# Análisis de Lenguajes de Programación
# Trabajo Práctico Final

Rinaldi, Lautaro

31 de Marzo de 2016

# 1 Motivación

En internet se encuentran muy pocos intepretes de Álgebra relaciónal, y la mayor parte de estos no son libres, no hacen el trabajo necesario y, de cierta forma, funcionan mal.
El objetivo de este trabajo es brindar una herramientas para que estudiantes puedan aprender como se trabaja con el algebra relaciónal y compararlo, utilizando otro software, con otros lenguajes de manejo de datos.

# 2 Base teórica y descripción del lenguaje

En bases de datos el álgebra relaciónal es un lenguaje de manejo de datos de tipo algebraico y procedimental, esto significa que trabaja con operaciones definidas entre relaciónes (lo que llamamos tablas).
El usuario da una secuencia de operaciones sobre las tablas que posee y espera el resultado. Estas operaciones pueden se unarias o binarias y producen una nueva tabla como resultado.

## 2.1 Operadores base

Los operadores básicos de álgebra relaciónal son lo suficientemente poderosos para expresar cualquier consulta. En los próximos items se dará una breve descripción y un ejemplo de cada operador base, estos ejemplos estarán basados en el formato que acepta el programa.

### 2.1.1 Selección

Dado un predicado lógico sobre las columnas se retorna todas las filas (o tuplas) de la tabla original que lo satisfacen.
Este predicado puede estar formado, en el caso de *Strins* por igualdades o desigualdades entre una columna y una palabra o frase, o entre una columna y otra columna. En el casos de los $Int$ las comparaciones pueden ser igualdad($=$), diferencia($\neq$), mayoreo($>$), minoreo($>$) o una combinación de éstas ($\leq$ y $\geq$). Además en ambos casos se pueden combinar comparaciones utilizando $AND$, $OR$ y $NOT$.

Ejemplo:
Sea la relación R la siguiente:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

El resultado de $Seleccionar\ OR\ A =' a1'\ B \neq' b2'$ R es:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |

### 2.1.2 Proyección

Dada una lista de columnas correspondientes a una tabla y la tabla en sí retorna la tabla en la cual cada tupla tiene solo las columnas pasadas.
Ejemplo:
Sea la relación R la siguiente:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

El resultado de $Proyectar\ [A, C]$ R es:

| A | C |
|---|---|
| a1 | c1 |
| a2 | c2 |

### 2.1.3 Renombre

En muchas ocasiones al operar dos tablas surgen problemas con los nombres. En una tabla cada columna es representada por un par ($NombreTabla$, $NombreColumna$) lo cual no soluciona el problema de que al operar una tabla consigo misma haya un conflicto con las columnas repetidas, para esto está el renombre que además de cambiar el nombre de la tabla en sí, también cambia la primer componente de cada identificador de columna, a partir de esto surge otro problema, al renombrar la tabla antes mencionada todas las columnas tendrán otra con el mismo identificador, por lo que también se creo el renombre de columnas a pesar de no ser un operador de álgebra relacional.

Ejemplo:
Sea la relación R la siguiente:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

El resultado de *Renombrar* S R es exactamente la misma tabla, pero cada columna tendra otro nombre el la primer componente de la tupla identidicatoria de la columna. En R las columnas son $[(R, A), (R, B), (R, C)]$ y en S $[(S, A), (S, B), (S, C)]$.

El resultado de *RenombrarCol* [(R.A,D)] R es:

| D | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

### 2.1.4 Unión

Para unir 2 tablas es necesario que posean las mismas columnas en el mismo órden.
Al unir dos tablas se genera una tercera tabla en la que aparecen todas las tuplas de las tablas operadas, sin embargo, a pesar de que una tupla aparezca en las dos tablas aparecerá solo una vez en la tabla resultante.

Ejemplo:
Sean las relaciónes R y S las siguientes:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

| A | B | C |
|---|---|---|
| a4 | b4 | c4 |
| a2 | b2 | c2 |
| a4 | b5 | c3 |

El resultado de *Unir* R S es:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |
| a4 | b4 | c4 |
| a4 | b5 | c3 |

### 2.1.5 Resta

Para restar 2 tablas es necesario que posean las mismas columnas en el mismo órden.
Al restar una tabla de otra se genera una tercer tabla en la que aparecen todas las tuplas de la primera tabla que no aparezca en la segunda.
Ejemplo:
Sean las relaciónes R y S las siguientes:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

| A | B | C |
|---|---|---|
| a4 | b4 | c4 |
| a2 | b2 | c2 |
| a4 | b5 | c3 |

El resuntado de *Restar* R S es:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a3 | b3 | c3 |

