



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

## Trabajo Práctico 2: Diseño

Primer cuatrimestre - 2016

Algoritmos y Estructuras de Datos II

### Grupo 22

Integrante	LU	Correo electrónico
BENZO, Mariano	198/14	marianobenzo@gmail.com
FARIAS, Mauro	821/13	farias.mauro@hotmail.com
GUTTMAN, Martin	686/14	mdg_92@yahoo.com.ar

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

<b>1. Tabla</b>	<b>2</b>
1.1. Interfaz . . . . .	2
1.2. Representación . . . . .	5
1.3. Algoritmos . . . . .	6
1.4. Algoritmos operaciones auxiliares . . . . .	14
<b>2. Base de Datos</b>	<b>14</b>
2.1. Interfaz . . . . .	14
2.2. Representación . . . . .	16
2.3. Algoritmos . . . . .	18

# 1 Tabla

## 1.1 Interfaz

se explica con TABLA

usa

géneros

DiccString(string, alfa), DiccNat(nat, beta), Nat, String, Dato, Campo, Tipo, Registro, ConjString, ConjNat.

### Operaciones

NOMBRE(**in**  $t : \text{tab}$ )  $\longrightarrow res : \text{string}$

**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.

CLAVES(**in**  $t : \text{tab}$ )  $\longrightarrow res : \text{ConjString}(\text{campo})$

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

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

**Descripción:** Devuelve un conjunto de campos clave en la tabla ingresada por parametro.

**Complejidad:**  $O(1)$

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

INDICES(**in**  $t : \text{tab}$ )  $\longrightarrow res : \text{ConjString}(\text{campo})$

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

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

**Descripción:** Devuelve un conjunto campos con los que se crearon los indices.

**Complejidad:**  $O(1)$

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

CAMPOS(**in**  $t : \text{tab}$ )  $\longrightarrow res : \text{ConjString}(\text{campo})$

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

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

**Descripción:** Devuelve un conjunto de todos los campos de la tabla.

**Complejidad:**  $O(1)$

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

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

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

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

**Descripción:** Devuelve el tipo del campo c en la tabla.

**Complejidad:**  $O(1)$

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

REGISTROS(**in**  $t : \text{tab}$ )  $\longrightarrow res : \text{itConj}(\text{registro})$

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

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

**Descripción:** Devuelve un conjunto a los registros de la tabla.

**Complejidad:**  $O(1)$

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

**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:**  $O(1)$

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

**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:**  $O(\text{Log}(n))$

**Aliasing:** Agrega el registro r por referencia.

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

**Pre**  $\equiv \{t\_0 = t \wedge \#(\text{campos}(r)) = 1 \wedge_{\text{L}} \text{Siguiente}(\text{CrearIt}(\text{campos}(\text{crit}))) \in \text{claves}(t)\}$

**Post**  $\equiv \{\text{borrarRegistro}(r, t\_0)\}$

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

**Complejidad:**  $O(n)$

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

**Pre**  $\equiv \{t\_0 = t \wedge \text{puedeIndexar}(c, t)\}$

**Post**  $\equiv \{\text{indexar}(c, t\_0)\}$

**Descripción:** Crea un indice en base al campo de crit.

**Complejidad:**  $O(n)$

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 respectos a los registros existentes, para los campos clave en la tabla pasada por parametro.

**Complejidad:**  $O(n)$

**Aliasing:** Retorna res por referencia, no es modificable.

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:**  $O(1)$

**Aliasing:** Retorna res por referencia, no es modificable.

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:**  $O(L + \text{Log}(n))$

**Aliasing:** Retorna res por referencia.

**MAXIMO**(in  $c : \text{campo}$ , 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:**  $O(L + \text{Log}(n))$

**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:**  $O(1)$

**Aliasing:** Retorna res por referencia, no es modificable.

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

**Pre**  $\equiv \{True\}$

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

**Descripción:** Devuelve el conjunto de registros de la tabla, que coinciden con los valores de r.

**Complejidad:**  $O(\text{Cardinal}(cj))$

**Aliasing:** Retorna res por referencia.

**HAYCOINCIDENCIA**(in  $r : \text{registro}$ , in  $cjc : \text{Conj}(\text{campo})$ , in  $cjr : \text{Conj}(\text{registro})$ )  $\longrightarrow res : \text{bool}$

**Pre**  $\equiv \{True\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{hayCoincidencia}(r, cjc, cjr)\}$

**Descripción:** Retorna true si algun registro del conjunto cjr, coincide con r en todos los valores de los campos de cjc.

