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 4 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

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

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	$\{\neg\emptyset?(cd)\}$
max	: conj(dato) cd	$\longrightarrow$	dato	$\{\neg\emptyset?(cd)\}$
$\bullet \leq \bullet$	: dato d_1 $\times$ dato d_2	$\longrightarrow$	bool	{mismoTipo?(d_1,d_2)}

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_1, d_2)	$\equiv$	(Nat?(d_1) $\equiv$ Nat?(d_2))
String?(d)	$\equiv$	$\neg$Nat?(d)

min(cd)	$\equiv$	**if** #(cd) = 1 $\vee_{\text{L}}$ dameUno(cd) $\leq$ min(sinUno(cd)) **then** dameUno(cd) **else** min(sinUno(cd)) **fi**
max(cd)	$\equiv$	**if** #(cd) = 1 $\vee_{\text{L}}$ dameUno(cd) $\leq$ max(sinUno(cd)) **then** max(sinUno(cd)) **else** dameUno(cd) **fi**

$\leq(d_1,d_2)$ $\equiv$ **if** String?(d_1) **then** valorStr(d_1) $\leq$ valorStr(d_2) **else** valorNat(d_1) $\leq$ valorNat(d_2) **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 *criterio* $\times$ registro *reg*	$\longrightarrow$	bool {#campos(criterio) $\equiv$ 1}
agregarCampos	:	registro $\times$ registro	$\longrightarrow$	registro
copiarCampos	:	conj(campo) $\times$ registro $\times$ registro	$\longrightarrow$	registro
coincideAlguno	:	registro $r1$ $\times$ conj(campo) cc $\times$ registro $r2$	$\longrightarrow$	bool {cc $\subseteq$ campos(r1) $\cap$ campos(r2)}
coincidenTodos	:	registro $r1$ $\times$ conj(campo) cc $\times$ registro $r2$	$\longrightarrow$	bool {cc $\subseteq$ campos(r1) $\cap$ campos(r2)}
mismosTipos	:	registro $r1$ $\times$ registro $r2$	$\longrightarrow$	bool {campos(r1) $\subseteq$ campos(r2)}
combinar	:	campo c $\times$ registro $r1$ $\times$ registro $r2$	$\longrightarrow$	registro {$c \in$ campos($r1$) $\cap$ campos($r2$)}