### 2.1.6 Producto Cartesiano

El producto cartesiano de dos tablas genera una tercer tabla con la concatenación de las columnas de ambas y en la cual cada tupla de la primer tabla aparece concatenada con cada componente de la segunda.
Ejemplo:
Sean las relaciónes R y S las siguientes:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

| D | E | F |
|---|---|---|
| d4 | e4 | f4 |
| d2 | e2 | f2 |
| d4 | e5 | f3 |

El resultado de *Cartesiano* R S es:

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| a1 | b1 | c1 | d4 | e4 | f4 |
| a1 | b1 | c1 | d2 | e2 | f2 |
| a1 | b1 | c1 | d4 | e5 | f3 |
| a2 | b2 | c2 | d4 | e4 | f4 |
| a2 | b2 | c2 | d2 | e2 | f2 |
| a2 | b2 | c2 | d4 | e5 | f3 |
| a3 | b3 | c3 | d4 | e4 | f4 |
| a3 | b3 | c3 | d2 | e2 | f2 |
| a3 | b3 | c3 | d4 | e5 | f3 |

## 2.2 Operadores compuestos

Si bien el conjunto operaciones del lenguaje puede expresar cualquier consulta, sin embargo, para facilitar algunas que realizadas con operadores base serían tediosas y aumentarían en gran medida la posibilidad de error se crearon funciones compuestas.

### 2.2.1 Intersección

Para intersecar 2 tablas es necesario que posean las mismas columnas en el mismo órden.
Al intersecar dos tablas se genera una tercera en la que aparecen todas las tuplas que aparecen en ambas relaciónes.
El resultado de *intersecar* R S es el mismo que el de *restar* R *restar* R S.
Ejemplo:
Sean las relaciones R y S las siguientes:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

| A | B | C |
|---|---|---|
| a2 | b2 | c2 |
| a4 | b4 | c4 |
| a4 | b5 | c3 |

El resultado de *Intersecar* R S es:

| A | B | C |
|---|---|---|
| a2 | b2 | c2 |

### 2.2.2 Producto Natural

El producto natural de dos tablas genera una tercer tabla en la que cada tupla es la concatenación de las tupas de cada tabla, con la limitación de que las columnas que se repiten tienen el mismo valor y solo aparecen una vez.
Realizar la operación *Natural* R S es equivalente a *Proyectar* list Seleccionar *com Cartesiano* R S ,donde list es la lista en la que aparece la unión (y no concatenación de las columnas de R y S) y *com* es el predicado resultante de igualar cada columna que este en las dos tablas por ejemplo si las columnas x e y se repiten el predicado sería
AND R.x=S.x R.y=S.y .
Ejemplo:
Sean las relaciones R y S las siguientes:

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |
| a3 | b3 | c3 |

| D | E | B |
|---|---|---|
| d4 | e4 | b1 |
| d2 | e2 | b2 |
| d4 | e5 | b5 |

El resultado de *Natural* R S es:

| A | B | C | D | E |
|---|---|---|---|---|
| a1 | b1 | c1 | d4 | e4 |
| a2 | b2 | c2 | d2 | e2 |

### 2.2.3 División

La división de se da entre dos tablas la tabla a dividir y la tabla divisora, las columnas de esta última deben estar contenidas es las columnas de la tabla a dividir. El resultado de esta operación es una tabla que por columnas tienen las tablas de la primer tabla que no estan en la segunda y como componentes a cada "subtupla" (la generada por la resta de columnas anterior) de la primer tabla que se relacionen con todas las tuplas de la segunda tabla.
*dividir* R S es equivalente a
*Restar* C1 *Proyectar* col *Restar Cartesiano* C1 S R,donde col son las columas de R que no estan en S, C1 es *Proyectar* col R.