**Complejidad:**  $O(\text{Cardinal}(cjr))$

**Aliasing:** Retorna res por referencia, no es modificable.

**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:**  $O(\text{Cardinal}(cj1) + \text{Cardinal}(cj2))$

**Aliasing:** Retorna res por referencia, es modificable.

**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:**  $O(\text{Cardinal}(cj) * \text{Log}(\text{Cardinal}(cj)))$

**Aliasing:** Retorna res por referencia, no es modificable.

**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:**  $O(1)$

**Aliasing:** Retorna res por referencia, no es modificable.

## 1.2 Representación

se representa con *Tabla*

donde *tab* es  $\text{tupla}\langle \text{Nombre} : \text{String},$   
 $\text{Registros} : \text{Conj}(\text{Registro}),$   
 $\text{Campos} : \text{DiccString}(\text{Campo}, \text{Tipo}),$   
 $\text{Claves} : \text{ConjString}(\text{Campo}),$   
 $\text{IndiceS} : \text{tupla}\langle \text{CampoI} : \text{campo},$   
 $\text{EnUso} : \text{bool},$   
 $\text{Indice} : \text{DiccString}(\text{string}, \text{Conj}(\text{ItConj}(\text{Registro}))),$   
 $\text{Min} : \text{Dato},$   
 $\text{Max} : \text{Dato}\rangle$   
 $\text{IndiceN} : \text{tupla}\langle \text{CampoI} : \text{campo},$   
 $\text{EnUso} : \text{bool},$   
 $\text{Indice} : \text{DiccNat}(\text{nat}, \text{Conj}(\text{ItConj}(\text{Registro}))),$   
 $\text{Min} : \text{Dato},$   
 $\text{Max} : \text{Dato}\rangle$   
 $\text{\#Accesos} : \text{Nat}\rangle$

### Invariante de representación

1. *t.Claves* esta incluido o es igual a *t.Campos*.
2. *t.Nombre* es un string acotado.
3. Para todo registro *r* de *t.Registros*, entonces *Campos(r)* es igual al *t.Campos*.
4. Para todo registro *r* de *t.Registros* y para todo campo *c* de *Campos(r)*, entonces  $\text{Tipo?}(\text{Significado}(r, c))$  es igual  $\text{Significado}(t.Campos, c)$ .
5. Si *t.IndiceS.EnUso* es true y *t.IndiceS.CampoI* pertenece a *t.Campos*, entonces para todo Dato *d*, si  $\text{Definido?}(t.IndiceS.Indice, d)$  es true, entonces  $\text{Significado}(t.IndiceS.Indice, d)$  esta incluido o es igual a *t.Registros*.
6. Si *t.IndiceS.EnUso* es true y *t.IndiceS.CampoI* pertenece a *t.Campos*, entonces para todo registro *r* de *t.Registros* entonces  $\text{Definido?}(t.IndiceS.Indice, \text{Significado}(r, t.IndiceS.CampoI))$  es true y *r* pertenece a  $\text{Significado}(t.IndiceS.Indice, \text{Significado}(r, t.IndiceS.CampoI))$ .
7. Lo anterior tambien aplica para *t.IndiceN.Indice*
8. El valor de *e.#Accesos* debe ser la cantidad de registros agregados, la cantidad de registros borrados

### Función de abstracción

$\text{Abs} : \widehat{\text{tab}} s \longrightarrow \widehat{\text{Tabla}} \quad \{\text{Rep}(s)\}$

$(\forall s : \widehat{\text{tab}})$   
 $\text{Abs}(s) \equiv t : \widehat{\text{Tabla}} \mid s.\text{Nombre} =_{\text{obs}} \text{nombre}(t) \wedge s.\text{Claves} =_{\text{obs}} \text{claves}(t) \wedge$   
 $s.\text{Indices} =_{\text{obs}} \text{indices}(t) \wedge s.\text{Registros} =_{\text{obs}} \text{registros}(t) \wedge s.\text{Campos}.\text{DiccClaves} =_{\text{obs}} \text{campos}(t)$   
 $\wedge s.\text{\#Accesos} =_{\text{obs}} \text{cantidadDeAccesos}(t) \wedge$   
 $((\forall c : \text{campo}) \text{Definido?}(s.\text{Campos}, c) \Rightarrow_{\text{L}} \text{Significado}(s.\text{Campos}, c) =_{\text{obs}} \text{tipoCampo}(c, t))$

### 1.3 Algoritmos

NOMBRE( <b>in</b> $t : \text{tab}$ ) $\longrightarrow res : \text{string}$ $res \leftarrow t.Nombre$	O(1)
	<hr/>
	O(1)
CLAVES( <b>in</b> $t : \text{tab}$ ) $\longrightarrow res : \text{ConjString}(\text{campo})$ $res \leftarrow t.Claves$	O(1)
	<hr/>
	O(1)
INDICES( <b>in</b> $t : \text{tab}$ ) $\longrightarrow res : \text{ConjString}(\text{campo})$ $\text{ConjString}(\text{campo})res \leftarrow \text{vacio}();$ <b>if</b> $t.IndiceS.EnUso$ <b>then</b> AgregarRapido( $res, t.IndiceS.CampoI$ ) <b>end if</b> <b>if</b> $t.IndiceN.EnUso$ <b>then</b> AgregarRapido( $res, t.IndiceN.CampoI$ ) <b>end if</b>	O(1)
	<hr/>
	O(1)
CAMPOS( <b>in</b> $t : \text{tab}$ ) $\longrightarrow res : \text{ConjString}(\text{campo})$ $res \leftarrow Claves(t.Campos)$	O(1)
	<hr/>
	O(1)
TIPOCAMPO( <b>in</b> $c : \text{campo}$ , <i>in</i> $t : \text{tab}$ ) $\longrightarrow res : \text{Tipo}$ $res \leftarrow Significado(t.Campos, c)$	O(1)
	<hr/>
	O(1)
REGISTROS( <b>in</b> $t : \text{tab}$ ) $\longrightarrow res : \text{Conj}(\text{registro})$ $res \leftarrow t.registros$ Se retorna el conjunto por referencia	$\theta(1)$
	<hr/>
	$\theta(1)$
CANTDEACCESOS( <b>in</b> $t : \text{tab}$ ) $\longrightarrow res : \text{nat}$ $res \leftarrow t.cantDeAccesos$	$\theta(1)$
	<hr/>
	$\theta(1)$

NUEVATABLA(**in** nombre : string, in claves : conj(campo), in columnas : registro)  $\longrightarrow$  res :  
tab

Conj(registro) Registros $\leftarrow$ Vacio()	O(1)
DiccString(campo, tipo) Campos $\leftarrow$ Vacio()	O(1)
ConjString(campo) Claves_ $\leftarrow$ Vacio()	O(1)
Dato d $\leftarrow$ Obtener(columnas, Siguiente(CrearIt(claves)))	
IndiceS $\leftarrow$ < Siguiente(CrearIt(claves)), False, Vacio(), d, d >	O(1)
IndiceN $\leftarrow$ < Siguiente(CrearIt(claves)), False, Vacio(), d, d >	O(1)
#Acessos $\leftarrow$ 0	O(1)
res $\leftarrow$ < nombre, Registros, Campos, Claves_, IndiceS, IndiceN, 0 >	O(1)
itcampos $\leftarrow$ crearItConj(Campos(columnas))	O(1)
<b>while</b> HaySiguiente(itcampos) <b>do</b>	O(1)
Este while es O(1) y se debe a que el cardinal de campos a iterar es acotado.	
valor $\leftarrow$ Significado(columnas, Siguiente(itcampos))	O(1)
DefinirRapido(res.Campos, Siguiente(itcampos), Tipo?(valor))	O(1)
Avanzar(itcampos)	O(1)
<b>end while</b>	
itclaves $\leftarrow$ crearItConj(claves)	O(1)
Paso los campos claves de un conj al conjString	
<b>while</b> HaySiguiente(itclaves) <b>do</b>	O(1)
Esto se debe a que # de campos a iterar es acotada.	
AgregarRapido(res.Claves, Siguiente(itclaves))	O(L)
Avanzar(itcampos)	O(1)
<b>end while</b>	
Donde L es la longitud de la cadena string mas larga y acotada del parametro claves.	
	<hr/>
	$\theta(1)$



AGREGARREGISTRO(**in**  $r$  : registro, *in/out*  $t$  : tab)

Para poder acceder al registro, guardo el iterador al elemento

itConj(Registro) nuevo  $\leftarrow$  AgregarRapido( $t$ .Registros, $r$ )  $\theta(1)$

$t$ .#Accesos++  $\theta(1)$

Este registro debe ser indexado, si estos indices existieran.

**if**  $t$ .IndiceS.EnUso **then**

Para el indice traigo el valor del campo con el que se creo.

valor  $\leftarrow$  Significado( $r$ ,  $t$ .IndiceS.CampoI)  $\theta(L)$

**if** Definido?( $t$ .IndiceS.Indice, valor) **then**  $\theta(L)$

viejo  $\leftarrow$  Significado( $t$ .IndiceS.Indice, valor)  $\theta(L)$

AgregarRapido(viejo, nuevo)  $\theta(L)$

**else**

viejo  $\leftarrow$  Vacio()  $\theta(1)$

AgregarRapido(viejo, nuevo)  $\theta(L)$

Definir( $t$ .IndiceS.Indice, ValorString(valor), viejo)  $\theta(L)$

Como ingresamos un nuevo valor, actualizamos el min y max

**if**  $t$ .IndiceS.Min  $\geq$  valor **then**  $\theta(L)$

$t$ .IndiceS.Min  $\leftarrow$  Copiar(valor)  $\theta(L)$

**end if**

**if** valor  $\geq t$ .IndiceS.Max **then**  $\theta(L)$

$t$ .IndiceS.Max  $\leftarrow$  Copiar(valor)  $\theta(L)$

**end if**

**end if**

Donde L es la longitud del valor string agregado a  $t$ .IndiceS

**end if**

**if**  $t$ .IndiceN.EnUso **then**

Para el indice traigo el valor del campo con el que se creo.

valor  $\leftarrow$  Significado( $r$ ,  $t$ .IndiceN.CampoI)  $\theta(\text{Log}(n))$

**if** Definido?( $t$ .IndiceN.Indice, valor) **then**  $\theta(\log(n))$

viejo  $\leftarrow$  Significado( $t$ .IndiceN.Indice, valor)  $\theta(1)$

AgregarRapido(viejo, nuevo)  $\theta(1)$

**else**

viejo  $\leftarrow$  Vacio()  $\theta(1)$

AgregarRapido(viejo, nuevo)  $\theta(1)$

Definir( $t$ .IndiceN.Indice, valor, viejo)  $\theta(\text{Log}(n))$

Como ingresamos un nuevo valor, actualizamos el min y max

**if**  $t$ .IndiceN.Min  $\geq$  valor **then**  $\theta(1)$

$t$ .IndiceS.Min  $\leftarrow$  Copiar(valor)  $\theta(1)$

**end if**

**if** valor  $\geq t$ .IndiceS.Max **then**  $\theta(1)$

$t$ .IndiceN.Max  $\leftarrow$  Copiar(valor)  $\theta(1)$

**end if**

**end if**

**end if**

Donde n es la cantidad de registros y L es la longitud maxima de un valor String.

