

## 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:** 1.0 del 6 de mayo de 2016 (ver TP2\_Changelog.txt)

### Especificación

#### 1. TAD DATO

##### TAD DATO

**géneros**      dato

**usa**            string, nat

**exporta**      generadores, observadores básicos y otras operaciones

**igualdad observacional**

$$(\forall d_1, d_2 : \text{dato}) \left( d_1 =_{\text{obs}} d_2 \iff \left( \text{mismoTipo?}(d_1, d_2) \wedge_L \left( (Nat?(d_1) \Rightarrow_L \text{valorNat}(d_1) =_{\text{obs}} \text{valorNat}(d_2)) \wedge (String?(d_1) \Rightarrow_L \text{valorStr}(d_1) =_{\text{obs}} \text{valorStr}(d_2)) \right) \right) \right)$$

**generadores**

datoString : string  $\longrightarrow$  dato

datoNat : nat  $\longrightarrow$  dato

**observadores básicos**

Nat? : dato  $\longrightarrow$  bool

valorNat : dato *d*  $\longrightarrow$  nat {Nat?(d)}

valorStr : dato *d*  $\longrightarrow$  string {String?(d)}

**otras operaciones**

mismoTipo? : dato  $\times$  dato  $\longrightarrow$  bool

String? : dato  $\times$  dato  $\longrightarrow$  bool

min : conj(dato) *cd*  $\longrightarrow$  dato {-∅?(cd)}

max : conj(dato) *cd*  $\longrightarrow$  dato {-∅?(cd)}

$\bullet \leq \bullet$  : dato *d*<sub>1</sub>  $\times$  dato *d*<sub>2</sub>  $\longrightarrow$  bool {mismoTipo?(d<sub>1</sub>, d<sub>2</sub>)}

**axiomas**

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

Nat?(datoNat(*n*))  $\equiv$  true

Nat?(datoString(*s*))  $\equiv$  false

valorNat(datoNat(*n*))  $\equiv$  *n*

valorStr(datoString(*s*))  $\equiv$  *s*

mismoTipo?(*d*<sub>1</sub>, *d*<sub>2</sub>)  $\equiv$  (Nat?(*d*<sub>1</sub>)  $\equiv$  Nat?(*d*<sub>2</sub>))

String?(*d*)  $\equiv$   $\neg$ Nat?(*d*)

min(*cd*)  $\equiv$  **if** #(*cd*) = 1  $\vee_L$  dameUno(*cd*)  $\leq$  min(sinUno(*cd*)) **then** dameUno(*cd*) **else** min(sinUno(*cd*)) **fi**

max(*cd*)  $\equiv$  **if** #(*cd*) = 1  $\vee_L$  dameUno(*cd*)  $\leq$  max(sinUno(*cd*)) **then** max(sinUno(*cd*)) **else** dameUno(*cd*) **fi**

$\leq(d_1, d_2) \equiv \text{if String?}(d_1) \text{ then valorStr}(d_1) \leq \text{valorStr}(d_2) \text{ else valorNat}(d_1) \leq \text{valorNat}(d_2) \text{ fi}$

**Fin TAD**

## 2. TAD REGISTRO

TAD CAMPO es STRING

**TAD REGISTRO**

TAD REGISTRO extiende a DICCIONARIO(CAMPO, DATO)

**géneros** registro

**usa** string, dato, campo, dicc

**exporta** otras operaciones

**otras operaciones**

campos	: registro	$\longrightarrow$ conj(campo)
borrar?	: registro <i>criterio</i> $\times$ registro <i>reg</i>	$\longrightarrow$ bool {#campos(criterio) $\equiv$ 1}
agregarCampos	: registro $\times$ registro	$\longrightarrow$ registro
copiarCampos	: conj(campo) $\times$ registro $\times$ registro	$\longrightarrow$ registro
coincideAlguno	: registro <i>r1</i> $\times$ conj(campo) <i>cc</i> $\times$ registro <i>r2</i>	$\longrightarrow$ bool { $cc \subseteq \text{campos}(r1) \cap \text{campos}(r2)$ }
coincidenTodos	: registro <i>r1</i> $\times$ conj(campo) <i>cc</i> $\times$ registro <i>r2</i>	$\longrightarrow$ bool { $cc \subseteq \text{campos}(r1) \cap \text{campos}(r2)$ }
mismosTipos	: registro <i>r1</i> $\times$ registro <i>r2</i>	$\longrightarrow$ bool { $\text{campos}(r1) \subseteq \text{campos}(r2)$ }
combinar	: campo <i>c</i> $\times$ registro <i>r1</i> $\times$ registro <i>r2</i>	$\longrightarrow$ registro { $c \in \text{campos}(r1) \cap \text{campos}(r2)$ }

