# Trabajo práctico 2: Especificación

# Base De Datos

## Normativa

**Límite de entrega**: Miércoles 25 de mayo hasta las 22:00 hs. Enviar PDF a algo2.dc+TP2@gmail.com con asunto "Grupo N" siendo N el número de grupo asignado.

Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.

(http://www.dc.uba.ar/materias/aed2/2016/1c/cursada)

Versión: 1.1 del 11 de mayo de 2016 (ver TP2 Changelog.txt)

# Especificación

# 1. TAD DATO

TAD TIPO es BOOL

TAD DATO

géneros dato

usa string, nat, tipo

exporta generadores, observadores básicos y otras operaciones

igualdad observacional

$$(\forall d_1, d_2 : \text{dato}) \left( d_1 =_{\text{obs}} d_2 \iff \begin{pmatrix} \text{tipo?}(d_1) \equiv \text{tipo?}(d_2) \land_{\text{L}} \\ ((\text{Nat?}(d_1) \Rightarrow_{\text{L}} \text{valorNat}(d_1) =_{\text{obs}} \text{valorNat}(d_2)) \land \\ ((\text{String?}(d_1) \Rightarrow_{\text{L}} \text{valorStr}(d_1) =_{\text{obs}} \text{valorStr}(d_2))) \end{pmatrix} \right)$$

generadores

 $\begin{array}{cccc} {\rm datoString} & : & {\rm string} & \longrightarrow & {\rm dato} \\ {\rm datoNat} & : & {\rm nat} & \longrightarrow & {\rm dato} \end{array}$ 

observadores básicos

tipo? : dato  $\longrightarrow$  tipo valorNat : dato d  $\longrightarrow$  nat  $\{Nat?(d)\}$ valorStr : dato d  $\longrightarrow$  string  $\{String?(d)\}$ 

otras operaciones

mismoTipo?: dato  $\times$  dato $\longrightarrow$  boolString?: dato $\longrightarrow$  boolNat?: dato $\longrightarrow$  boolmin: conj(dato) cd $\longrightarrow$  dato

 $1 \leq 0$  . dato  $a_1 \wedge dato a_2 \wedge a_1 \wedge a_2 \wedge a_2 \wedge a_1 \wedge a_2 \wedge a_$ 

axiomas

 $(\forall s: \text{string}, \forall n: \text{nat}, \forall d, d_1, d_2: \text{dato}, \text{paratodoconj}(\text{dato})\text{cd})$ 

 $\begin{array}{lll} tipo?(datoNat(n)) & \equiv true \\ tipo?(datoString(s)) & \equiv false \\ valorNat(datoNat(n)) & \equiv n \\ valorStr(datoString(s)) & \equiv s \end{array}$ 

mismoTipo? $(d_1, d_2)$   $\equiv (Nat?(d_1) \equiv Nat?(d_2))$ 

```
\begin{array}{lll} \min(cd) & \equiv & \mathbf{if} & \#(\mathrm{cd}) = 1 \ \lor_{\mathrm{L}} & \mathrm{dameUno}(cd) \le & \min(\sin \mathrm{Uno}(cd)) & \mathbf{then} & \mathrm{dameUno}(cd) & \mathbf{else} \\ & \min(\sin \mathrm{Uno}(cd)) & \mathbf{fi} & \\ \max(cd) & \equiv & \mathbf{if} & \#(\mathrm{cd}) = 1 \ \lor_{\mathrm{L}} & \mathrm{dameUno}(cd) \le & \max(\sin \mathrm{Uno}(cd)) & \mathbf{then} & \max(\sin \mathrm{Uno}(cd)) & \mathbf{else} \\ & & \mathrm{dameUno}(cd) & \mathbf{fi} & \\ \le (d_1, d_2) & \equiv & \mathbf{if} & \mathrm{String?}(d_1) & \mathbf{then} & \mathrm{valorStr}(d_1) \le & \mathrm{valorStr}(d_2) & \mathbf{else} & \mathrm{valorNat}(d_1) \le & \mathrm{valorNat}(d_2) & \mathbf{fi} & \\ \end{array}
```