---

$\theta(\text{Log}(n))$

```

BORRARREGISTRO(in crit : registro, in/out t : tab)
  Campo c ← Siguiente(CrearIt(Campos(crit)))           $\theta(1)$ 
  Dato valor ← Copiar(Significado(crit, c))            $\theta(L)$ 
  Sabemos que c esta incluido en claves(t)
  si hay un indice de un campo clave borrar es  $O(\log(n))$  u  $O(L)$  en peor caso
  if t.IndiceS.EnUso  $\wedge$  t.IndiceS.CampoI=c then
    String S ← ValorString(valor)
    if Definido?(t.IndiceS.Indice, S) then
      itConj(registro) itr ← Significado(t.IndiceS.Indice, S)     $\theta(1)$ 
      Borro el registro del conjunto de registros
      EliminarSiguiente(itr)                                      $\theta(1)$ 
      Borro el registro del indice,
      Borrar(t.IndiceS.Indice, S)                                 $\theta(1)$ 
      if t.IndiceS.Min=valor then
        t.IndiceS.Min← Minimo(Claves(t.IndiceS))
      end if
      if t.IndiceS.Max=valor then
        t.IndiceS.Max← Maximo(Claves(t.IndiceS))
      end if
      Dado que el crit solo tiene un campo clave, siempre elimino un registro. t.#Accesos++
    end if
  else
    if t.IndiceN.EnUso  $\wedge$  t.IndiceN.CampoI=c then
      N ← ValorNat(valor)
      if Definido?(t.IndiceN.Indice, valor) then
        itConj(registro) itr ← Significado(t.IndiceN.Indice, valor)
                                                                     $\theta(1)$ 
        EliminarSiguiente(itr)                                      $\theta(1)$ 
        Borrar(t.IndiceN.Indice, valor)                             $\theta(1)$ 
        if t.IndiceN.Min=valor then                                 $\theta(1)$ 
          t.IndiceS.Min← Minimo(Claves(t.IndiceN))                 $\theta(\log(n))$ 
        end if
        if t.IndiceN.Max=valor then                                 $\theta(1)$ 
          t.IndiceS.Max← Maximo(Claves(t.IndiceN))                 $\theta(\log(n))$ 
        end if
        Dado que el crit solo tiene un campo clave, siempre elimino un registro. t.#Accesos++
      end if
    else
      No hay indice para el campo clave de criterio pasado por parametro
      itConj(registro) cr ← CrearItConj(t.registros)               $\theta(1)$ 
      while HaySiguiente(cr) do                                     $\theta(\text{Cardinal}(t.\text{registros}))$ 
        valorR ← Significado(Siguiente(cr), c)                     $\theta(1)$ 
        if valorR=valor then
          EliminarSiguiente(cr);                                    $\theta(1)$ 
          Dado que el crit solo tiene un campo clave, siempre elimino solo un registro.
          t.#Accesos++
        end if
        Avanzar(cr)                                                 $\theta(1)$ 
      end while
      La complejidad de la operacion borrar depende de si hay o no indices
      para el campo del crit pasado por parametro.
      En caso de que exista dicho indice, en peor caso eliminar es  $O(\log(n))$  u  $O(L)$ 
    end if
  end if

```

siendo  $n$  la cantidad de registros de la tabla pasada por parametro y  $L$  el valor string mas largo definido en la tabla.

En caso de no haber tal indice es  $O(n)$ .

**end if**

**end if**

---

$O(n)$

```

INDEXAR(in c : campo, in/out t : tab)
  if tipoCampo(c,t) then
    t.IndiceN.EnUso  $\leftarrow$  True
  else
    t.IndiceS.EnUso  $\leftarrow$  True
  end if
  cr  $\leftarrow$  CrearItConj(t.registros)
  if tipoCampo?(c,t) then
    while HaySiguiente(cr) do
      Dato valor  $\leftarrow$  Significado(Siguiente(cr), c)
      itConj(registro) itr  $\leftarrow$  CrearItConj(Siguiente(cr))
      if Definido?(t.IndiceN.Indice, valor) then
        regviejos  $\leftarrow$  Significado(indC, valor)
        AgregarRapido(regviejos, itr)
      else
        conj(registro) nuevo  $\leftarrow$  Vacio()
        AgregarRapido(nuevo, itr)
        DefinirRapido(t.IndiceN.Indice, valor, nuevo)
      end if
      Avanzar(cr)
    end while
  end if
  if  $\neg$ tipoCampo?(c,t) then
    while HaySiguiente(cr) do
      Dato valor  $\leftarrow$  Significado(Siguiente(cr), c)
      itConj(registro) itr  $\leftarrow$  CrearItConj(Siguiente(cr))
      if Definido?(t.IndiceS.Indice, valor) then
        regviejos  $\leftarrow$  Significado(indC, valor)
        AgregarRapido(regviejos, itr)
      else
        conj(registro) nuevo  $\leftarrow$  Vacio()
        AgregarRapido(nuevo, itr)
        DefinirRapido(t.IndiceN.Indice, valor, nuevo)
      end if
      Avanzar(cr)
    end while
  end if