**axiomas**

$(\forall c: \text{campo}, \forall v, d_1, d_2: \text{dato}, \forall r, \text{crit}: \text{registro}, \forall cc: \text{conj}(\text{campo}))$

campos(*r*)  $\equiv$  claves(*r*)

borrar?(*crit*, *r*)  $\equiv$  coincidenTodos(*crit*, campos(*crit*), *r*)

agregarCampos(*r1*, *r2*)  $\equiv$  copiarCampos(campos(*r2*) – campos(*r1*), *r1*, *r2*)

copiarCampos(*cc*, *r1*, *r2*)  $\equiv$  if  $\emptyset?(cc)$  then *r1* else copiarCampos(sinUno(*cc*), definir(dameUno(*cc*), obtener(dameUno(campos), *r2*), *r1*)) fi

coincideAlguno(*r1*, *cc*, *r2*)  $\equiv$   $\neg \emptyset?(cc) \wedge_L (\text{obtener}(\text{dameUno}(cc), r1) = \text{obtener}(\text{dameUno}(cc), r2)) \vee \text{coincideAlguno}(r1, \text{sinUno}(cc), r2)$

coincidenTodos(*r1*, *cc*, *r2*)  $\equiv$   $\emptyset?(cc) \vee_L (\text{obtener}(\text{dameUno}(cc), r1) = \text{obtener}(\text{dameUno}(cc), r2)) \wedge \text{coincideAlguno}(r1, \text{sinUno}(cc), r2)$

mismosTipos(*r1*, *r2*)  $\equiv$   $\emptyset?(r1) \vee_L (\text{mismosTipos}(\text{borrar}(\text{dameUno}(\text{campos}(r1)), r1), r2)) \wedge \text{mismoTipo}(\text{obtener}(\text{dameUno}(r1), r1), \text{obtener}(\text{obtener}(\text{dameUno}(r1), r2))))$

combinar(*c*, *reg*, *cr*)  $\equiv$  if  $\emptyset?(cr)$  then  
     *reg*  
   else  
     if obtener(*c*, dameUno(*cr*)) = obtener(*c*, *reg*) then  
       agregarCampos(*reg*, dameUno(*cr*))  
     else  
       combinar(*c*, *reg*, sinUno(*cr*))  
     fi  
 fi

**Fin TAD**

## TAD TABLA

```

indices(borrarRegistro(crit, t))    ≡ indices(t)
indices(indexar(c, t))              ≡ Ag(c, indices(t))

columnas(nuevaTabla(n, cc, cp))    ≡ cp
columnas(agregarRegistro(r, t))    ≡ columnas(t)
columnas(borrarRegistro(crit, t))  ≡ columnas(t)
columnas(indexar(c, t))            ≡ columnas(t)

registros(nuevaTabla(n, cc, cp))  ≡ ∅
registros(agregarRegistro(r, t))    ≡ Ag(r, registros(t))
registros(borrarRegistro(crit, t))  ≡ registros(t) − coincidencias(crit, registros(t))
registros(indexar(c, t))            ≡ registros(t)

cantidadDeAccesos(nuevaTabla(n, cc, cp)) ≡ 0
cantidadDeAccesos(agregarRegistro(r, t)) ≡ 1 + cantidadDeAccesos(t)
cantidadDeAccesos(borrarRegistro(crit, t)) ≡ #coincidencias(crit, registros(t)) + cantidadDeAccesos(t)
cantidadDeAccesos(indexar(c, t))        ≡ cantidadDeAccesos(t)

puedoInsertar?(r, t)                ≡ compatible(r, t) ∧ ¬ hayCoincidencia(r, claves(t), registros(t))
compatible(r, t)                    ≡ campos(r) = campos(columnas(t)) ∧L mismosTipos(r, columnas(t))
puedeIndexar(c, t)                  ≡ c ∈ campos(columnas(t)) ∧L c ∉ indices(t) ∧ (#indices(t) ≤ 1 ∧ (#indices(t) = 1 ⇒L ¬ mismoTipo(obtener(c, columnas(t)), obtener(dameUno(indices(t)), columnas(t))))

combinarRegistros(c, cr1, cr2)    ≡ if ∅?(cr1) then ∅ else Ag(combinar(c, dameUno(cr1), cr2)),
                                     combinarRegistros(c, cr1, t2) fi
hayCoincidencia(r, cc, cr)        ≡ ¬∅?(cr) ∧L ( coincideAlguno(r, cc, dameUno(cr)) ∨ hayCoincidencia?(r, cc, sinUno(cr)) )
coincidencias(crit, cr)            ≡ if ∅?(cr) then
                                     ∅
                                     else
                                     if coincidenTodos(crit, crit, dameUno(cr)) then
                                     Ag(dameUno(cr), coincidencias(crit, sinUno(cr)))
                                     else
                                     coincidencias(crit, sinUno(cr))
                                     fi
                                     fi
minimo(c, t)                      ≡ min(dameColumna(c, registros(t)))
maximo(c, t)                      ≡ max(dameColumna(c, registros(t)))
dameColumna(c, cr)                ≡ if ∅?(cr) then ∅ else (if c ∈ campos(dameUno(cr)) then
                                     {obtener(c, dameUno(cr))} else ∅ fi) ∪ dameColumna(c, sinUno(cr))
                                     fi