#### Fin TAD

# 2. TAD REGISTRO

TAD CAMPO es STRING

```
TAD REGISTRO
```

```
TAD REGISTRO extiende a DICCIONARIO (CAMPO, DATO)
```

géneros registrousa string, dato, campo, dicc

exporta otras operaciones

#### otras operaciones

```
campos
                         : registro
                                                                                                 \rightarrow conj(campo)
borrar?
                         : registro criterio \times registro r
                                                                                                 \longrightarrow bool
                                                                                                                        \{\#\text{campos}(\text{criterio}) \equiv 1\}
agregar
Campos : registro r_1 \times registro r_2
                                                                                                \longrightarrow registro
copiar
Campos : conj(campo) cc \times registro r_1 \times registro r_2 \longrightarrow registro
                                                                                                                                  \{cc \in \operatorname{campos}(r_2)\}\
coincideAlguno : registro r_1 \times \text{conj}(\text{campo}) cc \times \text{registro} r_2 \longrightarrow \text{bool}
                                                                                                            \{cc \subseteq campos(r_1) \cap campos(r_2)\}\
coincidenTodos : registro r_1 \times \text{conj}(\text{campo}) cc \times \text{registro} r_2 \longrightarrow \text{bool}
                                                                                                            \{cc \subseteq campos(r_1) \cap campos(r_2)\}\
                         : campo c \times \text{conj}(\text{registro}) cr

ightarrow bool
combinarTodos : campo c \times \text{registro } r_1 \times \text{conj(registro)} cr \longrightarrow \text{conj(registro)}
                                                                                                          \{c \in \operatorname{campos}(r_1) \land \operatorname{enTodos}(c, cr)\}\
```

#### axiomas

```
(\forall c: \text{campo}, \forall v, d_1, d_2: \text{dato}, \forall r, r_1, r_2, crit: \text{registro}, \forall cc: \text{conj}(\text{campo}), \forall cr: \text{conj}(\text{registro}))
                                       \equiv claves(r)
  campos(r)
  borrar?(crit, r)
                                       \equiv coincidenTodos(crit, campos(crit), r)
                                       \equiv \operatorname{copiarCampos}(\operatorname{campos}(r_2) - \operatorname{campos}(r_1), r_1, r_2)
  agregarCampos(r_1, r_2)
  copiarCampos(cc, r_1, r_2)
                                       \equiv if \emptyset?(cc) then r_1 else copiarCampos(sinUno(cc), definir(dameUno(cc),
                                           obtener(dameUno(cc), r_2), r_1), r_2) fi
  coincideAlguno(r_1, cc, r_2)
                                       \equiv \neg \emptyset?(cc) \land_{L} (obtener(dameUno(cc), r_1) = obtener(dameUno(cc), r_2)) \lor
                                           coincideAlguno(r_1, sinUno(cc), r_2))
                                      \equiv \emptyset?(cc) \vee_{L}(obtener(dameUno(cc), r_1) = obtener(dameUno(cc), r_2) \wedge
  coincidenTodos(r_1, cc, r_2)
                                           coincidenTodos(r_1, sinUno(cc), r_2))
  enTodos(c,cr)
                                       \equiv \emptyset? (cr) \vee_{\mathsf{L}} (c \in \mathsf{campos}(\mathsf{dameUno}(cr)) \wedge \mathsf{enTodos}(c, \mathsf{sinUno}(cr)))
                                       \equiv if \emptyset? (cr) then
  combinarTodos(c, r, cr)
                                               Ø
                                           else
                                               combinarTodos(c, r, sinUno(cr)) \cup (if obtener(c, dameUno(cr)) =
                                               obtener(c, r) then {agregarCampos(r, dameUno(cr))} else \emptyset fi)
                                           fi
```

### Fin TAD

#### 3. TAD TABLA

```
TAD TABLA
```

tabla géneros

usa campo, clave, registro, conj, string

exporta Generadores, observadores, otras operaciones

### igualdad observacional

$$(\forall t_1, t_2 : \text{tabla}) \left( t_1 =_{\text{obs}} t_2 \iff \begin{pmatrix} \text{nombre}(t_1) = \text{nombre}(t_2) \land \text{campos}(t_1) =_{\text{obs}} \\ \text{campos}(t_2) \land_{\text{L}} (\forall c : \text{campo}) (c \in \text{campos}(t_1) \Rightarrow_{\text{L}} \\ \text{tipoCampo}(c, t_1) =_{\text{obs}} \text{tipoCampo}(c, t_2)) \land \text{claves}(t_1) \\ =_{\text{obs}} \text{claves}(t_2) \land \text{registros}(t_1) =_{\text{obs}} \text{registros}(t_2) \land \\ \text{indices}(t_1) =_{\text{obs}} \text{indices}(t_2) \land \text{cantidadDeAccesos}(t_1) \\ = \text{cantidadDeAccesos}(t_2) \end{pmatrix} \right)$$

