
Tema 4: Álgebra Relacional



Índice

- I. Introducción
- II. Operadores del Álgebra Relacional
- III. Operadores adicionales de consulta
- IV. Operadores adicionales de modificación
- V. Ejercicios
- VI. Bibliografía

4.1. Introducción

- En MD Relacional:
 - **Estructural:** Estructuras de datos que soporta.
 - **Semántica:** Definida mediante un conjunto de restricciones o reglas de integridad.
 - **Manipulativa:** permite modificar el estado de la estructura de datos relacional y realizar consultas de datos.
 - *Algebra Relacional (procedimental)*
 - Cálculo Relacional (no procedimental)
 - De Tuplas
 - De Dominios

4.1. Introducción

- El álgebra relacional es un **lenguaje de consulta procedimental** que dispone de un **conjunto de operadores** que operan sobre relaciones. Cada operador toma una o dos relaciones como entrada para producir una nueva relación como salida sin que cambien las relaciones originales.
- Así, tanto los operandos como los resultados **son** relaciones, de este modo, la salida de una operación puede convertirse en la entrada de otra operación.
Ejemplo: $R = ((R_1 \text{ op } R_2) \text{ op } R_3) \text{ op } R_4$
- **Expresión Relacional:** secuencia de operaciones del álgebra relacional cuyo resultado será también una relación.

4.1. Introducción

- El álgebra relacional es importante por varias razones:
 1. Proporciona un fundamento formal para las operaciones del modelo relacional.
 2. Se utiliza como base para la implementación y optimización de consultas en los SGBD.
 3. Algunos de sus conceptos se han incorporado al lenguaje estándar de consultas SQL para los SGBD.

4.1. Introducción

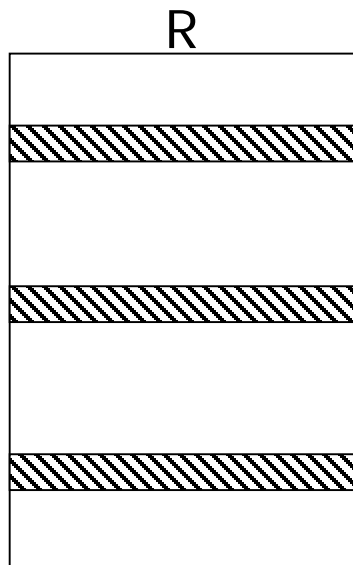
→ Operadores

- Codd define el primer conj. de operadores en 1972 y los clasifica en:
 - **Tradicionales** (*conjuntistas*): unión, intersección, diferencia, producto cartesiano.
 - **Propios** (*relacionales*): selección, proyección, reunión, división, renombrado.

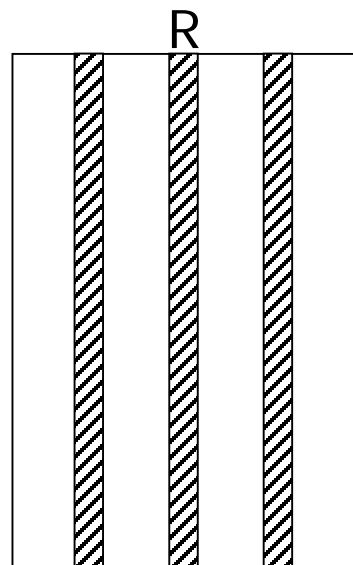
4.1. Introducción

→ Operadores

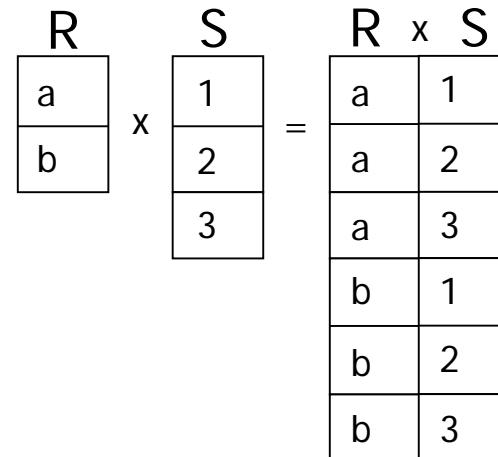
□ Definición intuitiva de los operadores:



Selección



Proyección

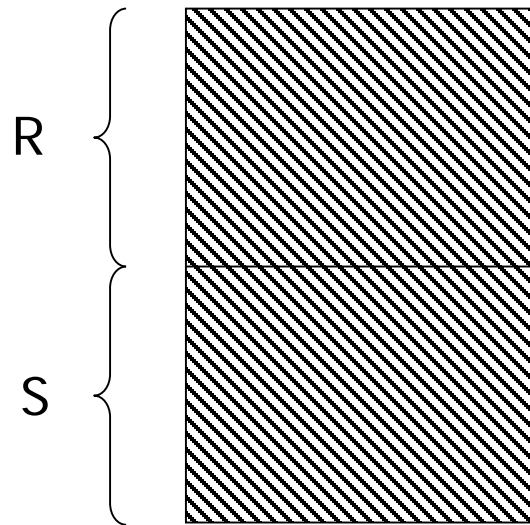


Producto Cartesiano

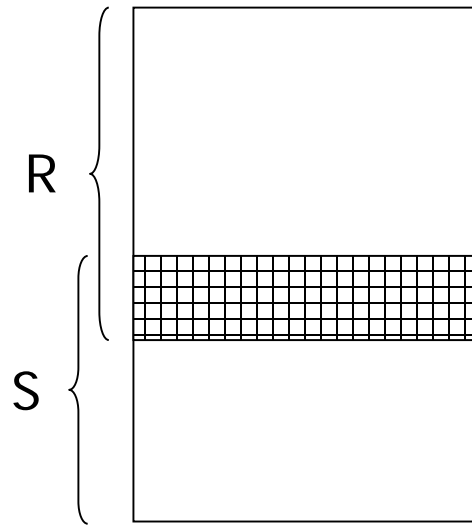
4.1. Introducción

→ Operadores

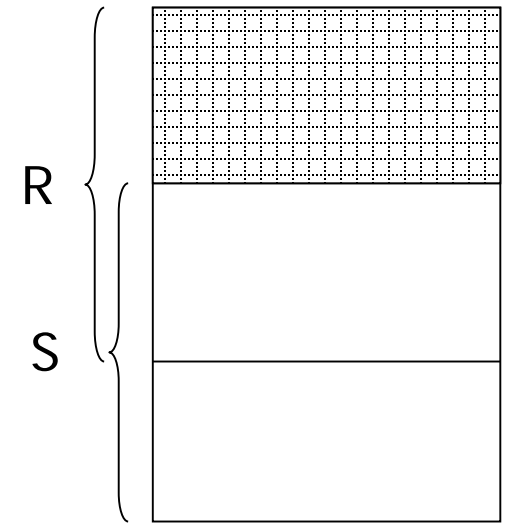
□ Definición intuitiva de los operadores:



Unión



Intersección



Diferencia

4.1. Introducción

→ Operadores

- Definición intuitiva de los operadores:

R		S				
A	B	B	C	A	B	C
a	1	1	x	a	1	x
b	2	1	y	a	1	y
		3	z			

Reunión

4.1. Introducción

→ Operadores

- Definición intuitiva de los operadores:

R

A	B
a	1
a	2
b	1
b	2
c	1

S

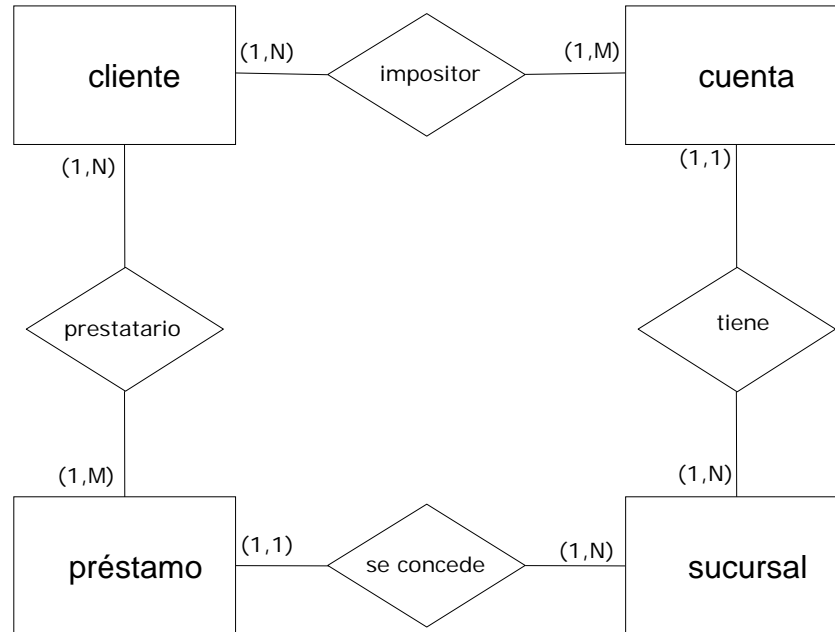
B
1
2

A
a
b

División

II. Operadores del Álgebra Relacional

➔ *Ejemplo*



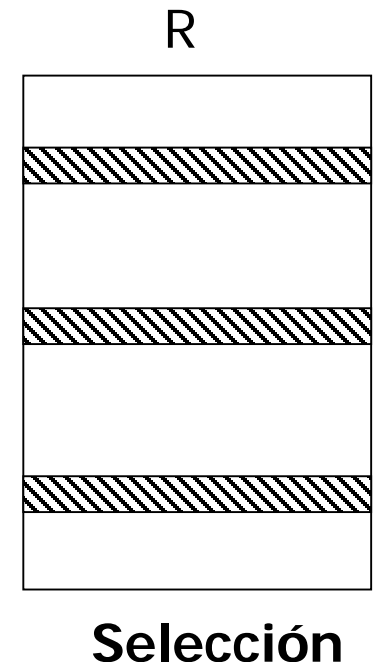
Ej: **cuenta** (num_cta, saldo_cta, nombre_suc)
sucursal (nombre_suc, ciudad_suc, activos_suc)
cliente (nombre_cli, calle_cli, ciudad_cli)
préstamo (num_pre, importe_pre, nombre_suc)
impositor (nombre_cli, num_cta)
prestario (nombre_cli, num_pre)

II. Operadores del Álgebra Relacional

➔ Selección

□ Selección (σ)

- Operador **unario**.
- Si θ es un predicado que se puede construir con los atributos de una relación $R \rightarrow \sigma_{\theta}(R)$ es la selección según el predicado θ . Es decir, se queda con las tuplas de la relación que hacen cierto el predicado θ .
- El predicado puede ser **compuesto** mediante los operadores: **AND**(\wedge), **OR**(\vee), **NOT**(\neg).
- Se permiten **comparaciones** que usan $=$, $<$, $>$, $<=$, \geq , $>$.
- Cualquier comparación que implique a un valor nulo se evalúa como **falsa**
- **SQL**: WHERE



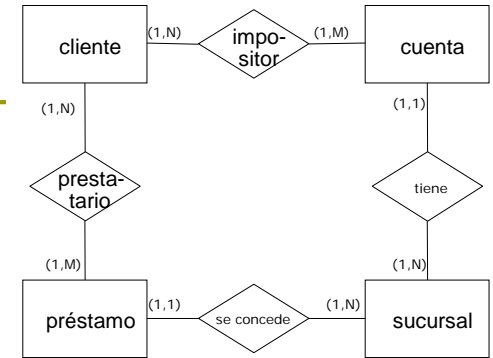
II. Operadores del Álgebra Relacional

→ Selección

□ Selección (σ)

sucursal

nombre_suc	ciudad_suc	activos_suc
SUC1	OURENSE	100.000
SUC2	VIGO	200.000
SUC3	LUGO	300.000
SUC4	OURENSE	250.000



“Sucursales con sede en Ourense”

$\sigma_{\text{ciudad_suc} = \text{“Ourense”}}(\text{sucursal})$



nombre_suc	ciudad_suc	activos_suc
SUC1	OURENSE	100.000
SUC4	OURENSE	250.000

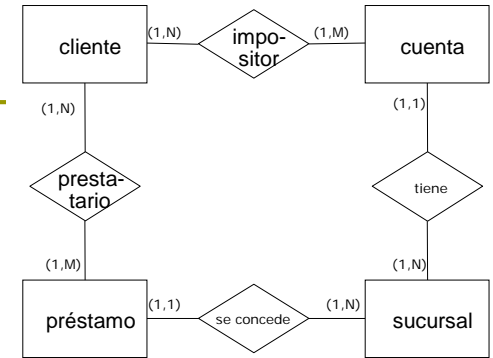
II. Operadores del Álgebra Relacional

➔ Selección

▣ Selección (σ)

sucursal

nombre_suc	ciudad_suc	activos_suc
SUC1	OURENSE	100.000
SUC2	VIGO	200.000
SUC3	LUGO	300.000
SUC4	OURENSE	250.000



“Sucursales con sede en Ourense y más de 150.000€ en activos”

$\sigma_{(ciudad_suc="OURENSE" \wedge activos_suc > 150000)}(sucursal)$

Sentencia SQL:

SELECT *

FROM sucursal

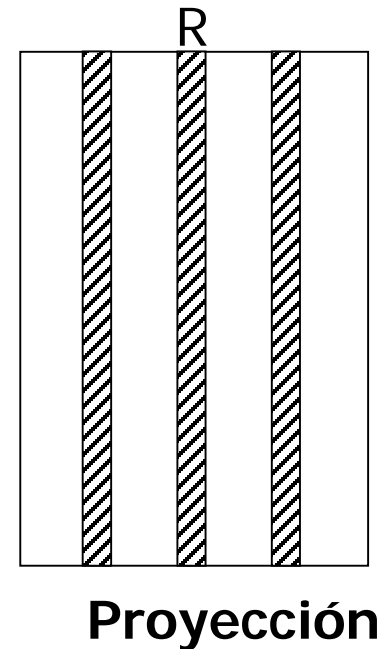
WHERE (ciudad_suc = "OURENSE") AND (activos_suc > 150000)

II. Operadores del Álgebra Relacional

➔ *Proyección*

□ Proyección (π)

- Operador **unario**.
- Si se tiene una relación R con atributos A_1, A_2, \dots, A_m , se dice que se ha proyectado R sobre el conjunto A de atributos A_1, A_2, \dots, A_m cuando se elimina de R todas las columnas que no están en el conjunto A y se eliminan las tuplas repetidas que puedan aparecer.
- **SQL**: SELECT



II. Operadores del Álgebra Relacional

➔ *Proyección*

□ Proyección (π)

sucursal

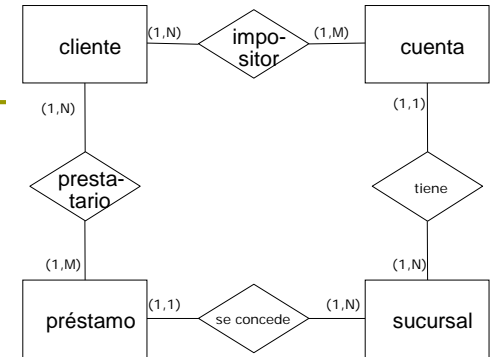
nombre_suc	ciudad_suc	activos_suc
SUC1	OURENSE	100.000
SUC2	VIGO	200.000
SUC3	LUGO	300.000
SUC4	OURENSE	250.000

“Lista con los nombres de las ciudades de las sucursales”

$\pi_{\text{ciudad_suc}}(\text{sucursal})$



ciudad_suc
OURENSE
VIGO
LUGO



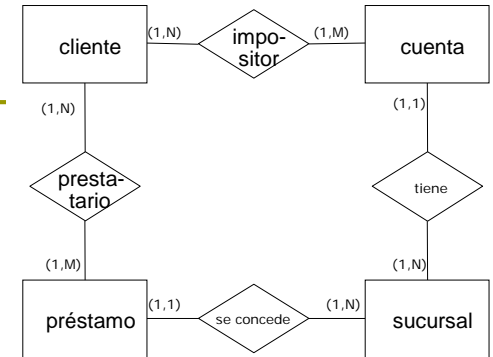
II. Operadores del Álgebra Relacional

→ *Proyección*

□ Proyección (π)

sucursal

nombre_suc	ciudad_suc	activos_suc
SUC1	OURENSE	100.000
SUC2	VIGO	200.000
SUC3	LUGO	300.000
SUC4	OURENSE	250.000



“Lista con los nombres de las ciudades y sucursales cuyos activos sean mayores de 220.000€”

$\pi_{\text{nombre_suc}, \text{ciudad_suc}} (\sigma_{\text{activos_suc} > 220000} (\text{sucursal}))$

Sentencia SQL:

```
SELECT nombre_suc, ciudad_suc
FROM sucursal
WHERE (activos_suc > 220000)
```

II. Operadores del Álgebra Relacional

→ Unión

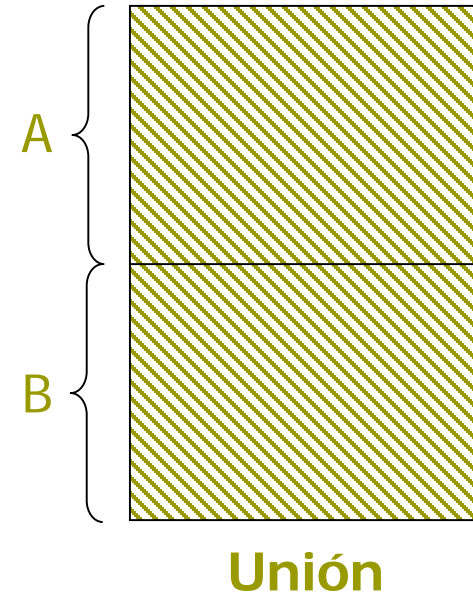
□ Unión (\cup)

- Operador binario.
- La unión de 2 relaciones A y B **con esquemas compatibles**, es una relación formada por todas las tuplas t pertenecientes **a A o a B**.

