Algoritmos y Estructuras de Datos II

Primer Cuatrimestre de 2016

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Recuperatorio - Trabajo Práctico 1

Especificación

Grupo 4

Integrante	LU	Correo electrónico
Borgna, Agustin	79/15	aborgna@dc.uba.ar
Salvador, Alejo	467/15	alelucmdp@hotmail.com
Tamborindeguy, Guido	584/13	guido@tamborindeguy.com.ar
Zdanovitch, Nikita	520/14	3hb.tch@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Renombres de TADs

TAD NombreTabla es String

TAD NombreJoin es String

TAD NombreTrigger es String

TAD Join es Tupla(NombreTabla,NombreTabla,Campo)

TAD Trigger es Tupla(NombreTabla,NombreTabla,Registro)

2. TAD Campo

TAD Campo

igualdad observacional

$$(\forall c, c' : \text{campo}) \left(c =_{\text{obs}} c' \iff \begin{pmatrix} \text{cNombre}(c) =_{\text{obs}} \text{cNombre}(c') \wedge \\ \text{cEsString?}(c) =_{\text{obs}} \text{cEsString?}(c') \end{pmatrix} \right)$$

géneros campo

exporta campo, generadores, observadores, otras operaciones

usa string, bool

observadores básicos

cNombre : campo $\longrightarrow$ string

cEsString? : campo $\longrightarrow$ bool

generadores

campoString : string $\longrightarrow$ campo

campoNat : string $\longrightarrow$ campo

otras operaciones

cEsNat? : campo $\longrightarrow$ bool

axiomas $\forall c$: campo

cNombre(campoString(s)) $\equiv$ s

cNombre(campoNat(s)) $\equiv$ s

cEsString?(campoString(s)) $\equiv$ true

cEsString?(campoNat(s)) $\equiv$ false

cEsNat?(c) $\equiv \neg$ cEsString?(c)

Fin TAD

3. TAD Valor

TAD VALOR

igualdad observacional

$$(\forall v, v' : \text{valor}) \left(v =_{\text{obs}} v' \iff \left(\begin{array}{l} \textbf{if } \text{vEsString?}(v) \textbf{ then} \\ \quad \text{vEsString?}(v') \quad \wedge_{\text{L}} \\ \quad \text{leerString}(v) \quad =_{\text{obs}} \\ \quad \text{leerString}(v') \\ \textbf{else} \\ \quad \text{vEsNat?}(v') \quad \wedge_{\text{L}} \\ \quad \text{leerNat}(v) \quad =_{\text{obs}} \\ \quad \text{leerNat}(v') \\ \textbf{fi} \end{array} \right) \right)$$

géneros valor

exporta valor, generadores, observadores, otras operaciones

usa string, nat, bool

observadores básicos

leerString : valor v $\longrightarrow$ string $\{\text{vEsString?}(v)\}$

leerNat : valor v $\longrightarrow$ nat $\{\text{vEsNat?}(v)\}$

vEsString? : valor $\longrightarrow$ bool

generadores

valorString : string $\longrightarrow$ valor

valorNat : nat $\longrightarrow$ valor

otras operaciones

vEsNat? : valor $\longrightarrow$ bool

axiomas $\forall v$: valor

vEsString?(valorString(s)) $\equiv$ true

vEsString?(valorNat(n)) $\equiv$ false

vEsNat?(v) $\equiv$ $\neg$vEsString?(v)

leerString(valorString(s)) $\equiv$ s

leerNat(valorNat(n)) $\equiv$ n

Fin TAD

4. TAD Registro

TAD REGISTRO

igualdad observacional

$$(\forall r, r' : \text{registro}) \left(r =_{\text{obs}} r' \iff \begin{pmatrix} \text{rCampos}(r) =_{\text{obs}} \text{rCampos}(r') \wedge_L \\ (\forall c\text{: campo})\ c \in \text{rCampos}(r) \Rightarrow_L \\ \quad \text{rValor}(r,\ c) =_{\text{obs}} \text{rValor}(r',\ c) \end{pmatrix} \right)$$

