GRAMATICAS

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 2

Carnet	Nombre
201503712	Oscar Rene Cuéllar Mancilla

29 DE SEPTIEMBRE DE 2019

Contenido

1. Gramática de CQL

```
#region Gramatica
BLOCK.Rule = MakePlusRule(BLOCK, GLOBAL);

GLOBAL.Rule = SENTENCIA
            | FUNCION
            | PROCEDURE
            | BATCH + puntocoma
            | TYPES + puntocoma
            | DECLARACION + puntocoma
            | INSTRUCCION + puntocoma
            | DML + puntocoma
            | DCL + puntocoma
            | TCL + puntocoma
            | DDL + puntocoma
            | CURSOR + puntocoma
            | SELECT + puntocoma;

CURSOR.Rule = res_cursor + arroba + id + res_is + SELECT
    | res_cursor + arroba + id + igual + E
    | res_open + arroba + id
    | res_close + arroba + id;

ASIG_CURSOR.Rule = SELECT
    | E;

PROCEDURE.Rule = res_procedure + id + l_parent + LISTA_PARAMETROS + r_parent + coma +
    l_parent + LISTA_PARAMETROS + r_parent + l_llave + BLOCK + r_llave;

FUNCION.Rule = TIPO + id + l_parent + LISTA_PARAMETROS + r_parent + l_llave + BLOCK + r_llave;
//FUNCION.NodeCaptionTemplate = "def #{1}(...)";
```

```
//================== OTROS CQL ====================================
BATCH.Rule = res_begin + res_bath + LISTA_DML + res_apply + res_bath;

FUN_AGR.Rule = TIPO_AGR + l_parent + decremento + SELECT + aumento + r_parent;

TIPO_AGR.Rule = res_count
    | res_min
    | res_max
    | res_sum
    | res_avg;

//==================== DDL =======================================
DDL.Rule = res_create + res_table + INE + id + l_parent + LISTA_COLDEF + r_parent
    | res_alter + res_table + id + res_add + LISTA_COLDEF
    | res_alter + res_table + id + res_drop + LISTA_IDS
    | res_create + res_database + INE + id
    | res_drop + res_table + IE + id
    | res_truncate + res_table + id
    | res_drop + res_database + id
    | res_use + id;

//================= DML ==========================================
LISTA_DML.Rule = MakePlusRule(LISTA_DML, DML2);

DML2.Rule = DML + puntocoma;

DML.Rule = res_insert + res_into + id + res_values + l_parent + LISTA_E + r_parent
    | res_insert + res_into + id + l_parent + LISTA_IDS + r_parent + res_values + l_parent + LISTA_E + r_parent
    | res_update + id + res_set + LISTA_ASIG_CQL + WHERE_Q
    | res_delete + ACCESO_ARR_Q + res_from + id + WHERE_Q;
```

```
LIMIT_Q.Rule = LIMIT
    | Empty;

LIMIT.Rule = res_limit + E;

ORDERBY_Q.Rule = ORDERBY
    | Empty;

ORDERBY.Rule = res_order + res_by + LISTA_ORDER;

LISTA_ORDER.Rule = MakePlusRule(LISTA_ORDER, coma, ORDER);

ORDER.Rule = id
    | id + res_desc
    | id + res_asc;

WHERE_Q.Rule = WHERE
    | Empty;

WHERE.Rule = res_where + E;

LISTA_ASIG_CQL.Rule = MakePlusRule(LISTA_ASIG_CQL,coma,ASIG_CQL);

ASIG_CQL.Rule = id + igual + E
    | ACCESO_ARR + igual + E
    | REFERENCIAS + igual + E;

ACCESO_ARR.Rule = id + l_corchete + E + r_corchete;

ACCESO_ARR_Q.Rule = ACCESO_ARR | id | Empty;
```

```
//=================== DCL ============================================
DCL.Rule = res_create + res_user + id + res_with + res_password + E
    | res_grant + id + res_on + id
    | res_revoke + id + res_on + id;

//=================== TCL ============================================
TCL.Rule = res_commit
    | res_rollback;

//=================== TYPES ==========================================