■ **Compatibilidad entre relaciones:** Dos relaciones A y B son compatibles para la unión, intersección y diferencia, si y sólo si:

1. $\text{Grado}(A) = \text{Grado}(B)$
2. Los dominios de los atributos *i*-ésimos de ambas relaciones son iguales para todo *i*.

■ **SQL:** UNION



II. Operadores del Álgebra Relacional

➔ *Unión*

□ Unión (\cup)

A

A1	A2
X	1
X	2
Y	1
Y	2

B

B1	B2
X	1
Y	2
Y	3
X	4

$A \cup B$



A1	A2
X	1
X	2
Y	1
Y	2
Y	3
X	4

II. Operadores del Álgebra Relacional

→ Unión

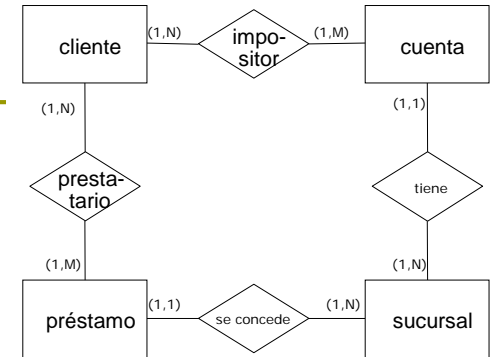
□ Unión (\cup)

impositor

nombre_cli	num_cuenta
CLIENTE1	C001
CLIENTE2	C002
CLIENTE3	C003
CLIENTE4	C004


prestatario

nombre_cli	num_pre
CLIENTE1	P001
CLIENTE5	P002



“Nombre de clientes que tienen una cuenta, un préstamo o ambas cosas”

$\pi_{\text{nombre_cli}} (\text{impositor}) \cup \pi_{\text{nombre_cli}} (\text{prestatario})$



nombre_cli
CLIENTE1
CLIENTE2
CLIENTE3
CLIENTE4
CLIENTE5

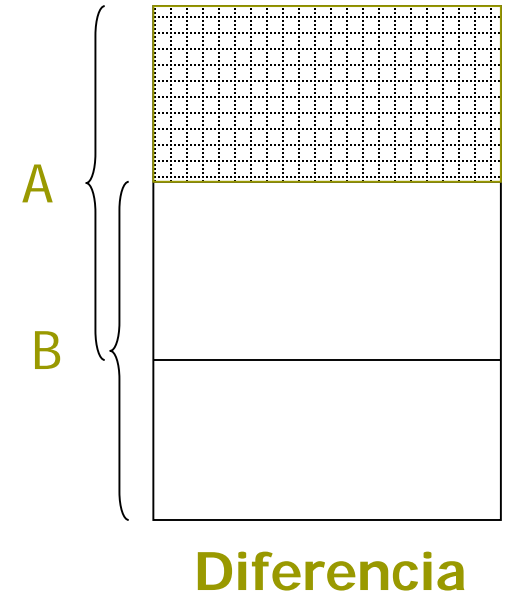
Sentencia SQL:

```
SELECT nombre_cli FROM impositor
UNION
SELECT nombre_cli FROM prestatario
```

II. Operadores del Álgebra Relacional

→ *Diferencia*

- Diferencia (—)
 - Operador binario.
 - La diferencia de 2 relaciones A y B **compatibles**, es una relación formada por todas las tuplas t pertenecientes **a A y no pertenecientes a B**.
 - **SQL**: EXCEPT, MINUS



II. Operadores del Álgebra Relacional

➔ *Diferencia*

□ Diferencia (—)

A

A1	A2
X	1
X	2
Y	1
Y	2

B

B1	B2
X	1
Y	2
Y	3
X	4

A — B



A1	A2
X	2
Y	1

II. Operadores del Álgebra Relacional

→ Diferencia

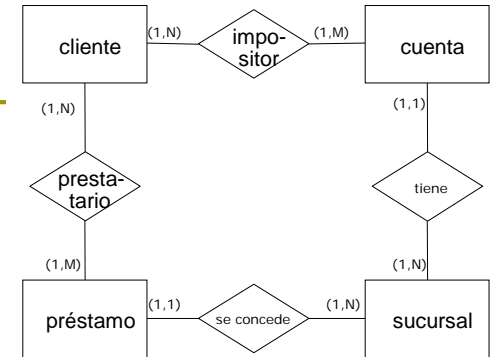
□ Diferencia (—)

impositor

nombre_cli	num_cuenta
CLIENTE1	C001
CLIENTE2	C002
CLIENTE3	C003
CLIENTE4	C004


prestatario

nombre_cli	num_pre
CLIENTE1	P001
CLIENTE5	P002



“Nombre de clientes que tienen una cuenta pero no un préstamo”

$$\pi_{\text{nombre_cli}} (\text{impositor}) - \pi_{\text{nombre_cli}} (\text{prestatario})$$



nombre_cli
CLIENTE2
CLIENTE3
CLIENTE4

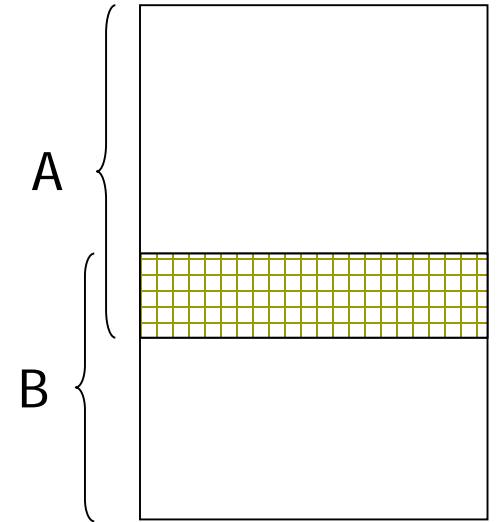
Sentencia SQL:

```
SELECT nombre_cli FROM impositor
MINUS
SELECT nombre_cli FROM prestatario
```

II. Operadores del Álgebra Relacional

➔ *Intersección*

- Intersección (\cap)
 - Operador binario.
 - La intersección de 2 relaciones A y B **compatibles**, es una relación formada por todas las tuplas t pertenecientes **a A y a B**.
 - **SQL**: INTERSECT



Intersección

II. Operadores del Álgebra Relacional

➔ *Intersección*

□ Intersección (\cap)

A

A1	A2
X	1
X	2
Y	1
Y	2

B

B1	B2
X	1
Y	2
Y	3
X	4

$A \cap B$



A1	A2
X	1
Y	2

4.2. Operadores del Álgebra Relacional

➔ *Intersección*

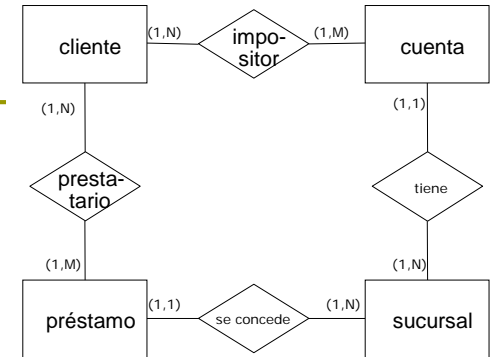
□ Intersección (\cap)

impositor

nombre_cli	num_cuenta
CLIENTE1	C001
CLIENTE2	C002
CLIENTE3	C003
CLIENTE4	C004


prestatario

nombre_cli	num_pre
CLIENTE1	P001
CLIENTE5	P002



“Nombre de clientes que tienen una cuenta y un préstamo”

