Módulo 5. Algebra Relacional

Introducción

Para los usuarios finales de una base de datos, el principal interés, con respecto a las consultas que se hagan sobre la base de datos es que dichas consultas respondan rápido y que entreguen la información solicitada. No importa dichas consultas **cómo** se hagan en el motor de la base de datos.

Pero desde el punto de vista de un profesional en sistemas, es muy importante conocer dichas consultas cómo se están llevando a cabo dentro del motor de la base de datos. Esto con el fin de poder darle respuestas concretas a un usuario acerca del rendimiento de las consultas que se están haciendo sobre una base de datos. No tiene presentación que un profesional en sistemas, ante el requerimiento de un usuario de encontrar la causa del bajo tiempo de respuesta de una consulta, no sepa dar los argumentos del por qué dicho tiempo de respuesta es tan bajo y, por ende, tampoco tendrá muchos criterios para optimizar dicha consulta.

En otras palabras, como se dijo en el módulo 1. de este libro, el lenguaje SQL es no procedimental, es decir, al escribir una instrucción de consulta, no interesa saber cómo ni dónde se va a llevar a cabo la consulta, sólo interesa expresar los datos que se quieren consultar.

El objetivo del álgebra relacional es entender cómo se ejecutan las instrucciones SQL. Es decir, si se quiere entender cómo se ejecuta una instrucción SQL de consulta, se debe entender álgebra relacional. Acá radica la importancia del tema del álgebra relacional.

Qué es Álgebra Relacional?

Álgebra Relacional es un lenguaje de consulta procedimental, es decir, sus instrucciones describen los datos que se van a consultar y la manera como dichos datos se van a traer de la base de datos.

Así como SQL es un lenguaje, el álgebra relacional también. Por lo tanto ambos se componen de instrucciones. Lo que sucede es que las instrucciones en álgebra relacional se conocen como **operaciones**. Además, la escritura de dichas operaciones tiene que respetar una sintaxis y una semántica, como en cualquier lenguaje de programación.

La forma como cualquier operación del álgebra relacional funciona se puede ver resumida en la siguiente gráfica. Como se puede observar, hay operaciones del álgebra relacional que reciben como parámetros de entrada los datos de una tabla (Tabla1) y hay otras operaciones que reciben los datos de dos tablas

(Tabla1 y Tabla 2). Pero independiente de la operación que sea, el resultado siempre será otra tabla (Tabla3).

Esta gráfica, que parece trivial, cobrará especial importancia cuando se esté aprendiendo a escribir operaciones de consulta en álgebra relacional.

Tipos de Operaciones del Álgebra Relacional

Existen dos formas de categorizar las operaciones del álgebra relacional:

- De acuerdo al número de tablas que recibe
  - Operaciones Unarias (U): Reciben una sola tabla.
  - Operaciones Binarias (B): Reciben dos tablas.

- De acuerdo a su posible implementación con otras operaciones.
  - Operaciones Básicas (o Primarias): Operaciones que NO se pueden implementar con base en otras operaciones, son las primitivas.
  - Operaciones Secundarias: Son operaciones que se pueden implementar con base en las operaciones básicas.

Operaciones del Álgebra Relacional

Las ocho principales operaciones que componen el álgebra relacional.

Operaciones Básicas (o Primarias):

- Selección
- Proyección
- Renombramiento
- Unión
- Diferencia
- Producto Cartesiano

Operaciones Secundarias:

- Intersección
- Reunión Natural

Operación Selección

Tipo: Unaria
Sintaxis:

$$\sigma_{\text{condición a cumplir}}(\text{Tabla})$$

Descripción: Operación que selecciona **tuplas completas** de una tabla y que cumplan con una condición dada.

Ejemplo:

Supongamos la siguiente tabla:

Carro = { **placa**, marca, modelo, capacidad }

Hacer la siguiente consulta en álgebra relacional: listar los datos de los carros cuyo modelo sea igual a 2001.

Solución:

$$\sigma_{\text{modelo} = 2001}(\text{Carro})$$

El resultado de la anterior consulta me muestra la placa, marca, modelo y capacidad de todos los carros modelo 2001. Recuerde que la selección trae tuplas completas, es decir, con todos sus atributos. Dicho de otra manera, la tabla3 de la anterior consulta tiene cuatro atributos (o columnas).