```

---

 $\theta(1)$ 

```

PUEDOINSERTAR?(in r : registro, in t : tab)  $\longrightarrow$  res : bool
  res  $\leftarrow$  compatible(r,t)  $\wedge$   $\neg$ hayCoincidencia( r, r.ClavesDicc, registros(t) )
   $\theta(\text{calcular})$ 

```

---

 $\theta(\text{calcular})$ 

```

COMPATIBLE(in r : registro, int t : tab)  $\longrightarrow$  res : bool
  bool valor  $\leftarrow$  True
  if Cardinal(campos(r))=Cardinal(t.Campos.DiccClaves) then
    itcampos  $\leftarrow$  CrearItString(t.Campos.DiccClaves)
    while valor  $\wedge$  HaySiguiente(itcampos) do
      Campo c  $\leftarrow$  Siguiente(itcampos)
      valor  $\leftarrow$  Definido?(r, c)
    end while

```

 $\theta(1)$ 
 $\theta(1)$ 
 $\theta(1)$

<b>else</b>	
valor $\leftarrow$ False	$\theta(1)$
<b>end if</b>	
res $\leftarrow$ valor $\wedge_L$ mismosTipos(r,t)	$\theta(1)$
El costo del While es $O(1)$ ya que la cantidad de campos de la tabla es acotado	
	<hr/>
	$O(1)$
MINIMO( <b>in</b> $c : \text{campo}$ , $in\ t : \text{tab}$ ) $\longrightarrow res : \text{dato}$	
Si hay indice en el campo c, debe ser de complejidad $O(1)$	
<b>if</b> t.IndiceS.EnUso $\wedge$ t.IndiceS.CampoI=c <b>then</b>	
Sabemos que hay un indice string para el campo c	
res $\leftarrow$ t.IndiceS.Min	$\theta(\text{Cardinal}(t.\text{registros}))$
<b>else</b>	
<b>if</b> t.IndiceN.EnUso $\wedge$ t.IndiceN.CampoI=c <b>then</b>	
Sabemos que hay un indice string para el campo c	
res $\leftarrow$ t.IndiceN.Min	$\theta(\text{Cardinal}(t.\text{registros}))$
<b>end if</b>	
<b>end if</b>	
	<hr/>
	$O(1)$
MAXIMO( <b>in</b> $c : \text{campo}$ , $in\ t : \text{tab}$ ) $\longrightarrow res : \text{dato}$	
Si hay indice en el campo c, debe ser de complejidad $O(1)$	
<b>if</b> t.IndiceS.EnUso $\wedge$ t.IndiceS.CampoI=c <b>then</b>	
Sabemos que hay un indice string para el campo c	
res $\leftarrow$ t.IndiceS.Max	$\theta(\text{Cardinal}(t.\text{registros}))$
<b>else</b>	
<b>if</b> t.IndiceN.EnUso $\wedge$ t.IndiceN.CampoI=c <b>then</b>	
Sabemos que hay un indice string para el campo c	
res $\leftarrow$ t.IndiceN.Max	$\theta(\text{Cardinal}(t.\text{registros}))$
<b>end if</b>	
<b>end if</b>	
	<hr/>
	$O(1)$
PUEDEINDEXAR( <b>in</b> $c : \text{campo}$ , $in\ t : \text{tab}$ ) $\longrightarrow res : \text{bool}$	
<b>if</b> TipoCampo(c, t) <b>then</b>	
res $\leftarrow$ $\neg(t.\text{IndiceN.EnUso})$	
<b>else</b>	
res $\leftarrow$ $\neg(t.\text{IndiceS.EnUso})$	
<b>end if</b>	
	<hr/>
	$O(1)$
HAYCOINCIDENCIA( <b>in</b> $r : \text{registro}$ , $in\ cc : \text{ConjString}(\text{campo})$ , $in\ cr : \text{Conj}(\text{registro})$ ) $\longrightarrow res : \text{bool}$	
itcr $\leftarrow$ CrearItConj(cr)	$\theta(1)$
res $\leftarrow$ false	$\theta(1)$
<b>while</b> HaySiguiente(itcr) <b>do</b>	$\theta(\text{Cardinal}(cr))$
res $\leftarrow$ coincideAlguno(r,cc,Siguiente(itcr)) $\vee$ res	$\theta(1)$
Avanzar(itcr)	$\theta(1)$
<b>end while</b>	
	<hr/>
	$O(\text{Cardinal}(cr))$
COINCIDENCIAS( <b>in</b> $crit : \text{registro}$ , $in\ cr : \text{Conj}(\text{registro})$ ) $\longrightarrow res : \text{Conj}(\text{registro})$	