$\pi_{\text{nombre_cli}} (\text{impositor}) \cap \pi_{\text{nombre_cli}} (\text{prestatario})$



nombre_cli
CLIENTE1

Sentencia SQL:

```
SELECT nombre_cli FROM impositor
INTERSECT
SELECT nombre_cli FROM prestatario
```

4.2. Operadores del Álgebra Relacional

➔ *Producto Cartesiano*

□ Producto Cartesiano (X)

- Operador binario.
- Permite combinar información de 2 relaciones.
- El producto cartesiano de 2 relaciones de cardinalidades **m1** y **m2** es una relación definida sobre la unión de los atributos de ambas relaciones y formada por las **m1 x m2** tuplas concatenando cada tupla de la primera relación con cada una de las tuplas de la segunda.
- Las relaciones **no han de ser compatibles**.
- El grado de la nueva relación será:
 - **Grado** (A x B) = **Grado(A)** + **Grado(B)**
- La cardinalidad de la nueva relación será:
 - **Card** (A x B) = **Card** (A) x **Card** (B)
- Es una operación asociativa y conmutativa.
- **SQL**: FROM A, B

4.2. Operadores del Álgebra Relacional

➔ *Producto Cartesiano*

□ Producto Cartesiano (X)

A

a
b

B

1
2
3

A x B



a	1
a	2
a	3
b	1
b	2
b	3

Grado (AxB)= Grado(A) + Grado(B)

Card (AxB)= Card(A) x Card(B)

4.2. Operadores del Álgebra Relacional

➔ *Producto Cartesiano*

□ Producto Cartesiano (X)

cliente

nombre_cli	calle_cli	ciudad_cli
CLIENTE1	rúa nova	OURENSE
CLIENTE2	otero pedrayo	OURENSE

cuenta

num_cuenta	saldo_cta	nombre_suc
C001	200	SUC1
C002	300	SUC1
C003	500	SUC2

“Clientes y cuentas”

cliente x cuenta



nombre_cli	calle_cli	ciudad_cli	num_cuenta	saldo_cta	nombre_suc
CLIENTE1	rúa nova	OURENSE	C001	200	SUC1
CLIENTE1	rúa nova	OURENSE	C002	300	SUC1
CLIENTE1	rúa nova	OURENSE	C003	500	SUC2
CLIENTE2	otero pedrayo	OURENSE	C001	200	SUC1
CLIENTE2	otero pedrayo	OURENSE	C002	300	SUC1
CLIENTE2	otero pedrayo	OURENSE	C003	500	SUC2

4.2. Operadores del Álgebra Relacional

➔ *Producto Cartesiano*

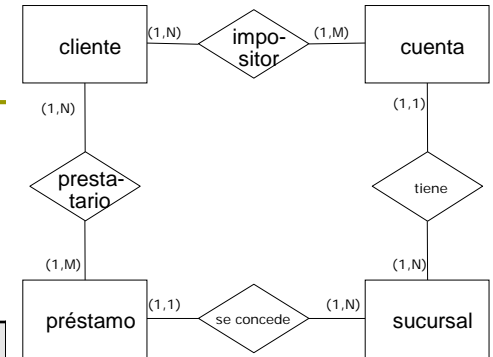
□ Producto Cartesiano (X)

prestamo

num_pre	nombre_suc	importe_pre
P1	SUC1	100
P2	SUC2	200
P3	SUC3	300
P4	SUC4	400

prestatario

nombre_cli	num_pre
Fdez.	P1
Glez	P2
Glez	P3
López	P4



2 atributos
con el mismo
nombre!!!

“Préstamos y prestatarios”

prestatario x prestamo

prestamo.num_pre	nombre_suc	importe_pre	nombre_cli	prestatario.num_pre
P1	SUC1	100	Fdez.	P1
P1	SUC1	100	Glez	P2
P1	SUC1	100	Glez	P3
P1	SUC1	100	López	P4
...
P4	SUC4	400	López	P4

4.2. Operadores del Álgebra Relacional

➔ *Producto Cartesiano*

□ Producto Cartesiano (X)

prestamo

num_pre	nombre_suc	importe_pre
P1	SUC1	100
P2	SUC2	200
P3	SUC3	300
P4	SUC4	400

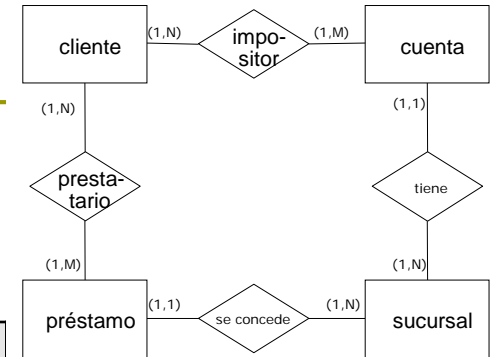
prestatario

nombre_cli	num_pre
Fdez.	P1
Glez	P2
Glez	P3
López	P4

“Nombre de Clientes e importe del préstamo de esos clientes”

$\pi_{\text{nombre_cli, importe_pre}} (\sigma_{\text{prestamo.num_pre=prestatario.num_pre}} (\text{prestatario} \times \text{prestamo}))$

importe_pre	nombre_cli
100	Fdez.
200	Glez
300	Glez
400	López



4.2. Operadores del Álgebra Relacional

➔ *Producto Cartesiano*

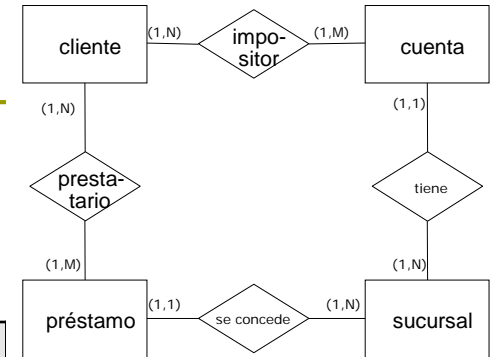
□ Producto Cartesiano (X)

prestamo

num_pre	nombre_suc	importe_pre
P1	SUC1	100
P2	SUC2	200
P3	SUC3	300
P4	SUC4	400

prestatario

nombre_cli	num_pre
Fdez.	P1
Glez	P2
Glez	P3
López	P4



“Clientes que tienen concedido un préstamo en la sucursal SUC1”

$\pi_{\text{nombre_cli}} (\sigma_{\text{prestamo.num_pre}=\text{prestatario.num_pre} \wedge \text{nombre_suc}=\text{“SUC1”}} (\text{prestatario} \times \text{prestamo}))$

prestamo.num_pre	nombre_suc	importe_pre	nombre_cli	prestatario.num_pre
P1	SUC1	100	Fdez.	P1
P1	SUC1	100	Glez	P2
P1	SUC1	100	Glez	P3
P1	SUC1	100	López	P4

4.2. Operadores del Álgebra Relacional

➔ *Producto Cartesiano*

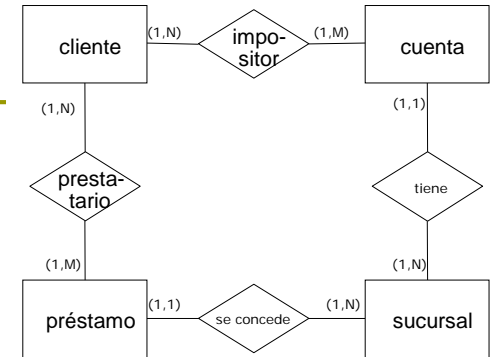
□ Producto Cartesiano (X)

cliente

nombre_cli	calle_cli	ciudad_cli
CLIENTE1	rúa nova	OURENSE
CLIENTE2	otero pedrayo	OURENSE
CLIENTE3	rep. argentina	VIGO

“Parejas de clientes”

¿ cliente X cliente ?



PROBLEMA: El producto cartesiano tiene un problema cuando:

- **se define sobre la misma relación**, ya que en la relación resultante se repetirían los nombres de los atributos y estos han de ser únicos.

SOLUCIÓN: Definir un alias (*operación de renombrado*) para la relación

4.2. Operadores del Álgebra Relacional

➔ *renombrado*

□ renombrado (ρ)

- Operador unario.
- Permite dar un nombre a una expresión del álgebra relacional, o aplicado sobre una relación, obtener la misma relación con un nombre nuevo.

