GRAMÁTICA

```
<INIT> ::= <INSTRUCTIONS> | ε

<INSTRUCTIONS> ::= <INSTRUCTIONS> <INSTRUCTION> | <INSTRUCTION>

<INSTRUCTION> ::= <CREATEDB> TK_semicolon
                | <USEDB> TK_semicolon
                | <CREATETABLE> TK_semicolon
                | <ALTERTAB> TK_semicolon
                | <DROPTAB> TK_semicolon
                | <INSERTREG> TK_semicolon
                | <UPDATETAB> TK_semicolon
                | <TRUNCATETAB> TK_semicolon
                | <DELETETAB> TK_semicolon
                | <SELECT> TK_semicolon
                | <DECLAREID> TK_semicolon
                | <ASIGNID> TK_semicolon
                | <IFSTRUCT> TK_semicolon
                | <CASESTRUCT_S> TK_semicolon
                | <WHILESTRUCT> TK_semicolon
                | <FORSTRUCT> TK_semicolon
                | <FUNCDEC> TK_semicolon
                | <CALLFUNC> TK_semicolon
                | <ENCAP> TK_semicolon
                | <PRINT> TK_semicolon
                | RW_break TK_semicolon
                | RW_continue TK_semicolon
                | RW_return <EXP> TK_semicolon
                | RW_return TK_semicolon

<CREATEDB> ::= RW_create RW_data RW_base TK_field

<USEDB> ::= RW_use TK_field

<DECLAREID> ::= RW_declare <DECLIDS>
              | RW_declare TK_id TYPE TK_equal <EXP>

<DECLIDS> ::= <DECLIDS> TK_comma <DECLID> | <DECLID>

<DECLID> ::= TK_id TYPE

<ASIGNID> ::= RW_set TK_id TK_equal <EXP>
            | TK_id inc
            | TK_id dec
```

```
<SELECT> ::= RW_select <FIELDS> RW_from TK_field RW_where <EXP>
          | RW_select <FIELDS> RW_from TK_field
          | RW_select <LIST_IDS>

<FIELDS> ::= <LIST_IDS> | TK_mult

<LIST_IDS> ::= <LIST_IDS> TK_comma <IDS> | <IDS>

<IDS> ::= <EXP> RW_as TK_nvarchar
        | <EXP> RW_as TK_field
        | <EXP>

<CREATETABLE> ::= RW_create RW_table TK_field TK_lpar <ATTRIBUTES>
TK_rpar

<ATTRIBUTES> ::= <ATTRIBUTES> TK_comma <ATTRIBUTE> | <ATTRIBUTE>

<ATTRIBUTE> ::= TK_field TYPE TK_lpar TK_int TK_rpar <PROPS>
          | TK_field TYPE <PROPS>
          | TK_field TYPE TK_lpar TK_int TK_rpar
          | TK_field TYPE

<PROPS> ::= RW_not RW_null RW_primary RW_key <FKEY>
          | RW_not RW_null RW_primary RW_key
          | RW_primary RW_key RW_not RW_null <FKEY>
          | RW_primary RW_key RW_not RW_null
          | RW_not RW_null <FKEY>
          | RW_not RW_null
          | RW_primary RW_key <FKEY>
          | RW_primary RW_key
          | <FKEY>

<FKEY> ::= RW_ref TK_field TK_lpar TK_field TK_rpar

<ALTERTAB> ::= RW_alter RW_table TK_field <ACTION>

<ACTION> ::= RW_add TK_field TYPE
          | RW_drop TK_field
          | RW_rename RW_to TK_field
          | RW_rename RW_column TK_field RW_to TK_field

<DROPTAB> ::= RW_drop RW_table TK_field

<INSERTREG> ::= RW_insert RW_into TK_field TK_lpar <LIST_ATTRIBS>
TK_rpar RW_values TK_lpar <LIST_EXPS> TK_rpar

<LIST_ATTRIBS> ::= <LIST_ATTRIBS> TK_comma TK_field | TK_field
```

```
<LIST_EXPS> ::= <LIST_EXPS> TK_comma <EXP> | <EXP>

<UPDATETAB> ::= RW_update TK_field RW_set <VALUESTAB> RW_where <EXP>

<VALUESTAB> ::= <VALUESTAB> TK_comma <VALUETAB> | <VALUETAB>

<VALUETAB> ::= TK_field TK_equal <EXP>

<TRUNCATETAB> ::= RW_truncate RW_table TK_field

<DELETETAB> ::= RW_delete RW_from TK_field RW_where <EXP>

<IFSTRUCT> ::= RW_if <EXP> RW_then <ENCAP> RW_else <ENCAP> RW_end
RW_if
           | RW_if <EXP> RW_then <ENCAP> RW_end RW_if
           | RW_if <EXP> RW_begin <ENCAP> RW_end

<CASESTRUCT_S> ::= RW_case <EXP> <WHENELSE> RW_end RW_as TK_field
             | RW_case <EXP> <WHENELSE> RW_end RW_as TK_nvarchar
             | RW_case <EXP> <WHENELSE> RW_end
             | RW_case <WHENELSE> RW_end RW_as TK_field
             | RW_case <WHENELSE> RW_end RW_as TK_nvarchar
             | RW_case <WHENELSE> RW_end

<WHENELSE> ::= <WHENS> <ELSE>
            | <WHENS>
            | <ELSE>

<WHENS> ::= <WHENS> <WHEN> | <WHEN>

<WHEN> ::= RW_when <EXP> RW_then <EXP>

<ELSE> ::= RW_else RW_then <EXP>

<PRINT> ::= RW_print <EXP>

<WHILESTRUCT> ::= RW_while <EXP> <ENCAP>

<FORSTRUCT> ::= RW_for TK_id RW_in <EXP> TK_dot <EXP> <ENCAP> RW_loop

<FUNCDEC> ::= RW_create RW_function TK_field TK_lpar <PARAMS> TK_rpar
RW_returns <TYPE> <ENCAP>
          | RW_create RW_function TK_field TK_lpar TK_rpar RW_returns
<TYPE> <ENCAP>
          | RW_create RW_procedure TK_field <PARAMS> RW_as <ENCAP>
          | RW_create RW_procedure TK_field RW_as <ENCAP>
          | RW_create RW_procedure TK_field TK_lpar <PARAMS> TK_rpar
<ENCAP>
```

```
          | RW_create RW_procedure TK_field <ENCAP>

<PARAMS> ::= <PARAMS> TK_comma <PARAM> | <PARAM>

<PARAM> ::= TK_id RW_as <TYPE>

<ENCAP> ::= RW_begin <INSTRUCTIONS> RW_end
          | RW_begin RW_end

<CALLFUNC> ::= TK_field TK_lpar <ARGS> TK_rpar
           | TK_field TK_lpar TK_rpar

<ARGS> ::= <ARGS> TK_comma <EXP> | <EXP>

<EXP> ::= <ARITHMETICS>
       | <RELATIONALS>
       | <LOGICS>
       | <CAST>
       | <NATIVEFUNC>
       | <CALLFUNC>
       | <TERNARY>
       | TK_id
       | TK_id TK_point TK_id
       | TK_field
       | TK_field TK_point TK_field
       | TK_nvarchar
       | TK_int
       | TK_decimal
       | TK_date
       | TK_datetime
       | RW_null
       | TK_lpar <EXP> TK_rpar

<ARITHMETICS> ::= <EXP> TK_plus <EXP>
               | <EXP> TK_minus <EXP>
               | <EXP> TK_mult <EXP>
               | <EXP> TK_div <EXP>
               | <EXP> TK_mod <EXP>
               | TK_minus <EXP> %prec TK_uminus

<RELATIONALS> ::= <EXP> TK_equalequal <EXP>
               | <EXP> TK_equal <EXP>
               | <EXP> TK_notequal <EXP>
               | <EXP> TK_lessequal <EXP>
               | <EXP> TK_greatequal <EXP>
               | <EXP> TK_less <EXP>
               | <EXP> TK_great <EXP>
```

```
<LOGICS> ::= <EXP> TK_and <EXP>
        | <EXP> TK_or <EXP>
        | TK_not <EXP>

<CAST> ::= RW_cast TK_lpar <EXP> RW_as <TYPE> TK_rpar

<NATIVEFUNC> ::= RW_concatenar TK_lpar <EXP> TK_comma <EXP> TK_rpar
            | RW_substraer TK_lpar <EXP> TK_comma <EXP> TK_comma <EXP>
TK_rpar
            | RW_hoy TK_lpar TK_rpar

<TERNARY> ::= RW_if TK_lpar <EXP> TK_comma <EXP> TK_comma <EXP>
TK_rpar

<TYPE> ::= RW_int
        | RW_bit
        | RW_decimal
        | RW_date
        | RW_datetime
        | RW_nchar
        | RW_nvarchar
```