Ejemplo: Listar los datos de los carros cuya capacidad sea menor a 5 y que sea de marca Renault.

Solución:

$$\sigma_{(\text{marca} = \text{“Renault”} \text{ AND } \text{capacidad} < 5)}(\text{Carro})$$

Ejemplo: Listar los datos de los carros cuya marca es Mazda o cuyo modelo es 2003.

$$\sigma_{(\text{marca} = \text{“Mazda”} \text{ OR } \text{modelo} = 2003)}(\text{Carro})$$

Operación Proyección

Tipo: Unaria
Sintaxis:

$$\Pi_{\text{atributo1, atributo2,……..,atributo N}}(\text{Tabla})$$

Descripción: Operación que selecciona valores de atributos específicos de una

tabla.

Ejemplo: Listar la placa y marca de todos los carros.

$\Pi_{placa,\ marca}$ (Carro)

La operación anterior da como resultado una tabla3 con dos atributos: placa y marca. Esa es la diferencia con la selección. La proyección selecciona atributos específicos que se necesitan imprimir.

Sugerencia: Siempre que se vaya a escribir una operación del álgebra relacional se debe respetar la sintaxis de la misma.

Ejemplo: Listar la placa y modelo de los carros que sean de marca Chevrolet.

En esta consulta surge una primera dificultad y es que se está pidiendo imprimir ciertos atributos pero condicionado a algo. Y como se puede ver, la proyección no permite poner condiciones. Por lo tanto, hay que empezar a combinar operaciones.

Para escribir la solución de esta consulta, hay dos posibilidades:

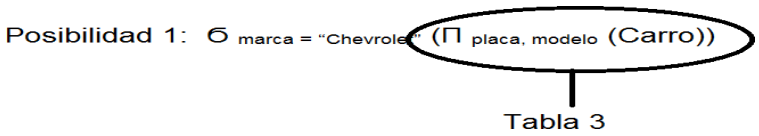
Posibilidad 1: $\sigma_{marca = \text{“Chevrolet”}} (\Pi_{placa,\ modelo}$ (Carro))

Posibilidad 2: $\Pi_{placa,\ modelo} (\sigma_{marca = \text{“Chevrolet”}}$ (Carro))

Se debe recalcar que ambas posibilidades están bien sintácticamente, pero hay una que hace lo que se pide y la otra no.

Por qué ambas están bien sintácticamente?

Como en toda operación aritmética, primero se hace lo que hay en paréntesis y luego el resto. Es decir, en la posibilidad 1, primero se hace la proyección dando como resultado una tabla3 y luego se hace la selección sobre ese resultado anterior. Como se puede ver, individualmente, no se está violando la sintaxis de la proyección ni de la selección.



Lo mismo sucede con la posibilidad 2, está bien escrita sintácticamente.

Como se dijo anteriormente, a pesar de que ambas están bien sintácticamente, solo una de las dos funciona para lo que se pide.

Analicemos la posibilidad 1,….primero se hace la proyección que es lo que está dentro del paréntesis. Esto genera una tabla con la placa y modelo de todos los carros (Tabla 3). Y luego se hace una selección a esta tabla resultante donde en la condición se pregunta por la marca y como se puede observar la marca no es un atributo que forma parte de la Tabla3 generada en la proyección. Luego acá se genera un error. (Es como si se hiciera un SELECT donde en el WHERE se invoca un campo que no existe en la tabla que está en el FROM).

En cambio, la posibilidad 2, primero se seleccionan tuplas completas de los carros marca Chevrolet y luego a dichas tuplas se les imprime su placa y su modelo. Lo cual es correcto y hace lo que se está pidiendo.

Por lo tanto, la solución a la consulta propuesta es la posibilidad 2. Este

resultado enseña que, teniendo el siguiente SELECT,….

```
SELECT ca.placa, ca.modelo
FROM carro as ca
WHERE ca.marca = “Chevrolet”
```

….. primero se ejecuta el WHERE y luego se hace lo de la cláusula SELECT. Es una muestra de que el álgebra relacional permite entender **“cómo”** se ejecuta una instrucción SELECT.