TYPES.Rule = res_create + res_type + INE + id + l_parent + LISTA_CQLTIPOS + r_parent;

INE.Rule = res_if + res_not + res_exists
    | Empty;

IE.Rule = res_if + res_exists
    | Empty;

LISTA_COLDEF.Rule = MakePlusRule(LISTA_COLDEF, coma, COLDEF);

COLDEF.Rule = id + TIPO + res_primary + res_key
    | id + TIPO
    | res_primary + res_key + l_parent + LISTA_IDS + r_parent;

//============ lista CQL TIPOS
LISTA_CQLTIPOS.Rule = MakeStarRule(LISTA_CQLTIPOS, coma, CQLTIPO);

CQLTIPO.Rule = id + TIPO;

//============ lista parametros
LISTA_PARAMETROS.Rule = MakeStarRule(LISTA_PARAMETROS, coma, UNPARAMETRO);

UNPARAMETRO.Rule = TIPO + arroba + id;
```

```
ID_ARROBA.Rule = arroba + id;

//============ LISTA IDS
LISTA_IDS.Rule = MakeStarRule(LISTA_IDS, coma, id);

//============= DECLARACION
DECLARACION.Rule = TIPO + LISTA_DECLARACION_E;

LISTA_DECLARACION_E.Rule = MakePlusRule(LISTA_DECLARACION_E, coma, DECLARACION_E);

DECLARACION_E.Rule = arroba + id + igual + E
    | arroba + id;
//===================================
```

```
INSTRUCCION.Rule = res_log + l_parent + E + r_parent
            //========== ver aquí ambiguedad entre referencias y reasignacion
            | REFERENCIAS + igual + E
            | res_return + LISTA_E
            | THROW
            | REASIGNACION
            | REASIGNACION2
            | ACTUALIZACION2
            | CORTE
            | REFERENCIAS;

CORTE.Rule = res_break
            | res_continue;

REASIGNACION.Rule = arroba + id + igual + E;

REASIGNACION2.Rule = LISTA_IDS_ARROBA + igual + E;

OPERADOR.Rule = mas | menos | modular | por | div;

TIPO.Rule = res_int
    | res_boolean
    | res_double
    | res_string
    | res_counter
    | res_set
    | res_map
    | res_list
    | id
    | res_date
    | res_time
    | res_cursor
    | res_list + menor_que + TIPO + mayor_que
    | res_set + menor_que + TIPO + mayor_que
    | res_map + menor_que + TIPO + coma + TIPO + mayor_que;
```

```
SENTENCIA.Rule = IF
               | FOR
               | WHILE
               | SWITCH
               | TRY;

IF.Rule = res_if + l_parent + E + r_parent + l_llave + BLOCK + r_llave + ELSEIFS + ELSE
               | res_if + l_parent + E + r_parent + l_llave + BLOCK + r_llave + ELSEIFS;

FOR.Rule = res_for + l_parent + FUENTE_FOR + puntocoma + E + puntocoma + ACTUALIZACION2 + r_parent + l_llave + BLOCK + r_llave
    | res_for + res_each + l_parent + LISTA_PARAMETROS + r_parent + res_in + arroba + id + l_llave + BLOCK + r_llave;

FUENTE_FOR.Rule = DECLARACION
               | REASIGNACION;

ACTUALIZAR.Rule = arroba + id + OPERADOR + igual + E;

ACTUALIZACION2.Rule = ACTUALIZACION
    | ACTUALIZAR;

ACTUALIZACION.Rule = arroba + id + mas_mas
    | arroba + id + menos_menos;

WHILE.Rule = res_while + l_parent + E + r_parent + l_llave + BLOCK + r_llave
    | res_do + l_llave + BLOCK + r_llave + res_while + l_parent + E + r_parent + puntocoma;

SWITCH.Rule = res_switch + l_parent + E + r_parent + l_llave + LISTA_CASOS + CASO_DEF + r_llave;

TRY.Rule = res_try + l_llave + BLOCK + r_llave + res_catch + l_parent + EXCEPTION + arroba
                    + id + r_parent + l_llave + BLOCK + r_llave;
```

```
//========================== EXPRESIONES ============================================
E.Rule = TERMINO | UNARIO
    | E + mas + E
    | E + menos + E
    | E + por + E
    | E + div + E
    | E + pot + E
    | E + modular + E
    | E + igual_igual + E
    | E + not_igual + E
    | E + mayor_igual + E
    | E + menor_igual + E
    | E + mayor_que + E
    | E + menor_que + E
    | E + xor + E
    | E + or + E
    | E + and + E
    | E + res_in + E
    | E + interrogacion + E + dospuntos + E;

TERMINO.Rule = PRIMITIVO | E_PARENT | NATIVAS | COLECCION | REFERENCIAS | CASTEOS | INSTANCIA | FUN_AGR | ACTUALIZACION;

INSTANCIA.Rule = res_new + id
    | res_new + res_list + menor_que + TIPO + mayor_que
    | res_new + res_set + menor_que + TIPO + mayor_que
    | res_new + res_map + menor_que + TIPO + coma + TIPO + mayor_que;

CASTEOS.Rule = l_parent + TIPO + r_parent + E;

NATIVAS.Rule = DATE_NOW;

DATE_NOW.Rule = res_today + l_parent + r_parent
    | res_now + l_parent + r_parent;

THROW.Rule = res_throw + res_new + EXCEPTION;
```

```
REFERENCIAS.Rule = MakePlusRule(REFERENCIAS, punto, REFERENCIA);

COLECCION.Rule = l_corchete + LISTA_E + r_corchete //instancia de list
    | l_corchete + KEY_VALUE_LIST + r_corchete //instancia de map
    | l_llave + KEY_VALUE_LIST + r_llave //instancia de map
    | l_llave + LISTA_E + r_llave //instancia de set
    | l_llave + LISTA_E + r_llave + res_as + id; //instancia de type

KEY_VALUE_LIST.Rule = MakePlusRule(KEY_VALUE_LIST, coma, KEY_VALUE);

KEY_VALUE.Rule = E + dospuntos + E;

E_PARENT.Rule = l_parent + E + r_parent;

UNARIO.Rule = mas + E | menos + E | not + E;

PRIMITIVO.Rule = id | entero | er_decimal | cadena | cadena2 | res_true | res_false | res_null | arroba + id;

//======================================================================

LISTA_E.Rule = MakeStarRule(LISTA_E, coma, E);

LLAMADA_FUNCION.Rule = id + l_parent + LISTA_E + r_parent
    | res_insert + l_parent + LISTA_E + r_parent
    | res_set + l_parent + LISTA_E + r_parent
    | res_call + id + l_parent + LISTA_E + r_parent;
//LLAMADA_FUNCION.NodeCaptionTemplate = "Llamada #{0}(...)";

this.Root = BLOCK;
#endregion
```

2. Gramática de Chison

```
#region Gramatica
S.Rule = dolar + menor_que + DATABASES + coma + USERS + mayor_que + dolar
    | Empty;

//============================ IMPORT ========================================
IMPORT.Rule = dolar + l_llave + id + punto + res_chison + r_llave + dolar;

//========================= USERS ============================================
USERS.Rule = res_users + igual + l_corchete + LISTA_USER + r_corchete;

LISTA_USER.Rule = MakeStarRule(LISTA_USER, coma, USER)
    | IMPORT;

USER.Rule = menor_que + LISTA_DATA_USER + mayor_que;

LISTA_DATA_USER.Rule = MakePlusRule(LISTA_DATA_USER, coma,DATA_USER);

DATA_USER.Rule = res_name + igual + cadena
    | res_password + igual + cadena
    | res_permissions + igual + l_corchete + LISTA_PERMISOS + r_corchete;

LISTA_PERMISOS.Rule = MakeStarRule(LISTA_PERMISOS,coma, PERMISO)
    | IMPORT;

PERMISO.Rule = menor_que + res_name + igual + cadena + mayor_que;

//========================= DATABASES ========================================

DATABASES.Rule = res_databases + igual + l_corchete + LISTA_BASES + r_corchete;

LISTA_BASES.Rule = MakeStarRule(LISTA_BASES,coma,BASE)
    | IMPORT;

BASE.Rule = menor_que + res_name + igual + cadena + coma +
    res_data + igual + l_corchete + LISTA_DATA_BASE + r_corchete + mayor_que ;
```

```
DATA_BASE.Rule = menor_que + LISTA_CUERPO_DB + mayor_que;

LISTA_CUERPO_DB.Rule = MakePlusRule(LISTA_CUERPO_DB,coma,CUERPO_DB);

CUERPO_DB.Rule = res_name + igual + cadena
    | res_cql_type + igual + cadena
    | res_columns + igual + l_corchete + LISTA_COLUMNAS + r_corchete
    | res_data + igual + l_corchete + LISTA_DATA_COLUMNAS1 + r_corchete
    | res_attrs + igual + l_corchete + LISTA_ATTRS + r_corchete
    | res_parameters + igual + l_corchete + LISTA_PARAMETERS + r_corchete
    | res_instr + igual + cadena3;

LISTA_PARAMETERS.Rule = MakeStarRule(LISTA_PARAMETERS,coma,PARAMETER)
    | IMPORT;

PARAMETER.Rule = menor_que + LISTA_CARACT_PARAMETER + mayor_que;

LISTA_CARACT_PARAMETER.Rule = MakeStarRule(LISTA_CARACT_PARAMETER,coma,CARACT_PARAMETER);

CARACT_PARAMETER.Rule = res_name + igual + cadena
    | res_type + igual + cadena
    | res_as + igual + res_in
    | res_as + igual + res_out;

LISTA_ATTRS.Rule = MakeStarRule(LISTA_ATTRS,coma , ATTRS)
    | IMPORT;

ATTRS.Rule = menor_que + LISTA_ATTRS_VALS + mayor_que;

LISTA_COLUMNAS.Rule = MakeStarRule(LISTA_COLUMNAS,coma,COLUMNA)
    | IMPORT;

COLUMNA.Rule = menor_que + LISTA_CARACT_COLUMNA + mayor_que;
```

```
DATA_COLUMNAS.Rule = cadena + igual + VALOR;

LISTA_VALORES.Rule = MakeStarRule(LISTA_VALORES, coma, VALOR);

VALOR.Rule = PRIMITIVO
    | LISTA
    | MAP;

LISTA.Rule = l_corchete + LISTA_VALORES + r_corchete
    | l_llave + LISTA_VALORES + r_llave;

MAP.Rule = menor_que + LISTA_KEY_VALUE_PAIR + mayor_que
    | IMPORT;

LISTA_ATTRS_VALS.Rule = MakeStarRule(LISTA_ATTRS_VALS, coma, ATTRS_VALS);
```

```
ATTRS_VALS.Rule = cadena + igual + cadena;

LISTA_KEY_VALUE_PAIR.Rule = MakeStarRule(LISTA_KEY_VALUE_PAIR,coma,KEY_VALUE_PAIR);

KEY_VALUE_PAIR.Rule = VALOR + igual + VALOR;

PRIMITIVO.Rule = numero
    | cadena
    | cadena2
    | res_true
    | res_false
    | res_null;

this.Root = S;
//RECUPERACION
//BLOCK.ErrorRule = SyntaxError + puntocoma;
//BLOCK.ErrorRule = SyntaxError + r_llave;
#endregion
```

3. Gramática LUP (Servidor)

```
#region Gramatica
S.Rule = LOGIN
    | LOGOUT
    | STRUCT
    | QUERY;

LOGIN.Rule = res_loginOpen + res_userOpen + id + res_userClose + res_pass + res_loginClose;

LOGOUT.Rule = res_logoutOpen + res_userOpen + id + res_userClose + res_logoutClose;

QUERY.Rule = res_queryOpen + res_userOpen + id + res_userClose + res_data + res_queryClose;

STRUCT.Rule = res_structOpen + res_userOpen + id + res_userClose + res_structClose;

this.Root = S;
#endregion
```

4. Gramática LUP (Cliente)

```
%% /* Definición de la gramática */

S : LIST_BLOCK EOF{
    return ast;
}
;

LIST_BLOCK : LIST_BLOCK BLOCK
    | BLOCK
;

BLOCK : MENSAJE {ast.mensajes[ast.contMess++] = $1;}
    | DATA {ast.data[ast.contData++] = $1;}
    | ERROR {ast.errores[ast.contErr++] = $1;}
    | DBMS
    | LOGOUT
    | LOGIN
;

LOGIN : 'res_loginOpen' STATUS 'res_loginClose'
;

LOGOUT : 'res_logoutOpen' STATUS2 'res_logoutClose'
;

STATUS2 : res_success {ast.logout = true;}
    | res_fail {ast.logout = false;}
;

STATUS : res_success {ast.login = true;}
    | res_fail {ast.login = false;}
;

MENSAJE : 'res_messageOpen' T1 'res_messageClose'
    {$$ = $2;}
;
```

```
DATA : 'res_dataOpen' T1 'res_dataClose'
    {$$=$2;}
;

ERROR : 'res_errorOpen'
        'res_lexemaOpen' T1 'res_lexemaClose'
        'res_lineOpen' 'numero' 'res_lineClose'
        'res_columnOpen' 'numero' 'res_columnClose'
        'res_typeOpen'  T1 'res_typeClose'
        'res_descripcionOpen' T1 'res_descripcionClose'
        'res_errorClose'
        {var error = new TokenError(); error.lexema = $3; error.fila = $6; error.columna=$9;
            error.tipo = $12; error.descripcion = $15;
            $$ = error;}
;

DBMS : res_dataBasesOpen DATA_BASES res_dataBasesClose
    | res_dataBasesOpen res_dataBasesClose
;

DATA_BASES : DATA_BASES res_dataBaseOpen CONTENIDO res_dataBaseClose
    | res_dataBaseOpen CONTENIDO res_dataBaseClose
;

CONTENIDO : res_nameOpen id res_nameClose res_tablesOpen TABLES res_typesOpen TYPES res_proceduresOpen PROCEDURES {
    var db = {};
    db.name = $2;
    db.tables = $5;
    db.types = $7;
    db.procedures = $9;
    ast.dbms.dataBases[ast.dbms.contDataBase++] = db;
}
;
```

```
TABLES : TABLE res_tablesClose {
    $$ = $1;
}
    | res_tablesClose {
        $$ = {};
    }
;

TABLE : TABLE res_tableOpen res_nameOpen id res_nameClose COLUMNAS res_tableClose {
        var tab = {};
        var tables = $1;
        tab.columns = $6;
        tab.name = $4;
        tables[ast.dbms.contTab++] = tab;
        $$ = tables;
    }
    | res_tableOpen res_nameOpen id res_nameClose COLUMNAS res_tableClose {
        var tab = {};
        var tables = {};
        tab.columns = $5;
        tab.name = $3;
        ast.dbms.contTab = 0;
        tables[ast.dbms.contTab++] = tab;
        $$ = tables;
    }
;
```

```
COLUMNAS : COLUMNAS res_columnOpen id res_columnClose {
        var columns = $1;
        columns[ast.dbms.contColumns++] = $3;
        $$ = columns;
    }
    | res_columnOpen id res_columnClose {
        var columns = {};
        ast.dbms.contColumns = 0;
        columns[ast.dbms.contColumns++] = $2;
        $$ = columns;
    }
;

TYPES : TYPE res_typesClose {
    $$ = $1;
}
    | res_typesClose {
        $$ = {};
    }
;
```

```
TYPE : TYPE res_typeCQLOpen res_nameOpen id res_nameClose ATRIBUTOS res_typeCQLClose {
    var typ = {};
    var types = $1;
    typ.atrs = $6;
    typ.name = $4;
    types[ast.dbms.contTypes++] = typ;
    $$ = types;
}
    | res_typeCQLOpen res_nameOpen id res_nameClose ATRIBUTOS res_typeCQLClose {
        var typ = {};
        var types = {};
        typ.atrs = $5;
        typ.name = $3;
        ast.dbms.contTypes = 0;
        types[ast.dbms.contTypes++] = typ;
        $$ = types;
    }
;

PROCEDURES : PROCEDURE res_proceduresClose {
    $$ = $1;
}
    | res_proceduresClose {
        $$ = {};
    }
;
```

```
PROCEDURE : PROCEDURE res_procedureOpen id res_procedureClose {
    var procedures = $1;
    procedures[ast.dbms.contProcedures++] = $3;
    $$ = procedures;
}
    | res_procedureOpen id res_procedureClose {
        var procedures = {};
        ast.dbms.contProcedures = 0;
        procedures[ast.dbms.contProcedures++] = $2;
        $$ = procedures;
    }
;

ATRIBUTOS : ATRIBUTOS res_atributtesOpen id res_atributtesClose {
    var atrs = $1;
    atrs[ast.dbms.contAtrs++] = $3;
    $$ = atrs;
}
    | res_atributtesOpen id res_atributtesClose {
        var atrs = {};
        ast.dbms.contAtrs = 0;
        atrs[ast.dbms.contAtrs++] = $2;
        $$ = atrs;
    }
;

T1 : T1 'TODO' {$$ = $1+$2;}
    | T1 'WS' {$$ = $1+$2;}
    | 'TODO' {$$=$1;}
    | 'WS' {$$=$1;}
;
```