$\rho_x (E) \rightarrow \mathbf{E}$ con el nombre \mathbf{x}

$\rho_{x(A1, A2, \dots, An)} (E) \rightarrow \mathbf{E}$ con el nombre \mathbf{x} y sus atributos como $A1, A2, \dots, An$

- **SQL:** ALIAS

4.2. Operadores del Álgebra Relacional

➔ *renombrado*

□ renombrado (ρ)

$\rho_x(E) \rightarrow E$ con el nombre x

$\rho_{x(A_1, A_2, \dots, A_n)}(E) \rightarrow E$ con el nombre x y sus atributos como A_1, A_2, \dots, A_n

cliente

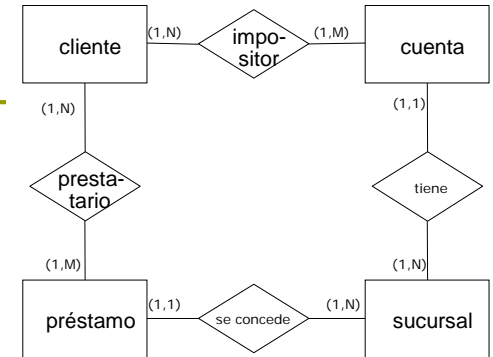
nombre_cli	calle_cli	ciudad_cli
CLIENTE1	rúa nova	OURENSE
CLIENTE2	otero pedrayo	OURENSE
CLIENTE3	rep. argentina	VIGO

“Parejas de clientes”

$\pi_{\text{cliente1.nombre_cli}, \text{cliente2.nombre_cli}} (\sigma_{\text{cliente1.nombre_cli} < \text{cliente2.nombre_cli}} (\rho_{\text{cliente1}}(\text{cliente}) \times \rho_{\text{cliente2}}(\text{cliente})))$

c1.nombre_cli	cliente2.nombre_cli
CLIENTE1	CLIENTE1
CLIENTE1	CLIENTE2
CLIENTE2	CLIENTE1
....

SIN
Predicado



Sentencia SQL:

```
SELECT cliente1.nombre_cli, cliente2.nombre_cli
FROM cliente as cliente1, cliente as cliente2
```

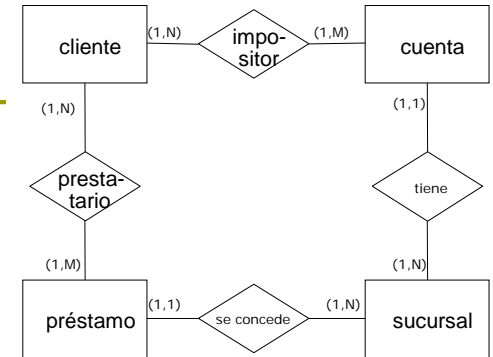
4.2. Operadores del Álgebra Relacional

➔ *renombrado*

□ renombrado (ρ)


cliente

nombre_cli	calle_cli	ciudad_cli
CLIENTE1	rúa nova	OURENSE
CLIENTE2	otero pedrayo	OURENSE
CLIENTE3	rep. argentina	VIGO



“Nombres de parejas de clientes que no viven en la misma ciudad”

$\pi_{c1.nombre_cli, c2.nombre_cli} (\sigma_{c1.nombre_cli < c2.nombre_cli \wedge c1.ciudad_cli \neq c2.ciudad_cli} (\rho_{c1}(cliente) \times \rho_{c2}(cliente)))$



c1.nombre_cli	c2.nombre_cli
CLIENTE1	CLIENTE3
CLIENTE2	CLIENTE3

Sentencia SQL:

```
SELECT c1.nombre_cli, c2.nombre_cli
FROM cliente as c1, cliente as c2
WHERE c1.ciudad_cli <> c2.ciudad_cli
```

4.2. Operadores del Álgebra Relacional

➔ *Join Natural*

□ Join Natural (\Join)

- Operador binario.

- Sean A y B dos relaciones, la reunión natural (o join natural)

(**A** \Join **B**) obtiene una relación cuyas tuplas son todas las tuplas de A concatenadas con todas las tuplas de B que **en los atributos comunes** (los que se llaman igual) **tienen los mismos valores**. Estos atributos comunes aparecen **una sola vez** en el resultado.

- $r \Join s = \pi_{R \cup S} (\sigma_{r.A1=s.A1 \wedge r.A2=s.A2 \wedge \dots \wedge r.An=s.An} (R \times S))$
donde $R \cap S = \{A_1, A_2, \dots, A_n\}$

R

A	B
a	1
b	2

S

B	C
1	x
1	y
3	z

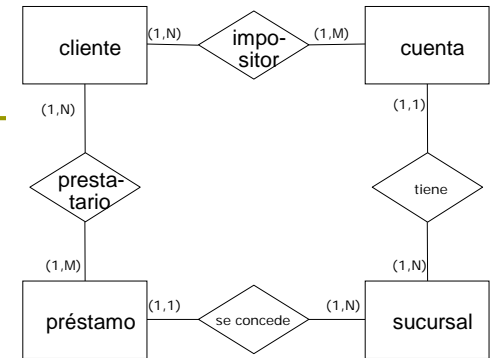
A	B	C
a	1	x
a	1	y

Reunión

4.2. Operadores del Álgebra Relacional

→ *Join Natural*

□ Join Natural (∞)



prestamo


num_pre	nombre_suc	importe_pre
P1	SUC1	100
P2	SUC2	200
P3	SUC3	300
P4	SUC4	400

prestatario

nombre_cli	num_pre
Fdez.	P1
Glez	P2
Glez	P3
López	P4

“Clientes que tienen concedido un préstamo y el importe del mismo”

$\pi_{\text{nombre_cli, importe_pre}} (\text{prestamo} \infty \text{prestatario})$



importe_pre	nombre_cli
100	Fdez.
200	Glez
300	Glez
400	López

Sentencia SQL

```
SELECT num_pre, nombre_cli  
FROM prestatario NATURALJOIN prestamo
```

4.2. Operadores del Álgebra Relacional

→ *Join Natural*

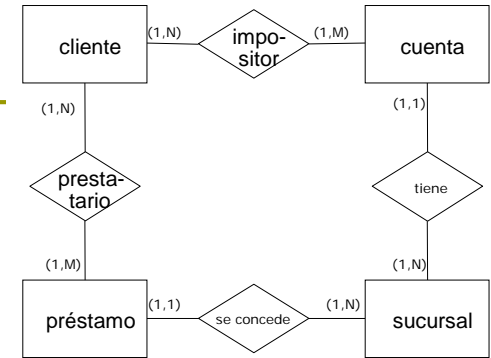
□ Join Natural (∞)

prestatario

nombre_cli	num_pre
Glez	P2
Glez	P3
Rguez	P4

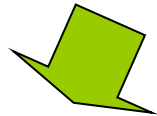
impositor

nombre_cli	num_cta
Fdez.	C1
Glez	C3
López	C2



“Hallar los clientes que tienen una cuenta y un préstamo”

$\pi_{\text{nombre_cli}} (\text{impositor} \infty \text{prestatario})$



nombre_cli
Glez

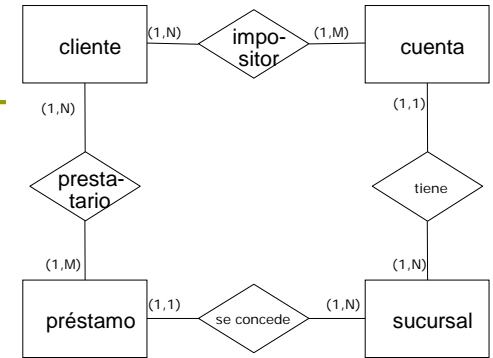
Sentencia SQL:

```
SELECT nombre_cli  
FROM prestatario NATURALJOIN impositor
```

4.2. Operadores del Álgebra Relacional

→ *Join Natural*

□ Join Natural (∞)



cliente

nombre_cli	calle_cli	ciudad_cli
Fdez	calleA	Madrid
López	calleB	Orense
Glez	calleC	Madrid
Glez	calleC	Madrid

cuenta

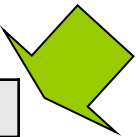
num_cta	nombre_suc	saldo
C1	Suc1	500
C2	Suc2	400

impositor

nombre_cli	num_cta
Fdez.	C1
López	C2

“Nombres de sucursales con clientes que tienen cuenta y viven en Madrid”

$\pi_{\text{nombre_suc}} (\sigma_{\text{ciudad_cli} = \text{“Madrid”}} (\text{cliente}) \infty \text{cuenta} \infty \text{impositor})$



nombre_suc
Suc1

Sentencia SQL:

```
SELECT nombre_suc
FROM cliente NATURALJOIN cuenta NATURALJOIN impositor
WHERE ciudad_cli=Madrid
```


4.2. Operadores del Álgebra Relacional

➔ *Join - Variantes*

□ Variantes:

- **THETAJOIN**: El atributo común no tiene el mismo nombre en las relaciones.
- **SEMIJOIN**: Se queda con las tuplas de la primera relación que participan en el join.
- **EQUIJOIN**: El predicado sólo consiste en condiciones de igualdad para el o los atributos especificados.
- **JOIN Natural**: Es un *equijoin* en el que las condiciones de igualdad se especifican para todos los atributos que tienen el mismo nombre en las relaciones.