Explicación de la Gramática:

La gramática proporcionada está diseñada para describir la sintaxis de un lenguaje de programación o consulta de bases de datos. A continuación, se presenta una descripción de las principales construcciones y reglas gramaticales:

Estructura del Programa:

Un programa comienza con <INIT>, que puede ser un conjunto de instrucciones o estar vacío (ε).

Instrucciones:

Las instrucciones están definidas por la regla <INSTRUCTIONS>, que puede ser una instrucción única o una secuencia de instrucciones.

Tipos de Instrucciones:

La gramática cubre una variedad de instrucciones, como la creación y uso de bases de datos, manipulación de tablas (crear, modificar, eliminar), inserción y actualización de

registros, consultas SELECT, declaraciones de variables, estructuras de control de flujo (if, case, while, for), definición y llamada de funciones y procedimientos, entre otros.

Expresiones y Operadores:

Las expresiones (<EXP>) incluyen aritmética, relaciones lógicas y comparaciones, así como funciones nativas y llamadas a funciones definidas por el usuario.

Operaciones en Tablas:

Las operaciones en tablas incluyen la selección de datos (SELECT), inserción de registros (INSERT), actualización de registros (UPDATE), truncado de tablas (TRUNCATE), y eliminación de registros (DELETE).

Estructuras de Control de Flujo:

Se admiten estructuras de control de flujo como IF, CASE, WHILE y FOR, cada una con su propia sintaxis y reglas.

Declaración y Uso de Funciones/Procedimientos:

La gramática permite la definición de funciones y procedimientos con parámetros y retorno de valores.

Expresiones Nativas y Conversiones:

Incluye funciones nativas como concatenación y extracción de substrings, así como la capacidad de realizar conversiones de tipo (CAST).

Gestión de Errores y Excepciones:

La gramática no aborda específicamente la gestión de errores y excepciones, pero esto podría implementarse según las necesidades del lenguaje.