



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2: Diseño

Primer cuatrimestre - 2016

Algoritmos y Estructuras de Datos II

Grupo XXXX

Integrante	LU	Correo electrónico
BENZO, Mariano	198/14	marianobenzo@gmail.com
FARIAS, Mauro	821/13	farias.mauro@hotmail.com
GUTTMAN, Martin	686/14	haris@live.com.ar
MOSQUEIRA C., Edgardo Ramon	808/13	edgarcab666@hotmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria – Pabellón I (Planta Baja)

Intendente Güiraldes 2160 – C1428EGA

Ciudad Autónoma de Buenos Aires – Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

Índice

1 Tabla

1.1 Interfaz

se explica con TABLA

usa

géneros nat, dato, campo, tipo, registro, conjTrie, string, diccTrie(string, alfa), diccAVL

Operaciones

NOMBRE(in t : tab) \longrightarrow res : string

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ nombre(t)}

Descripción: Devuelve el nombre de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se retorna res por copia, por ser un tipo basico.

CLAVES(in t : tab) \longrightarrow res : itConjTrie(campo)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ claves(t)}

Descripción: Devuelve un conjunto de campos que son claves en la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve un iterador al conjunto claves por referencia.

INDICES(in t : tab) \longrightarrow res : itConjTrie(campo)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ indices(t)}

Descripción: Devuelve un conjunto de los indices de la tabla ingresada por parametro.

Complejidad: $O(\text{calcular})$

Aliasing: Se devuelve res por referencia y no es modificable.

CAMPOS(in t : tab) \longrightarrow res : itConjTrie(campo)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ campos(t)}

Descripción: Devuelve un conjunto a los campos de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia.

TIPOCAMPO(in c : campo, in t : tab) \longrightarrow res : tipo

Pre \equiv { $c \in \text{campos}(t)$ }

Post \equiv { $res =_{\text{obs}}$ tipoCampo(t)}

Descripción: Devuelve el tipo del campo c en la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia, no es modificable.

REGISTROS(in t : tab) \longrightarrow res : itConj(registro)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ registros(t)}

Descripción: Devuelve un conjunto a los registros de la tabla ingresada por parametro.

Complejidad: $O(L + \log(n))$

Aliasing: Se devuelve res referencia

CANTIDADDEACCESOS(**in** $t : \text{tab}$) $\longrightarrow res : \text{nat}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{cantidadDeAccesos}(t)\}$

Descripción: Devuelve la cantidad de modificaciones de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por copia.

NUEVATABLA(**in** $\text{nombre} : \text{string}$, $\text{in claves} : \text{conjTrie}(\text{campo})$, $\text{in columnas} : \text{registro}$) $\longrightarrow res : \text{tab}$

Pre $\equiv \{\neg \emptyset?(\text{claves}) \wedge \text{claves} \subseteq \text{campos}(\text{columnas})\}$

Post $\equiv \{res =_{\text{obs}} \text{nuevaTabla}(t)\}$

Descripción: Crea una tabla sin registros.

Complejidad: $O(\text{calcular})$

AGREGARREGISTRO(**in** $r : \text{registro}$, $\text{in } t : \text{tab}$)

Pre $\equiv \{t_0 = t \wedge \text{campos}(r) =_{\text{obs}} \text{campos}(t) \wedge \text{puedoInsertar?}(r, t)\}$

Post $\equiv \{\text{agregarRegistro}(r, t_0)\}$

Descripción: Agrega un registro a la tabla pasada por parametro.

Complejidad: $O(L + in)$

Aliasing: Agrega el registro r por referencia.

BORRARREGISTRO(**in** $\text{crit} : \text{registro}$, $\text{in } t : \text{tab}$)

Pre $\equiv \{t_0 = t \wedge \#(\text{campos}(r)) = 1 \wedge_L \text{dameUno}(\text{campos}(\text{crit})) \in \text{claves}(t)\}$

Post $\equiv \{\text{borrarRegistro}(r, t_0)\}$

Descripción: Borra los registros que cumplan el criterio pasado por parametro.

Complejidad: $O(L + in)$

INDEXAR(**in** $\text{crit} : \text{registro}$, $\text{in } t : \text{tab}$)

Pre $\equiv \{t_0 = t \wedge \text{puedeIndexar}(c, t)\}$

Post $\equiv \{\text{indexar}(c, t_0)\}$

Descripción: Borra los registros que cumplan el criterio pasado por parametro.

Complejidad: $O(L + in)$

PUEDOIINSERTAR?(**in** $r : \text{registro}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{puedoInsertar?}(r, t)\}$

Descripción: Informa si el registro pasado por parametro no tiene valores repetidos con respecto a los registros existentes, para los campos clave en la tabla pasada por parametro.

Complejidad: $O(T * L + in)$

COMPATIBLE(**in** $r : \text{registro}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{compatible}(r, t)\}$

Descripción: Informa si el registro pasado por parametro tiene correspondencia en los tipos de los campos de tabla pasada por parametro.