Conj(registro) salida $\leftarrow$ Vacio()	
Debemos comparar todos los registros de cr.	
y agregarlos al conjunto de registros salida	
itcr $\leftarrow$ CrearItConj(cr)	
<b>while</b> HaySiguiente(cr) <b>do</b>	$\theta(\text{Cardinal}(\text{cr}))$
<b>if</b> coincidenTodos(crit,campos(crit),Siguiente(itcr)) <b>then</b>	$\theta(1)$
AgregarRapido(salida,Siguiente(itcr))	$\theta(1)$
<b>end if</b>	
Avanzar(itcr);	$\theta(1)$
<b>end while</b>	
	<hr/>
	$O(\text{Cardinal}(\text{cr}))$
COMBINARREGISTROS( <b>in</b> c : campo, in cr1 : Conj(registro), in cr2 : Conj(registro)) $\rightarrow$ res : Conj(registros)	
itcr1 $\leftarrow$ CrearItConjString(cr1)	$\theta(1)$
copiacr2 $\leftarrow$ Copiar(cr2)	$\theta(\text{Cardinal}(\text{cr2}))$
<b>while</b> HaySiguiente(itcr1) <b>do</b>	$\theta(\text{Cardinal}(\text{cr1}))$
combinarTodos(c,Siguiente(itcr1),copiacr2)	$\theta(1)$
Avanzar(itcr1)	$\theta(1)$
<b>end while</b>	
res $\leftarrow$ copiacr2	$\theta(1)$
	<hr/>
	$O(\text{Cardinal}(\text{cr1}))$
DAMECOLUMNA( <b>in</b> c : campo, in cr : Conj(registro)) $\rightarrow$ res : Conj(dato)	
Conj(Dato) cj $\leftarrow$ vacio();	$\theta(1)$
La idea es no agregar el mismo dato dos veces, para eso uso un conj del tipo de dato de la columna para hacer consulta.	
<b>if</b> Cardinal(cr) $\geq 1$ <b>then</b>	$\theta(1)$
Tvalor $\leftarrow$ Tipo?(Significado(Siguiente(CrearIt(cr))), c)	$\theta(1)$
<b>if</b> Tvalor <b>then</b>	
ConjLog(nat) cj $\leftarrow$ Vacio()	$\theta(1)$
<b>else</b>	
ConjString(string) cj $\leftarrow$ Vacio()	$\theta(1)$
<b>end if</b>	
itcr $\leftarrow$ CrearItConj(cr)	$\theta(1)$
cjd $\leftarrow$ Vacio()	
<b>while</b> HaySiguiente(itcr) <b>do</b>	$\theta(\text{Cardinal}(\text{cr}))$
Dato data $\leftarrow$ Significado(Siguiente(itcr), c)	
<b>if</b> Tvalor <b>then</b>	
<b>if</b> $\neg$ Pertenece?(cj, valorNat(data)) <b>then</b>	$\theta(\text{Log}(n))$
AgregarRapido(cjd, data)	$\theta(1)$
AgregarRapido(cj, valorNat(data))	$\theta(1)$
<b>end if</b>	
<b>else</b>	
<b>if</b> $\neg$ Pertenece?(cj, valorString(data)) <b>then</b>	$\theta(1)$
AgregarRapido(cjd, data)	$\theta(1)$
AgregarRapido(cj, valorString(data))	$\theta(1)$
<b>end if</b>	
<b>end if</b>	
Avanzar(itcr);	$\theta(1)$
<b>end while</b>	
<b>end if</b>	

$res \leftarrow cjd$

Si la columna es de tipo String, la complejidad es  $O(n)$ , en caso de ser de tipo Nat la complejidad es  $O(n \log(n))$ .

El cardinal de res es la cantidad de datos distintos.

---

$O(n \log(n))$

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

$valor \leftarrow \text{True}$

$\theta(1)$

$itconjClaves \leftarrow \text{CrearItConj}(r.ClavesDicc)$

$\theta(1)$

**while**  $valor \wedge \text{HaySiguiente}(itconjClaves)$  **do**

$\theta(1)$

$val1 \leftarrow \text{tipo?}(\text{Significado}(r, \text{Siguiente}(itconjClaves)))$

$\theta(1)$

$val2 \leftarrow \text{tipoCampo}(\text{Siguiente}(itconjClaves), t)$

$\theta(1)$

$valor \leftarrow (val1 = val2)$

$\theta(1)$

$\text{Avanzar}(cr);$

$\theta(1)$

**end while**

$res \leftarrow valor$

---

$O(1)$

## 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 indices de la tabla ingresada por parametro.

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

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

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

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

**Pre**  $\equiv \{b_0 = b \wedge t \in \text{tablas}(b) \wedge_L \text{puedoInsertar?}(\text{dameTabla}(t)\ reg)\}$

**Post**  $\equiv \{\text{insertarEntrada}(r\ t\ b_0)\}$

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

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

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

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

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

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

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

**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.

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

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

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

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

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

**Aliasing:** Se retorna res por referencia.

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

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

**Post**  $\equiv \{res =_{\text{obs}} \text{tablaMaxima}(tb)\}$

**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.

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

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

**Post**  $\equiv \{res =_{\text{obs}} \text{tablaMaxima}(tb)\}$

**Descripción:** Retorna ...

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

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

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

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

**Post**  $\equiv \{res =_{\text{obs}} \text{tablaMaxima}(tb)\}$

**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  $\text{tupla}\langle \text{TablaMaxima} : \text{Tmax},$

$\text{Tablas} : \text{DiccTrie}(\text{NombreTabla}; \text{info\_tabla})\rangle$

donde info\_tabla es  $\text{tupla}\langle \text{TActual} : \text{tabla},$

$\text{Joins} : \text{DiccTrie}(\text{NombreTabla}; \text{info\_join})\rangle$

donde info\_join es  $\text{tupla}\langle \text{Rcambios} : \text{Cola}(\text{DatoCambio}),$