axiomas

($\forall c$: campo, $\forall v, d_1, d_2$: dato, $\forall r, crit$: registro, $\forall cc$: conj(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) $\wedge_{\text{L}}$ (obtener(dameUno(cc), r1) = obtener(dameUno(cc), r2)) $\vee$ coincideAlguno(r1, sinUno(cc), r2))
coincidenTodos(r1, cc, r2)	$\equiv$	$\emptyset$?(cc) $\vee_{\text{L}}$(obtener(dameUno(cc), r1) = obtener(dameUno(cc), r2)) $\wedge$ coincideAlguno(r1, sinUno(cc), r2)
mismosTipos(r1, r2)	$\equiv$	$\emptyset$?(r1) $\vee_{\text{L}}$ (mismosTipos(borrar(dameUno(campos(r1)),r1),r2)) $\wedge$ mismoTipo(obtener(dameUno($r1$),$r1$), obtener(obtener(dameUno($r1$),$r2$))))

```
combinar(c, reg, cr) ≡ if ∅?(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

3. TAD Tabla

TAD Tabla

géneros tabla

usa campo, clave, registro, conj, string

exporta Generadores, observadores, otras operaciones

igualdad observacional

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

observadores básicos

nombre	: tabla	$\longrightarrow$	string
claves	: tabla	$\longrightarrow$	conj(campo)
indices	: tabla	$\longrightarrow$	conj(campo)
columnas	: tabla	$\longrightarrow$	registro
registros	: tabla	$\longrightarrow$	conj(registro)
cantidadDeAccesos	: tabla	$\longrightarrow$	nat

generadores

nuevaTabla	: string *nombre* $\times$ conj(campo) *claves* $\times$ registro *columnas*	$\longrightarrow$	tabla $\{\text{claves} \neq \emptyset \land \text{claves} \subseteq \text{campos}(\text{columnas})\}$
agregarRegistro	: registro r $\times$ tabla t	$\longrightarrow$	tabla $\{\text{campos}(r) =_{\text{obs}} \text{campos}(\text{columnas}(t)) \land \text{puedoInsertar?}(r, t)\}$
borrarRegistro	: registro *crit* $\times$ tabla t	$\longrightarrow$	tabla $\{\#\text{campos}(crit) = 1 \land_{\text{L}} \text{dameUno}(\text{campos}(crit)) \in \text{claves}(t)\}$
indexar	: campo c $\times$ tabla t	$\longrightarrow$	tabla $\{\text{puedeIndexar}(c,t)\}$

otras operaciones

puedoInsertar?	: registro r $\times$ tabla t	$\longrightarrow$	bool
compatible	: registro $\times$ tabla	$\longrightarrow$	bool
minimo	: campo c $\times$ tabla t	$\longrightarrow$	dato $\{\neg\emptyset?(\text{registros}(t)) \land c \in \text{indices}(t)\}$
maximo	: campo c $\times$ tabla t	$\longrightarrow$	dato $\{\neg\emptyset?(\text{registros}(t)) \land c \in \text{indices}(t)\}$
puedeIndexar	: campo c $\times$ tabla t	$\longrightarrow$	bool
hayCoincidencia	: registro $\times$ conj(campo) $\times$ conj(registro)	$\longrightarrow$	bool
coincidencias	: registro $\times$ conj(registro)	$\longrightarrow$	conj(registro)
combinarRegistros	: campo c $\times$ conj(registro) $cr1$ $\times$ conj(registro) $cr2$	$\longrightarrow$	conj(registro)
dameColumna	: campo c $\times$ conj(registro) cr	$\longrightarrow$	conj(dato)

axiomas

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

nombre(nuevaTabla(n, cc, cp))	$\equiv$	n
nombre(agregarRegistro(r, t))	$\equiv$	nombre(t)
nombre(borrarRegistro($crit$, t))	$\equiv$	nombre(t)
nombre(indexar(c, t))	$\equiv$	nombre(t)
claves(nuevaTabla(cc, cp))	$\equiv$	cc
claves(agregarRegistro(r, t))	$\equiv$	claves(t)
claves(borrarRegistro($crit$, t))	$\equiv$	claves(t)
claves(indexar(c, t))	$\equiv$	claves(t)
indices(nuevaTabla(n, cc, cp))	$\equiv$	$\emptyset$
indices(agregarRegistro(r, t))	$\equiv$	indices(t)

indices(borrarRegistro($crit$, t)) $\equiv$ indices(t)
indices(indexar(c, t)) $\equiv$ Ag(c,indices(t))

columnas(nuevaTabla(n, cc, cp)) $\equiv$ cp
columnas(agregarRegistro(r, t)) $\equiv$ columnas(t)