4.2. Operadores del Álgebra Relacional

→ *Join - Variantes*

■ **THETAJOIN (θ)**: El atributo común no tiene el mismo nombre en las relaciones.

- Se debe especificar la condición de combinación.
- θ es el operador de combinación. Si la operación es "=" equivale a un EQUIJOIN
- Ejemplo:

$$\pi_{\text{nombre_cli}} (\text{sucursal} \bowtie_{\text{ciudad_suc } \theta \text{ ciudad_cli}} \text{cliente})$$

- Dado que los atributos de join son diferentes (no tienen el mismo nombre) el resultado no elimina ninguna de las columnas.
- θ puede combinar diferentes operaciones relacionándolas mediante $\wedge, \vee, \circ, \neg$.

4.2. Operadores del Álgebra Relacional

➔ *Join - Variantes*

- ▣ **SEMIJOIN** (\bowtie): Se queda con las tuplas de la primera relación que participan en el JOIN.

R1		R2			R1 \bowtie R2				R1 \bowtie R2	
a	b	b	c	d	a	b	c	d	a	b
1	A	A	4	X	1	A	4	X	1	A
2	B	C	5	Y	3	A	4	X	3	A
3	A				4	C	5	Y	4	C
4	C									

- ▣ **SQL**: `SELECT A.* FROM A, B WHERE A.campo=B.campo`

4.2. Operadores del Álgebra Relacional

→ División

□ División (\div)

- Operador binario.
- Sea **R** una relación de grado **m+n** (*dividendo*), **S** otra relación de grado **n** (*divisor*), **tal que los atributos del esquema de S están en el esquema de R**, el operador división produce una nueva relación de grado **m** (contendrá los atributos de **R** que no están en **S**), tal que al realizar el producto cartesiano con el divisor (**S**), todas las tuplas resultantes se encuentran en el dividendo.
- De manera informal: Para cada valor **x** de **R**, considérese el conjunto de valores **y** que aparecen en las tuplas de **R** con ese valor **x**. Si ese conjunto contiene todos los valores de **S**, el valor **x** se halla en el resultado de **R \div S**.

R		S
x	y	y
a	1	1
a	2	2
b	1	
b	2	
c	1	

R \div S

x
a
b

4.2. Operadores del Álgebra Relacional

➔ División

□ División (\div)

A	
P	R
p1	r1
p1	r2
p1	r3
p1	r4
p2	r1
p2	r2
p3	r2
p4	r2
p4	r4

B1	
R	
r2	

B2	
R	
r1	
r2	

B3	
R	
r1	
r2	
r4	

$A \div B1$	
P	
p1	
p2	
p3	
p4	

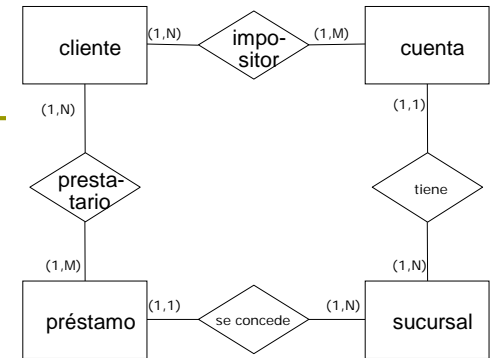
$A \div B2$	
P	
p1	
p2	

$A \div B3$	
P	
p1	

4.2. Operadores del Álgebra Relacional

➔ División

□ División (\div)



sucursal

nombre_suc	ciudad_suc	activos_suc
SUC1	OURENSE	100.000
SUC2	VIGO	200.000
SUC3	LUGO	300.000
SUC4	OURENSE	250.000

cuenta

num_cta	nombre_suc	saldo
C1	SUC1	500
C8	SUC4	900
C2	SUC2	400

impositor

nombre_cli	num_cta
Fdez.	C1
López	C2
Fdez.	C8

“Clientes que tengan abierta una cuenta en todas las sucursales ubicadas en OURENSE”

$\pi_{\text{nombre_cli, nombre_suc}} (\text{cuenta} \bowtie \text{impositor}) \div \pi_{\text{nombre_suc}} (\sigma_{\text{ciudad_suc}=\text{“OURENSE”}} (\text{sucursal}))$

Clientes con cuenta en una sucursal

nombre_cli	nombre_suc
Fdez.	SUC1
Fdez.	SUC4
López	SUC2

Sucursales de OURENSE

nombre_suc
SUC1
SUC4

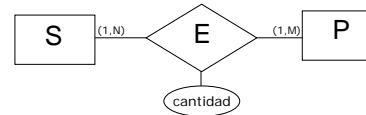
Resultado

nombre_cli
Fdez.

4.2. Operadores del Álgebra Relacional

➔ División

□ División (\div)



Proveedores: **S**(S#, NOMS, CIUDAD)
 Piezas: **P**(P#, NOMP, COLOR, PESO)
 Envíos: **E**(S#, P#, CANTIDAD)

E

S#	P#	CANTIDAD
PROV1	P1	20
PROV1	P2	30
PROV2	P1	15
PROV4	P2	50

P

P#	NOMP	COLOR	PESO
P1	Pieza1	ROJO	2
P2	Pieza2	VERDE	7

“Hallar los códigos de los proveedores que suministran todas las piezas”

$$\pi_{S\#, P\#} (E) \div \pi_{P\#} (P)$$

$\pi_{S\#, P\#} (E)$

S#	P#
PROV1	P1
PROV1	P2
PROV2	P1
PROV4	P2

\div

$\pi_{P\#} (P)$

P#
P1
P2

=

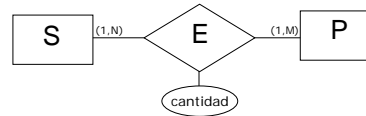
Resultado:

S#
PROV1

4.2. Operadores del Álgebra Relacional

➔ División

□ División (÷)



Proveedores: **S**(S#, NOMS, CIUDAD)
 Piezas: **P**(P#, NOMP, COLOR, PESO)
 Envíos: **E**(S#, P#, CANTIDAD)

E

S#	P#	CANTIDAD
PROV1	P1	20
PROV1	P2	30
PROV2	P1	15
PROV4	P2	50

P

P#	NOMP	COLOR	PESO
P1	Pieza1	ROJO	2
P2	Pieza2	VERDE	7

“Hallar los códigos de los proveedores que suministran todas las piezas rojas”

$$\pi_{S\#, P\#} (E) \div \pi_{P\#} (\sigma_{\text{color}=\text{"ROJO"}}(P))$$

$\pi_{S\#, P\#} (E)$

S#	P#
PROV1	P1
PROV1	P2
PROV2	P1
PROV4	P2

$\pi_{P\#} (\sigma_{\text{color}=\text{"ROJO"}}(P))$

÷

P#
P1

=

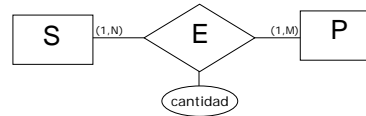
Resultado:

S#
PROV1
PROV2

4.2. Operadores del Álgebra Relacional

➔ División

□ División (÷)



Proveedores: **S**(S#, NOMS, CIUDAD)
 Piezas: **P**(P#, NOMP, COLOR, PESO)
 Envíos: **E**(S#, P#, CANTIDAD)

E

S#	P#	CANTIDAD
PROV1.	P1	20
PROV1	P2	30
PROV2	P1	15
PROV4	P2	50

P

P#	NOMP	COLOR	PESO
P1	Pieza1	ROJO	2
P2	Pieza2	VERDE	7

S

S#	NOMS	CIUDAD
PROV1	Juan Pérez	OU
PROV2	Raúl Blanco	PO
PROV4	Pedro Guerra	LU

“Hallar los nombres de los proveedores que suministran todas las piezas rojas”

$$\pi_{\text{NOMS}}((\pi_{\text{S\#, P\#}}(E) \div \pi_{\text{P\#}}(\sigma_{\text{color}=\text{"ROJO"}}(P))) \bowtie S)$$

Resultado:

NOMS
Juan Pérez
Raúl Blanco

4.2. Operadores del Álgebra Relacional

➔ *Asignación*

□ Asignación (\leftarrow)

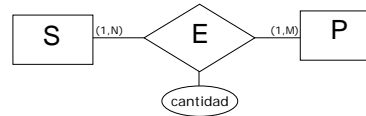
- Se utiliza para:

- Almacenar el resultado de una consulta en una nueva relación.
- Denominar resultados intermedios cuando se desea dividir una única operación compleja en una secuencia de operaciones más simples.
- Asignar un nuevo nombre a una relación existente.