Complejidad: $O(1)$

MINIMO(**in** $c : \text{campo}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{dato}$

Pre $\equiv \{\neg \emptyset?(\text{registro}(t)) \wedge c \in \text{indices}(t)\}$

Post $\equiv \{res =_{\text{obs}} \text{minimo}(c, t)\}$

Descripción: Retorna el minimo entre los valores de la tabla para el campo c.

Complejidad: $O(L + in)$

Aliasing: Retorna res por referencia.

MAXIMO(**in** $c : \text{campo}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{dato}$

Pre $\equiv \{\neg \emptyset?(\text{registro}(t)) \wedge c \in \text{indices}(t)\}$

Post $\equiv \{res =_{\text{obs}} \text{maximo}(c, t)\}$

Descripción: Retorna el maximo entre los valores de la tabla para el campo c.

Complejidad: $O(L + in)$

Aliasing: Retorna res por referencia.

PUEDEINDEXAR(**in** $c : \text{campo}$, **in** $t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{puedeIndexar}(c, t)\}$

Descripción: Informa si se puede crear un nuevo indice.

Complejidad: $O(L + in)$

COINCIDENCIAS(**in** $r : \text{registro}$, **in** $cj : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{Conj}(\text{registro})$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{coincidencias}(r, cj)\}$

Descripción: Compara el valor del registro con el conjunto de registros y retorna la interseccion.

Complejidad: $O(L + in)$

Aliasing: Retorna res por referencia.

HAYCOINCIDENCIA(**in** $r : \text{registro}$, **in** $cj1 : \text{ConjTrie}(\text{campo})$, **in** $cj2 : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{hayCoincidencia}(r, cj1, cj2)\}$

Descripción: Compara los valores del registro para los campos dados por parametro, con el conjunto de registros.

Complejidad: $O(L + in)$

COMBINARREGISTROS(**in** $c : \text{campo}$, **in** $cj1 : \text{Conj}(\text{registro})$, **in** $cj2 : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{conj}(\text{registro})$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{combinarRegistros}(c, cj1, cj2)\}$

Descripción: Combina los valores de los registros para el campo dado por parametro.

Complejidad: $O(L + in)$

Aliasing: Retorna res por copia.

DAMECOLUMNA(**in** $c : \text{campo}$, **in** $cj : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{conj}(\text{dato})$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{dameColumna}(c, cj1, cj2)\}$

Descripción: Reune en un conjunto los valores del campo pasado por parametro.

Complejidad: $O(T * L + in)$

Aliasing: Retorna res por referencia.

MISMOSTIPOS(**in** $r : \text{registro}$, **in** $t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{\text{campos}(r) \subseteq \text{campos}(t)\}$

Post $\equiv \{res =_{\text{obs}} \text{mismosTipos}(r, t)\}$

Descripción: Compara los tipos correspondientes a los campos del registro y la tabla.

Complejidad: $O(1)$

1.2 Representación

se representa con *Tabla*

donde *tab* es $\text{tupla}(\text{Nombre} : \text{String},$
 $\text{Indices} : \text{DiccTrie}(\text{campo}, \text{Indice}),$
 $\text{Registros} : \text{Conj}(\text{Registro}),$
 $\text{Campos} : \text{DiccTrie}(\text{Campo}, \text{Tipo}),$
 $\# \text{Accesos} : \text{Nat})$
donde *Indice* es $\text{tupla}(\text{PorNat} : \text{DiccTrie}(\text{Dato Conj}(\text{Registro})),$
 $\text{PorString} : \text{DiccTrie}(\text{Dato Conj}(\text{Registro})))$

Invariante de representación

1. Para todos los registros de *r*, el tipo de los datos de las columnas de *r*, deben coincidir con los tipos de las columnas en *e.campos*.
2. Todas las columnas de *e.campos* y su tipo, deben coincidir con los campos y tipo de todos los registros de *e.registros*. Es decir no debe haber campos de mas.'
3. El nombre de la tabla que figura en *e.nombre*, es un string de longitud acotada.
4. Para todo registro *r* de *e.registros* y para todo campo *c* de *e.Indices.DiccClaves*, se debe cumplir que si tenemos $\text{valor} \leftarrow \text{Obtener}(r, c)$ y $\text{ind} \leftarrow \text{Obtener}(e.\text{Indices}, c)$. Al evaluar que $r \in \text{Obtener}(\text{ind}, \text{valor})$ y deben ser del mismo tipo.
5. Para todo campo *c*, que pertenece a *e.Indices.DiccClaves*, si tenemos que $\text{ind} \leftarrow \text{Obtener}(e.\text{Indices}, c)$, y para todo dato *d* perteneciente a *ind.DiccClaves* entonces $\text{Obtener}(\text{ind}, d)$ esta incluido o es igual a *e.registros*.
6. Para todo registro *r* perteneciente a *e.registros* *r.DiccClaves* es igual a *e.campos.DiccClaves*.
7. El valor de *e.#Accesos* debe ser la cantidad de registros agregados, la cantidad de registros borrados, mas la cantidad de indices creados.