#### observadores básicos

nombre : tabla  $\rightarrow$  string claves : tabla  $\rightarrow$  conj(campo) indices : tabla  $\rightarrow \text{conj}(\text{campo})$ : tabla  $\rightarrow$  conj(campo) campos

tipoCampo : campo  $c \times \text{tabla } t$  $\longrightarrow$  tipo  $\{c \in \operatorname{campos}(t)\}\$ registros : tabla  $\rightarrow$  conj(registro)

cantidadDeAccesos: tabla  $\rightarrow$  nat

### generadores

nuevaTabla : string  $nombre \times \text{conj}(\text{campo}) \ claves \times \text{registro} \ columnas \longrightarrow \text{tabla}$ 

 $\{\text{claves} \neq \emptyset \land \text{claves} \subseteq \text{campos}(\text{columnas})\}$ 

agregar Registro : registro  $r \times$  tabla t $\rightarrow$  tabla

 $\{\operatorname{campos}(r) =_{\operatorname{obs}} \operatorname{campos}(t) \land \operatorname{puedoInsertar}(r, t)\}$ 

borrar Registro : registro  $crit \times tabla t \longrightarrow tabla$ 

 $\{\# \operatorname{campos}(crit) = 1 \land_{\mathsf{L}} \operatorname{dameUno}(\operatorname{campos}(crit)) \in \operatorname{claves}(t)\}$ 

{puedeIndexar(c,t)} indexar : campo  $c \times \text{tabla } t$  $\longrightarrow$  tabla

### otras operaciones

puedoInsertar? : registro r  $\times$  tabla t compatible : registro  $\times$  tabla → bool

minimo : campo  $c \times \text{tabla } t$  $\longrightarrow dato$  $\{\neg\emptyset?(\operatorname{registros}(t)) \land c \in \operatorname{indices}(t)\}$ : campo  $c \times$  tabla t $\{\neg\emptyset?(\operatorname{registros}(t)) \land c \in \operatorname{indices}(t)\}$ maximo  $\longrightarrow$  dato

puedeIndexar : campo  $c \times \text{tabla } t$  $\rightarrow$  bool

hayCoincidencia : registro  $\times$  conj(campo)  $\times$  conj(registro) ightarrow bool

 $\longrightarrow$  conj(registro) coincidencias : registro  $\times$  conj(registro) combinarRegistros : campo  $c \times \text{conj}(\text{registro}) cr1 \times \text{conj}(\text{registro}) cr2)$  $\longrightarrow$  conj(registro) dameColumna : campo  $c \times \text{conj(registro)} cr$  $\longrightarrow \text{conj}(\text{dato})$ mismosTipos: registro  $r \times \text{tabla } t$  $\rightarrow$  bool  $\{\operatorname{campos}(r) \subseteq \operatorname{campos}(t)\}$ 

#### axiomas

 $(\forall cc, cp: \text{conj}(\text{campo}), \forall t: \text{tabla}, \forall r: \text{registro}, \forall cr: \text{conj}(\text{registro}) \forall n: \text{string})$ 

nombre(nuevaTabla(n, cc, r))

nombre(agregarRegistro(r, t)) $\equiv$  nombre(t) nombre(borrarRegistro(crit, t)) $\equiv$  nombre(t) nombre(indexar(c, t)) $\equiv \text{nombre}(t)$ 