4.2. Operadores del Álgebra Relacional

➔ *Asignación*

□ Asignación (\leftarrow)



Proveedores: **S**(S#, NOMS, CIUDAD)
Piezas: **P**(P#, NOMP, COLOR, PESO)
Envíos: **E**(S#, P#, CANTIDAD)

E		
S#	P#	CANTIDAD
PROV1.	P1	20
PROV1	P2	30
PROV2	P1	15
PROV4	P2	50

P			
P#	NOMP	COLOR	PESO
P1	Pieza1	ROJO	2
P2	Pieza2	VERDE	7

S		
S#	NOMS	CIUDAD
PROV1	Juan Pérez	OU
PROV2	Raúl Blanco	PO
PROV4	Pedro Guerra	LU

“Hallar los nombres de los proveedores que suministran todas las piezas rojas”

$$\text{temp1} \leftarrow \pi_{S\#, P\#} (E) \div \pi_{P\#} (\sigma_{\text{color}=\text{"ROJO"}}(P))$$
$$\text{resul} \leftarrow \pi_{\text{NOMS}}(\text{temp1} \bowtie S)$$

resul

NOMS
Juan Pérez
Raúl Blanco

4.3. Operadores Adicionales de consulta

- Las operaciones básicas se han extendido de varias formas:
 1. Permitiendo **operaciones aritméticas** en los atributos proyectados (*proyección generalizada*).
 2. Permitiendo **operaciones de agregación**, como el cálculo de la suma de los elementos de un conjunto, o su media.

4.3. Operadores Adicionales de consulta

→ *Proyección generalizada*

□ Proyección generalizada informacion_credito

nombre_cli	limite	saldo_credito
CLI1	100	20
CLI2	100	30
CLI3	100	15

“Importe disponible para cada cliente”

$\pi_{\text{nombre_cli}, (\text{limite} - \text{saldo_credito})}(\text{informacion_credito})$

El atributo resultante de la expresión *limite – saldo_credito* no tiene nombre asignado => Se puede aplicar la operación de **renombrado** al resultado de la proyección generalizada para darle un nombre.

Como convenio, se utiliza la siguiente notación:

$\pi_{\text{nombre_cli}, (\text{limite} - \text{saldo_credito})} \underline{\text{AS}} \text{ credito_disponible}(\text{informacion_credito})$
resultado

nombre_cli	credito_disponible
CLI1	80
CLI2	70
CLI3	85

4.3. Operadores Adicionales de consulta

➔ *Funciones de Agregación*

❑ Funciones de Agregación

- Toman un conjunto de valores y devuelven como resultado **un único valor**.
- Las funciones disponibles son:
 - ❑ **sum**: devuelve la suma de un conj. de valores
 - ❑ **avg**: devuelve la media de un conj. de los valores
 - ❑ **count**: devuelve el n° de elementos de un conj. de valores. **Count distinct**: devuelve los distintos.
 - ❑ **min**: devuelve el valor mínimo de un conj de valores
 - ❑ **max**: devuelve el valor máximo de un conj. de valores
- Los conjuntos sobre los que actúan pueden contener valores repetidos y el orden de los elementos no importa, son ***multiconjuntos***.

4.3. Operadores Adicionales de consulta

➔ *Funciones de Agregación*

□ Funciones de Agregación

$G_{F1(A1), F2(A2), \dots, Fm(Am)}(E)$ ➔ Funciones Agregación

trabajo_por_horas

nombre_empleado	nombre_sucursal	sueldo
Cana	Leganés	1500
Cascallar	Navacerrada	5300
Catalán	Leganés	1600
Díaz	Centro	1300
Fernández	Navacerrada	1500

“Suma total de los sueldos de los empleados del banco”

$G_{\text{sum(sueldo) as suma_sueldos}}(\text{trabajo_por_horas})$

resultado

suma_sueldos
11200

Sentencia SQL:

```
SELECT sum(sueldo) FROM trabajo_por_horas
```

4.3. Operadores Adicionales de consulta

➔ *Funciones de Agregación*

□ Funciones de Agregación

$G_{F1(A1), F2(A2), \dots, Fm(Am)}(E)$ ➔ **Funciones Agregación**

trabajo_por_horas

nombre_empleado	nombre_sucursal	sueldo
Cana	Leganés	1500
Cascallar	Navacerrada	5300
Catalán	Leganés	1600
Díaz	Centro	1300
Fernández	Navacerrada	1500

“Número de sucursales distintas que aparecen en trabajo_por_horas”

$G_{\text{count-distinct(nombre_sucursal) as num_sucursales}}(\text{trabajo_por_horas})$
resultado

mum_sucursales
3

Sentencia SQL:

```
SELECT count (DISTINCT nombre_sucursal) as num_sucursales  
FROM trabajo_por_horas
```


4.3. Operadores Adicionales de consulta

➔ *Funciones de Agregación*

▣ Funciones de Agregación

Atributos Agrupación ← G_1, G_2, \dots, G_n G $F_1(A_1), F_2(A_2), \dots, F_m(A_m)$ (E) Funciones Agregación

trabajo por horas

nombre_empleado	nombre_sucursal	sueldo
Cana	Leganés	1500
Cascallar	Navacerrada	5300
Catalán	Leganés	1600
Díaz	Centro	1300
Fernández	Navacerrada	1500

trabajo por horas

nombre_empleado	nombre_sucursal	sueldo
Cana	Leganés	1500
Catalán	Leganés	1600
Díaz	Centro	1300
Cascallar	Navacerrada	5300
Fernández	Navacerrada	1500

“Suma total de los sueldos de los empleados del banco en cada sucursal”

resultado nombre_sucursal G $\text{sum(sueldo) as suma_sueldos}$ (trabajo_por_horas)

nombre_sucursal	suma_sueldos
Leganés	3100
Centro	1300
Navecerrada	6800

Sentencia SQL:

```
SELECT nombre_sucursal, sum(sueldo) as suma_sueldos
FROM trabajo_por_horas
GROUP BY nombre_sucursal
```

4.4. Operadores adicionales de modificación

- ❑ Hasta el momento el Álgebra Relacional se ha utilizado para extraer información de la base de datos.
- ❑ El Álgebra Relacional también permite modificar información de la base de datos:
 - Insertar, Eliminar, Modificar
 - Se lleva a cabo mediante el operador de asignación

4.4. Operadores adicionales de modificación

➔ *Borrado*

□ Borrado

- Las solicitudes de borrado se expresan igual que las consultas, pero en lugar de *mostrar* las tuplas, se eliminan.
- La expresión que permite borrar tuplas es la siguiente:
 - $r \leftarrow r - E$
 - **r**: Relación
 - **E**: Consulta del álgebra relacional

4.4. Operadores adicionales de modificación

➔ Borrado

□ Borrado

impositor

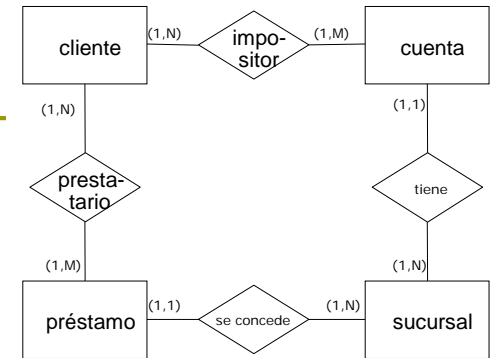
nombre_cli	num_cta
Fdez.	C1
Glez	C3
López	C2
Fdez.	C8

“Borrar todas las cuentas de Fdez.”