Ejemplo:
Sean R y S las siguientes relaciones:

| A | B |
|---|---|
| a1 | b1 |
| a1 | b2 |
| a1 | b3 |
| a2 | b1 |
| a2 | b2 |
| a2 | b3 |
| a2 | b4 |
| a3 | b2 |
| a3 | b3 |
| a3 | b5 |
| a4 | b2 |

| B |
|---|
| b1 |
| b2 |
| b3 |

El resultado de *Dividir* R S es:

| A |
|---|
| a1 |
| a2 |

# 3 Sintaxis

## 3.1 Sintaxis de un archivo

$< archivo >$ ::= $< tabla > < EOL > [< archivo >]$
$< tabla >$ ::= '@' $< Nombre >$ '(' $< columnas >$ ')' $< EOL > [< contenidos >]$
$< columnas >$ ::= $< Nombre >$ '/' $< tipo >$ [ ',' $< columnas >$]
$< contenidos >$ ::= $< contenido > < EOL > [< contenidos >]$
$< contenido >$ ::= $< campo >$ [',' $< contenido >$]
$< campo >$ ::= ' ' ' $< string >$ ' ' ' | ' " ' $< string >$ ' " ' | $< int >$
$< Nombre >$ es cualquier string que no sea una palabra reservada o $< string >$.
$< string >$ es cualquier string que no contenga un $< EOL >$.
$< int >$ es cualquier entero.

### 3.2 Sintaxis de las operaciones

$< operacion >$ ::= $< Nombre >$
| "*Seleccionar*" $< Predicado >$ $< Operacion >$
| "*Proyectar*" $< NombreColumnas >$ $< Operacion >$
| "*Renombrar*" $< Nombre >$ $< Operacion >$
| "*RenombrarCol*" $< CR >$ $< Operacion >$
| "*Unir*" $< Operacion >$ $< Operacion >$
| "*Restar*" $< Operacion >$ $< Operacion >$
| "*Instersecar*" $< Operacion >$ $< Operacion >$
| "*Carteciano*" $< Operacion >$ $< Operacion >$
| "*Natural*" $< Operacion >$ $< Operacion >$
| "*Dividir*" $< Operacion >$ $< Operacion >$

$< CR >$ ::= '[' '(' $< NombreColumna >$ ',' $< Nombre >$')' ',' [$< CR >$] ]'

$< NombreColumna >$ ::= $< Nombre >$ '.' $< Nombre >$
| $< Nombre >$

$< Predicado >$ ::= NOT $< Predicado >$
| $< Predicado >$ AND $< Predicado >$
| $< Predicado >$ OR $< Predicado >$
| $< NombreColumna >$ $OP$ $< ArgComp >$

$< ArgComp >$ ::= $< NombreColumna > | < String > | < Int >$

$< Nombre >$ es cualquier string que no sea una palabra reservada o $< string >$.
$< string >$ es cualquier string que no contenga un $< EOL >$.
$< int >$ es cualquier entero.

# 4 Decisiones de diseño

Una de las decisiones de diseño fundamental fue la utilización o no de clave en las relaciones, lo que permite la clave es darle más relevancia a ciertos campos de la relación (columnas) y con esto hacer que la unicidad de las tuplas dependa solo de estos, es decir, viendo las relaciones como en la matemática las limita a funciones. Se decidió no utilizar claves ya que además de no generar demasiada diferencia deja muchas cosas a criterio del programador (sobre todo como tratar las tablas en cada operación) , ya que no hay teoría sobre eso.
Otra decisión importante fue que hacer con el TAD de tablas ya que existía un inconveniente a la hora de representar el contenido.
En un inicio se tuvo en cuenta la utilización de un solo TAD para representar las tablas, el problema fue que, o bien, el parser se tenía que encargar de una

trasformación importante en la representación de los datos ($Tables$), o bien, la estructura aumentaría a la posibilidad de más errores ya que cada elemento lleva consigo su tipo, lo cual es motivo suficiente para rechazarla para su uso en operaciones($Tablas$), por lo que se eligío hacer dos TAD uno para parsear y corroborar que la tabla este bien formada y otro TAD en el que el tipo lo lleva cada columna y que se utiliza en las operaciones. El inconveniente que surgió de este último es que para facilitar la definición de operaciones, al operar las columnas se vuelven a transformar en filas y después de operarlas estas se vuelven a transformar en columnas.
La diferencia fundamental entre los dos TAD's elegidos es que el primero(Tabla) tiene un formato más cercano a la realidad pero es menos elegante en cuanto al manejo de información mientras que el segundo(Table) es totalmente lo contrario.

