Unidad 3.2: BBDD relacionales

Álgebra relacional y SQL 1

INDICE

- Algebra relacional
  - Selección
  - Proyección
  - Unión
  - Diferencia de conjuntos
  - Producto Cartesiano
  - Renombramiento
  - Intersección
  - Join
  - Asignación
  - Modificaciones
- SQL
  - SELECT
  - Renombramiento
  - Operaciones sobre conjuntos: union, intersect, except
  - Join
  - Modificación de la base de datos

Referencias: Silberschatz – Capítulos 2

Lenguajes de consulta

- Lenguaje de consulta es un lenguaje en el que el usuario solicita información de la base de datos.
- Dos tipos:
  - Procedimentales ⇒ usuario especifica las operaciones a realizar
    - Ej.: Algebra relacional
  - No procedimentales ⇒ usuario describe "lo que necesita", no el modo de conseguirlo.
    - Ej.: Cálculo relacional de tuplas y el de dominios
- Lenguaje → también incluye componentes para modificación de la base de datos.

Algebra relacional

- Conjunto de operaciones básicas del modelo relacional.
- Lenguaje consulta procedimental r(R)
- El resultado de cualquier operación (unaria o binaria) es una nueva relación, se trata, por tanto, de operaciones cerradas que se pueden componer.
- Seis operadores básicos
  - **Selección:** $\sigma_{p}(r)$, siendo P predicado de la selección y r relación.
  - **Proyección:** $\prod_{A1, A2, \ldots, Ak}(r)$, siendo A_1, $A_{2\ldots}$, A_k atributos y r relación
  - **Unión:** $r \cup s$
  - **Diferencia de conjuntos:** $r - s$
  - **Producto Cartesiano:** $r \times s$
  - **Renombramiento:** $\rho_{x\ (A1, A2, \ldots, An)}(E)$
  - **Asignación:** $\leftarrow$

.

Álgebra relacional
Operación de selección

- Notación:
  - $\sigma_p(r)$, donde
  - p predicado de la selección y r relación
  - p es una expresión lógica: $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
- Se define como:

  $\sigma_p(r) = \{t \mid t \in r$ **y** $p(t)\}$, donde p está compuesto por un término lógico con formato:

  <atributo> op (<atributo> o <constante>)

  donde op es: =, ≠, >, ≥. <. ≤
- Funcionamiento:
  - Se aplica p a cada tupla t de r
  - Si p(t) VERDADERO, la tupla t se **selecciona.**
  - Si el resultado de la operación de comparación es un valor nulo ⇒ p(t) se evalúa como FALSO.
- La selección:
  - Es conmutativa: $\sigma_{\text{<condición1>}}(\sigma_{\text{<condición2>}}(R)) = \sigma_{\text{<condición2>}}(\sigma_{\text{<condición1>}}(R))$
  - Se pueden combinar en cascada: $\sigma_{\text{<condición1>}}(\sigma_{\text{<condición2>}}(\ldots(\sigma_{\text{<condiciónn>}}(R))\ldots)) = \sigma_{\text{<condición1>}}$ **AND** $\sigma_{\text{<condición2>}}$ **AND** . . . **AND** $\sigma_{\text{<condiciónn>}}(R)$

Álgebra relacional
Operación selección

- Ejemplo: Información sobre préstamos de la sucursal de Navacerrada

préstamo

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

σ nombre_sucursal="Navacerrada"(préstamo)

número-préstamo	*nombre-sucursal*	*importe*
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300

Álgebra relacional
Operación de proyección

- Notación:
  - $\Pi_{A1, A2, \ldots, Ak}\ (r)$, donde A_1, $A_{2\ldots}$, A_k son atributos y r es la relación.
- El resultado es una relación de k columnas eliminando de r las que no están en la lista
- Elimina cualquier fila duplicada (son conjuntos).
  - Realiza una ordenación de tuplas para detectar duplicados -> capacidad de procesamiento
- Ejemplo: Importe de cada uno de los préstamos.

préstamo

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

$\Pi_{numero\_prestamo, importe}$ (préstamo)

número-préstamo	*importe*
P-11	900
P-14	1.500
P-15	1.500
P-16	1.300
P-17	1.000
P-23	2.000
P-93	500

Álgebra relacional
Operación proyección

- Ejemplo: Sexo y sueldo de los empleados

EMPLEADO

Nombre	Apellido1	Apellido2	Dni	FechaNac	Dirección	Sexo	Sueldo	SuperDni	Dno
José	Pérez	Pérez	123456789	01-09-1965	Eloy I, 98	H	30000	333445555	5
Alberto	Campos	Sastre	333445555	08-12-1955	Avda. Ríos, 9	H	40000	888665555	5
Alicia	Jiménez	Celaya	999887777	12-05-1968	Gran Vía, 38	M	25000	987654321	4
Juana	Sainz	Oreja	987654321	20-06-1941	Cerquillas, 67	M	43000	888665555	4
Fernando	Ojeda	Ordóñez	666884444	15-09-1962	Portillo, s/n	H	38000	333445555	5
Aurora	Oliva	Avezuela	453453453	31-07-1972	Antón, 6	M	25000	333445555	5
Luis	Pajares	Morera	987987987	29-03-1969	Enebros, 90	H	25000	987654321	4
Eduardo	Ochoa	Paredes	888665555	10-11-1937	Las Peñas, 1	H	55000	NULL	1