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

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

$\text{JoinS} : \text{ConjTrie}(\text{registro}),$

$\text{ValoresS} : \text{ConjTrie}(\text{String}),$

$\text{JoinN} : \text{ConjNat}(\text{registro}),$

$\text{ValoresN} : \text{ConjNat}(\text{Nat})\rangle$

donde Tmax es  $\text{tupla}\langle \text{NomTabla} : \text{NombreTabla},$

$\text{\#Modif} : \text{Nat}\rangle$

donde DatoCambio es  $\text{tupla}\langle \text{Reg} : \text{Registro},$

$\text{NomOrigen} : \text{NombreTabla},$

$\text{Accion} : \text{Bool}\rangle$

### **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**

## 2.3 Algoritmos

TABLAS( <b>in</b> $b : \text{estr}$ ) $\longrightarrow res : \text{ConjTrie}(\text{string})$ $res \leftarrow b.\text{tablas}.\text{DiccClaves}$	$O(1)$
	<hr/>
	$O(1)$
HAYJOIN?( <b>in</b> $t1 : \text{string}$ , $in$ $t2 : \text{string}$ , $in$ $b : \text{estr}$ ) $\longrightarrow res : \text{bool}$ $res \leftarrow \text{Definido?}(\text{Obtener}(b, t1).\text{Joins}, t2) \leftarrow \text{Definido?}(\text{Obtener}(b, t2).\text{Joins}, t1)$	$O(1)$
	<hr/>
	$O(1)$
CAMPOJOIN( <b>in</b> $t1 : \text{string}$ , $in$ $t2 : \text{string}$ , $in$ $b : \text{estr}$ ) $\longrightarrow res : \text{campo}$ $res \leftarrow \text{Obtener}(\text{Obtener}(b, t1).\text{Joins}, t2).\text{campoJ}$	$O(1)$
	<hr/>
	$O(1)$
NUEVADB() $\longrightarrow res : \text{estr}$ String s Nat n $\leftarrow 0$ $res \leftarrow \langle \langle s, 0 \rangle, \text{vacio}() \rangle$	$O(1)$
	<hr/>
	$O(1)$
AGREGARTABLA( <b>in</b> $t : \text{tabla}$ , $in/out$ $b : \text{estr}$ ) $\text{info\_tabla} \leftarrow \langle t.\text{cantidadDeAccesos}, t, \text{vacio}() \rangle$ Definir( $b.\text{tablas}$ , nombre( $t$ ), $\text{info\_tabla}$ )	$O(1)$
	$O(1)$
	<hr/>
	$O(1)$
INSERTARENTRADA( <b>in</b> $reg : \text{registro}$ , $in$ $t : \text{string}$ , $in/out$ $b : \text{estr}$ ) Obtenemos la tabla es $O(1)$ porque su nombre esta acotado. $\text{info\_tabla infoT} \leftarrow \text{Obtener}(b.\text{tablas}, t).\text{TActual}$ Agrego el registro a la tabla. $\text{Tabla T} \leftarrow \text{infoT}.\text{TActual}$ agregarRegistro( $reg$ , $T$ ) Ahora si hay Joins actualizo la informacion temporal de cada Join. <b>if</b> $\neg \emptyset?(\text{infoT}.\text{Joins})$ <b>then</b> $\text{ItConjString}(\text{String}) \text{ itNomTab} \leftarrow \text{CrearIt}(\text{Claves}(\text{infoT}.\text{Joins}))$ <b>while</b> HaySiguiente?( $\text{itNomTab}$ ) <b>do</b> $\text{info\_join infoJ} \leftarrow \text{Obtener}(\text{infoT}.\text{Joins}, \text{Siguiente}(\text{NomTab}))$ Encolar( $\text{infoJ}.\text{Rcambios}$ , $\langle reg, \text{Siguiente}(\text{NomTab}), true \rangle$ ) Encolar es $O(L)$ porque se copia un registro con su cantidad de campos acotada y los valores string copiarlos tiene costo $O(L)$ , siendo $L$ el valor string mas largo. Avanzar( $\text{itClaves}$ ) <b>end while</b> <b>end if</b> <b>if</b> CantidadDeAccesos( $T$ ) $b.\text{TablaMaxima}.\#Modif$ <b>then</b> $b.\text{TablaMaxima}.\text{NomTabla} \leftarrow \text{Copiar}(\text{Nombre}(T))$ $b.\text{TablaMaxima}.\#Modif \leftarrow \text{Copiar}(\text{CantidadDeAccesos}(T))$ <b>end if</b>	$O(1)$ Por referencia $O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(L)$ $O(L)$ $O(1)$ $O(L)$ $O(L)$ $O(1)$
	<hr/>
	$O(T*L + \text{Log}(n))$ siendo $n$ la cantidad de r
BORRAR( <b>in</b> $cr : \text{registro}$ , $in$ $t : \text{string}$ , $in/out$ $b : \text{estr}$ )	

```

info.tabla infoT ← Obtener(b.tablas, t).TActual          O(1) por ref
Tabla T ← infoT.TActual                                O(1) por ref
NombreTabla NomTab ← NombreTabla(T)
La eliminacion en primera etapa depende de si hay
joins con la tabla pasada por parametro
if  $\neg \emptyset?$ (Claves(infoT.Joins)) then                O(1)
    Creo el iterador, para navegar los nombres de tablas con los que tiene Join
    itNom ← CrearIt(Claves(infoT.Joins))                O(1)
    while HaySiguiente?(itNom) do                      O(Cant de tablas)
        info_join infoJ ← Siguiente(itNom)              O(1)
        Verifico si el Join esta creado en base al
        campo del cr pasado por parametro.
        if infoJ.campoJ=DameUno(Campos(cr)) then        O(1)
            Entonces solo actualizo la cola temporal del join
            Registro reg ← Buscar(cr, T)                  O(L + Log(n))
            Encolar(infoJ.Rcambios,  $\langle reg, Siguiente(itNom), False \rangle$ )
                                                         O(L)
        else
            Si el campo del criterio de borrado, es distinto que el campo del Join.
            Primero busco que registros coinciden con el criterio
            en el peor caso con complejidad O(cantidad de registros de t) sin indices.
            Conj(Registro) cjc ← Buscar(cr, T)            O(L + Log(n))
            ItConj(Registro) itReg ← CrearIt(cjc)
            Luego agrego estos registros a la cola temporal de cambios del join
            while HaySiguiente?(itReg) do
                Encolar(infoJ.Rcambios,  $\langle Siguiente(itReg), NomTab, False \rangle$ )
                                                         O(L)
                Avanzar(itReg)                             O(1)
            end while
        end if
    end while
end if
Habiendo actualizado las colas temporales de los joins con los
registros que cumplen con el criterio de borrado
de los joins correspondientes.
Elimino del conjunto de registros, aquellos que cumplen el criterio de borrado.
Siendo n la cantidad de registros de t y T la cantidad de tablas en la base
En peor caso con costo O(n)
borrarRegistro(r, T_actual)                             O(T*L + n)
if CantidadDeAccesos(T) b.TablaMaxima.#Modif then
    b.TablaMaxima.NomTabla ← Copiar(Nombre(T))          O(L)
    b.TablaMaxima.#Modif ← Copiar(CantidadDeAccesos(T))
                                                         O(1)
end if


---


O(T*L + n)