# 5 Organización de archivos

Los árboles de sintáxis abstracta fueron escritos en archivos separados dependiendo de que se encarga. AST.hs contiene la definición de las operaciones mientras que en ASTTablas.hs está la definición del de las tablas que se parsean y de las tablas que se operan.
El parser para las tablas (ParseTablas.y ,ParseTablas.hs) y el Parser para las operaciones (Parse.y , Parse.hs) están separados. Ambos se hicieron con Happy y las funciones y definiciones que tienen en común se llaman con el archivo ParseR.hs.
El archivos Tablas.hs se encarga de la de la transformación del Tabla (dentro de Archivo) a Table (dentro de BD) y brinda algunas aplicaciones fundamentales.
Operaciones.hs tiene definida cada una de las operaciones del álgebra relacional e incluye funciones auxiliares que se usan en más de una operación.
PrettiPrinter.hs contiene todas las funciones encargadas de mostrar las tablas de la manera más estética posible.
Main.hs se encarga principalemente del modo interactivo, además de manejar la baja, alta y asignacion de tablas.

# 6 Instalación

Se requiere tener alguna version de ghc instalada (se está probando sobre GHC Versión 7.10.3). Una vez asegurados de poseer una versión de GHC se accede a la carpera contenedora y se escribe

```
cabal install
```

Una vez descargadas todas las dependencias solicitadas e instalalado el paquete, para ejecutar se escribe

```
cabal run
```

# 7 Ejemplos

## 7.1 De la carga de archivos

Los archivos se carga escribiendo :l < *documento* >.
Los archivos palRes, long, noPar, Repe, RepeCol,tiposS,tiposI harán saltar algun error, en cambio spj cargará las tablas correctamente.

## 7.2 De las operaciones

Los ejemplos de operaciones se pueden copiar del archivo EjemploOP.txt de a linea (en el caso en que aparezca la tabla ans es necesario realizar esta operación justo después de la anterior, pues ans representa este resultado.

# 8 Posibles mejoras

## 8.1 En el código

- Mejorar la modularización sería lo ideal, pués como está programado ahora es posible que al necesitar efectuar un mínimo cambio se tenga que modificar gran parte del código.
- Limitar las funciones que se pueden usar cuando se llama un módulo, si esto se implementa bien y junto con la primer mejora haría que al cambiar la estructura los cambios sean no sólo mínimos sino también muy simples.
- División de los archivos en subcarpetas para no exponer información innecesaria, si bien esto está más ligado a la organización que al código en sí, lo afectafia minimamente.
- Que los errores sean más explicativos, sobre todo los de parseo.
- Hacer que ande, si hay alguna manera, en todas las versiones de ghc al momento.

## 8.2 En cuanto a funcionamiento

- Agregar la posibilidad de trabajar con y sin clave, para que el usuario pueda elegir.
- Agregar junto con el primer item la posibilidad de elegir el comportamiento de la clave según la operación, y tener una por defecto.
- Agregar la posibildad de ejecutar operaciones desde un archivo.
- Agregar la posibilidad de definir nuevos tipos como ser pares o estructuras similares a los Data de Haskell además de poder trabajar con números no enteros.
- Extender las operaciones para poder realizar operaciones sobre las columnas como ser agrupamiento, promedio, cálculo de extremos, etc. y con esto dar la posibilidad de definir funciones matemáticas para poder aplicarse a los elementos de las columnas o sobre éstas.
- Dar la posibilidad de imprimir la base de datos, o ciertas tablas, en un archivo.

# 9 Bibliografía


- http://hackage.haskell.org/package/base-4.8.2.0/docs/Prelude.html
- http://hackage.haskell.org/package/base-4.8.2.0/docs/Data-Char.html
- http://hackage.haskell.org/package/base-4.8.2.0/docs/Data-List.html
- http://hackage.haskell.org/package/base-4.8.2.0/docs/Data-Maybe.html
- http://hackage.haskell.org/package/base-4.8.2.0/docs/Data-Either.html
- http://hackage.haskell.org/package/base-4.8.2.0/docs/Control-Monad.html
- http://hackage.haskell.org/package/base-4.8.2.0/docs/Control-Exception.html
- http://hackage.haskell.org/package/control-monad-exception
- http://hackage.haskell.org/package/base-4.8.2.0/docs/Control-Applicative.html
- http://hackage.haskell.org/package/readline
- http://hackage.haskell.org/package/base-4.8.2.0/docs/System-Environment.html
- http://hackage.haskell.org/package/base-4.8.2.0/docs/System-IO.html
- https://downloads.haskell.org/ ghc/6.2/docs/html/libraries/base/Text.PrettyPrint.Hughes
- https://www.haskell.org/cabal/users-guide
- http://www.dsi.fceia.unr.edu.ar/downloads/base_de_datos/Algebrarelaciónal.pdf
- https://es.wikipedia.org/wiki/%C3%81lgebra_relaciónal