UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA
LENGUAJES FORMALES DE PROGRAMACIÓN

MANUAL TÉCNICO

DIEGO ALESSANDRO GONZALEZ BRAVO / 202230939

Introducción

El proyecto desarrollado consiste en un analizador léxico que procesa código fuente en HTML, CSS, y JavaScript. El objetivo principal es identificar tokens clave en cada uno de estos lenguajes y proporcionar la capacidad de optimizar el código eliminando elementos repetidos. El sistema puede generar reportes detallados de los tokens encontrados y presentar los resultados de forma clara en una interfaz gráfica. También incluye una función de optimización de código, donde se eliminan líneas repetidas o innecesarias para hacer el código más limpio y eficiente.

Herramientas de desarrollo.
Las herramientas (IDES, lenguajes de programación) utilizadas para el desarrollo de la aplicacion Manejo de tarjetas son:

- JAVA
    - Version openjdk version "21.0.4"
- Apache NetBeans
    - Version 22
- Sistema operativo de desarrollo
    - Ubuntu 24.04 LTS
- JFLEX
    - Version 1.9.1

Descripción del Análisis Léxico

El analizador léxico que se presenta a continuación está diseñado para interpretar y descomponer consultas SQL, facilitando su análisis y manejo. Este analizador reconoce diferentes componentes de una consulta SQL y los clasifica en categorías específicas. A continuación, se describe cómo funciona el analizador y sus principales características.

Estructura General:

El analizador está basado en un sistema de reglas que utiliza expresiones regulares para identificar diferentes elementos dentro del texto de la consulta SQL. Cada vez que encuentra un fragmento que coincide con una de estas reglas, genera un "token", que es una representación de ese fragmento, junto con su tipo correspondiente.

En el analizador léxico que hemos diseñado para interpretar consultas SQL, se utilizan expresiones regulares para identificar diferentes tipos de elementos dentro del texto. Las expresiones regulares son patrones que permiten buscar y clasificar cadenas de texto de manera eficiente. A continuación, se describen las expresiones regulares que se emplean en nuestro analizador:

- PALABRA:
    - `PALABRA = [a-zA-Z]+`

    - Descripción: Esta expresión regular identifica cualquier cadena de texto que consista únicamente en letras del alfabeto, ya sea en mayúsculas o minúsculas. Se utiliza para reconocer palabras individuales que forman parte de la sintaxis SQL.
- IDENTIFICADOR
    - `IDENTIFICADOR = [a-zA-Z_][a-zA-Z_0-9]*`

    - Descripción: Esta expresión se utiliza para identificar los identificadores que pueden ser nombres de tablas, columnas u otros elementos en SQL. Un identificador debe comenzar con una letra o un guión bajo y puede estar seguido por letras, números o guiones bajos. Esto asegura que se capturen adecuadamente los nombres válidos en SQL.
- NÚMERO:
    - `NUMERO = "-"?[0-9]+`

    - Descripción: Esta expresión reconoce números enteros. Permite números positivos y negativos. La parte `"-"?` indica que puede haber un signo negativo opcional antes de la secuencia de dígitos, que se representa por `[0-9]+`.
- DECIMAL:
    - `DECIMAL = "-"?[0-9]+("." [0-9]+)?`

    - Descripción: Esta expresión es utilizada para identificar números decimales. Al igual que la anterior, permite un signo negativo opcional y requiere al menos un dígito antes del punto decimal, seguido de cualquier cantidad de dígitos después del punto. Esto permite que se reconozcan correctamente los valores decimales en las consultas SQL.
- ESPACIO:
    - `ESPACIO = [" "\r\t\n\f]`
    - Descripción: Esta expresión regular se utiliza para reconocer y omitir los espacios en blanco, incluyendo espacios, tabulaciones, saltos de línea y otros caracteres de espacio. Es fundamental para garantizar que el analizador no se vea afectado por el formato del texto y pueda centrarse únicamente en el contenido relevante de la consulta.

Reglas de Reconocimiento:

- Palabras Reservadas:
    - El analizador identifica palabras clave de SQL, como "CREATE", "SELECT", "INSERT", etc. Cada vez que encuentra una de estas palabras, retorna un token que indica su tipo. Por ejemplo:
    - "CREATE" { return token(yytext(), TokenType.CREATE); }

- Tipos de Datos:
    - También reconoce los tipos de datos SQL, como "INTEGER", "VARCHAR" y "DATE". Al igual que las palabras reservadas, se devuelve un token para cada tipo de dato encontrado:
    - "INTEGER" { return token(yytext(), TokenType.INTEGER_TYPE); }

- Constantes y Literales:
    - El analizador puede identificar números enteros, decimales, fechas y cadenas. Por ejemplo, las cadenas se manejan de la siguiente manera:
    - \'([^\']*)\' { return token(yytext(), TokenType.CADENA); }
    -
- Booleanos:
    - También reconoce los valores booleanos "TRUE" y "FALSE", generando tokens correspondientes.