Operación Renombramiento

Tipo: Unaria
Sintaxis:

$\rho_{\text{Nombre Nuevo Tabla}}(\text{Tabla})$

Descripción: Operación que renombra, a nivel lógico, una tabla. Es decir, algunas veces es necesario renombrar a una tabla para poder escribir una operación del álgebra relacional.

El ejemplo de la utilización de esta operación se verá con más detalle cuando se explique la operación de producto cartesiano.

Operación Unión

Tipo: Binaria

Sintaxis:

Tabla1 U Tabla2

Descripción: Esta operación reúne o junta las tuplas de Tabla1 con las de Tabla2 en un mismo resultado, eliminando registros duplicados.

Ejemplo: Para hacer un ejemplo de unión, por ser una operación binaria, se

necesitan dos tablas. Por lo tanto, a la tabla que se viene trabajando (Carro) se va a adicionar la siguiente tabla:

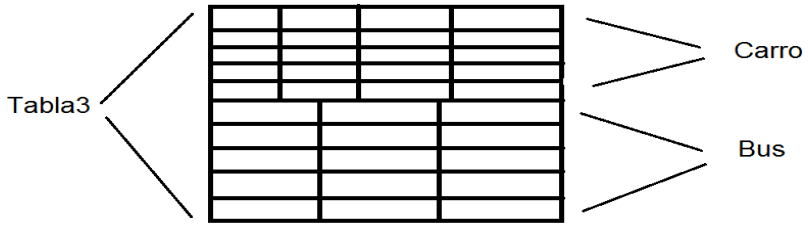
Bus = { **placa**, marca, color }. Se supone color como univalorado.

Se desea listar los datos de los automóviles en general (carros y buses).

Lo que se pide, se ajusta perfectamente a lo que hace la operación de Unión. Se necesita poner los datos de los carros y de los buses en un solo resultado. Por lo tanto, se puede pensar que la solución es la siguiente:

Carro U Bus

Analizando el posible resultado de dicha unión, y teniendo en cuenta que el resultado de la operación es otra tabla (Tabla3), dicho resultado tendría la siguiente apariencia:



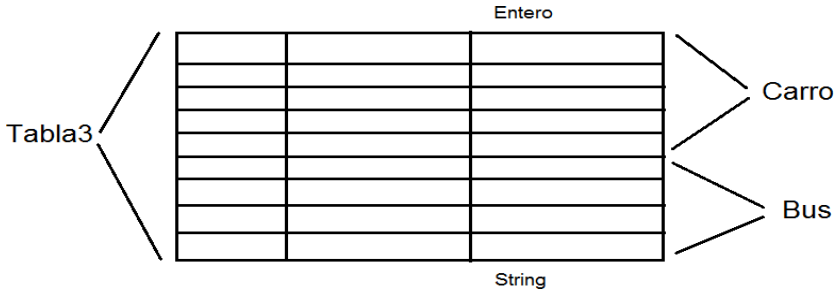
Carro U Bus

Como se puede observar, la Tabla3 resultante tendría una característica que no es válida en el modelo relacional: una tabla no puede tener en ciertas tuplas cuatro atributos y en otras tuplas tres atributos.

Por lo tanto, para hacer la unión de dos tablas, se TIENE que cumplir la condición de que ambas tablas tengan la misma aridad (grado), es decir, la misma cantidad de atributos.

Para poder solucionar el problema y hacer que ambas tablas cumplan con la condición, supongamos la tabla CARRO sin el campo capacidad. Es decir, ambas tablas tendrían tres atributos.

Al hacer la unión entre CARRO y BUS, el resultado (Tabla3) quedaría con el siguiente aspecto:



Se soluciona el problema de que ya TODA la Tabla3 quedaría con tres atributos. Pero si se analiza bien, hay otra situación que no puede pasar en una tabla del modelo relacional. El tercer atributo de la tabla CARRO es el modelo, el cual es de tipo entero. Y el tercer atributo de la tabla BUS es el color, el cual es de tipo string. Es decir, en la Tabla3 hay ciertas tuplas donde el tercer atributo es entero y otras tuplas donde ese mismo atributo es string. Eso no es posible en el modelo relacional.

Por lo tanto, para hacer la unión de dos tablas se tiene que cumplir una segunda condición: los atributos correspondientes en las dos tablas deben ser del mismo tipo.