géneros registro

exporta registro, generadores, observadores, otras operaciones

usa CAMPO, VALOR, CONJ

observadores básicos

rValor : registro r × campo c $\longrightarrow$ valor $\{c \in \text{rCampos}(r)\}$

rCampos : registro r $\longrightarrow$ conj(campo)

generadores

nuevoRegistro : $\longrightarrow$ registro

agregarValor : registro r × campo c × valor v $\longrightarrow$ registro
$\{\neg$ rContieneNombre?(r, cNombre(c)) $\wedge$ (cEsString?(c) $\iff$ vEsString?(v))$\}$

otras operaciones

unionRegistros : registro × registro $\longrightarrow$ registro

rVacio? : registro $\longrightarrow$ bool

mismosValores : registro $r1$ × registro $r2$ × conj(campo) cs $\longrightarrow$ bool
$\{cs \subseteq \text{rCampos(r1)} \wedge cs \subseteq \text{rCampos(r2)}\}$

rContieneNombre? : registro r × string *nombre* $\longrightarrow$ bool

mismosCampos : conj(registro) rs $\longrightarrow$ bool

todosLosCampos : conj(registro) rs $\longrightarrow$ conj(campo)

rFiltrarPorCriterio : conj(registro) rs × registro *criterio* $\longrightarrow$ conj(registro)
$\{$mismosCampos(rs) $\wedge$ rCampos(criterio) $\subseteq$ todosLosCampos(rs)$\}$

axiomas $\forall r$: registro, $\forall v$: valor, $\forall c$: campo

rValor(agregarValor(r, c, v), c') $\equiv$ **if** c = c' **then** v **else** rValor(r, c') **fi**

rCampos(nuevoRegistro) $\equiv$ $\emptyset$

rCampos(agregarValor(r, c, v)) $\equiv$ Ag(c, rCampos(r))

unionRegistros(r, nuevoRegistro) $\equiv$ r

unionRegistros(r, agregarValor(r',c,v)) $\equiv$ **if** $\neg$ rContieneNombre(r, cNombre(c)) **then**
agregarValor(unionRegistros(r,r'),c,v)
else
unionRegistros(r,r')
fi

rVacio?(nuevoRegistro) $\equiv$ true

rVacio?(agregarValor(r, c, v)) $\equiv$ false

mismosValores(r1,r2,cs) $\equiv$ **if** $\emptyset$?(cs) **then**
true
else
rValor(r1,dameUno(cs)) = rValor(r2,dameUno(cs))
$\wedge$ mismosValores(r1,r2,sinUno(cs))
fi

rContieneNombre?(nuevoRegistro, nom) $\equiv$ false

rContieneNombre?(agregarValor(r, c, v), nom) $\equiv$ cNombre(c) $=_{obs}$ nom $\vee$ rContieneNombre?(r, nom)

mismosCampos(rs) $\equiv$ **if** #(rs) $<$ 2 **then**
    true
else
    rCampos(dameUno(rs)) $=_{obs}$ rCampos(dameUno(sinUno(rs)))
    $\wedge$ mismosCampos(sinUno(rs))
fi

todosLosCampos(rs) $\equiv$ **if** $\emptyset$(rs) **then** $\emptyset$ **else** rCampos(dameUno(rs)) $\cup$ todosLosCampos(sinUno(rs)) **fi**

rFiltrarPorCriterio(rs,criterio) $\equiv$ **if** $\emptyset$(rs) **then**
    $\emptyset$
else
    if mismosValores(dameUno(rs), criterio, rCampos(criterio)) **then**
        Ag(dameUno(rs), rFiltrarPorCriterio(sinUno(rs)))
    else
        rFiltrarPorCriterio(sinUno(rs))
    fi
fi

Fin TAD

5. TAD Tabla

TAD TABLA

igualdad observacional

$$(\forall t, t' : \text{tabla}) \left(t =_{\text{obs}} t' \iff \begin{pmatrix} \text{tRegistros(t)} =_{\text{obs}} \text{tRegistros(t')} \wedge \\ \text{tModificaciones(t)} =_{\text{obs}} \text{tModificaciones(t')} \wedge \\ \text{tCampos(t)} =_{\text{obs}} \text{tCampos(t')} \wedge_L \\ (\forall c\text{: campo})\ \text{c} \in \text{tCampos(t)} \Rightarrow_L \\ \quad \text{tEsClave(t, c)} =_{\text{obs}} \text{tEsClave(t', c)} \end{pmatrix} \right)$$

géneros tabla

exporta tabla, generadores, observadores, otras operaciones

usa CONJUNTO(α), CAMPO, REGISTRO, VALOR

observadores básicos

tCampos : tabla $\longrightarrow$ conj(campo)

tEsClave : tabla t × campo c $\longrightarrow$ bool {c ∈ tCampos(t)}

tRegistros : tabla t $\longrightarrow$ conj(registro)

tModificaciones : tabla t $\longrightarrow$ nat

generadores

nuevaTabla : conj(campo) *campos* × conj(campo) *claves* $\longrightarrow$ tabla {¬∅?(claves) ∧ (claves ⊆ campos)}

insertarEnTabla : tabla t × registro r $\longrightarrow$ tabla
{tCampos(t) = rCampos(r) ∧ ¬tRegistroRepetido?(t, r)}

borrarDeTabla : tabla t × campo c × valor v $\longrightarrow$ tabla
{c ∈ tCampos(t) ∧ (cEsString?(c) $\iff$ vEsString?(v))}

otras operaciones

tieneCampo? : tabla t × campo c $\longrightarrow$ bool

tRegistroRepetido? : tabla t × registro r $\longrightarrow$ bool {tCampos(t) ⊆ rCampos(r)}

tClaves : tabla t $\longrightarrow$ conj(campo)

tieneRegistro : tabla t × campo c × valor v $\longrightarrow$ bool {c ∈ tClaves(t)}

tieneRegistroAux : tabla t × campo c × valor v × conj(registro) *tRegs* $\longrightarrow$ bool
{mismosCampos(rRegs) ∧ c ∈ todosLosCampos(rRegs)}

dameRegistro : tabla t × campo c × valor v $\longrightarrow$ registro {c ∈ tClaves(t) $\wedge_L$ tieneRegistro(t,c,v)}

dameRegistroAux : tabla t × campos c × valor v × conj(registro) *tRegs* $\longrightarrow$ registro
{mismosCampos(rRegs) ∧ c ∈ todosLosCampos(rRegs)}

tablasVirgenes : conj(tabla) *ts* $\longrightarrow$ bool

Buscar : tabla t × registro *criterio* × conj(campo) *campos* $\longrightarrow$ conj(registro)
{campos ⊆ tCampos(t) ∧ rCampos(criterio) ⊆ tCampos(t)}

quitarRegistros : conj(registro) rs × campo c × valor v $\longrightarrow$ conj(registro)
{mismosCampos(rs) ∧ c ∈ todosLosCampos(rs) ∧ (cEsString?(c) $\iff$ vEsString?(v))}

recortaR : registro r × conj(campo) *cs* $\longrightarrow$ registro {cs ⊆ rCampos(r) }

recortaCR : conj(registro) rs × conj(campo) *cs* $\longrightarrow$ conj(registro)
{mismosCampos(rs) ∧ cs ⊆ todosLosCampos(rs)}

axiomas $\forall t$: tabla

tCampos(nuevaTabla(campos, claves)) $\equiv$ campos

tCampos(insertarEnTabla(t, r)) $\equiv$ tCampos(t)

tCampos(borrarDeTabla(t, c, v)) $\equiv$ tCampos(t)

tEsClave(nuevaTabla(campos, claves), c) $\equiv$ c $\in$ claves

tEsClave(insertarEnTabla(t, r), c) $\equiv$ tEsClave(t, c)

tEsClave(borrarDeTabla(t, c, v), c) $\equiv$ tEsClave(t, c)

tRegistros(nuevaTabla(campos, claves)) $\equiv$ $\emptyset$

tRegistros(insertarEnTabla(t, r)) $\equiv$ Ag(r, tRegistros(t))

tRegistros(borrarDeTabla(t, c, v)) $\equiv$ quitarRegistros(tRegistros(t), c, v)