claves(nuevaTabla(n, cc, r))  $\equiv cc$ claves(agregarRegistro(r, t)) $\equiv \text{claves}(t)$ claves(borrarRegistro(crit, t)) $\equiv \text{claves}(t)$ 

```
claves(indexar(c, t))
                                                      \equiv claves(t)
indices(nuevaTabla(n, cc, r))
                                                         Ø
indices(agregarRegistro(r, t))
                                                      \equiv indices(t)
indices(borrarRegistro(crit, t))
                                                      \equiv indices(t)
indices(indexar(c, t))
                                                          Ag(c, indices(t))
campos(nuevaTabla(n, cc, r))
                                                      \equiv \operatorname{campos}(r)
campos(agregarRegistro(r, t))
                                                      \equiv \text{campos}(t)
campos(borrarRegistro(crit, t))
                                                          campos(t)
campos(indexar(c, t))
                                                          campos(t)
tipoCampo(c, nuevaTabla(n, cc, r))
                                                      \equiv tipo?(obtener(c,r))
tipoCampo(c, agregarRegistro(r, t))
                                                      \equiv \operatorname{tipoCampo}(c,t)
                                                      \equiv \operatorname{tipoCampo}(c,t)
tipoCampo(c, borrarRegistro(crit, t))
tipoCampo(c, indexar(c, t))
                                                      \equiv \operatorname{tipoCampo}(c,t)
                                                      \equiv \emptyset
registros(nuevaTabla(n, cc, r))
registros(agregarRegistro(r, t))
                                                      \equiv \operatorname{Ag}(r, \operatorname{registros}(t))
registros(borrarRegistro(crit, t))
                                                      \equiv \operatorname{registros}(t) - \operatorname{coincidencias}(crit, \operatorname{registros}(t))
registros(indexar(c, t))
                                                      \equiv \operatorname{registros}(t)
                                                            \equiv 0
cantidadDeAccesos(nuevaTabla(n, cc, r))
cantidadDeAccesos(agregarRegistro(r, t)
                                                            \equiv 1 + \text{cantidadDeAccesos}(t)
cantidadDeAccesos(borrarRegistro(crit, t))
                                                           \equiv \# \operatorname{coincidencias}(crit, \operatorname{registros}(t)) + \operatorname{cantidadDeAccesos}(t)
cantidadDeAccesos(indexar(c, t))
                                                            \equiv cantidadDeAccesos(t)
puedoInsertar?(r, t)
                                           \equiv compatible(r,t) \land \neg hayCoincidencia(r, claves(t), registros(t))
compatible(r, t)
                                           \equiv \operatorname{campos}(r) = \operatorname{campos}(t) \wedge_{\scriptscriptstyle L} \operatorname{mismosTipos}(r,t)
puedeIndexar(c,t)
                                           \equiv c \in \operatorname{campos}(t) \wedge_{\operatorname{L}} c \notin \operatorname{indices}(t) \wedge (\#\operatorname{indices}(t) \leq 1 \wedge (\#\operatorname{indices}(t) = 1))
                                                \Rightarrow_{\text{L}} \neg \text{tipoCampo}(c, t) \equiv \text{tipoCampo}(\text{dameUno}(\text{indices}(t)), t)))
                                           \equiv if \emptyset?(cr_1) then \emptyset else combinarTodos(c, dameUno(cr_1), cr_2) \cup
combinarRegistros(c, cr_1, cr_2)
                                                combinarRegistros(c, \sin U \operatorname{no}(cr_1), t_2) fi
hayCoincidencia(r, cc, cr)
                                           \equiv \neg \emptyset?(cr) \land_{\mathsf{L}} (\mathsf{coincideAlguno}(\mathsf{r}, \mathsf{cc}, \mathsf{dameUno}(cr)) \lor \mathsf{hayCoinciden}
                                                cia?(r, cc, sinUno(cr)))
coincidencias(crit, cr)
                                           \equiv if \emptyset?(cr) then
                                                    Ø
                                                else
                                                    if coinciden Todos(crit, campos(crit), dame Uno(cr)) then
                                                         Ag(dameUno(cr), coincidencias(crit, sinUno(cr)))
                                                    else
                                                         coincidencias(crit, sinUno(cr))
                                                fi
\min (c,t)
                                           \equiv \min(\text{dameColumna}(c, \text{registros}(t)))
\operatorname{maximo}(c,t)
                                           \equiv \max(\text{dameColumna}(c, \text{registros}(t)))
dameColumna(c,cr)
                                           \equiv if \emptyset? (cr) then \emptyset else (if c \in \text{campos}(\text{dameUno}(cr)) then
                                                \{\text{obtener}(c, \text{dameUno}(cr))\}\ else \emptyset fi) \cup \text{dameColumna}(c, \sin \text{Uno}(cr))
                                                fi
mismosTipos(r, t)
                                                                           (mismosTipos(borrar(dameUno(campos(r)),r),t)
                                               \emptyset?(r)
                                                                   tipo? (obtener (dame Uno (campos (r_1)), r_1)
                                                tipoCampo(dameUno(campos(r_1)),t)))
```