> Importante: Para poder hacer Tabla1 U Tabla2 estas tablas deben ser unión compatible. Es decir, se tienen que cumplir dos condiciones: Tabla1 y Tabla2 tienen que ser de la misma aridad (grado) y los atributos correspondientes (que se van a unir) de Tabla1 con los de Tabla2 toman sus valores del mismo dominio.

Es así, entonces, que cuando se va a hacer la unión de dos tablas, se proyectan los campos en común de ambas tablas. Por lo tanto, para listar los datos de todos los automóviles se haría lo siguiente:

$\Pi_{placa,\ marca}$ (Carro) U $\Pi_{placa,\ marca}$ (Bus)

Lo anterior es lo mismo que $\Pi_{placa,\ marca}$ (Carro U Bus)?

Operación Diferencia

Tipo: Binaria
Sintaxis:

Tabla1 - Tabla2

Descripción: Lista las tuplas que están en Tabla1 y **NO ESTÁN** en Tabla2.

Ejemplo: Para el ejemplo de la diferencia supongamos las siguientes dos tablas:

Todos_Los_Estudiantes = { **cédula**, nombre, dirección, edad }
Estudiantes_Becados = { **cédula**, nombre, edad, raza, teléfono }

Se supone que en la primera tabla están absolutamente todos los estudiantes, becados y no becados. Y en la segunda solamente están los datos de los estudiantes que están becados.

Listar los datos de los estudiantes que no están becados.

Para resolver esta consulta se puede utilizar la operación diferencia. De tal

manera que se escribiría:

Todos_Los_Estudiantes - Estudiantes_Becados

Si se analiza bien el resultado de la anterior operación, todas las tuplas que están en Todos_Los_Estudiantes y NO están en Estudiantes_Becados, no daría como resultado lo que se está pidiendo. De hecho:

Todos_Los_Estudiantes-Estudiantes_Becados = Todos_Los_Estudiantes

Lo cual no es lo que se está pidiendo. Es decir, cuando en una diferencia, las dos tablas no cumplen con la condición de unión compatible, la consulta funciona pero no da el resultado esperado.

Por lo tanto, mientras en la unión las dos condiciones expuestas se **TIENEN** que cumplir para que la consulte funcione, en la diferencia las dos condiciones se **DEBEN** cumplir para lograr el resultado esperado.

Para hacer la diferencia entre dos tablas, se procede, como en la unión, a proyectar los atributos comunes. Entonces, la solución a la anterior consulta es la siguiente:

$\Pi_{cedula,\ nombre,\ edad}$(Todos_Los_Estudiantes) - $\Pi_{cedula,\ nombre,\ edad}$(Estudiantes_Becados)

Operación Producto Cartesiano

Tipo: Binaria
Sintaxis:

Tabla1 X Tabla2

Descripción: Concatena cada una de las tuplas de Tabla1 con cada una de las tuplas de Tabla2, una a una. Para entender mejor esta operación, suponga la siguiente situación expresada:

Tabla1: Tiene 10 tuplas y 8 atributos.
Tabla2: Tiene 7 tuplas y 9 atributos.

Entonces,

Tabla1 X Tabla2 tiene 70 tuplas y cada tupla tendrá 17 atributos.

Ejemplo: Suponga las siguientes dos tablas:

Avión = { **matrícula**, marca, No.horasvuelo, código_aerolinea }
código_aerolínea referencia a Aerolinea (código)
Aerolínea = { **código**, nombre, añofundación }

Y los datos de ambas tablas son los siguientes:

Tabla Avión			
Matrícula	**Marca**	**No.horasvuelo**	**Código_aerolínea**
HK7800	Boeing	520	10
HK8000	Airbus	410	10
HK2300	Concorde	215	20
HK7400	Boeing	500	30

Tabla Aerolínea		
Código	**Nombre**	**AñoFundación**
10	Avianca	1950
20	Copa	1966
30	Satena	1945

Entonces el resultado de Avión X Aerolínea es el siguiente:

Avión X Aerolínea						
Matrícula	**Marca**	**No.HorasV**	**Código_Aerolínea**	**Código**	**Nombre**	**AñoFund.**
HK7800	Boeing	520	10	10	Avianca	1950
HK7800	Boeing	520	10	20	Copa	1966
HK7800	Boeing	520	10	30	Satena	1945
HK8000	Airbus	410	10	10	Avianca	1950
HK8000	Airbus	410	10	20	Copa	1966
HK8000	Airbus	410	10	30	Satena	1945
HK2300	Concorde	215	20	10	Avianca	1950
HK2300	Concorde	215	20	20	Copa	1966
HK2300	Concorde	215	20	30	Satena	1945
HK7400	Boeing	500	30	10	Avianca	1950
HK7400	Boeing	500	30	20	Copa	1966
HK7400	Boeing	500	30	30	Satena	1945

A cualquier par de tablas se les puede hacer un producto cartesiano. Lo que sucede es que hay ocasiones en las cuales el resultado no tiene sentido.