tModificaciones(nuevaTabla(campos, claves)) $\equiv$ 0

tModificaciones(insertarEnTabla(t, r)) $\equiv$ 1 + tModificaciones(t)

tModificaciones(borrarDeTabla(t, c, v)) $\equiv$ 1 + tModificaciones(t)

tieneCampo?(t, c) $\equiv$ c $\in$ tCampos(t)

tRegistroRepetido?(t, r) $\equiv$ recortaR(r, tClaves(t)) $\in$ recortaCR(tRegistros(t), tClaves(t))

tClaves(nuevaTabla(campos, claves)) $\equiv$ claves

tClaves(insertarEnTabla(t, r)) $\equiv$ tClaves(t)

tClaves(borrarDeTabla(t, c, v)) $\equiv$ tClaves(t)

tieneRegistro(t, c, v) $\equiv$ tieneRegistrosAux(t, c, v, tRegistros(t))

tieneRegistroAux(t, c, v, tRegs) $\equiv$ **if** $\varnothing$?(tRegs) **then**
false
else
rValor(dameUno(tRegs), c) == v $\vee$ dameRegistroAux(t, c , v, sinUno(tRegs))
fi

dameRegistro(t, c, v) $\equiv$ dameRegistrosAux(t, c, v, tRegistros(t))

dameRegistroAux(t, c, v, tRegs) $\equiv$ **if** rValor(dameUno(tRegs), c) == v **then**
dameUno(tRegs)
else
dameRegistroAux(t, c , v, sinUno(tRegs))
fi

tablasVirgenes(ts) $\equiv$ **if** $\emptyset$?(ts) **then**
true
else
tModificaciones(dameUno(ts)) = 0 $\wedge$ tablasVirgenes(sinUno(ts))
fi

tBuscar(t, criterio, campos) $\equiv$ recortaCR(rFiltrarPorCriterio(rRegistros(t), criterio), campos)

quitarRegistros(rs, c, v) $\equiv$ **if** $\emptyset$?(rs) **then**
$\emptyset$
else
if c $\in$ rCampos(dameUno(rs)) $\wedge_L$ v = rValor(dameUno(rs),c) **then**
Ag(dameUno(rs),quitarRegistros(sinUno(rs), c, v))
else
quitarRegistros(sinUno(rs), c, v)
fi
fi

recortaR(r, cs) $\equiv$ **if** $\emptyset$?(cs) **then**
nuevoRegistro
else
agregarValor(recortaR(r, sinUno(cs)), rValor(r, dameUno(cs)), rValor(r, dameUno(cs)))
fi

recortaCR(rs, cs) $\equiv$ **if** $\emptyset$?(rs) **then** $\emptyset$ **else** Ag(recortaR(dameUno(rs), cs), recortaCR(sinUno(rs), cs)) **fi**

Fin TAD

6. TAD Database

TAD DATABASE

igualdad observacional

$$(\forall db, db' : \text{database}) \left(db =_{\text{obs}} db' \iff \left(\begin{array}{l} (\forall nt\text{: nombreTabla}) \\ \quad (\text{dbTieneTabla(db,nt)} =_{\text{obs}} \text{dbTieneTabla(db',nt)}) \\ \wedge\ \text{dbJoins(db)} =_{\text{obs}} \text{dbJoins(db')} \\ \wedge\ \text{dbTriggers(db)} =_{\text{obs}} \text{dbTriggers(db')} \\ \wedge_L\ (\forall nt\text{: nombreTabla})\ \text{dbTieneTabla(db,nt)} \Rightarrow_L \\ \quad (\text{dbTabla(db,nt)} =_{\text{obs}} \text{dbTabla(db',nt)}) \\ \wedge\ (\forall nj\text{: nombreJoin})\ \text{nj} \in \text{claves(dbJoins(db,nj))} \Rightarrow_L \\ \quad (\text{dbVerJoin(db,nj)} =_{\text{obs}} \text{dbVerJoin(db',nj)}) \end{array} \right) \right)$$

géneros database

exporta database, generadores, observadores, masModificada

usa DICCIONARIO(α), CONJUNTO(α), REGISTRO, CAMPO, VALOR, NOMBRETABLA, TABLA, NOMBREJOIN, JOIN, NOMBRETRIGGER, TRIGGER, NAT