```

**Fin TAD**

## 4. TAD BASEDEDATOS

**TAD BASEDEDATOS**

**géneros**      base

**usa**           NAT, STRING, TABLA, REGISTRO, CAMPO, DATO

**exporta**      generadores, observadores básicos y otras operaciones

**igualdad observacional**

$$(\forall b_1, b_2 : \text{base}) \left( b_1 =_{\text{obs}} b_2 \iff \left( \begin{array}{l} \text{tablas}(b_1) =_{\text{obs}} \text{tablas}(b_2) \wedge_L (\forall t: \text{string}) \ t \in \text{tablas}(b_1) \Rightarrow_L \text{dameTabla}(t, b_1) =_{\text{obs}} \text{dameTabla}(t, b_2) \wedge (\forall t_1, t_2, c: \text{string}) \ \{t_1, t_2\} \subseteq \text{tablas}(b_1) \Rightarrow_L \text{hayJoin?}(t_1, t_2, c, b_1) =_{\text{obs}} \text{hayJoin?}(t_1, t_2, c, b_2) \end{array} \right) \right)$$

**observadores básicos**

tablas	: base $db$	$\longrightarrow$ conj(string)	
dameTabla	: string $t \times$ base $db$	$\longrightarrow$ tabla	$\{t \in \text{tablas}(db)\}$
hayJoin?	: string $t_1 \times$ string $t_2 \times$ base $db$	$\longrightarrow$ bool	$\{t_1 \neq t_2 \wedge \{t_1, t_2\} \subseteq \text{tablas}(db)\}$
campoJoin	: string $t_1 \times$ string $t_2 \times$ base $db$	$\longrightarrow$ campo	$\{\text{hayJoin?}(t_1, t_2, db)\}$

**generadores**

nuevaDB	:	$\longrightarrow$ base	
agregarTabla	: tabla $ta \times$ base $db$	$\longrightarrow$ base	$\{\emptyset?(\text{registros}(ta))\}$
insertarEntrada	: registro $reg \times$ string $t \times$ base $db$	$\longrightarrow$ base	$\{t \in \text{tablas}(db) \wedge_L \text{puedoInsertar?}(reg, t)\}$
borrar	: registro $cr \times$ string $t \times$ base $db$	$\longrightarrow$ base	$\{\#campos(cr)=1 \wedge t \in \text{tablas}(db)\}$
generarVistaJoin	: string $t_1 \times$ string $t_2 \times$ campo $c \times$ base $db$	$\longrightarrow$ base	$\left\{ \begin{array}{l} t_1 \neq t_2 \wedge \{t_1, t_2\} \subseteq \text{tablas}(db) \wedge_L \\ c \in \text{claves}(\text{dameTabla}(t_1, db)) \wedge c \in \text{claves}(\text{dameTabla}(t_2, db)) \wedge \neg \text{hayJoin?}(t_1, t_2, db) \end{array} \right\}$

**otras operaciones**

registros	: string $t \times$ base $db$	$\longrightarrow$ conj(registro)	$\{t \in \text{tablas}(db)\}$
vistaJoin	: string $t_1 \times$ string $t_2 \times$ base $db$	$\longrightarrow$ conj(registro)	$\{\text{hayJoin?}(t_1, t_2, db)\}$
cantidadDeAccesos	: string $t \times$ base $db$	$\longrightarrow$ nat	$\{t \in \text{tablas}(db)\}$
tablaMaxima	: base $db$	$\longrightarrow$ string	$\{\text{tablas}(db) \neq \emptyset\}$
encontrarMaximo	: string $t \times$ conj(string) $ct \times$ base $db$	$\longrightarrow$ string	$\{\{t\} \cup ct \subseteq \text{tablas}(db)\}$
buscar	: registro $criterio \times$ string $t \times$ base $db$	$\longrightarrow$ conj(registro)	$\{t \in \text{tablas}(db)\}$