#### Fin TAD

#### 4. TAD BASEDEDATOS

hayJoin? $(t_1, t_2, \text{nuevaDB})$ 

```
TAD BASEDEDATOS
```

```
géneros
usa
                      NAT, STRING, TABLA, REGISTRO, CAMPO, DATO
                      generadores, observadores básicos y otras operaciones
exporta
igualdad observacional
                     (\forall b_1, b_2 : \text{base}) \quad \left(b_1 =_{\text{obs}} b_2 \iff \begin{pmatrix} \text{tablas}(b_1) =_{\text{obs}} \text{ tablas}(b_2) \land_{\text{L}} (\forall t : \text{ string}) \ t \in \text{tablas}(b_1) \Rightarrow_{\text{L}} \text{ dameTabla}(t, b_1) =_{\text{obs}} \text{ dameTabla}(t, b_2) \land (\forall t_1, t_2, c : \text{ string}) \ \{t_1, t_2\} \subseteq \text{ tablas}(b_1) \Rightarrow_{\text{L}} \text{ hayJoin?}(t_1, t_2, c, b_1) =_{\text{obs}} \text{ hayJoin?}(t_1, t_2, c, b_2) \end{pmatrix}
observadores básicos
   tablas
                       : base db
                                                                                  \rightarrow conj(string)
   dameTabla : string t \times base db
                                                                                  \rightarrow tabla
                                                                                                                                                 \{t \in \text{tablas}(db)\}\
                     : string t_1 \times \text{string } t_2 \times \text{base } db \longrightarrow \text{bool}
   campoJoin : string t_1 \times \text{string } t_2 \times \text{base } db \longrightarrow \text{campo}
                                                                                                                                        \{\text{hayJoin}?(t_1, t_2, db)\}
generadores
   nuevaDB
                                                                                                               \rightarrow base
                                : tabla ta \times base db
                                                                                                               → base
                                                                                                                                             \{\emptyset?(\text{registros}(\text{ta}))\}
   agregarTabla
   insertar
Entrada : registro reg \times string \ t \times base \ db
                                                                                                               \rightarrow base
                                                                                                       \{t \in \text{tablas}(db) \land_{\text{\tiny L}} \text{puedoInsertar}?(reg, t)\}
                                                                                                               \rightarrow base
   borrar
                                : registro cr \times string t \times base db
                                                                                                                  \{\#\operatorname{campos}(cr)=1 \land t \in \operatorname{tablas}(db)\}\
   generarVistaJoin : string t_1 \times \text{string } t_2 \times \text{campo } c \times \text{base } db \longrightarrow \text{base}
                          \int t_1 \neq t_2 \land \{t_1, t_2\} \subseteq \operatorname{tablas}(db) \land_{\scriptscriptstyle L}
                          c \in \text{claves}(\text{dameTabla}(t_1, db)) \land c \in \text{claves}(\text{dameTabla}(t_2, db)) \land \neg \text{hayJoin}(t_1, t_2, db)
   borrarJoin
                                : string t_1 \times \text{string } t_2 \times \text{base } db
                                                                                                              \rightarrow base
                                                                                                                                         \{\text{hayJoin}?(t_1,t_2,db)\}
otras operaciones
                                                                                                                                                \{t \in \operatorname{tablas}(db)\}\
   registros
                                    : string t \times \text{base } db
                                                                                                       \rightarrow \text{conj}(\text{registro})
                                                                                                                                        \{\text{hayJoin}?(t_1, t_2, db)\}
   vistaJoin
                                    : string t_1 \times \text{string } t_2 \times \text{base } db
                                                                                                       \rightarrow conj(registro)
   cantidadDeAccesos : string t \times base db
                                                                                                     \rightarrow nat
                                                                                                                                                \{t \in \operatorname{tablas}(db)\}\
                                                                                                                                                 \{ \operatorname{tablas}(db) \neq \emptyset \}
   tablaMaxima
                                    : base db
                                                                                                      \rightarrow string
                                                                                                                                     \{\{t\} \cup ct \subseteq \operatorname{tablas}(db)\}\
   encontrarMaximo : string t \times \text{conj}(\text{string}) ct \times \text{base } db \longrightarrow \text{string}
                                    : registro criterio \times string t \times base db \longrightarrow conj(registro)
                                                                                                                                                 \{t \in \operatorname{tablas}(db)\}\
   buscar
axiomas
(\forall cp: \text{conj}(\text{campo}), \forall t, t_1, t_2, t_3, t_4: \text{string}, \forall tbl: \text{tabla}, \forall reg: \text{registro}, \forall nom: \text{string}, \forall cr_1, cr_2: \text{conj}(\text{registro}))
   tablas(nuevaDB)
   tablas(agregarTabla(tbl, db))
                                                                   \equiv Ag(nombre(tbl), tablas(db))
   tablas(insertarEntrada(reg, t, db))
                                                                   \equiv \text{tablas}(db)
   tablas(borrar(cr, t, db))
                                                                   \equiv \text{tablas}(db)
   tablas(generarVistaJoin(t_1, t_2, c, db)) \equiv tablas(db)
   tablas(borrarJoin(t_1, t_2, db))
                                                                   \equiv \text{tablas}(db)
   dameTabla(t, agregarTabla(tbl, db))
                                                                               \equiv if nombre(tbl) \equiv t then tbl else dameTabla(t,db)
   dameTabla(t_1, insertarEntrada(reg, t_2, db))
                                                                              \equiv if t_1 = t_2 then agregarRegistro(reg, dameTabla(t_1,
                                                                                    db)) else dameTabla(t_1, db) fi
   dameTabla(t_1, borrar(cr, t_2, db))
                                                                              \equiv if t_1 = t_2 then borrarRegistro(cr, dameTabla(t_1,
                                                                                    db)) else dameTabla(t_1, db) fi
   dameTabla(t, generarVistaJoin(t_1, t_2, c, db))
                                                                              \equiv \text{dameTabla}(t, db)
   dameTabla(t, borrarJoin(t_1, t_2, db))
                                                                               \equiv \text{dameTabla}(t, db)
```

 $\equiv$  false

```
hayJoin?(t_1, t_2, agregarTabla(tbl, db))
                                                                 \equiv \text{ hayJoin?}(t_1, t_2, db)
hayJoin?(t_1, t_2, insertarEntrada(reg, t_3, d_b))
                                                                 \equiv \text{ hayJoin?}(t_1, t_2, db)
hayJoin?(t_1, t_2, borrar(cr, c, db))
                                                                 \equiv \text{ hayJoin?}(t_1, t_2, db)
hayJoin?(t_1, t_2, \text{ generarVistaJoin}(t_3, t_4, c_2, db)) \equiv (t_1 = t_3 \land t_2 = t_4) \lor \text{ hayJoin?}(t_1, t_2, db)
hay
Join?(t_1, t_2, borrarJoin(t_3, t_4, c_2, db))
                                                                \equiv \neg(t_1 = t_3 \land t_2 = t_4) \land \text{hayJoin?}(t_1, t_2, db)
campoJoin(t_1, t_2, agregarTabla(tbl, db))
                                                                  \equiv \text{campoJoin}(t_1, t_2, db)
campoJoin(t_1, t_2, insertarEntrada(reg, t, db))
                                                                 \equiv \text{campoJoin}(t_1, t_2, db)
campoJoin(t_1, t_2, borrar(cr, t, db))
                                                                  \equiv \text{campoJoin}(t_1, t_2, db)
campoJoin(t_1, t_2, \text{generarVistaJoin}(t_3, t_4, c, db)) \equiv \text{if } t_1 = t_3 \land t_2 = t_4 \text{ then}
                                                                      else
                                                                          campoJoin(t_1, t_2, db)
                                                                      fi
campoJoin(t_1, t_2, borrarJoin(t_3, t_4, db))
                                                                  \equiv \text{campoJoin}(t_1, t_2, db)
registros(t, db)
                                          \equiv \text{registros}(\text{dameTabla}(t, db))
\operatorname{cantidadDeAccesos}(t, db)
                                         \equiv cantidadDeAccesos(dameTabla(t, db))
tablaMaxima(db)
                                          \equiv dameTabla(encontrarMaximo(dameUno(tablas(db))),
                                              \sin U_{\text{no}}(\text{tablas}(db)), db), db)
                                          \equiv if \emptyset?(ct) then
encontrarMaximo(t, ct, db)
                                              else
                                                  if cantidadDeAccesos(t, db) \ge \text{cantidadDeAccesos}(\text{dameUno}(ct), db)
                                                      encontrarMaximo(t, sinUno(ct), db)
                                                  else
                                                      \operatorname{encontrarMaximo}(\operatorname{dameUno}(ct), \sin \operatorname{Uno}(ct), db)
                                              fi
                                         \equiv combinarRegistros( campoJoin(t_1, t_2, db), registros(dameTabla(t_1, t_2, db))
vistaJoin(t_1, t_2, db)
                                              db)), registros( dameTabla(t_2, db) )
buscar(r,t,db)
                                         \equiv coincidencias(r, registros(dameTabla(t, db)))
```

Fin TAD