Ejemplo: Listar la matrícula y marca de cada avión junto con el nombre de la aerolínea a la cual pertenecen.

Para solucionar esto en SQL, se sabe que se ejecuta la siguiente instrucción:

```
SELECT av.matricula, av.marca, ae.nombre
FROM Avion av INNER JOIN Aerolínea ae ON
av.codigo_aerolinea = ae.codigo
```

Si se analiza el producto cartesiano entre Avión y Aerolínea, el resultado son 12 tuplas, de las cuales 8 son tuplas espurias y 4 son tuplas verdaderas. Cuáles son las 8 tuplas espurias, y cuáles son las 4 verdaderas?

Entonces, para hacer la anterior consulta en álgebra relacional, la operación es la siguiente:

$\Pi_{\text{matricula, marca, nombre}} (\sigma_{\text{código\_aerolinea = codigo}}(\text{Avión X Aerolínea}))$

Primero se hace el producto cartesiano, el cual da como resultado 12 tuplas. Luego se hace la selección, la cual selecciona las 4 tuplas reales y de estas 4 tuplas se proyectan los atributos a imprimir.

Como conclusión a este ejemplo, se puede afirmar que todo INNER JOIN es hecho a través de un producto cartesiano.

Ejemplo: Listar el número de horas de vuelo del avión que MÁS horas de vuelo tiene.

Hacer esto en SQL es muy sencillo:

SELECT max(No.horasvuelo) FROM
Avión

La explicación siguiente muestra cómo se llevaría a cabo esta instrucción en álgebra relacional.

Para uno saber cuál es el máximo número de horas de vuelo, se deben comparar TODOS los números de horas de vuelo. Eso está disponible en la tabla AVION, pero cada No.horasvuelo está en una tupla distinta, lo cual se constituye en un problema ya que no se tiene una operación que compare valores de un mismo campo en tuplas distintas.

El problema se solucionaría si tuviéramos el par de valores a comparar del atributo No.horasvuelo en la misma tupla. Cómo se logra esto?

Si se hace un producto cartesiano entre la tabla Avión consigo misma, cada tupla tendría dos No.horasvuelo y se tendría todas las parejas que habría que comparar. El producto cartesiano mencionado se muestra a continuación.

Avión X Avión

Matricula	Marca	No.HorasV	Codigo_A	Matricula	Marca	No.HorasV	Codigo_A
HK7800	Boeing	520	10	HK7800	Boeing	520	10
HK7800	Boeing	520	10	HK8000	Airbus	410	10
HK7800	Boeing	520	10	HK2300	Concorde	215	20
HK7800	Boeing	520	10	HK7400	Boeing	500	30
HK8000	Airbus	410	10	HK7800	Boeing	520	10
HK8000	Airbus	410	10	HK8000	Airbus	410	10
HK8000	Airbus	410	10	HK2300	Concorde	215	20
HK8000	Airbus	410	10	HK7400	Boeing	500	30
HK2300	Concorde	215	20	HK7800	Boeing	520	10
HK2300	Concorde	215	20	HK8000	Airbus	410	10
HK2300	Concorde	215	20	HK2300	Concorde	215	20
HK2300	Concorde	215	20	HK7400	Boeing	500	30
HK7400	Boeing	500	30	HK7800	Boeing	520	10
HK7400	Boeing	500	30	HK8000	Airbus	410	10
HK7400	Boeing	500	30	HK2300	Concorde	215	20
HK7400	Boeing	500	30	HK7400	Boeing	500	30

Lo que sigue es empezar a comparar cada par de valores del campo No.horasvuelo que hay en cada tupla. Pero surge un nuevo problema: en Avión X Avión, en cada tupla hay dos atributos llamados No.horasvuelo, por lo tanto, si necesitamos comparar ese par de valores hay que diferenciar, de alguna

manera, los nombres de los atributos.