observadores básicos

dbTabla : database db × nombreTabla nt $\longrightarrow$ tabla {tieneTabla(db,nt)}

dbTieneTabla : database × nombreTabla $\longrightarrow$ bool

dbJoins : database $\longrightarrow$ dicc(nombreJoin,join)

dbVerJoin : database db × nombreJoin nj $\longrightarrow$ conj(registro) {def?(nj,dbJoins(db))}

dbTriggers : database $\longrightarrow$ dicc(nombreTrigger,trigger)

generadores

nuevaDB : dicc(nombreTabla × tabla) $tablas$ $\longrightarrow$ database
{¬∅?(claves(tablas)) ∧ tablasVirgenes(valores(tablas))}

insertar : database db × nombreTabla nt × registro r $\longrightarrow$ database
$$\left\{ \begin{array}{l} \text{dbTieneTabla(db,nt)} \wedge_L \text{tCampos(dbTabla(db, nt))} = \text{rCampos(r)} \\ \wedge\ \neg\ \text{tRegistroRepetido?(dbTabla(db, nt), r)} \end{array} \right\}$$

borrar : database db × nombreTabla nt × campo c × valor v $\longrightarrow$ database
{(cEsString?(c) $\iff$ vEsString?(v)) ∧ dbTieneTabla(nt) $\wedge_L$ c ∈ tCampos(dbTabla(db,nt))}

crearJoin : database db × nombreJoin nj × join j $\longrightarrow$ database
$$\left\{ \begin{array}{l} \neg\text{def?(nj,dbJoins(db))} \wedge \neg(\Pi_1(\text{j}) = \Pi_2(\text{j})) \wedge (\\ \quad \text{dbTieneTabla(db, } \Pi_1\text{(j))} \wedge_{\text{L}} \\ \quad \text{tieneCampo?(dbTabla(db, } \Pi_1\text{(j)), } \Pi_3\text{(j))} \wedge_{\text{L}} \\ \quad \text{tEsClave(dbTabla(db, } \Pi_1\text{(j)), } \Pi_3\text{(j))} \\ ) \wedge (\\ \quad \text{dbTieneTabla(db, } \Pi_2\text{(j))} \wedge_{\text{L}} \\ \quad \text{tieneCampo?(dbTabla(db, } \Pi_2\text{(j)), } \Pi_3\text{(j))} \wedge_{\text{L}} \\ \quad \text{tEsClave(dbTabla(db, } \Pi_2\text{(j)), } \Pi_3\text{(j))} \\ ) \end{array} \right\}$$

borrarJoin : database db × nombreJoin nj $\longrightarrow$ database {def?(nj,dbJoins(db))}

crearTrigger : database db × nombreTrigger ntg × trigger tg $\longrightarrow$ database
$$\left\{ \begin{array}{l} \neg\text{def?(ntg,dbTriggers(db))} \wedge \neg(\Pi_1(\text{tg}) = \Pi_2(\text{tg})) \\ \wedge\ \text{dbTieneTabla(db, } \Pi_1\text{(tg))} \wedge \text{dbTieneTabla(db, } \Pi_2\text{(tg))} \\ \wedge_{\text{L}}\ \text{tClaves(dbTabla(db, } \Pi_2\text{(tg))) } \subseteq \text{tClaves(dbTabla(db, } \Pi_1\text{(tg)))} \\ \wedge\ \text{tCampos(dbTabla(db, } \Pi_2\text{(tg))) } \subseteq \text{tCampos(dbTabla(db, } \Pi_1\text{(tg))) } \cup \text{rCampos(}\Pi_3\text{(tg)))} \end{array} \right\}$$

borrarTrigger : database db × nombreTrigger ntg $\longrightarrow$ database {def?(ntg,dbTriggers(db))}

otras operaciones

masModificada : database db $\longrightarrow$ nombreTabla

masModificadaAux : database db $\times$ conj(nombreTabla) nts $\longrightarrow$ nombreTabla
$\{\neg\emptyset?(\text{nts}) \wedge nts \subseteq \text{nombresTablas}(db)\}$