**axiomas**

$(\forall cp: \text{conj}(\text{campo}), \forall t, t_1, t_2: \text{string}, \forall tbl: \text{tabla}, \forall reg: \text{registro}, \forall nom: \text{string}, \forall cr1, cr2: \text{conj}(\text{registro}))$

tablas(nuevaDB)	$\equiv \emptyset$
tablas(agregarTabla(tbl, db))	$\equiv \text{Ag}(\text{nombre}(tbl), \text{tablas}(db))$
tablas(insertarEntrada(reg, t, db))	$\equiv \text{tablas}(db)$
tablas(borrar(cr, t, db))	$\equiv \text{tablas}(db)$
tablas(generarVistaJoin(t <sub>1</sub> , t <sub>2</sub> , c, db))	$\equiv \text{tablas}(db)$
dameTabla(t, agregarTabla(tbl, db))	$\equiv \text{if } \text{nombre}(tbl) \equiv t \text{ then } tbl \text{ else dameTabla}(t, db) \text{ fi}$
dameTabla(t <sub>1</sub> , insertarEntrada(reg, t <sub>2</sub> , db))	$\equiv \text{if } t_1 = t_2 \text{ then } \text{agregarRegistro}(\text{reg}, \text{dameTabla}(t_1, db)) \text{ else dameTabla}(t_1, db) \text{ fi}$
dameTabla(t <sub>1</sub> , borrar(cr, t <sub>2</sub> , db))	$\equiv \text{if } t_1 = t_2 \text{ then } \text{borrarRegistro}(cr, \text{dameTabla}(t_1, db)) \text{ else dameTabla}(t_1, db) \text{ fi}$
dameTabla(t, generarVistaJoin(t <sub>1</sub> , t <sub>2</sub> , c, db))	$\equiv \text{dameTabla}(t, db)$
hayJoin?(t <sub>1</sub> , t <sub>2</sub> , nuevaDB)	$\equiv \text{false}$
hayJoin?(t <sub>1</sub> , t <sub>2</sub> , agregarTabla(tbl, db))	$\equiv \text{hayJoin?}(t_1, t_2, db)$
hayJoin?(t <sub>1</sub> , t <sub>2</sub> , insertarEntrada(reg, t <sub>3</sub> , db))	$\equiv \text{hayJoin?}(t_1, t_2, db)$
hayJoin?(t <sub>1</sub> , t <sub>2</sub> , borrar(cr, c, db))	$\equiv \text{hayJoin?}(t_1, t_2, db)$
hayJoin?(t <sub>1</sub> , t <sub>2</sub> , generarVistaJoin(t <sub>3</sub> , t <sub>4</sub> , c <sub>2</sub> , db))	$\equiv (t_1 = t_3 \wedge t_2 = t_4) \vee \text{hayJoin?}(t_1, t_2, db)$
campoJoin(t <sub>1</sub> , t <sub>2</sub> , agregarTabla(tbl, db))	$\equiv \text{campoJoin}(t_1, t_2, db)$
campoJoin(t <sub>1</sub> , t <sub>2</sub> , insertarEntrada(reg, t, db))	$\equiv \text{campoJoin}(t_1, t_2, db)$
campoJoin(t <sub>1</sub> , t <sub>2</sub> , borrar(cr, t, db))	$\equiv \text{campoJoin}(t_1, t_2, db)$
campoJoin(t <sub>1</sub> , t <sub>2</sub> , generarVistaJoin(t <sub>3</sub> , t <sub>4</sub> , c, db))	$\equiv \text{if } t_1 = t_3 \wedge t_2 = t_4 \text{ then } \begin{array}{c} c \\ \text{else} \\ \text{campoJoin}(t_1, t_2, db) \end{array} \text{ fi}$
registros(t, db)	$\equiv \text{registros}(\text{dameTabla}(t, db))$

```

cantidadDeAccesos(t, db)      ≡ cantidadDeAccesos(dameTabla(t, db))

tablaMaxima(db)                ≡ dameTabla(encontrarMaximo(dameUno(tablas(db)),
                                                                sinUno(tablas(db)), db), db)
encontrarMaximo(t, ct, db)   ≡ if  $\emptyset?(ct)$  then
                                t
                                else
                                if cantidadDeAccesos(t, db) ≥ cantidadDeAccesos(dameUno(ct), db)
                                then
                                encontrarMaximo(t, sinUno(ct), db)
                                else
                                encontrarMaximo(dameUno(ct), sinUno(ct), db)
                                fi
                                fi
vistaJoin(t1, t2, db)      ≡ combinarRegistros( campoJoin(t1, t2, db), registros(dameTabla(t1,
                                                                db)), registros( dameTabla(t2, db) ) )
buscar(r, t, db)            ≡ coincidencias(r, registros(dameTabla(t, db)))

```

**Fin TAD**