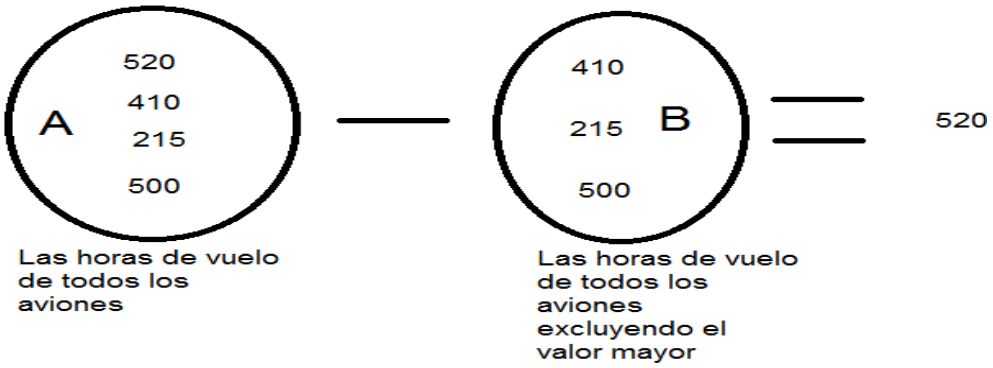
Es ahí donde entra a funcionar la operación de renombramiento. Si al hacer la operación de Avión X Avión, se renombra alguna de las dos "tablas" (Avión de la izquierda del producto o Avión de la derecha del producto), ya habría alguna manera de identificar los dos atributos No.horasvuelo.

Por lo tanto, con la explicación dada hasta el momento, tenemos la siguiente expresión del álgebra:

Avión X ρ_{Av} (Avión)

lo cual da como resultado la tabla anterior.

El mecanismo para lograr la consulta requerida es la siguiente:



Entonces,

A = $\Pi_{No.horasvuelo}$ (Avión)

C = $\sigma_{Avion.No.horasvuelo < Av.No.horasvuelo}$ (Avión X ρ_{Av} (Avión))

B = $\Pi_{Avion.No.horasvuelo}$ (C)

Las tuplas seleccionadas en C se muestran resaltadas en amarillo.

Avión X Avión							
Matricula	**Marca**	**No.HorasV**	**Codigo_A**	**Matricula**	**Marca**	**No.HorasV**	**Codigo_A**
HK7800	Boeing	520	10	HK7800	Boeing	520	10
HK7800	Boeing	520	10	HK8000	Airbus	410	10
HK7800	Boeing	520	10	HK2300	Concorde	215	20
HK7800	Boeing	520	10	HK7400	Boeing	500	30
HK8000	Airbus	410	10	HK7800	Boeing	520	10
HK8000	Airbus	410	10	HK8000	Airbus	410	10
HK8000	Airbus	410	10	HK2300	Concorde	215	20
HK8000	Airbus	410	10	HK7400	Boeing	500	30
HK2300	Concorde	215	20	HK7800	Boeing	520	10
HK2300	Concorde	215	20	HK8000	Airbus	410	10
HK2300	Concorde	215	20	HK2300	Concorde	215	20
HK2300	Concorde	215	20	HK7400	Boeing	500	30
HK7400	Boeing	500	30	HK7800	Boeing	520	10
HK7400	Boeing	500	30	HK8000	Airbus	410	10
HK7400	Boeing	500	30	HK2300	Concorde	215	20
HK7400	Boeing	500	30	HK7400	Boeing	500	30

Y al hacer B, es decir, al proyectar los valores de No.horasvuelo de la izquierda de dichas tuplas, obtenemos B.

Por lo tanto, la solución a la consulta es A – B.

Qué ocurre, si lo que se necesita es listar el número de horas de vuelo del avión que MENOS horas de vuelo tiene?

Operación Intersección

Tipo: Binaria
Sintaxis:

Tabla1 ∩ Tabla2

Descripción: Selecciona las tuplas de Tabla1 que también están en Tabla2. En otras palabras, selecciona las tuplas comunes en las dos tablas.