columnas(borrarRegistro($crit$, t)) $\equiv$ columnas(t)
columnas(indexar(c, t)) $\equiv$ columnas(t)

registros(nuevaTabla(n, cc, cp)) $\equiv$ $\emptyset$
registros(agregarRegistro(r, t)) $\equiv$ Ag(r, registros(t))
registros(borrarRegistro($crit$, t)) $\equiv$ registros(t) $-$ coincidencias($crit$, registros(t))
registros(indexar(c, t)) $\equiv$ registros(t)

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

puedoInsertar?(r, t) $\equiv$ compatible(r,t) $\wedge \neg$ hayCoincidencia(r, claves(t), registros(t))
compatible(r, t) $\equiv$ campos(r) = campos(columnas(t)) $\wedge_{\text{L}}$ mismosTipos(r,columnas(t))
puedeIndexar(c,t) $\equiv$ $c \in$ campos(columnas(t)) $\wedge_{\text{L}}$ $c \notin$ indices(t) $\wedge$ (#indices(t) $\leq$1 $\wedge$ (#indices(t)=1 $\Rightarrow_{\text{L}} \neg$ mismoTipo(obtener(c, columnas(t)), obtener(dameUno(indices(t)), columnas(t)))))
combinarRegistros(c, cr_1, cr_2) $\equiv$ **if** $\emptyset$?(cr_1) **then** $\emptyset$ **else** Ag(combinar(c, dameUno(cr_1), cr_2)), combinarRegistros(c, cr_1, t_2) **fi**
hayCoincidencia(r, cc, cr) $\equiv$ $\neg\emptyset$?(cr) $\wedge_{\text{L}}$ (coincideAlguno(r, cc, dameUno(cr)) $\vee$ hayCoincidencia?(r, cc, sinUno(cr)))
coincidencias($crit$, cr) $\equiv$ **if** $\emptyset$?(cr) **then**
    $\emptyset$
  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) $\equiv$ min(dameColumna(c,registros(t)))
maximo(c,t) $\equiv$ max(dameColumna(c,registros(t)))
dameColumna(c,cr) $\equiv$ **if** $\emptyset$?(cr) **then** $\emptyset$ **else** (**if** $c \in$ campos(dameUno(cr)) **then** {obtener(c,dameUno(cr))} **else** $\emptyset$ **fi**) $\cup$ 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

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

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

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

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

tablas(nuevaDB)	$\equiv \emptyset$
tablas(agregarTabla(tbl, db))	$\equiv$ Ag(nombre(tbl), tablas(db))
tablas(insertarEntrada(reg, t, db))	$\equiv$ tablas(db)
tablas(borrar(cr, t, db))	$\equiv$ tablas(db)
tablas(generarVistaJoin(t_1, t_2, c, db))	$\equiv$ tablas(db)

dameTabla(t, agregarTabla(tbl, db))	$\equiv$ **if** nombre(tbl) $\equiv t$ **then** tbl **else** dameTabla(t,db) **fi**
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))	$\equiv$ dameTabla(t, db)

hayJoin?(t_1, t_2, nuevaDB)	$\equiv$ false
hayJoin?(t_1, t_2, agregarTabla(tbl, db))	$\equiv$ hayJoin?(t_1, t_2, db)
hayJoin?(t_1, t_2, insertarEntrada(reg, $t3$, db))	$\equiv$ hayJoin?(t_1, t_2, db)
hayJoin?(t_1, t_2, borrar(cr, c, db))	$\equiv$ hayJoin?(t_1, t_2, db)
hayJoin?(t_1, t_2, generarVistaJoin(t_3, t_4, c_2, db))	$\equiv$ $(t_1 = t_3 \wedge t_2 = t_4) \vee$ hayJoin?(t_1, t_2, db)

campoJoin(t_1, t_2, agregarTabla(tbl, db))	$\equiv$ campoJoin(t_1, t_2, db)
campoJoin(t_1, t_2, insertarEntrada(reg, t, db))	$\equiv$ campoJoin(t_1, t_2, db)
campoJoin(t_1, t_2, borrar(cr, t, db))	$\equiv$ campoJoin(t_1, t_2, db)
campoJoin(t_1, t_2, generarVistaJoin(t_3, t_4, c, db))	$\equiv$ **if** $t_1 = t_3 \wedge t_2 = t_4$ **then** c **else** campoJoin(t_1, t_2, db) **fi**

registros(t, db)	$\equiv$ registros(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(t_1, t_2, db) ≡ combinarRegistros(campoJoin(t_1, t_2, db), registros(dameTabla(t_1, db)), registros(dameTabla(t_2, db)))

buscar(r,t,db) ≡ coincidencias(r,registros(dameTabla(t,db)))

Fin TAD