- Funciones de Agregación:
    - Funciones comunes en SQL, como "SUM", "AVG", y "COUNT", son también reconocidas y clasificadas.

- Operadores y Símbolos:
    - El analizador detecta operadores matemáticos y de comparación, así como símbolos como paréntesis y comas, generando tokens para cada uno:
    - "+" { return token(yytext(), TokenType.SUMA); }
    -

- Comentarios:
    - Los comentarios en línea que comienzan con "--" son gestionados para evitar que interfieran con la ejecución de las consultas. Se cambian al estado de comentario:

    - "--" { yybegin(COMMENT); }

- Identificadores:
    - Se reconocen identificadores que pueden incluir letras, números y guiones bajos. Se generan tokens para estos elementos:
    - {PALABRA}({PALABRA}|{NUMERO}|"_")* { return token(yytext(), TokenType.ID); }
- Espacios y Errores:
    - Los espacios son ignorados en el análisis, mientras que cualquier símbolo que no coincida con las reglas definidas se clasifica como un error:
    - {ESPACIO} {}
      . { return token(yytext(), TokenType.ERROR); }

Cuando el analizador se ejecuta, comienza a leer el texto de la consulta SQL. A medida que procesa cada parte del texto, aplica las reglas definidas. Cuando encuentra un patrón que coincide con una regla, genera un token correspondiente y continúa con el siguiente fragmento. Este proceso se repite hasta que se ha analizado toda la consulta.

El resultado es una secuencia de tokens que representan la estructura y los componentes de la consulta. Estos tokens pueden ser utilizados posteriormente por otras partes del sistema para validar la consulta, ejecutar operaciones o mostrar información al usuario.

Descripción del Analizador Sintáctico

El analizador sintáctico es una parte esencial del sistema diseñado para procesar consultas SQL. Su función principal es tomar la secuencia de tokens generados por el analizador léxico y organizarla en una estructura lógica, conocida como árbol de sintaxis, que representa la gramática de las consultas. Este árbol permite interpretar correctamente la intención del usuario al escribir una consulta SQL. A continuación se describe cómo funciona el analizador sintáctico en nuestro proyecto.

- Recepción de Tokens
    - El analizador sintáctico recibe una serie de tokens, cada uno representando una parte de la consulta SQL, como palabras reservadas, identificadores, operadores y literales. Estos tokens son generados previamente por el analizador léxico a partir del texto de entrada.

- Estructura del Árbol de Sintaxis
    - El árbol de sintaxis es una representación jerárquica de la consulta SQL. Cada nodo en este árbol representa un componente de la consulta, como una cláusula o un elemento específico. Por ejemplo:

        - Un nodo puede representar una instrucción SELECT.
        - Otros nodos pueden representar las columnas seleccionadas, la tabla de origen y las condiciones de búsqueda.
        -
- Esta estructura permite que el sistema entienda cómo se relacionan entre sí los diferentes componentes de la consulta.

- Reglas Gramaticales
    - El analizador sintáctico utiliza un conjunto de reglas gramaticales que definen la estructura correcta de las consultas SQL. Estas reglas son fundamentales para asegurar que las consultas sean válidas y que sigan la sintaxis del lenguaje SQL. Por ejemplo:

        - Una consulta SELECT debe tener la forma SELECT <columnas> FROM <tabla>.
        - Las condiciones pueden incluir cláusulas WHERE que especifican criterios adicionales.
- Verificación de la Sintaxis
    - A medida que el analizador procesa los tokens, verifica si la secuencia de estos cumple con las reglas definidas. Si encuentra que la estructura de la consulta es incorrecta, el analizador genera un mensaje de error que indica el problema encontrado. Esto permite al usuario corregir la consulta antes de enviarla a la base de datos.

- Construcción del Árbol
    - Durante el proceso de análisis, el analizador construye el árbol de sintaxis de manera dinámica. A medida que se reconocen componentes válidos, se crean nodos en el árbol que representan estos elementos. Este enfoque facilita la construcción de la consulta de manera lógica y estructurada.

- Salida del Análisis
    - Al finalizar el análisis, el resultado es un árbol de sintaxis que representa la consulta SQL de forma jerárquica. Este árbol se puede utilizar posteriormente por otras partes del sistema, como el ejecutor de consultas, para llevar a cabo la operación deseada en la base de datos.

Diagrama de clases: