Trabajo práctico 1: **Álgebra relacional**

Normativa

Límite de entrega: Domingo 22 de septiembre, 23:59hs. Enviar el zip al mail: `algo2.dc+TP1@gmail.com`
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://campus.exactas.uba.ar`)
Versión: 1.0 del 3 de septiembre de 2019

Enunciado

Una **base de datos** es un sistema para el almacenamiento de información. Se caracteriza por mantener la información de forma estructurada y en un solo lugar. Además están preparadas para guardar grandes volúmenes de información. La estructura de los datos permite hacer consultas sofisticadas, que incluyen filtrar datos, transformarlos y contabilizarlos. Una de las principales utilidades de las bases de datos es la de guardar información de elementos que se encuentran relacionados y poder acceder a estas relaciones de forma muy eficiente. Un ejemplo de elementos relacionados sería guardar información de Proveedores y Productos, donde los Proveedores venden distintos Productos. Las bases de datos permiten hacer consultas del tipo *¿Cuales son los productos que vende el proveedor X?* de forma muy rápida, incluso con una cantidad enorme de datos. El objetivo de este TP es diseñar la interfaz de una **base de datos**.

- Una base de datos tiene **tablas**. Cada tabla suele representar un concepto. Las tablas se identifica con un nombre único.

  Por ejemplo, una base de datos podría contar con tres tablas: Empleados, Proveedores y Productos.

- Cada tabla tiene **campos** o "columnas" que se identifican con un nombre. Cada tabla tiene exactamente un campo que se declara como la **clave** de la tabla. (En los ejemplos de abajo la clave se subraya).

  Por ejemplo, la base de datos podría tener las siguientes tres tablas:

Empleados	campos: {`nombre`, `apellido`, `cuit`}	la clave es el `cuit`.
Proveedores	campos: {`razón_social`, `cuit`}	la clave es el `cuit`.
Productos	campos: {`nombre`, `precio`}	la clave es el `nombre`.

- Un **registro** es una asociación entre nombres de campos y valores. Los valores son siempre **cadenas de caracteres** (*strings*).

  Por ejemplo, {`nombre`: "Mona", `apellido`: "Lisa", `cuit`: "12345678"} es un registro.

- Cada tabla tiene varios registros o "filas". Por ejemplo, las tablas de la base de datos podrían tener los siguientes registros:

  Empleados

`nombre`	`apellido`	`cuit`
Mona	Lisa	12345678
Papá	Noel	12345679
Mona	Simpson	12345680
King	Kong	12345681
King	King	12345682
Mona	Jiménez	12345683
King	Lear	12345684

  Proveedores

`razón_social`	`cuit`
Acme	98765432
Xanadu	98765431

  Productos

`nombre`	`precio`
caramelo	1
chocolatín	1
paleta	2
alfajor	2
alfajor triple	3

  Una tabla no puede tener dos registros que coincidan en el valor del campo clave. Por ejemplo, en la tabla Productos puede haber dos registros con el mismo precio, pero no puede haber dos registros con el mismo nombre.

- Se deben permitir las siguientes operaciones sobre la base de datos:

  1. **Agregar una tabla**, indicando cuáles son sus campos y cuál es el campo clave.
  2. **Eliminar una tabla**.
  3. **Agregar un registro** a una tabla, dándole valor a todos (y solamente a) los campos de la tabla.
  4. **Eliminar un registro** de una tabla, identificándolo por su clave.
  5. **Realizar una consulta**. Las consultas se describen a continuación.

Consultas

Las consultas están basadas en un formalismo conocido como **álgebra relacional**[1]. El resultado de hacer una consulta sobre una base de datos es un conjunto de registros. Las consultas se construyen recursivamente de la siguiente manera:

1. **Tabla completa.** Si t es el nombre de una tabla, `FROM`(t) es la consulta que devuelve todos los registros de la tabla t. Nota: si t no es el nombre de una tabla definida en la base de datos, la consulta deberá comportarse de alguna manera que ustedes deben especificar.

2. **Filtrar por valor de un campo.** Si q es una consulta, c es el nombre de un campo y v es un valor, `SELECT`(q, c, v) es una nueva consulta que devuelve los registros de la consulta q tales que el campo c tiene valor v.

   Por ejemplo, `SELECT(FROM(`EMPLEADOS`), nombre, "Mona")` devuelve el conjunto:

   {`nombre`: "Mona", `apellido`: "Lisa", `cuit`: "12345678"}
   {`nombre`: "Mona", `apellido`: "Simpson", `cuit`: "12345680"}
   {`nombre`: "Mona", `apellido`: "Jiménez", `cuit`: "12345683"}

   Nota: si hay registros que no tienen el campo c, la consulta deberá comportarse de alguna manera que ustedes deben especificar.

3. **Filtrar por coincidencia de campos.** Si q es una consulta y c_1 y c_2 son nombres de campos, `MATCH`(q, c_1, c_2) es una nueva consulta que devuelve los registros de la consulta q tales que los campos c_1 y c_2 tienen el mismo valor.

   Por ejemplo, `MATCH(FROM(`EMPLEADOS`), nombre, apellido)` devuelve el conjunto:

   {`nombre`: "King", `apellido`: "King", `cuit`: "12345682"}

   Nota: si hay registros que no tienen el campo c_1 o que no tienen el campo c_2, la consulta deberá comportarse de alguna manera que ustedes deben especificar.

4. **Proyección.** Si q es una consulta y C es un conjunto de campos `PROJ`(q, C) es una nueva consulta que devuelve los mismos registros que la consulta q, pero que incluyen solamente los campos del conjunto C.

   Por ejemplo, `PROJ(SELECT(FROM(`EMPLEADOS`), nombre, "Mona"), {apellido, cuit})` devuelve el conjunto:

   {`apellido`: "Lisa", `cuit`: "12345678"}
   {`apellido`: "Simpson", `cuit`: "12345680"}
   {`apellido`: "Jiménez", `cuit`: "12345683"}

5. **Renombre.** Si q y es una consulta y c_1 y c_2 son nombres de campos, `RENAME`(q, c_1, c_2) renombra el campo c_1 en el resultado de la consulta q para que pase a llamarse c_2.

   Por ejemplo `RENAME(FROM(`PROVEEDORES`), razón_social, nombre_proveedor)` devuelve el conjunto:

   {`nombre_proveedor`: "Acme", `cuit`: "98765432"}
   {`nombre_proveedor`: "Xanadu", `cuit`: "98765431"}

   Nota: si hay registros que no tienen definido el campo c_1 o que ya tienen definido el campo c_2, la consulta deberá comportarse de alguna manera que ustedes deben especificar.

6. **Intersección.** Si q_1 y q_2 son consultas, `INTER`(q_1, q_2) es una nueva consulta que devuelve la intersección entre los registros de las consultas q_1 y q_2.

   Por ejemplo la consulta:

   `INTER(SELECT(FROM(`EMPLEADOS`), nombre, "Mona"), SELECT(FROM(`EMPLEADOS`), apellido, "Lisa"))`

   devuelve el conjunto:

   {`nombre`: "Mona", `apellido`: "Lisa", `cuit`: "12345678"}

[1] `https://en.wikipedia.org/wiki/Relational_algebra`

7. **Unión.** Si q_1 y q_2 son consultas, UNION(q_1, q_2) es una nueva consulta que devuelve la unión entre los registros de las consultas q_1 y q_2.

   Por ejemplo, la consulta:

   UNION(SELECT(FROM(EMPLEADOS), nombre, "Papá"), SELECT(FROM(PRODUCTOS), precio, 3))

   devuelve el conjunto:

   {nombre: "Papá", apellido: "Noel", cuit: "12345679"}
   {nombre: "alfajor triple", precio: "3"}

   Nota: observar que el resultado de una consulta puede mezclar diferentes "tipos" de registros.

8. **Producto cartesiano.** Si q_1 y q_2 son consultas, PRODUCT(q_1, q_2) es una nueva consulta que devuelve el producto cartesiano entre los registros de las consultas q_1 y q_2.

   Por ejemplo, la consulta:

   PRODUCT(FROM(PROVEEDORES), SELECT(FROM(PRODUCTOS), precio, 2))

   devuelve el conjunto:

   {razón_social: "Acme", cuit: "98765432", nombre: "paleta", precio: "2"}
   {razón_social: "Acme", cuit: "98765432", nombre: "alfajor", precio: "2"}
   {razón_social: "Xanadu", cuit: "98765431", nombre: "paleta", precio: "2"}
   {razón_social: "Xanadu", cuit: "98765431", nombre: "alfajor", precio: "2"}

   Nota: si hay registros provenientes de la consulta q_1 que tienen campos en común con registros provenientes de la consulta q_2, la consulta deberá comportarse de alguna manera que ustedes deben especificar.

Entrega

La entrega consistirá de un único documento digital con la descripción de todos los módulos y sus interfaces. Se debe diseñar el módulo principal (BASEDEDATOS) y todos los módulos auxiliares. De definirse módulos que no pueden ser apropiadamente explicados por un TAD existente, deberá desarrollarse un TAD para describir el comportamiento del mismo. Esto aplica principalmente para el módulo BASEDEDATOS. La única excepción son los módulos disponibles en el Apunte de Módulos Básicos, que se pueden utilizar sin diseñarlos: lista enlazada, pila, cola, vector, diccionario lineal, conjunto lineal y conjunto acotado de naturales.

En este TP se debe diseñar únicamente la **interfaz** de los módulos, que constará de las siguientes partes:

1. **Descripción.** Descripción del papel que cumple el módulo en la solución del problema, cuáles son sus principales utilidades, qué información almacena y cómo se espera que sea usado.
2. **Tipo abstracto ("se explica con ..."). ** Género (TAD) que sirve para explicar las instancias del módulo, escrito en el lenguaje de especificación **formal** de la materia. Pueden utilizar la especificación que se incluye en el apéndice.
3. **Signatura.** Listado de todas las operaciones públicas que provee el módulo, escrita con la notación de módulos de la materia, por ejemplo:

   apilar(**in/out** pila : PILA, **in** x : ELEMENTO)

4. **Contrato.** Precondición y postcondición de todas las operaciones públicas. Las precondiciones de las operaciones deben estar expresadas **formalmente** en lógica de primer orden.

La corrección se realizará con los siguientes criterios en mente:

- **Completitud.** Los módulos deben tener operaciones en la interfaz que permitan operar con ellos de las formas esperadas.
- **Correctitud.** Las precondiciones y postcondiciones de las operaciones deben dar buenas descripciones de los requisitos y efectos de las operaciones. En relación a esto, se debe contar con las funciones y TADs de forma que puedan describirse las operaciones correctamente.
- **Interpretabilidad.** Es fundamental para que un módulo sea usable que su interfaz sea entendible, por lo que se espera que la selección de operaciones permita comprender cómo y porqué se utiliza el módulo.

Se recomienda el uso de los paquetes de LaTeX de la cátedra para lograr una mejor visualización del informe.

La entrega se realizará por mail a la dirección `algo2.dc+tp1@gmail.com`. El documento desarrollado se entregará como un archivo en formato pdf hasta el día 22 de septiembre a las 23:59hs. El mail deberá tener como **Asunto** los números de libreta separados por punto y coma (;). Por ejemplo:

To: algo2.dc+tp1@gmail.com
From: alumno-algo2@dc.uba.ar
Subject: 123/45; 67/8; 910/11; 12/13
Adjunto: tp1.pdf

Sobre el recuperatorio: dado que el TP2 consistirá en extender el TP1, agregando las estructuras de datos y algoritmos necesarios para implementar los módulos diseñados, el **recuperatorio** del TP1 se entregará como parte del informe del TP2.

A. Tipos abstractos de datos

Para escribir formalmente las precondiciones y postcondiciones de las operaciones, pueden utilizar los siguientes tipos abstractos de datos. Si es necesario, pueden extender estos tipos de datos con más operaciones, o definir nuevos tipos abstractos de datos (p.ej. pueden definir el TAD BaseDeDatos si lo necesitan).

Usaremos los siguientes tipos auxiliares. El tipo String es un arreglo dimensionable de caracteres, extendido con una operación $\bullet = \bullet : \text{string} \times \text{string} \to \text{bool}$ para compararlas por igualdad. Cada caracter es un número entre 0 y 255.

- **TAD** Char es Enum(0, 1, .., 255)
- **TAD** String es Arreglo dimensionable(Char)
- **TAD** NombreCampo es String
- **TAD** NombreTabla es String
- **TAD** Valor es String

A.1. Registro

TAD Registro

géneros registro

exporta observadores, generadores

usa NombreCampo, Valor, Conjunto

igualdad observacional

$$(\forall r, r' : \text{registro}) \left(r =_{\text{obs}} r' \iff \left(\begin{array}{l} \text{campos}(r) =_{\text{obs}} \text{campos}(r') \wedge_{\text{L}} \\ (\forall c : \text{nombre\_campo})(c \in \text{campos}(r) \Rightarrow_{\text{L}} \\ \qquad r[c] =_{\text{obs}} r'[c]) \end{array} \right) \right)$$

observadores básicos

campos	: registro	$\longrightarrow$	conj(nombre_campo)	
$\bullet[\bullet]$	: registro r $\times$ nombre_campo c	$\longrightarrow$	valor	$\{c \in \text{campos}(r)\}$

generadores

nuevo	:	$\longrightarrow$	registro
definir	: registro $\times$ nombre_campo $\times$ valor	$\longrightarrow$	registro

axiomas $\forall r : \text{registro}, \forall c, c' : \text{nombre\_campo}, \forall v : \text{valor}$

campos(nuevo)	$\equiv$	$\emptyset$
campos(definir(r, c, v))	$\equiv$	Ag$(c, \text{campos}(r))$
definir$(r, c, v)[c']$	$\equiv$	**if** $c = c'$ **then** v **else** $r[c']$ **fi**

Fin TAD

A.2. Tabla

TAD Tabla

géneros tabla

exporta observadores, generadores

usa NombreCampo, Valor, Conjunto, Registro

igualdad observacional

$$(\forall t, t' : \text{tabla}) \left(t =_{\text{obs}} t' \iff \begin{pmatrix} \text{campos}(t) =_{\text{obs}} \text{campos}(t') \wedge \text{clave}(t) =_{\text{obs}} \text{clave}(t') \wedge \\ \text{registros}(t) =_{\text{obs}} \text{registros}(t') \end{pmatrix} \right)$$

observadores básicos

campos	: tabla	$\longrightarrow$	conj(nombre_campo)
clave	: tabla	$\longrightarrow$	nombre_campo

registros : tabla $\longrightarrow$ conj(registro)

generadores

nueva : conj(nombre_campo) cs × nombre_campo k $\longrightarrow$ tabla $\{k \in cs\}$

insertar : tabla t × registro r $\longrightarrow$ tabla $\{\text{campos}(t) =_{\text{obs}} \text{campos}(r)\}$

borrar : tabla × valor $\longrightarrow$ tabla

otras operaciones

insertarRegistro : nombre_campo k × registro r × conj(registro) rs $\longrightarrow$ conj(registro) $\{k \in \text{campos}(r) \land (\forall r' : \text{registro})(r' \in rs \Rightarrow k \in \text{campos}(r'))\}$

borrarRegistro : nombre_campo k × valor v × conj(registro) rs $\longrightarrow$ conj(registro) $\{(\forall r : \text{registro})(r \in rs \Rightarrow k \in \text{campos}(r))\}$

axiomas $\forall t : \text{tabla}, k : \text{nombre\_campo}, cs : \text{conj(nombre\_campo)}, r : \text{registro}, rs : \text{conj(registro)}, v : \text{valor}$

$\text{campos}(\text{nueva}(cs, k)) \equiv cs$

$\text{campos}(\text{insertar}(t, r)) \equiv \text{campos}(t)$

$\text{campos}(\text{borrar}(t, v)) \equiv \text{campos}(t)$

$\text{clave}(\text{nueva}(cs, k)) \equiv k$

$\text{clave}(\text{insertar}(t, r)) \equiv \text{clave}(t)$

$\text{clave}(\text{borrar}(t, v)) \equiv \text{clave}(t)$

$\text{registros}(\text{nueva}(cs, k)) \equiv \emptyset$

$\text{registros}(\text{insertar}(t, r)) \equiv \text{insertarRegistro}(\text{clave}(t), r, \text{registros}(t))$

$\text{registros}(\text{borrar}(t, v)) \equiv \text{borrarRegistro}(\text{clave}(t), v, \text{registros}(t))$

$\text{insertarRegistro}(k, r, rs) \equiv$ **if** $\emptyset?(rs)$ **then** $\text{Ag}(r, \emptyset)$ **else** **if** $\text{dameUno}(rs)[k] = r[k]$ **then** $\text{Ag}(r, \text{sinUno}(rs))$ **else** $\text{Ag}(\text{dameUno}(rs), \text{insertarRegistro}(k, r, \text{sinUno}(rs)))$ **fi** **fi**

$\text{borrarRegistro}(k, v, rs) \equiv$ **if** $\emptyset?(rs)$ **then** $\emptyset$ **else** **if** $\text{dameUno}(rs)[k] = v$ **then** $\text{sinUno}(rs)$ **else** $\text{Ag}(\text{dameUno}(rs), \text{borrarRegistro}(k, v, \text{sinUno}(rs)))$ **fi** **fi**

Fin TAD

A.3. Consulta

TAD TipoConsulta es Enum(FROM, SELECT, MATCH, PROJ, RENAME, INTER, UNION, PRODUCT)

TAD Consulta

géneros consulta

exporta observadores, generadores

usa NombreCampo, NombreTabla, Valor, Conjunto

igualdad observacional

$$(\forall q, q' : \text{consulta})\ q =_{\text{obs}} q' \iff \text{tipo\_consulta}(q) =_{\text{obs}} \text{tipo\_consulta}(q') \wedge_{\text{L}}$$
$$\left(\begin{array}{lll}
& (\text{tipo_consulta}(q) \in \{\underline{\texttt{FROM}}\} & \Rightarrow_{\text{L}}\ \text{nombre_tabla}(q) =_{\text{obs}} \text{nombre_tabla}(q')) \\
\wedge & (\text{tipo_consulta}(q) \in \{\underline{\texttt{SELECT}}, \underline{\texttt{MATCH}}, \underline{\texttt{RENAME}}\} & \Rightarrow_{\text{L}}\ \text{campo}_1(q) =_{\text{obs}} \text{campo}_1(q')) \\
\wedge & (\text{tipo_consulta}(q) \in \{\underline{\texttt{MATCH}}, \underline{\texttt{RENAME}}\} & \Rightarrow_{\text{L}}\ \text{campo}_2(q) =_{\text{obs}} \text{campo}_2(q')) \\
\wedge & (\text{tipo_consulta}(q) \in \{\underline{\texttt{SELECT}}\} & \Rightarrow_{\text{L}}\ \text{valor}(q) =_{\text{obs}} \text{valor}(q')) \\
\wedge & (\text{tipo_consulta}(q) \in \{\underline{\texttt{PROJ}}\} & \Rightarrow_{\text{L}}\ \text{conj_campos}(q) =_{\text{obs}} \text{conj_campos}(q')) \\
\wedge & (\text{tipo_consulta}(q) \in \{\underline{\texttt{SELECT}}, \underline{\texttt{MATCH}}, \underline{\texttt{PROJ}}, \underline{\texttt{RENAME}}, \underline{\texttt{INTER}}, \underline{\texttt{UNION}}, \underline{\texttt{PRODUCT}}\} & \\
& & \Rightarrow_{\text{L}}\ \text{subconsulta}_1(q) =_{\text{obs}} \text{subconsulta}_1(q')) \\
\wedge & (\text{tipo_consulta}(q) \in \{\underline{\texttt{INTER}}, \underline{\texttt{UNION}}, \underline{\texttt{PROJ}}\} & \Rightarrow_{\text{L}}\ \text{subconsulta}_2(q) =_{\text{obs}} \text{subconsulta}_2(q'))
\end{array}\right)$$

generadores

FROM	: nombre_tabla	$\longrightarrow$	consulta
SELECT	: consulta $\times$ nombre_campo $\times$ valor	$\longrightarrow$	consulta
MATCH	: consulta $\times$ nombre_campo $\times$ nombre_campo	$\longrightarrow$	consulta
PROJ	: consulta $\times$ conj(nombre_campo)	$\longrightarrow$	consulta
RENAME	: consulta $\times$ nombre_campo $\times$ nombre_campo	$\longrightarrow$	consulta
INTER	: consulta $\times$ consulta	$\longrightarrow$	consulta
UNION	: consulta $\times$ consulta	$\longrightarrow$	consulta
PRODUCT	: consulta $\times$ consulta	$\longrightarrow$	consulta

observadores básicos

tipo_consulta	: consulta	$\longrightarrow$	tipo_consulta	
nombre_tabla	: consulta q	$\longrightarrow$	nombre_tabla	$\{\text{tipo\_consulta}(q) \in \{\underline{\texttt{FROM}}\}\}$
campo_1	: consulta q	$\longrightarrow$	nombre_campo	$\{\text{tipo\_consulta}(q) \in \{\underline{\texttt{SELECT}}, \underline{\texttt{MATCH}}, \underline{\texttt{RENAME}}\}\}$
campo_2	: consulta q	$\longrightarrow$	nombre_campo	$\{\text{tipo\_consulta}(q) \in \{\underline{\texttt{MATCH}}, \underline{\texttt{RENAME}}\}\}$
valor	: consulta q	$\longrightarrow$	valor	$\{\text{tipo\_consulta}(q) \in \{\underline{\texttt{SELECT}}\}\}$
conj_campos	: consulta q	$\longrightarrow$	conj(nombre_campo)	$\{\text{tipo\_consulta}(q) \in \{\underline{\texttt{PROJ}}\}\}$
subconsulta_1	: consulta q	$\longrightarrow$	consulta	$\{\text{tipo\_consulta}(q) \in \{\underline{\texttt{SELECT}}, \underline{\texttt{MATCH}}, \underline{\texttt{PROJ}}, \underline{\texttt{RENAME}}, \underline{\texttt{INTER}}, \underline{\texttt{UNION}}, \underline{\texttt{PRODUCT}}\}\}$
subconsulta_2	: consulta q	$\longrightarrow$	consulta	$\{\text{tipo\_consulta}(q) \in \{\underline{\texttt{INTER}}, \underline{\texttt{UNION}}, \underline{\texttt{PROJ}}\}\}$

axiomas $\forall r : \text{registro}, \forall c, c' : \text{nombre\_campo}, \forall v : \text{valor}$

$\text{tipo\_consulta}(\texttt{FROM}(t))$	$\equiv$	$\underline{\texttt{FROM}}$
$\text{tipo\_consulta}(\texttt{SELECT}(q, c, v))$	$\equiv$	$\underline{\texttt{SELECT}}$
$\text{tipo\_consulta}(\texttt{MATCH}(q, c_1, c_2))$	$\equiv$	$\underline{\texttt{MATCH}}$
$\text{tipo\_consulta}(\texttt{PROJ}(q, cs))$	$\equiv$	$\underline{\texttt{PROJ}}$
$\text{tipo\_consulta}(\texttt{RENAME}(q, c_1, c_2))$	$\equiv$	$\underline{\texttt{RENAME}}$
$\text{tipo\_consulta}(\texttt{INTER}(q_1, q_2))$	$\equiv$	$\underline{\texttt{INTER}}$
$\text{tipo\_consulta}(\texttt{UNION}(q_1, q_2))$	$\equiv$	$\underline{\texttt{UNION}}$
$\text{tipo\_consulta}(\texttt{PRODUCT}(q_1, q_2))$	$\equiv$	$\underline{\texttt{PRODUCT}}$
$\text{nombre\_tabla}(\texttt{FROM}(t))$	$\equiv$	t
$\text{campo}_1(\texttt{SELECT}(q, c, v))$	$\equiv$	c
$\text{campo}_1(\texttt{MATCH}(q, c_1, c_2))$	$\equiv$	c_1
$\text{campo}_1(\texttt{RENAME}(q, c_1, c_2))$	$\equiv$	c_1
$\text{campo}_2(\texttt{MATCH}(q, c_1, c_2))$	$\equiv$	c_2
$\text{campo}_2(\texttt{RENAME}(q, c_1, c_2))$	$\equiv$	c_2
$\text{valor}(\texttt{SELECT}(q, c, v))$	$\equiv$	v
$\text{conj\_campos}(\texttt{PROJ}(q, cs))$	$\equiv$	cs
$\text{subconsulta}_1(\texttt{SELECT}(q, c, v))$	$\equiv$	q
$\text{subconsulta}_1(\texttt{MATCH}(q, c_1, c_2))$	$\equiv$	q
$\text{subconsulta}_1(\texttt{PROJ}(q, cs))$	$\equiv$	q
$\text{subconsulta}_1(\texttt{RENAME}(q, c_1, c_2))$	$\equiv$	q
$\text{subconsulta}_1(\texttt{INTER}(q_1, q_2))$	$\equiv$	q_1

$$\begin{aligned}
&\text{subconsulta}_1(\texttt{UNION}(q_1, q_2)) &&\equiv\ q_1\\
&\text{subconsulta}_1(\texttt{PRODUCT}(q_1, q_2)) &&\equiv\ q_1\\
&\text{subconsulta}_2(\texttt{INTER}(q_1, q_2)) &&\equiv\ q_2\\
&\text{subconsulta}_2(\texttt{UNION}(q_1, q_2)) &&\equiv\ q_2\\
&\text{subconsulta}_2(\texttt{PRODUCT}(q_1, q_2)) &&\equiv\ q_2
\end{aligned}$$

Fin TAD