

```

public class InsertTrans extends TranslatorStm{

    public InsertTrans(AdmiFiles admiFiles, List<String> semanticErrosList)
    {
        super.semanticErrors = semanticErrosList;
        super.separator = new SeparatorElements();
        super.admiFiles = admiFiles;
        super.errorMss = ERROR_MSS + "insert";
    }

    @Override
    public String translate(List<Token> tokens, Index index) {
        InsertModel model = (InsertModel) this.separateTkns(tokens, index);
        try {
            admiFiles.openFile( pathWithDots: model.getPath().getPathWithDots()
            if(model.getColumns() == null) {
                insertAll(model);
            } else {
                admiFiles.getCSVinterpretor().setPosColumns(
                    columnsList: model.getColumns(), errorsList: semanticError
                return "Aun no implementado, actualizacion en la v 1.2";
            }
        }
        return "Insercion exitosa";
    }
}

```

# SQL\_emulator

Manual técnico

06/03/2024

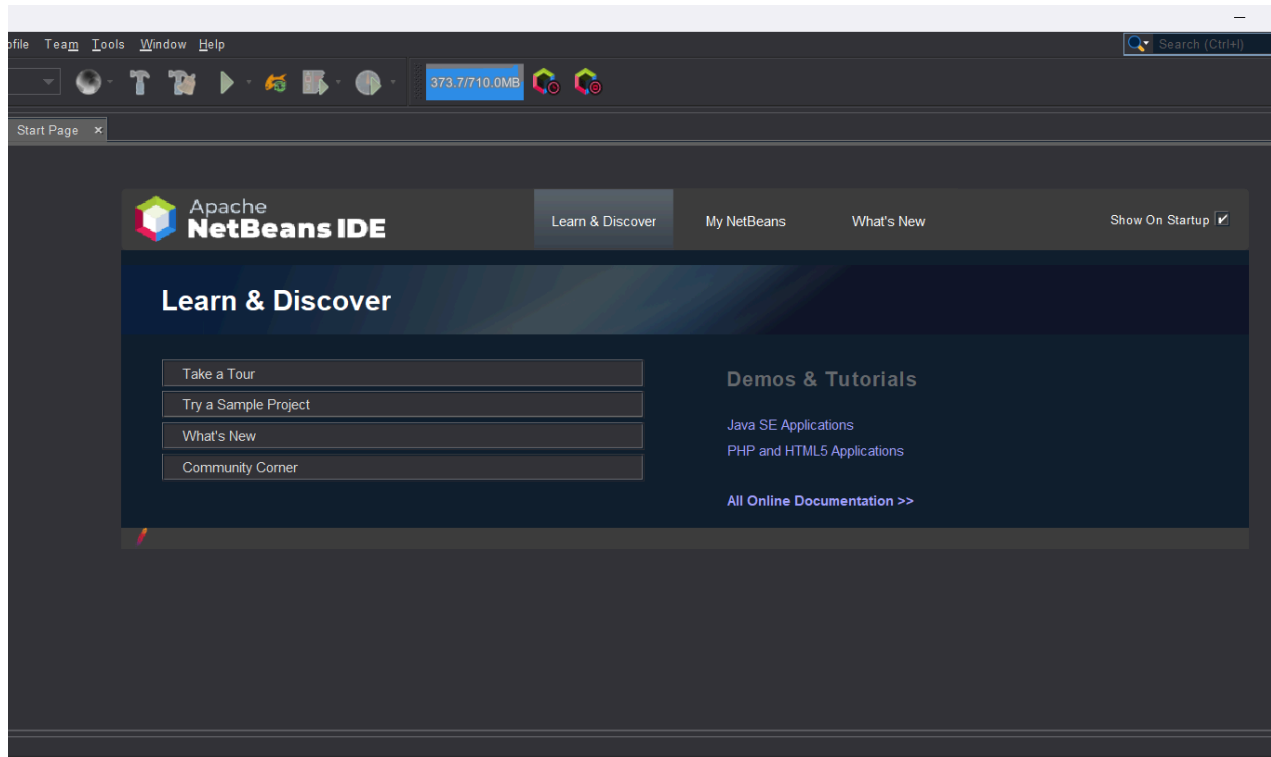
---

Yennifer Maria de León Samuc

Organización de Lenguajes y Compiladores 1

## TECNOLOGÍA USADA EN EL PROYECTO

- Ide: ApacheNetBeans IDE 16



- JFlex 1.9.1
  - Enlace para la descarga: <https://jflex.de/download.html>
- Java-cup-11b
  - Enlace para la descarga: <https://www2.cs.tum.edu/projects/cup/>

## CONFIGURACIONES PARA EL LEXER Y EL PARSER

En el repositorio del proyecto (<https://github.com/Yennifer017/sql-emulator>) se encontrara una carpeta con los nombres Lexer.jflex y Parser.cup, que sirvieron para compilar los archivos Lexer.java y parser.java y sym.java respectivamente para el análisis léxico y sintáctico.

A continuación se dan detalles del análisis léxico y sintáctico.

## Análisis Léxico

- **Constantes**

LineTerminator =  $\backslash r | \backslash n | \backslash r \backslash n$

WhiteSpace =  $\{ \text{LineTerminator} \} | [ \backslash t \backslash f ]$

L =  $[a-zA-Z\_]+$

DecIntegerLiteral =  $[0-9]^+$

Identifier =  $[ \text{letra} : ] ( [ \text{digito} : ] | "-" | "_" | "@" | "+" | "*" | "#" )^*$

- **Tokens**

- **Palabras reservadas**

SELECCIONAR, FILTRAR, INSERTAR, ACTUALIZAR, ASIGNAR, ELIMINAR, EN, VALORES.