correrTriggers : database $\times$ conj(nombreTrigger) $\times$ nombreTabla ins $\times$ nombreTabla tgt $\times$ registro $\longrightarrow$ tabla
$\{\neg(\text{ins} =_{\text{obs}} \text{tgt})\}$

nombresTablas : database db $\longrightarrow$ conj(nombreTabla)

verJoinAux : tabla t $\times$ tabla t' $\times$ campo c $\longrightarrow$ conj(registro)

joinRegistros : conj(registro) $\times$ tabla $\times$ campo $\longrightarrow$ conj(registro)

valores : diccionario(α) $\longrightarrow$ conj(α)

axiomas $\forall db$: database, $\forall nt, nt', ntInsert, ntTarget$: nombreTabla, $\forall r$: registro, $\forall c$: campo, $\forall v$: valor, $\forall ntg$: nombreTrigger, $\forall nj, nj'$: nombreJoin, $\forall tablas$: conj(tabla), $\forall j$: join, $\forall tg$: trigger, $\forall t, t'$: tabla, $\forall rsTabla, rsSuma$: conj(registro)

dbTabla(nuevaDB(tablas),nt) $\equiv$ obtener(nt,tablas)

dbTabla(insertar(db,nt',r),nt) $\equiv$ **if** nt' = nt **then**
insertarEnTabla(dbTabla(db,nt),r)
else
correrTriggers(insertarEnTabla(dbTabla(db,nt),r),
claves(dbTriggers(db)), nt', nt, r)
fi

dbTabla(borrar(db,nt',c,v),nt) $\equiv$ **if** nt' = nt **then**
borrarDeTabla(dbTabla(db,nt),c,v)
else
dbTabla(db,nt)
fi

dbTabla(crearJoin(db,nj,j),nt) $\equiv$ dbTabla(db,nt)

dbTabla(borrarJoin(db,nj),nt) $\equiv$ dbTabla(db,nt)

dbTabla(crearTrigger(db,ntg,tg),nt) $\equiv$ dbTabla(db,nt)

dbTabla(borrarTrigger(db,ntg),nt) $\equiv$ dbTabla(db,nt)

dbTieneTabla(nuevaDB(tablas),nt) $\equiv$ def?(nt,tablas)

dbTieneTabla(insertar(db,nt',r),nt) $\equiv$ dbTieneTabla(db,nt)

dbTieneTabla(borrar(db,nt',c,v),nt) $\equiv$ dbTieneTabla(db,nt)

dbTieneTabla(crearJoin(db,nj,j),nt) $\equiv$ dbTieneTabla(db,nt)

dbTieneTabla(borrarJoin(db,nj),nt) $\equiv$ dbTieneTabla(db,nt)

dbTieneTabla(crearTrigger(db,ntg,tg),nt) $\equiv$ dbTieneTabla(db,nt)

dbTieneTabla(borrarTrigger(db,ntg),nt) $\equiv$ dbTieneTabla(db,nt)

dbJoins(nuevaDB(tablas)) $\equiv$ vacio

dbJoins(insertar(db,nt,r)) $\equiv$ dbJoins(db)

dbJoins(borrar(db,nt,c,v)) $\equiv$ dbJoins(db)

dbJoins(crearJoin(db,nj,j)) $\equiv$ definir(nj,j,dbJoins(db))

dbJoins(borrarJoin(db,nj)) $\equiv$ borrar(nj,dbJoins(db))

dbJoins(crearTrigger(db,ntg,tg)) $\equiv$ dbJoins(db)

dbJoins(borrarTrigger(db,ntg)) $\equiv$ dbJoins(db)

dbVerJoin(insertar(db,nt,r), nj) $\equiv$ verJoinAux(
dbTabla(insertar(db, nt, r), Π_1(obtener(dbJoins(db), nj))),
dbTabla(insertar(db, nt, r), Π_2(obtener(dbJoins(db), nj))),
Π_3(obtener(dbJoins(db), nj))
)

dbVerJoin(borrar(db, nt, c, v), nj) ≡ verJoinAux(
dbTabla(borrar(db, nt, c, v), Π_1(obtener(dbJoins(db), nj))),
dbTabla(borrar(db, nt, c, v), Π_2(obtener(dbJoins(db), nj))),
Π_3(obtener(dbJoins(db), nj))
)