$\text{impositor} \leftarrow \text{impositor} - \sigma_{\text{nombre_cli}=\text{“Fdez.”}}(\text{impositor})$

resultado

nombre_cli	num_cta
Glez	C3
López	C2



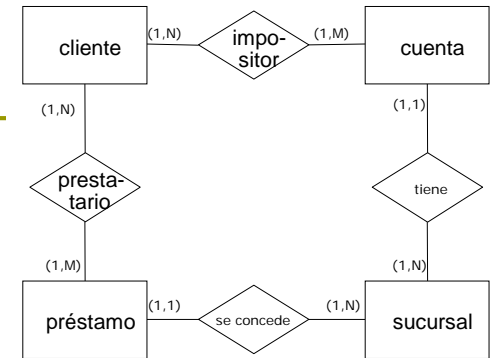
4.4. Operadores adicionales de modificación

➔ Borrado

□ Borrado

prestamo

num_pre	nombre_suc	importe_pre
P1	SUC1	10
P2	SUC2	200
P3	SUC3	30
P4	SUC4	400



“Borrar todos los préstamos con importes entre 0 y 50”

$\text{prestamo} \leftarrow \text{prestamo} - \sigma_{\text{importe_pre} \geq 0 \wedge \text{importe_pre} \leq 50}(\text{prestamo})$

resultado

num_pre	nombre_suc	importe_pre
P2	SUC2	200
P4	SUC4	400

4.4. Operadores adicionales de modificación

➔ *Inserción*

□ Inserción

- Para insertar tuplas en una relación hay que especificar la **tupla** que se va a insertar o escribir una consulta cuyo resultado sea el **conjunto de tuplas** que se va a insertar
- La expresión que permite insertar tuplas es la siguiente:
 - $r \leftarrow r \cup E$
 - **r**: Relación
 - **E**: Consulta del álgebra relacional
- Se ha de tener en cuenta que:
 - El valor de los atributos de las tuplas insertadas debe ser miembro del dominio de cada atributo.
 - Las tuplas han de tener el grado correcto.

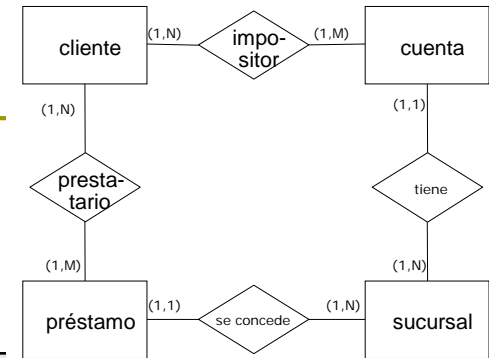
4.4. Operadores adicionales de modificación

➔ *Inserción*

□ Inserción

impositor	
nombre_cli	num_cta
Fdez.	C1
Fdez.	C8

cuenta		
num_cta	nombre_suc	saldo
C1	Suc1	500
C8	Suc2	400



“Insertar el hecho de que Glez tiene 1000 euros en la cuenta C27 de la sucursal X”

$\text{cuenta} \leftarrow \text{cuenta} \cup \{(C27, \text{"X"}, 1000)\}$
 $\text{impositor} \leftarrow \text{impositor} \cup \{(\text{Glez}, C27)\}$

Relación constante

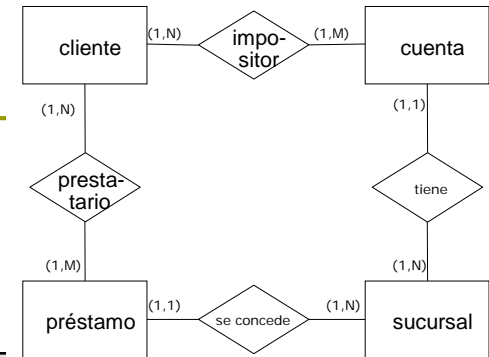
4.4. Operadores adicionales de modificación

➔ *Inserción*

□ Inserción

impositor	
nombre_cli	num_cta
Fdez.	C1
Fdez.	C8

cuenta		
num_cta	nombre_suc	saldo
C1	Suc1	500
C8	Suc2	400



“Insertar cuentas de ahorro con 200 euros como regalo a todos los clientes con prestamos concedidos en la sucursal X. El número de la cuenta de ahorro será el mismo que el número de préstamo”

$$r1 \leftarrow \sigma_{\text{nombre_suc}="X"} (\text{prestatarario} \bowtie \text{préstamo})$$
$$\text{cuenta} \leftarrow \text{cuenta} \cup (\pi_{\text{num_cta}} (r1) \times (200) \times \pi_{\text{nombre_suc}} (r1))$$
$$\text{impositor} \leftarrow \text{impositor} \cup \pi_{\text{nombre_cli}, \text{num_pre}} (r1)$$

4.4. Operadores adicionales de modificación

➔ *Actualización*

□ Actualización

- Es útil cuando se desea modificar un valor de una tupla sin modificar todos los valores de esa tupla.
- La actualización se lleva a cabo haciendo uso del operador de **proyección generalizada**. La expresión que permite **insertar** tuplas es la siguiente:

$$□ r \leftarrow \pi_{F_1, F_2}(r)$$

- **r**: Relación
- **F_i**: es el i-ésimo atributo de r

4.4. Operadores adicionales de modificación

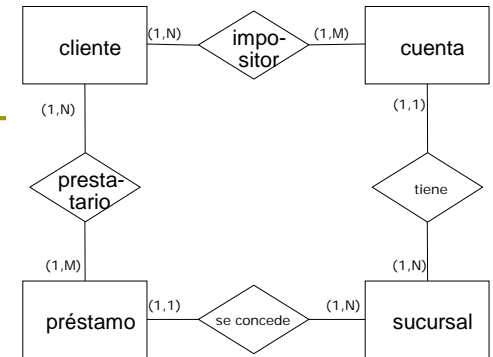
➔ *Actualización*

□ Actualización

cuenta		
num_cta	nombre_suc	saldo
C1	Suc1	500
C8	Suc2	400

“Aumentar los saldos de los clientes en un 5%”

$\text{cuenta} \leftarrow \pi_{\text{nombre_suc}, \text{num_cta}, (\text{saldo} * 1.05)} (\text{cuenta})$



4.4. Operadores adicionales de modificación

➔ *Actualización*

□ Actualización

- Si en lugar de actualizar todas las tuplas, sólo se desea actualizar ciertas tuplas, se utiliza la siguiente expresión:

- $$r \leftarrow (\pi_{F_1, F_2, \dots, F_n} (\sigma_P(r))) \cup (r - \sigma_P(r))$$

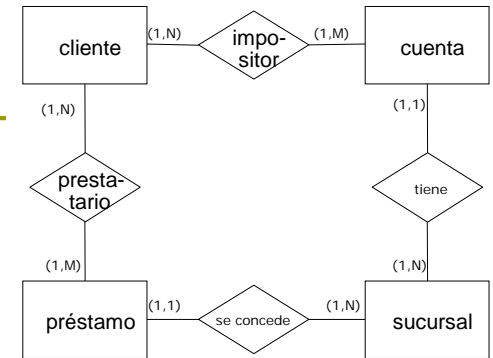
- r : relación
- P : condición de selección
- Es decir, es la unión de lo que modificó más lo que NO es modificado

4.4. Operadores adicionales de modificación

→ Actualización

□ Actualización

cuenta		
num_cta	nombre_suc	saldo
C1	Suc1	500
C8	Suc2	400



“Aumentar los saldos de los clientes en un 6% si el saldo es superior a 10000€”

$$\text{cuenta} \leftarrow \pi_{\text{nombre_suc}, \text{num_cta}, (\text{saldo} * 1.06)} (\sigma_{\text{saldo} > 10000} (\text{cuenta}))$$

$$\cup (\text{cuenta} - \sigma_{\text{saldo} > 10000} (\text{cuenta}))$$

VI. Bibliografía

- ▣ [deMP99] de Miguel, A.; Piattini, M. **Fundamentos y modelos de bases de datos** (2ª edición). Madrid. Rama, 1999.
[cap. 6]
- ▣ [EN07] Ramez A. Elmasri, Shamkant B. Navathe. **Fundamentos de Sistemas de Bases de Datos** (5ª edic.). Prentice-Hall. 2007.
[cap. 6]