- **Símbolos reservados**

"."      "\*"      "("      ")"      ","      ";"

- **Operadores Relacionales**

"="      ">"      "<"      ">="      "<="      "<>"

- **Operadores Lógicos**

AND, OR

- **Otras variables**

$\{ \text{DecIntegerLiteral} \} \rightarrow$  Entero

$\{ \text{Identifier} \} \rightarrow$  Identificador

$\{ \text{DecIntegerLiteral} \} . \{ \text{DecIntegerLiteral} \} \rightarrow$  Decimal

$\backslash ( \{ L \} | \{ \text{DecIntegerLiteral} \} | " " )^* \backslash$  → Cadena

$\{ \text{WhiteSpace} \} \rightarrow$  Espacio vacío (Es ignorado)

## Análisis Sintáctico

*Nota: los símbolos terminales estan en mayúscula y los no terminales en minúscula.*

- **Símbolo inicial: "Instrucciones"**

relational\_complete\_operator ::= IGUAL | DIFERENTE;

relational\_operator ::= MAYOR\_QUE | MENOR\_QUE | MAYOR\_IGUAL\_QUE |  
MENOR\_IGUAL\_QUE ;

number\_values ::= ENTERO | DECIMAL;

values ::= CADENA | number\_values;

instrucciones ::= instruccion FIN\_INSTRUCCION more\_instruccions;

more\_instruccions ::= instrucciones | /\*empty\*/;

instruccion ::= select\_stm | insert\_stm | update\_stm | delete\_stm;

select\_stm ::= SELECCIONAR selects EN path filtros\_opt;

insert\_stm ::= INSERTAR EN path columns\_opt VALORES PARENTESIS\_L insertables  
PARENTESIS\_R;

update\_stm ::= ACTUALIZAR EN path ASIGNAR asignations filtros\_opt;

delete\_stm ::= ELIMINAR EN path filtros\_opt;

selects ::= ASTERISCO

| columns ;

columns ::= IDENTIFICADOR more\_columns ;

more\_columns ::= COMA columns

| /\*empty\*/ ;

path ::= IDENTIFICADOR carpetas;

carpetas ::= PUNTO path

| /\*empty\*/;



```
filtros_opt ::= filtros | /*empty*/;
```

```
filtros ::= FILTRAR conditions;
```

```
conditions ::= condition logic;
```

```
logic ::= and_stm | or_stm | /*empty*/;
```

```
and_stm ::= AND condition and_stm2 ;
```

```
and_stm2 ::= AND condition and_stm2 | /*empty*/;
```

```
or_stm ::= OR condition or_stm2 ;
```

```
or_stm2 ::= OR condition or_stm2 | /*empty*/;
```

```
condition ::= IDENTIFICADOR comparative;
```

```
comparative ::= relational_complete_operator values  
               | relational_operator number_values;
```

```
insertables ::= values more_insertables;
```

```
more_insertables ::= COMA insertables  
                  | /*empty*/;
```

```
columns_opt ::= PARENTESIS_L columns PARENTESIS_R  
               | /*empty*/;
```

```
asignations ::= IDENTIFICADOR IGUAL values more_asignations;
```

```
more_asignations ::= COMA asignations  
                  | /*empty*/;
```

## ALGORITMO DE FUNCIONES IMPORTANTES

### I. Traducción del método Delete

Verificar si el path existe

Si no hay filtros

Setear en el display solo el contenido de las primeras columnas

De lo contrario

pedir a la lectura de archivos que mande un int[] con el código de posición de las columnas

Leer el archivo por columnas

Mientras haya columnas

Cuando es un and

VERIFICAR AND()

Cuando es un or o cuando solo sea una condición

VERIFICAR OR()

Dendiendo del veredicto final

si es falso

no eliminar fila

si es verdadero

eliminar fila

```

-----VERIFICAR AND
resultado = falso;
para cada condición
    resultado = verificar la
    condición actual
    si es falso
        parar
fin para
  
```

```

-----VERIFICAR OR
resultado = falso;
para cada condición
    resultado = verificar la
    condicion actual
    si es verdadero
        parar
fin para
  
```

-----VERIFICAR LA CONDICIÓN ACTUAL

si el valor es null o ni siquiera está presente

retornar falso

de lo contrario

si el operador relacional es igual o diferente

comparar value con comparable dependiendo del operador, estos pueden ser tratados como strings

si el operador es mayor, mayor\_que, menor, menor que

probar:

convertir valor en un float y también comparable

comparar dependiendo del símbolo con comparable

retornar la comparación

si ocurre error:

lanzar una excepción de data invalida

retornar falso como opción default

## II. Traducción del método Select

Arreglar el path y obtener uno absoluto

ABRIR EL ARCHIVO

SEPARAR TOKENS DEL MODELO DEL SELECT

Seterar las columnas correspondientes

Comenzar con el filtrado de columnas

...cuando la línea filtrada de como respuesta verdadero:

verificar si se deben considerar todas las columnas

Si es verdadero

al contenido actual añadirle el contenido de la columna actual


Si es falso

para cada columna que debe ser seleccionada

obtener el índice de dicho valor de la línea actual y añadirlo al

contenido

si no es la última columna



```
        añadir coma
    si es la última
        añadir salto de línea
    fin para
```

### III. Traducción del método Update

Usa los métodos descritos anteriormente, pero para poder actualizar:

Leer cada línea

Si tenemos filtros

Editar las columnas especificadas

De lo contrario

Guardar la columna tal y como está

Por último setear el nuevo contenido en el archivo