ', Z) \to (q_2, Z), (q_2, c, Z) \to (q_2, Z), (q_2, ', Z) \to (q_3, Z)\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

Diagrama: c = cualquier caracter.



Decimal. $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{(0-9), .\}$

$\Gamma = \{Z\}$

$\delta = \{(q_0, \lambda, \lambda) \to (q_1, Z), (q_1, d, Z) \to (q_1, Z), (q_1, ., Z) \to (q_2, Z), (q_2, d, Z) \to (q_3, Z), (q_3, d, Z) \to (q_3, Z), (q_3, \lambda, Z) \to (q_4, Z)\}$

$q_0 = \{q_0\}$

$F = \{q_4\}$

Diagrama: