MANUAL TECNICO

José Carlos Soberanis Ramírez
201730246

Analisis de .CSV

Analisis Lexico

Se usaron las siguientes macros y expresiones regulares para identificar los tokens.

```
"," {}
   {ID}            {Token ID}
   ({ID}|("\ "))*  {Token TEXTO}
   "\n"            {Token FINLINEA}
   {WHITE}         {Ignorar los espacios en blanco}
   .               {Cualquier otra cosa será un error}
```

Analisis Sintactico

Se usaron las siguientes reglas de produccion empezando por s0.

G={N,T,P,S}
N={s0, s1, s2,s3,s4}
T={ID, FINLINEA, COMA, CADENA}
S={s0}
P={

s0 ::= ID s1 FINLINEA s2 |error FINLINEA s2;

s1 ::= | COMA ID s1 |error FINLINEA s2;

s2 ::= s4 s3 FINLINEA |error FINLINEA s2;

s3 ::= | COMA s4 s3 |error FINLINEA s2;

s4 ::= ID | CADENA ;

}

Analisis de .IDE

Analisis lexico

Se usaron los siguientes macros y expresiones regulares para identificar los Tokens

```
L = [a-zA-Z]
S = [-_@+*#]
D = [0-9]
ID = ({L}|{D}|{S})+
WHITE = ([\n\r\t\f])+
EXT = ".csv"
```

"PROYECTO" {Token Palabra_Reservada_Proyecto}
"CARPETA" {Token Palabra_Reservada_Carpeta}
"ARCHIVO" {Token Palabra_Reservada_Archivo}
("<")("\ ")*("/")("\ ")*("PROYECTO")("\ ")*(">") {Token FinProyecto}
("<")("\ ")*("/")("\ ")*("CARPETA")("\ ")*(">") {Token FinCarpeta}
"nombre" {Token Palabra_Reservada_nombre}
"<" {Token MENOR}
">" {Token MAYOR}
"=" {Token ASIGNACION}
("\"")((("/")({ID})(("\ ")+({ID}))*)+)({EXT})("\"") {Token PATH}
("\"")({ID}|("\ "))*("\"") {Token CADENA}
"/" {Token SLASH}
{WHITE} {/* ignore */}
("\ ") {/*ignore*/}
. {Todo lo no definido se considera un error}

Analisis sintactico

Se uso la siguiente gramatica para analizar el .ide
G={N,T,P,S}
N={s0,s1,s2}
T={PROYECTO, FINPROYECTO,CARPETA, FINCARPETA, ARCHIVO, MENOR, MAYOR, ASIGNACION, nombre, ubicación}
P={

s0 ::= MENOR PROYECTO nombre ASIGNACION CADENA MAYOR s1 FINPROYECTO
| error MENOR s2;

s1 ::= MENOR s2
| error MENOR s2
| ;

s2 ::= ARCHIVO nombre ASIGNACION CADENA ubicacion ASIGNACION PATH SLASH MAYOR s1
| CARPETA nombre ASIGNACION CADENA MAYOR s1 FINCARPETA s1
| error MENOR s2 ;

}

Analisis de instrucción SQL

Analisis lexico

Se usaron los siguientes macros y expresiones regulares para identificar los tokens
L = [a-zA-Z]
S = [-_@+*#]
TD = [0-9]
D = [1-9]
C = ["<=","<",">",">=","<>"]

```
ID = ({L})({L}|{D}|{S})*
WHITE = ([\n\r\t\f])+

"SELECCIONAR" {Palabra reservada SELECCIONAR}
   "INSERTAR" {Palabra reservada INSERTAR}
   "ELIMINAR" {Palabra reservada ELIMINAR}
   "EN" {Palabra reservada EN}
   "FILTRAR" {Palabra reservada FILTRAR}
   "VALORES" {Palabra reservada VALORES}
   "ACTUALIZAR" {Palabra reservada ACTUALIZAR}
   "ASIGNAR" {Palabra reservada ASIGNAR}
   "AND" {Palabra reservada AND}
   "OR" {Palabra reservada OR}
   "(" {PARENTESISA}
   ")" {PARENTESISC}
   ("\"")({L}|{C}|{TD}|{S}|("\ "))*("\"") {CADENA}
   "," {COMA}
   "*" {TODO}
   ";" {FINLINEA}
   "=" {ASIGNACION}
   ({D})({TD})* {NUM}
   {ID} {ID}
   ({ID})(("."){ID}))* {PATH}
   {C} {OPERADOR}
   {WHITE} {/* ignore */}
   ("\ ") {/*ignore*/}
   . {Deteccion de errores}
```

Analisis sintactico

Se uso la siguiente gramatica para el reconocimiento y analisis sintactico

```
G={N,T,P,S}
N={inicio, orden, listColumn, listColumnP, condiciones, comparacionList,
      comparacionListP, listColumn2, valueList, valueListP, listAsigValue, listAsigValueP, value,
      comparacionListAnd, comparacionListOr, operadorCondicion}
T={SELECCIONAR, EN, FILTRAR, INSERTAR, VALORES, ACTUALIZAR,
      ASIGNAR,ASIGNACION, ELIMINAR, PARENTESISA, PARENTESISC,
     COMA, FINLINEA, AND, OR, TODO, CADENA, ID, NUM, OPERADOR, PATH}
P={

inicio ::= orden FINLINEA;

orden ::= SELECCIONAR listColumn EN PATH condiciones
       | INSERTAR EN PATH listColumn2 VALORES PARENTESISA valueList PARENTESISC
       | ACTUALIZAR EN PATH ASIGNAR listAsigValue condiciones
       | ELIMINAR EN PATH condiciones;
```

listColumn ::= TODO
| ID listColumnP;

listColumnP ::= | COMA ID listColumnP;

condiciones ::= | FILTRAR comparacionList;

comparacionList ::= ID operadorCondicion value comparacionListP;

comparacionListP ::= | AND ID operadorCondicion value comparacionListAnd
| OR ID operadorCondicion value comparacionListOr;

listColumn2 ::= | PARENTESISA ID listColumnP PARENTESISC;

valueList ::= value valueListP;

valueListP ::= | COMA value valueListP;

listAsigValue ::= ID ASIGNACION value listAsigValueP;

listAsigValueP ::= | COMA listAsigValue;

value ::= CADENA | NUM;

comparacionListAnd ::= |AND ID operadorCondicion value comparacionListAnd;

comparacionListOr ::= | OR ID operadorCondicion value comparacionListOr;

operadorCondicion ::= ASIGNACION | OPERADOR;

}