dbVerJoin(crearJoin(db,nj',j), nj) ≡ dbVerJoin(db, nj)

dbVerJoin(borrarJoin(db,nj'), nj) ≡ dbVerJoin(db, nj)

dbVerJoin(crearTrigger(db,ntg,tg), nj) ≡ dbVerJoin(db, nj)

dbVerJoin(borrarTrigger(db,ntg), nj) ≡ dbVerJoin(db, nj)

dbTriggers(nuevaDB(tablas)) ≡ vacio

dbTriggers(insertar(db,nt,r)) ≡ dbTriggers(db)

dbTriggers(borrar(db,nt,c,v)) ≡ dbTriggers(db)

dbTriggers(crearJoin(db,nj,j)) ≡ dbTriggers(db)

dbTriggers(borrarJoin(db,nj)) ≡ dbTriggers(db)

dbTriggers(crearTrigger(db,ntg,tg)) ≡ definir(ntg,tg,dbTriggers(db))

dbTriggers(borrarTrigger(db,ntg)) ≡ borrar(ntg,dbTriggers(db))

masModificada(db) ≡ masModificadaAux(db, nombresTablas(db))

masModificadaAux(db, nts) ≡ **if** ∅?(sinUno(nts)) **then**
dameUno(nts)
else
if tModificaciones(dbTabla(db, dameUno(nts)))
≥ tModoficaciones(dbTabla(db, masModificadaAux(db, sinUno(nts))))
then
dameUno(nts)
else
masModificadaAux(db, sinUno(nts))
fi
fi

correrTriggers(db, ntgs, ntInsert, ntTarget, r) ≡ **if** ∅?(ntgs) **then**
dbTabla(db,ntTarget)
else
if
Π_1(obtener(dameUno(ntgs),dbTriggers(db)))
$=_{obs}$ ntInsert ∧
Π_2(obtener(dameUno(ntgs),dbTriggers(db)))
$=_{obs}$ ntTarget ∧
¬ tRegistroRepetido?(
correrTriggers(db,sinUno(ntgs),ntInsert,
ntTarget, r),
unionRegistros(r,Π_3(obtener(dameUno(ntgs),
dbTriggers(db))))
)
then
insertarEnTabla(
correrTriggers(db,sinUno(ntgs),ntInsert,
ntTarget, r),
unionRegistros(r,Π_3(obtener(dameUno(ntgs),
dbTriggers(db))))
)
else
correrTriggers(db,sinUno(ntgs),ntInsert, ntTarget, r)
fi
fi

nombresTablas(nuevaDB(tablas)) ≡ claves(tablas)

nombresTablas(insertar(db,nt,r)) ≡ nombresTablas(db)

nombresTablas(borrar(db,nt,c,v)) ≡ nombresTablas(db)

nombresTablas(crearJoin(db,nj,j)) ≡ nombresTablas(db)

nombresTablas(borrarJoin(db,nj)) ≡ nombresTablas(db)

nombresTablas(crearTrigger(db,ntg,tg)) ≡ nombresTablas(db)

nombresTablas(borrarTrigger(db,ntg)) ≡ nombresTablas(db)

verJoinAux(t, t', campo) ≡ joinRegistros(tRegistros(t), t', campo)

```
joinRegistros(rsTabla, t, campo) ≡ if ∅?(rsTabla) then
                                       ∅
                                   else
                                       if tieneRegistro(t, campo,
                                                rValor(dameUno(rsTabla), campo)) then
                                           Ag(unionRegistros(dameUno(rsTabla),
                                                   dameRegistro(t, campo,
                                                   rValor(dameUno(rsTabla), campo))),
                                              joinRegistros(sinUno(rsTabla), t, campo)
                                           )
                                       else
                                           joinRegistros(sinUno(rsTabla), t, campo)
                                       fi
                                   fi
valores(d) ≡ if vacio(d) then
                 ∅
             else
                 Ag(obtener(dameUno(claves(d)), d), valores(borrar(dameUno(claves(d)), d)))
             fi
```

Fin TAD