```

**GENERARVISTAJOIN**(**in** t1 : string, **in** t2 : string, **in** c : campo, **in/out** b : estr)
 Join ← vacio()
 T\_actual1 ← Obtener(b.tablas, t1).Tactual O(1)
 T\_actual2 ← Obtener(b.tablas, t2).Tactual O(1)
 **if** Pertenece?(Indices(T\_actual1), c)  $\wedge$  Pertenece?(Indices(T\_actual1), c) **then**
 ind1 ← Obtener(T\_actual1.Indices, c) O(1)
 O(calcular)

```

if tipoCampo(T_actual1, c) then
  ConjNat(Nat) cjNat  $\leftarrow$  vacio()
  itvalores  $\leftarrow$  CrearItConjNat(ind1.PorNat.DiccClaves)
  while HaySiguiente(itvalores) do
    Registro r  $\leftarrow$  Obtener(ind1.PorNat, Siguiente(itvalores))
    Nat n  $\leftarrow$  ValorNat(Obtener(r, c))
    if  $\neg$  Pertenece?(cjNat, n) then
      AgregarRapido(cjNat, r)
    end if
    Avanzar(itvalores)
  end while
  ind2  $\leftarrow$  Obtener(T_actual2.Indices, c)
  itvalores  $\leftarrow$  CrearItConjNat(ind2.PorNat.DiccClaves)
  while HaySiguiente(itvalores) do
    Registro r  $\leftarrow$  Obtener(ind2.PorNat, Siguiente(itvalores))
    Nat n  $\leftarrow$  ValorNat(Obtener(r, c))
    if  $\neg$  Pertenece?(cjNat, n) then
      AgregarRapido(cjNat, r)
    end if
    Avanzar(itvalores)
  end while
else
  itvalores  $\leftarrow$  CrearItConjString(ind1.PorString.DiccClaves)
  while HaySiguiente(itvalores) do
    r1  $\leftarrow$  Obtener(ind1.PorString, Siguiente(itvalores))
    r2  $\leftarrow$  Obtener(ind2.PorString, Siguiente(itvalores))
    cj1  $\leftarrow$  AgregarRapido(vacio(), r1)
    cj2  $\leftarrow$  AgregarRapido(vacio(), r2)
    nuevor  $\leftarrow$  combinarRegistros(c, cj1, cj2)
    AgregarRapido(Join, DameUno(nuevor))
    Avanzar(itvalores)
  end while
  end if
else
  cjr1  $\leftarrow$  T_actual1.registros
  cjr2  $\leftarrow$  T_actual1.registros
  Join  $\leftarrow$  combinarRegistros(c, cjr1, cjr2)
end if
info_join  $\leftarrow$   $\langle 0, vacio(), vacio(), c, tipoCampo(T\_actual1, c), Join \rangle$ 
Definir(b.Joins,  $\langle t1, t2 \rangle$ , info_join)

```

---

O(1)

```

BORRARJOIN(in t1 : string, in t2 : string, in/out b : estr)
if Pertenece?(b.Joins,  $\langle t1, t2 \rangle$ ) then
  Borrar(b.Joins,  $\langle t1, t2 \rangle$ )
else
  if Pertenece?(b.Joins,  $\langle t2, t1 \rangle$ ) then
    Borrar(b.Joins,  $\langle t2, t1 \rangle$ )
  end if
end if

```

---

O(1)