Función de abstracción

$\text{Abs} : \widehat{\text{sistema}} s \longrightarrow \widehat{\text{CampusSeguro}} \quad \{\text{Rep}(s)\}$

$(\forall s : \widehat{\text{sistema}})$
 $\text{Abs}(s) \equiv cs : \widehat{\text{CampusSeguro}} \mid s.\text{campus} =_{\text{obs}} \text{campus}(cs) \wedge$
 $s.\text{estudiantes} =_{\text{obs}} \text{estudiantes}(cs) \wedge$
 $s.\text{hippies} =_{\text{obs}} \text{hippies}(cs) \wedge$
 $s.\text{agentes} =_{\text{obs}} \text{agentes}(cs) \wedge$
 $((\forall n : \text{nombre}) s.\text{hippies}.\text{definido}(n) \Rightarrow_{\text{L}} s.\text{hippies}.\text{obtener}(n) =_{\text{obs}} \text{posEstYHippie}(n, cs) \vee$
 $(\forall n : \text{nombre}) s.\text{estudiantes}.\text{definido}(n) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(n) =_{\text{obs}} \text{posEstYHippie}(n, cs))$
 $(\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{pos} =_{\text{obs}} \text{posAgente}(pl, cs))$
 $(\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{cantSanciones} =_{\text{obs}} \text{cantSanciones}(pl, cs))$
 $(\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{cantCapturas} =_{\text{obs}} \text{cantCapturas}(pl, cs))$

1.3 Algoritmos

NOMBRE(in $t : \text{tab}$) $\longrightarrow res : \text{string}$ $res \leftarrow t.nombre$	O(1)
	O(1)
CLAVES(in $t : \text{tab}$) $\longrightarrow res : \text{ConjTrie}(\text{campo})$ $res \leftarrow t.Campos.ClavesDicc$	O(1)
	O(1)
INDICES(in $t : \text{tab}$) $\longrightarrow res : \text{ConjTrie}(\text{campo})$ $res \leftarrow t.ClavesDicc$	O(1)
	O(1)
CAMPOS(in $t : \text{tab}$) $\longrightarrow res : \text{ConjTrie}(\text{campo})$ $res \leftarrow t.Campos.ClavesDicc$	O(1)
	O(1)
TIPOCAMPO(in $c : \text{campo}$, $in t : \text{tab}$) $\longrightarrow res : \text{Tipo}$ $res \leftarrow \text{Significado}(t.Campos, c)$	O(1)
	O(1)
REGISTROS(in $t : \text{tab}$) $\longrightarrow res : \text{Conj}(\text{registro})$ $res \leftarrow t.registros$	$\theta(L + \log(n))$
	$\theta(L + \log(n))$
CANTDEACCESOS(in $t : \text{tab}$) $\longrightarrow res : \text{nat}$ $res \leftarrow t.cantDeAccesos$	$\theta(1)$
	$\theta(1)$
NUEVATABLA(in $nombre : \text{string}$, $in claves : \text{conjTrie}(\text{campo})$, $in columnas : \text{registro}$) $\longrightarrow res : \text{tab}$ $itcampos \leftarrow \text{crearItTrie}(\text{Campos}(\text{columnas}))$ O(1) $res \leftarrow < nombre \text{ Vacio}() \text{ Vacio}() 0 >$ O(1) while HaySiguiente(itcampos) do O(1) Esto se debe a que # de campos a iterar es acotada. $valor \leftarrow \text{Significado}(r, \text{Siguiente}(itcampos))$ O(1) DefinirRapido(res.Campos, Siguiente(itcampos), valor) O(1) if Pertenece?(claves, Siguiente(itcampos)) then O(1) $val \leftarrow \text{Significado}(r, \text{Siguiente}(itcampos))$ AgregarRapido(res.Campos.CamposClave, val) O(1) end if Avanzar(itcampos) O(1) end while	
	$\theta(1)$
AGREGARREGISTRO(in $r : \text{registro}$, $in t : \text{tab}$) $nuevo \leftarrow \text{AgregarRapido}(t.Registros, r)$ $\theta(1)$ $t.\#Accesos++$ $\theta(1)$ if Cardinal(t.Indices.ClavesDicc) ≥ 1 then $\theta(1)$	

itInd \leftarrow crearItConjTrie(t.Indices.ClavesDice)	$\theta(1)$
while HaySiguiente(itInd) do	$\theta(1)$
indiceC \leftarrow Obtener(t.Indices, Siguiente(itInd))	$\theta(1)$
valorC \leftarrow Obtener(r, Siguiente(itInd))	$\theta(1)$
AgregarRapido(Obtener(indiceC, valorC), nuevo)	$\theta(1)$
Avanzar(itInd)	$\theta(1)$
end while	
end if	
<hr/>	
	$\theta(1)$
BORRARREGISTRO(in crit : registro, in t : tab)	
c \leftarrow Siguiente(Campos(crit))	$\theta(1)$
valor \leftarrow Obtener(crit, c)	$\theta(1)$
if Definido?(t.Indices, c) then	$\theta(1)$
indiceC \leftarrow Obtener(t.Indices, c)	$\theta(1)$
itcjr \leftarrow CrearItConj(Obtener(indiceC, valor))	$\theta(1)$
while HaySiguiente(itcjr) do	$\theta(\log(n))$
EliminarSiguiente(Siguiente(itcjr))	$\theta(1)$
tiene sentido???	
EliminarSiguiente(itcjr)	$\theta(1)$
end while	
else	
cr \leftarrow Coincidencias(crit, t.registros)	$\theta(\text{Cardinal}(t.\text{registros}))$
while HaySiguiente(cr) do	
EliminarSiguiente(Siguiente(cr))	
tiene sentido???	
EliminarSiguiente(cr)	
end while	
end if	
<hr/>	
	$\theta(\text{Calcular despues de consulta})$
INDEXAR(in c : campo, in t : tab)	
if tipoCampo(c,t) then	
conjLog(registro) nuevo \leftarrow vacio()	
else	
conjTrie(registro) nuevo \leftarrow vacio()	
end if	
indC \leftarrow Siguiente(DefinirRapido(t.Indices, c, nuevo))	
cr \leftarrow t.registros	
while HaySiguiente(cr) do	
valor \leftarrow Obtener(Siguiente(cr), c)	
if Definido?(indC, valor) then	
regviejos \leftarrow Obtener(indC, valor)	
AgregarRapido(regviejos, Siguiente(cr))	
else	
DefinirRapido(indC, valor, Siguiente(cr))	
end if	
Avanzar(cr)	
end while	
<hr/>	
	$\theta(1)$
PUEDOINSERTAR?(in r : registro, in t : tab) \rightarrow res : bool	

$\text{res} \leftarrow \text{compatible}(r,t) \wedge \neg \text{hayCoincidencia}(r, r.\text{ClavesDicc}, \text{registros}(t))$	$\theta(L+\log(n))$
	<hr/>
	$\theta(L+\log(n))$
$\text{COMPATIBLE}(\text{in } r : \text{registro}, \text{ in } t : \text{tab}) \longrightarrow \text{res} : \text{bool}$	
$\text{res} \leftarrow \text{compatible}(r,t) \wedge_L \text{mismosTipos}(r,t)$	$\theta(1)$
	<hr/>
	$O(1)$
$\text{PUEDEINDEXAR}(\text{in } c : \text{campo}, \text{ in } t : \text{tab}) \longrightarrow \text{res} : \text{bool}$	
$\text{res} \leftarrow \text{Definido?}(t.\text{campos},c) \wedge_L \neg \text{Definido?}(t.\text{Indices},c) \wedge (\text{Cardinal}(t.\text{Indices}) \leq 1)$	
	<hr/>
	$O(\text{calcular})$
$\text{COMBINARREGISTROS}(\text{in } c : \text{campo}, \text{ in } cr1 : \text{Conj}(\text{registro}), \text{ in } cr2 : \text{Conj}(\text{registro})) \longrightarrow \text{res} : \text{Conj}(\text{registros})$	
$\text{itcr1} \leftarrow \text{CrearItConjTrie}(cr1)$	$\theta(1)$
$\text{copiacr2} \leftarrow \text{Copiar}(cr2)$	$\theta(\text{Cardinal}(cr2))$
while $\text{HaySiguiente}(\text{itcr1})$ do	$\theta(\text{Cardinal}(cr1))$
$\text{combinarTodos}(c, \text{Siguiente}(\text{itcr1}), \text{copiacr2})$	$\theta(1)$
$\text{Avanzar}(\text{itcr1});$	$\theta(1)$
end while	
$\text{res} \leftarrow \text{copiacr2};$	$\theta(1)$
	<hr/>
	$O(\text{Cardinal}(cr1))$
$\text{HAYCOINCIDENCIA}(\text{in } r : \text{registro}, \text{ in } cc : \text{ConjTrie}(\text{campo}), \text{ in } cr : \text{Conj}(\text{registro})) \longrightarrow \text{res} : \text{bool}$	
$\text{itcr} \leftarrow \text{CrearItConj}(cr);$	$\theta(1)$
$\text{res} \leftarrow \text{false};$	$\theta(1)$
while $\text{HaySiguiente}(\text{itcr})$ do	$\theta(\text{Cardinal}(cr))$
$\text{res} \leftarrow \text{coincideAlguno}(r, cc, \text{Siguiente}(\text{itcr})) \vee \text{res};$	$\theta(1)$
$\text{Avanzar}(\text{itcr});$	$\theta(1)$
end while	
	<hr/>
	$O(\text{Cardinal}(cr))$
$\text{COINCIDENCIAS}(\text{in } crit : \text{registro}, \text{ in } cr : \text{Conj}(\text{registro})) \longrightarrow \text{res} : \text{Conj}(\text{registro})$	
$\text{res} \leftarrow \text{Vacio}();$	$\theta(1)$
$\text{itcr} \leftarrow \text{CrearItConj}(cr)$	
while $\text{HaySiguiente}(cr)$ do	$\theta(\text{Cardinal}(cr))$
if $\text{coincidenTodos}(crit, \text{campos}(crit), \text{Siguiente}(\text{itcr}))$ then	$\theta(1)$
$\text{AgregarRapido}(\text{res}, \text{Siguiente}(\text{itcr}))$	$\theta(1)$
end if	
$\text{Avanzar}(\text{itcr});$	$\theta(1)$
end while	
	<hr/>
	$O(\text{Cardinal}(cr))$
$\text{MINIMO}(\text{in } c : \text{campo}, \text{ in } t : \text{tab}) \longrightarrow \text{res} : \text{dato}$	
$\text{res} \leftarrow \text{min}(\text{dameColumna}(c, t.\text{registros}))$	$\theta(\text{Cardinal}(t.\text{registros}))$
	<hr/>
	$O(\text{Cardinal}(t.\text{registros}))$
$\text{MAXIMO}(\text{in } c : \text{campo}, \text{ in } t : \text{tab}) \longrightarrow \text{res} : \text{dato}$	
$\text{res} \leftarrow \text{max}(\text{dameColumna}(c, t.\text{registros}))$	$\theta(\text{Cardinal}(t.\text{registros}))$

	<hr/>
	$O(\text{Cardinal}(t.\text{registros}))$
DAMECOLUMNA(in $c : \text{campo}$, <i>in</i> $cr : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{Conj}(\text{dato})$	
$itcr \leftarrow \text{CrearItConj}(cr);$	$\theta(1)$
$res \leftarrow \text{vacio}();$	$\theta(1)$
while HaySiguiente($itcr$) do	$\theta(\text{Cardinal}(cr))$
if $\neg \text{Pertenece}(res, \text{Siguiente}(itcr))$ then	$\theta(????)$
$\text{AgregarRapido}(res, \text{Siguiente}(itcr))$	
end if	
$\text{Avanzar}(itcr);$	$\theta(1)$
end while	
	<hr/>
	$O(\text{calcular})$
MISMOSTIPOS(in $r : \text{registro}$, <i>in</i> $t : \text{tab}$) $\longrightarrow res : \text{bool}$	
$res \leftarrow \text{True};$	$\theta(1)$
$itconjClaves \leftarrow \text{CrearItConj}(r.\text{ClavesDicc});$	$\theta(1)$
while HaySiguiente($itconjClaves$) do	$\theta(1)$
$val1 \leftarrow \text{tipo?}(\text{Obtener}(r, \text{Siguiente}(itconjClaves)))$	$\theta(\text{Cardinal}(t.\text{registros}))$
$val2 \leftarrow \text{tipoCampo}(\text{Siguiente}(itconjClaves), t)$	$\theta(1)$
$res \leftarrow res \wedge val1 = val2$	$\theta(1)$
$\text{Avanzar}(cr);$	$\theta(1)$
end while	
	<hr/>
	$O(\text{calcular})$

1.4 Algoritmos operaciones auxiliares

2 Base de Datos

2.1 Interfaz

se explica con BASE

usa

géneros nat, string, tabla, registro, campo, dato

Operaciones

TABLAS(**in** $b : \text{base}$) $\longrightarrow res : \text{conj}(\text{string})$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{nombre}(t)\}$

Descripción: Devuelve el nombre de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se retorna res por copia, por ser un tipo basico.

DAMETABLA(**in** $b : \text{base}$) $\longrightarrow res : \text{tabla}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{claves}(t)\}$

Descripción: Devuelve un conjunto de campos que son claves en la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve un iterador al conjunto claves por referencia.

HAYJOIN?(**in** $t1 : \text{string}$, $in\ t2 : \text{string}$, $in\ t : \text{base}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{indices}(t)\}$

Descripción: Devuelve un conjunto de los índices de la tabla ingresada por parametro.

Complejidad: $O(\text{calcular})$

Aliasing: Se devuelve res por referencia y no es modificable.

CAMPOJOIN(**in** $t1 : \text{string}$, $in\ t2 : \text{string}$, $in\ t : \text{base}$) $\longrightarrow res : \text{itConjTrie}(\text{campo})$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{campos}(t)\}$

Descripción: Devuelve un conjunto a los campos de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia.

NUEVADB() $\longrightarrow res : \text{base}$

Pre $\equiv \{\text{True}\}$

Post $\equiv \{res =_{\text{obs}} \text{nuevaDB}()\}$

Descripción: Crea una base sin tablas.

Complejidad: $O(\text{calcular})$

AGREGARTABLA(**in** $t : \text{tabla}$, $in\ b : \text{base}$)

Pre $\equiv \{b.0=b \wedge \text{nombre}(t) \notin \text{tablas}(b) \wedge \text{Vacio?}(t.\text{registros})\}$

Post $\equiv \{\text{agregarTabla}(t\ b.0)\}$

Descripción: Agrega una tabla a la base de datos.

Complejidad: $O(\text{calcular})$

Aliasing: Agrega tabla por referencia.

INSERTARENTRADA(**in** $reg : \text{registro}$, $in\ t : \text{string}$, $in\ b : \text{base}$)

Pre $\equiv \{b.0=b \wedge t \in \text{tablas}(b) \wedge_L \text{puedoInsertar?}(\text{dameTabla}(t)\ reg)\}$

Post $\equiv \{\text{insertarEntrada}(rt\ b.0)\}$

Descripción: Inserta el registro a la tabla que corresponde al string pasado por parametro.

Complejidad: $O(\text{calcular})$

BORRAR(**in** $cr : \text{registro}$, $in\ t : \text{string}$, $in\ b : \text{base}$)

Pre $\equiv \{b.0=b \wedge t \in \text{tablas}(b) \wedge \#(cr.\text{DiccClaves})\}$

Post $\equiv \{\text{borrar}(cr\ t\ b.0)\}$

Descripción: Borra los registros que cumplan el criterio cr pasado por parametro.

Complejidad: $O(\text{calcular})$

GENERARVISTAJOIN(**in** $t1 : \text{string}$, $in\ t2 : \text{string}$, $in\ c : \text{campo}$, $in\ b : \text{base}$)

Pre $\equiv \{b.0=b \wedge t1 \sqsubseteq t2 \wedge \{t1\ t2\} \subseteq \text{tablas}(b) \wedge_L (c \in \text{dameTabla}(t1\ b).\text{diccClaves} \wedge c \in \text{dameTabla}(t2\ b).\text{diccClaves})\}$

Post $\equiv \{\text{generarVistaJoin}(cr, t, b.0)\}$

Descripción: Borra los registros que cumplan el criterio cr pasado por parametro.

Complejidad: $O(\text{calcular})$

BORRARJOIN(**in** $t1 : \text{string}$, $in\ t2 : \text{string}$, $in\ b : \text{base}$)

Pre $\equiv \{b.0=b \wedge \text{hayJoin?}(t1\ t2\ b)\}$

Post $\equiv \{\text{borrarJoin}(t1\ t2\ b.0)\}$

Descripción: Borra correspondiente a los nombres de tablas, pasados por parametro.

Complejidad: $O(\text{calcular})$

REGISTROS(**in** $t : \text{string}$, $in\ b : \text{base}$) $\longrightarrow res : \text{conj}(\text{registro})$

Pre $\equiv \{t \in \text{tablas}(b)\}$

Post $\equiv \{res =_{\text{obs}} \text{registros}(t\ b)\}$

Descripción: Retorna el conjunto de registros correspondientes al nombre de tabla pasado por parametro

Complejidad: $O(\text{calcular})$

Aliasing: Se retorna el conjunto de registros por referencia.

VISTAJOIN(**in** $t1 : \text{string}$, **in** $t2 : \text{string}$, **in** $b : \text{base}$) $\longrightarrow res : \text{conj}(\text{registro})$

Pre $\equiv \{\{t1 \ t2\} \subseteq \text{tablas}(b) \wedge \text{hayJoin?}(t1 \ t2 \ b)\}$

Post $\equiv \{res =_{\text{obs}} \text{vistaJoin}(t1 \ t2 \ b)\}$

Descripción: Retorna el conjunto de registros correspondientes al nombre de tabla pasado por parametro

Complejidad: $O(\text{calcular})$

Aliasing: Se retorna el conjunto de registros por referencia.

CANTIDADDEACCESOS(**in** $t : \text{string}$, **in** $b : \text{base}$) $\longrightarrow res : \text{nat}$

Pre $\equiv \{t \in \text{tablas}(b)\}$

Post $\equiv \{res =_{\text{obs}} \text{cantidadDeAccesos}(t \ b)\}$

Descripción: Retorna la cantidad de modificaciones correspondientes al nombre de tabla pasado por parametro.

Complejidad: $O(\text{calcular})$

Aliasing: Se retorna res por referencia.

