Universidad Simón Bolívar
Dpto. de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Abril-Julio 2012

**Gramática de Atributos para SuperQuery**

# 1. Introducción

Esta miniguía sirve al estudiante del curso no sólo como apoyo a la teoría sino que sirve de apoyo para la entrega de análisis sintáctico del lenguaje que se les haya planteado, enfoncándose en la construcción de una Gramática de Atributos que permite la construcción de un Arbol Sintáctico Abstracto.

*SuperQuery* es un lenguaje cuyas instrucciones realizan manipulaciones sobre una base de datos. *Nota:* Esta guía no asume que el estudiante posea conocimiento alguno sobre base de datos, no es necesario para resolver el problema.

# 2. Definición de *SuperQuery*

*SuperQuery* posee 5 instrucciones básicas que pueden realizarse sobre bases de datos, a continuación se listan éstas junto a su definición:

## 2.1. Creación

Instrucción para la creación de una tabla sobre la base de datos. Se define la siguiente forma:

```
table <nombre_tabla>{
   unique => <nombre_columna1>:<tipo>;
   [<nombre_columna2>:<tipo>; ... <nombre_columnan>:<tipo>;]
}
```

Se define una tabla como una lista ordenada de una o varias columnas de la cuales sólo una es *clave primaria*, es decir, que existe sólo una fila en la tabla cuyo valor es único para la columna.

Los tipos de datos que pueden poseer los columnas de las tablas pueden ser: `int`, `bool` y `str`.

## 2.2. Inserción

Instrucción para la insertar una fila sobre alguna tabla de la base de datos. Se define la siguiente forma:

```
insert into <nombre_tabla> row {valor1 [,valor2, ... , valorn]}
```

Los valores a insertar deben estar en el orden en que se declararon las columnas de la tabla y sus dominios corresponder a los tipos de datos en que fueron declaradas éstas.

## 2.3. Borrado

Instrucción para borrar una fila de alguna alguna tabla de la base de datos. Se define la siguiente forma:

```
delete from <nombre_tabla> row <columna_unique>
```

El valor de `<columna_unique>` corresponde al valor de la columna declarada como `unique` en la tabla.

## 2.4. Actualización

Instrucción para actualizar los valores de una fila de alguna tabla de la base de datos. Se define la siguiente forma:

```
update in <nombre_tabla> row <columna_unique> to {valor1 [,valor2, ... , valorn]}
```

Se combinan las restricciones sobre la lista de valores y el valor de `<columna_unique>` dadas en las secciones 2.2 y 2.3.

## 2.5. Consulta

Instrucción para consultar los valores de una fila de alguna tabla de la base de datos. Se define la siguiente forma:

```
fetch from <nombre_tabla> where <condicion>
```

`<condicion>` es una expresión booleana cuyos operadores son: `and`, `or` y `not`. Los operandos de expresiones de este tipo son expresiones relaciones cuyos operadores relaciones son: `>`, `>=`, `<`, `<=`, `=` y `!=`. Los operandos de estas expresiones relaciones son expresiones aritméticas cuyos operadores aritméticos son: `+`, `-`, `*` y `/`. Los operadores prefijos y unarios para las expresiones aritméticas y booleanas son el `-` y el `not`, respectivamente.

La precendecia y asociatividades se listan a continuación en el siguiente orden:

- `*` y `/` Asocian hacia la izquierda.
- `+` y `-` Asocian hacia la derecha.
- `>`, `>=`, `<`, `<=`, `=` y `!=` No son asociativos.
- `and` Asocia hacia la izquierda.
- `or` Asocia hacia la izquierda.

# 3. Gramática de Atributos

## 3.1. Gramática

| | | |
|---|---|---|
| S | → | SI |
| | \| | I |
| I | → | `table` TkIdent { C } |
| | \| | `insert into` TkIdent `row` { V } |
| | \| | `delete from` TkIdent `row` V' |
| | \| | `update in` TkIdent `row` V' `to` { V } |
| | \| | `fetch from` TkIdent `where` E |
| C | → | CC' |
| | \| | C' |
| C' | → | U TkIdent `:` T `;` |
| U | → | `unique =>` |
| | \| | $\lambda$ |
| T | → | `int` |
| | \| | `bool` |
| | \| | `str` |
| V | → | V`,`V' |
| | \| | V' |
| V' | → | TkNum |
| | \| | TkString |
| | \| | `true` |
| | \| | `false` |
| E | → | E `<` E |
| | \| | E `<=` E |
| | \| | E `>` E |
| | \| | E `>=` E |
| | \| | E `=` E |
| | \| | E `!=` E |
| | \| | E `and` E |
| | \| | E `or` E |
| | \| | `not` E |
| | \| | E `+` E |
| | \| | E `-` E |
| | \| | E `*` E |
| | \| | E `/` E |
| | \| | `-` E |
| | \| | TkIdent |
| | \| | V' |

*Nótese* que la gramática es ambigüa, es intencional para que el lector la pueda mejorar, e incluso si desea implementar este lenguaje haga uso de las herramientas para desambigüarla.

### 3.2. Atributos

Se definen los siguientes atributos para la gramática, en su momento se ha de notar el uso:

| Atributo | Tipo |
|---|---|
| ins, cols, rows | sequence |
| ast | AST |
| val | string, bool, int |
| atr | COL |
| unique | bool |
| exp | EXP |
| tipo | string |

Donde los tipos de datos `AST`, `COL` y `EXP` se definen como:

$$AST : table : string \begin{cases} TABLE : \\ \quad cols : sequence \\ INSERT : \\ \quad rows : sequence \\ DELETE : \\ \quad val : (int \mid string) \\ UPDATE : \\ \quad val : (int \mid string) \\ \quad rows : sequence \\ FETCH : \\ \quad exp : EXP \end{cases}$$

$$EXP : op1 : EXP \begin{cases} GT : \\ \quad op2 : EXP \\ GTE : \\ \quad op2 : EXP \\ LT : \\ \quad op2 : EXP \\ LTE : \\ \quad op2 : EXP \\ EQ : \\ \quad op2 : EXP \\ NEQ : \\ \quad op2 : EXP \\ OR : \\ \quad op2 : EXP \\ AND : \\ \quad op2 : EXP \\ NOT : no\ tiene\ atributos. \\ SUM : \\ \quad op2 : EXP \\ MIN : \\ \quad op2 : EXP \\ TIM : \\ \quad op2 : EXP \\ DIV : \\ \quad op2 : EXP \\ NEG : no\ tiene\ atributos. \end{cases}$$

COL:
- unique : bool
- id : string
- tipo : string

**Nota:** Los tipos de datos AST y EXP son tipos abstractos, no pueden instanciarse directamente.

Se define la gramática de atributos que construye el arbol de sintaxis de la siguiente manera:

| | |
|---|---|
| $S \to SI$ | $\{S_0.ins = S_1.ins + + < I.ast >\}$ |
| $S \to I$ | $\{S.ins = < I.ast >\}$ |
| $I \to$ `table` $TkIdent$ $\{C\}$ | $\{I.ast = TABLE(TkIdent.val, C.cols)\}$ |
| $I \to$ `insert into` $TkIdent$ `row` $\{V\}$ | $\{I.ast = INSERT(TkIdent.val, V.rows)\}$ |
| $I \to$ `delete from` $TkIdent$ `row` $V'$ | $\{I.ast = DELETE(TkIdent.val, V'.val)\}$ |
| $I \to$ `update in` $TkIdent$ `row` $V'$`to`$\{V\}$ | $\{I.ast = UPDATE(TkIdent.val, V'.val, V.rows)\}$ |
| $I \to$ `fetch from` $TkIdent$ `where` $E$ | $\{I.ast = FETCH(TkIdent.val, E.exp)\}$ |
| $C \to CC'$ | $\{C_0.cols = C_1.cols + + < C'.atr >\}$ |
| $C \to C'$ | $\{C.cols =< C'.atr >\}$ |
| $C' \to U$ $TkIdent$ `:` $T$`;` | $\{C'.atr = COL(U.unique, TkIdent.val, T.tipo\}$ |
| $U \to$ `unique =>` | $\{U.unique = true\}$ |
| $U \to \lambda$ | $\{U.unique = false\}$ |
| $T \to$ `int` | $\{T.tipo = int\}$ |
| $T \to$ `bool` | $\{T.tipo = bool\}$ |
| $T \to$ `str` | $\{T.tipo = string\}$ |
| $V \to V$`,`$V'$ | $\{V_0.rows = V_1.rows + + < V'.val >\}$ |
| $V \to V'$ | $\{V.rows =< V'.val >\}$ |
| $V' \to TkNum$ | $\{V'.val = TkNum.val\}$ |
| $V' \to TkString$ | $\{V'.val = TkString.val\}$ |
| $V' \to$ `true` | $\{V'.val = true\}$ |
| $V' \to$ `false` | $\{V'.val = false\}$ |
| $E \to E$ `<` $E$ | $\{E_0.exp = LT(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `<=` $E$ | $\{E_0.exp = LTE(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `>` $E$ | $\{E_0.exp = GT(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `>=` $E$ | $\{E_0.exp = GTE(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `=` $E$ | $\{E_0.exp = EQ(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `!=` $E$ | $\{E_0.exp = NEQ(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `and` $E$ | $\{E_0.exp = AND(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `or` $E$ | $\{E_0.exp = OR(E_1.exp, E_2.exp)\}$ |
| $E \to$ `not`$E$ | $\{E_0.exp = NOT(E_1.exp)\}$ |
| $E \to E$ `+` $E$ | $\{E_0.exp = SUM(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `-` $E$ | $\{E_0.exp = MIN(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `*` $E$ | $\{E_0.exp = TIM(E_1.exp, E_2.exp)\}$ |
| $E \to E$ `/` $E$ | $\{E_0.exp = DIV(E_1.exp, E_2.exp)\}$ |
| $E \to$ `-`$E$ | $\{E_0.exp = NEG(E_1.exp)\}$ |
| $E \to TkIdent$ | $\{E.exp = TkIdent.val\}$ |
| $E \to V'$ | $\{E.exp = V'.val\}$ |

[0]Esta guía ha sido realizada, revisada y modificada por: Hancel González y José A. Goncalves