Por qué esta operación es secundaria? Porque, según la teoría de conjuntos, la intersección se podría implementar con la operación diferencia, así:

A ∩ B = A – (A – B).

Ejemplo: Supongamos las siguientes dos tablas:

Profesor = { **cedula**, nombre, edad }
Estudiante = { **cedula**, nombre, dirección, teléfono }

Se desea listar los datos de los estudiantes que a la vez son profesores.

La operación intersección suple está necesidad porque se necesitan los datos

que estén a la vez en las dos tablas. Por lo tanto, la aparente solución es:

Estudiante ∩ Profesor

Si aplicamos exactamente la definición de intersección, las tuplas que están en Estudiante y a la vez están en Profesor, nos da como resultado un conjunto vacío, es decir, no trae ninguna tupla, lo cual no es el resultado esperado. Ninguna tupla de Estudiante está también en Profesor ya que las tuplas de Estudiante tienen 4 atributos y las de Profesor tienen 3 atributos.

Es así que se llega a la misma conclusión a la que se llegó en la operación diferencia. Para que la operación intersección de un resultado coherente, se DEBEN cumplir las mismas dos condiciones de la unión y la diferencia.

Por lo tanto, la solución a la consulta es la siguiente:

$[\Pi_{\text{cedula, nombre}}(\text{Estudiante})] \cap [\Pi_{\text{cedula, nombre}}(\text{Profesor})]$

Operación Reunión Natural

Tipo: Binaria
Sintaxis:

Tabla1 **|X|** Tabla2

Descripción: La Reunión natural hace automáticamente las siguientes dos operaciones y en dicho orden:

- Tabla1 X Tabla2
- Selecciona de dicho producto cartesiano las tuplas donde la clave foránea de Tabla1 es igual a la clave primaria de Tabla2.

Es decir, la reunión natural implementa automáticamente el INNER JOIN.

Ejemplo: Rehacer la misma consulta expresada en la operación producto cartesiano, pero a través de una reunión natural. La consulta decía: Listar la matrícula y marca de cada avión junto con el nombre de la aerolínea a la cual pertenecen.

La forma de hacer la consulta con el producto cartesiano es la siguiente: (ya estaba expresada en apartados anteriores)

$\Pi_{\text{matricula, marca, nombre}}(\sigma_{\text{código-aerolinea = codigo}}(\text{Avión X Aerolínea}))$

La manera de hacer la misma consulta, pero a través de una reunión natural es

la siguiente:

$\Pi_{matricula,\ marca,\ nombre}$ (Avión **IXI** Aerolínea)

Es decir, el operador **IXI** automáticamente hace el producto cartesiano entre Avión y Aerolínea y luego hace la selección de las tuplas donde el código de aerolínea en Avión sea igual al código de aerolínea en Aerolínea.

Conclusión: Sabiendo expresar una consulta en álgebra relacional, se tiene una visión muy clara de cómo ejecuta dicha consulta el motor de la base de datos y, por lo tanto, se tiene un criterio mucho más amplio para decidir si hay una manera de optimizar dicha consulta.
Ahí radica la importancia del álgebra relacional.

Operación División:

Tipo: Binaria

Sintaxis:

$Tabla1_{(x,y)} \div Tabla2_{(z)}$

Descripción: Tomamos como dividendo la relación binaria Tabla1 con atributos *x, y*. El divisor Tabla2 con atributo *z*. Los atributos *y, z* tienen el mismo dominio. Entonces:

$Tabla1_{(x,y)} \div Tabla2_{(z)}$ produce un cociente con dominio igual al de *x.*

Un valor *x* aparecerá solo cuando Tabla1 contenga pares (*x,y*) para todos los valores de *y* en Tabla2.

Ejemplo: Sean las siguientes tablas, listar las tallas que se presenten en colores azul y rojo.

Tabla1

Talla	Color
32	Gris
32	Azul
32	Rojo
34	Gris
34	Negro
35	Azul
36	Rojo

Tabla2

Color
Azul
Rojo

Para encontrar la solución realizamos una división así:

Tabla1 ÷ Tabla2 =

Talla
32

Se observa que la talla 32 se encuentra en los dos colores, azul y rojo.

Ejemplo: Listar las tallas que se presenten en color rojo.

Tabla1 ÷

Color
Rojo

=

Talla
32
36