TABLAMAXIMA(**in** $b : \text{base}$) $\longrightarrow res : \text{string}$

Pre $\equiv \{\neg \emptyset?(\text{tablas}(b))\}$

Post $\equiv \{res =_{\text{obs}} \text{tablaMaxima}(t \ b)\}$

Descripción: Retorna el nombre de la tabla con la mayor cantidad de modificaciones.

Complejidad: $O(\text{calcular})$

Aliasing: Se retorna el nombre de la tabla por referencia.

ENCONTRARMAXIMO(**in** $t : \text{string}$, **in** $ct : \text{conj}(\text{string})$, **in** $b : \text{base}$) $\longrightarrow res : \text{string}$

Pre $\equiv \{\{t\} \cup ct \subseteq \text{tablas}(b)\}$

Post $\equiv \{res =_{\text{obs}} \text{tablaMaxima}(t \ b)\}$

Descripción: Retorna ...

Complejidad: $O(\text{calcular})$

Aliasing: Se retorna el nombre de la tabla por referencia.

BUSCAR(**in** $\text{criterio} : \text{registro}$, **in** $t : \text{string}$, **in** $b : \text{base}$) $\longrightarrow res : \text{conj}(\text{registro})$

Pre $\equiv \{t \in \text{tablas}(b)\}$

Post $\equiv \{res =_{\text{obs}} \text{tablaMaxima}(t \ b)\}$

Descripción: Retorna ...

Complejidad: $O(\text{calcular})$

Aliasing: Se retorna el nombre de la tabla por referencia.

2.2 Representación

se representa con Base

donde estr es tupla $\langle \text{Tablas} : \text{DiccTrie}(\text{Campo}, \text{info_tabla}) \rangle$

donde info_tabla es tupla $\langle \# \text{Accesos} : \text{nat},$

$\text{TActual} : \text{tabla},$

$\text{Joins} : \text{DiccTrie}(\text{string } \text{info_join}) \rangle$

donde info_join es tupla $\langle R : \text{nat},$

$R\text{borrados} : \text{Conj}(\text{registro}),$

$R\text{agregados} : \text{Conj}(\text{registro}),$

$\text{campoJ} : \text{campo},$

$\text{campoT} : \text{tipo},$

$\text{Join} : \text{Conj}(\text{registro}) \rangle$

Invariante de representación

1. El Nombre de la tabla es un String acotado.
2. Indices es un arreglo de tamaño 2, que aloja el Indice correspondiente segun el orden de creacion.
3. Para toda Dato que es clave en Indice, su significado llamemoslo sign esta incluido en Registros.
- 4.

Función de abstracción

$$\begin{aligned}
 \text{Abs} : \widehat{\text{sistema}} s &\longrightarrow \widehat{\text{CampusSeguro}} && \{\text{Rep}(s)\} \\
 (\forall s : \widehat{\text{sistema}}) & \\
 \text{Abs}(s) \equiv cs : \widehat{\text{CampusSeguro}} & \mid s.\text{campus} =_{\text{obs}} \text{campus}(cs) \wedge \\
 s.\text{estudiantes} =_{\text{obs}} \text{estudiantes}(cs) \wedge & \\
 s.\text{hippies} =_{\text{obs}} \text{hippies}(cs) \wedge & \\
 s.\text{agentes} =_{\text{obs}} \text{agentes}(cs) \wedge & \\
 ((\forall n : \text{nombre}) s.\text{hippies}.\text{definido}(n) \Rightarrow_{\text{L}} s.\text{hippies}.\text{obtener}(n) =_{\text{obs}} \text{posEstYHippie}(n, cs) \vee & \\
 (\forall n : \text{nombre}) s.\text{estudiantes}.\text{definido}(n) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(n) =_{\text{obs}} \text{posEstYHippie}(n, cs)) & \\
 (\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{pos} =_{\text{obs}} \text{posAgente}(pl, cs)) & \\
 (\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{cantSanciones} =_{\text{obs}} \text{cantSanciones}(pl, cs)) & \\
 (\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{cantCapturas} =_{\text{obs}} \text{cantCapturas}(pl, cs)) &
 \end{aligned}$$

2.3 Algoritmos

TABLAS(in <i>b</i> : estr) \longrightarrow <i>res</i> : ConjTrie(string)	
<i>res</i> \leftarrow <i>b.tablas.DiccClaves</i>	O(1)
	<hr/>
	O(1)
HAYJOIN?(in <i>t1</i> : string , <i>in</i> <i>t2</i> : string , <i>in</i> <i>b</i> : estr) \longrightarrow <i>res</i> : bool	
<i>res</i> \leftarrow Pertenece?(<i>b</i> .Joins, $\langle t1, t2 \rangle$) \vee Pertenece?(<i>b</i> .Joins, $\langle t2, t1 \rangle$)	O(1)
	<hr/>
	O(1)
CAMPOJOIN(in <i>t1</i> : string , <i>in</i> <i>t2</i> : string , <i>in</i> <i>b</i> : estr) \longrightarrow <i>res</i> : campo	
if Pertenece?(<i>b</i> .Joins, $\langle t1, t2 \rangle$) then	O(1)
<i>res</i> \leftarrow Obtener(<i>e</i> .Joins, $\langle t1, t2 \rangle$).campoJ	O(1)
else	
<i>res</i> \leftarrow Obtener(<i>e</i> .Joins, $\langle t2, t1 \rangle$).campoJ	O(1)
end if	
	<hr/>
	O(1)
NUEVADB() \longrightarrow <i>res</i> : estr	
<i>res</i> \leftarrow $\langle vacio(), vacio() \rangle$	O(1)
	<hr/>
	O(1)
AGREGARTABLA(in <i>t</i> : tabla , <i>in</i> <i>b</i> : estr)	
<i>info_tabla</i> \leftarrow $\langle t.cantidadDeAccesos, t \rangle$	O(1)
Definir(<i>b</i> .tablas, nombre(<i>t</i>), <i>info_tabla</i>)	O(1)
Falta hacer algo?	
	<hr/>
	O(1)
INSERTARENTRADA(in <i>reg</i> : registro , <i>in</i> <i>t</i> : string , <i>in</i> <i>b</i> : estr)	
<i>T_actual</i> \leftarrow Obtener(<i>b</i> .tablas, <i>t</i>).TActual	O(1)
agregarRegistro(<i>reg</i> , <i>T_actual</i>)	O(1)
Falta hacer algo?	
	<hr/>
	O(1)
BORRAR(in <i>cr</i> : registro , <i>in</i> <i>t</i> : string , <i>in</i> <i>b</i> : estr)	
<i>T_actual</i> \leftarrow Obtener(<i>b</i> .tablas, <i>t</i>).TActual	O(1)
borrarRegistro(<i>r</i> , <i>T_actual</i>)	O(1)
Falta hacer algo?	
	<hr/>
	O(1)
GENERARVISTAJOIN(in <i>t1</i> : string , <i>in</i> <i>t2</i> : string , <i>in</i> <i>c</i> : campo , <i>in/out</i> <i>b</i> : estr)	
Join \leftarrow vacio()	
<i>T_actual1</i> \leftarrow Obtener(<i>b</i> .tablas, <i>t1</i>).Tactual	O(1)
<i>T_actual2</i> \leftarrow Obtener(<i>b</i> .tablas, <i>t2</i>).Tactual	O(1)
if Definido?(<i>T_actual1</i> .Indices, <i>c</i>) \wedge Definido?(<i>T_actual2</i> .Indices, <i>c</i>) then	
<i>ind1</i> \leftarrow Obtener(<i>T_actual1</i> .Indices, <i>c</i>)	
<i>ind2</i> \leftarrow Obtener(<i>T_actual2</i> .Indices, <i>c</i>)	
if tipoCampo(<i>T_actual1</i> , <i>c</i>) then	
<i>itvalores</i> \leftarrow CreaItConjNat(<i>ind1</i> .PorNat.DiccClaves)	
while HaySiguiente(<i>itvalores</i>) do	

```

    r1 ← Obtener(ind1.PorNat, Siguiente(itvalores))
    r2 ← Obtener(ind2.PorNat, Siguiente(itvalores))
    cj1 ← AgregarRapido(vacio(), r1)
    cj2 ← AgregarRapido(vacio(), r2)
    nuevor ← combinarRegistros(c, cj1, cj2)
    AgregarRapido(Join, DameUno(nuevor))
    Avanzar(itvalores)
  end while
else
  itvalores ← CrearItConjString(ind1.PorString.DiccClaves)
  while HaySiguiente(itvalores) do
    r1 ← Obtener(ind1.PorString, Siguiente(itvalores))
    r2 ← Obtener(ind2.PorString, Siguiente(itvalores))
    cj1 ← AgregarRapido(vacio(), r1)
    cj2 ← AgregarRapido(vacio(), r2)
    nuevor ← combinarRegistros(c, cj1, cj2)
    AgregarRapido(Join, DameUno(nuevor))
    Avanzar(itvalores)
  end while
end if
else
  cjr1 ← T_actuall.registros
  cjr2 ← T_actuall.registros
  Join ← combinarRegistros(c, cjr1, cjr2)
end if
info_join ← ⟨0, vacio(), vacio(), c, tipoCampo(T_actuall, c), Join⟩
Definir(b.Joins, ⟨t1, t2⟩, info_join)

```

O(1)

```

BORRARJOIN(in t1 : string, in t2 : string, in/out b : estr)
  if Pertenece?(b.Joins, ⟨t1, t2⟩) then
    Borrar(b.Joins, ⟨t1, t2⟩)
  else
    if Pertenece?(b.Joins, ⟨t2, t1⟩) then
      Borrar(b.Joins, ⟨t2, t1⟩)
    end if
  end if
end if

```

O(1)