$\Pi_{\text{Sexo, Sueldo}}$(EMPLEADO)

Sexo	Sueldo
H	30000
H	40000
M	25000
M	43000
H	38000
H	25000
H	55000

Álgebra relacional
Composición de operaciones relacionales

- Podemos anidar expresiones de álgebra relacional en una única expresión
- El resultado de una operación relacional es otra relación
- Ejemplo: Encontrar los nombres de los clientes que viven en Peguerinos.

cliente

nombre_cliente	*calle_cliente*	*ciudad_cliente*
Abril	Preciados	Valsaín
Amo	Embajadores	Arganzuela
Badorrey	Delicias	Valsaín
Fernández	Jazmín	León
Gómez	Carretas	Cerceda
González	Arenal	La Granja
López	Mayor	Peguerinos
Pérez	Carretas	Cerceda
Rodríguez	Yeserías	Cádiz
Rupérez	Ramblas	León
Santos	Mayor	Peguerinos
Valdivieso	Goya	Vigo

1. $\sigma_{\text{ciudad\_cliente="Peguerinos"}}(\text{cliente})$

Nombre_cliente	Calle_cliente	Ciudad_cliente
López	Mayor	Peguerinos
Santos	Mayor	Peguerinos

2. $\Pi_{\text{nombre\_cliente}}(\sigma_{\text{ciudad\_cliente="Peguerinos"}}(\text{cliente}))$

Nombre_cliente
López
Santos

Proyección generalizada

- Extiende la operación de proyección permitiendo funciones aritméticas en la lista de proyección

$$\Pi_{F1, F2, \ldots, Fn}(E)$$, siendo

  - E expresión del álgebra relacional
  - F_1, F_2, …, F_n expresiones aritméticas que involucran constantes y atributos en el esquema de E

- Ejemplo: Determinar el importe disponible de cada persona

Información_crédito

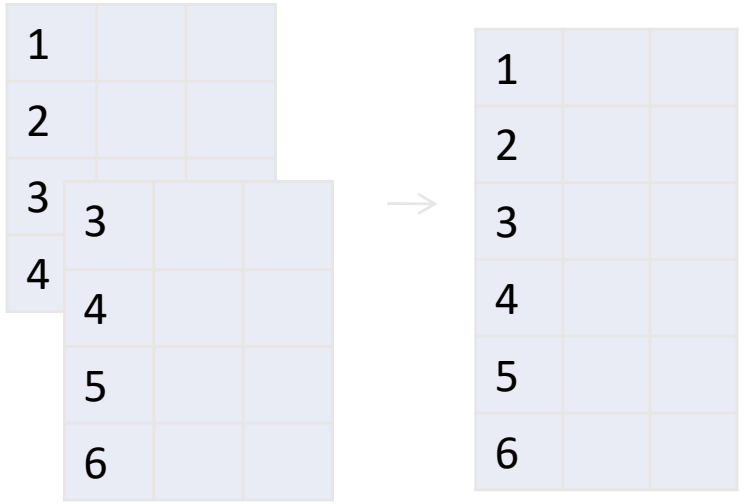
nombre-cliente	*límite*	*saldo-crédito*
Gómez	2.000	400
López	1.500	1.500
Pérez	2.000	1.750
Santos	6.000	700

  -
  - $\Pi_{\text{nombre-cliente, limite - saldo-crédito as crédito-disponible}}$ (información_crédito)

crédito-disponible

nombre-cliente	*crédito-disponible*
Gómez	1.600
López	0
Pérez	250
Santos	5.300

Operación unión



- Notación: $r \cup s$
- Se define como:

$$r \cup s = \{t \mid t \in r \text{ o } t \in s\}$$

- Para $r \cup s$ sea válida, r y s deben ser **compatibles**:
  1. Tienen el mismo número de atributos
  2. El dominio del atributo i-ésimo de r y s es igual, ~~∀~~i
- No hay valores duplicados

Operación unión

- Ejemplo: Nombre de todos los clientes, ya sea que tengan un préstamo o una cuenta
nombre cliente	número préstamo
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

nombre cliente	número cuenta
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

  or

$$\Pi_{\text{nombre\_cliente}}(\text{prestatario}) \cup \Pi_{\text{nombre\_cliente}}(\text{impositor})$$

nombre-cliente
Abril
Fernández
Gómez
González
López
Pérez
Rupérez
Santos
Sotoca
Valdivieso

Operación diferencia de conjuntos

- Notación r − s
- Definido como:

  $r - s = \{t \mid t \in r \textbf{ y } t \notin s\}$

- Las relaciones deben de ser **compatibles**
- Ejemplo: Clientes que tienen una cuenta, pero no un préstamo.

pr

nombre cliente	*número cuenta*
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

$\Pi_{nombre\_cliente}$ (

nombre-cliente
Abril
González
Rupérez

$\Pi_{nombre\_cliente}$ (prestatario)

Operación producto cartesiano

- Notación r x s
- Se define como:

  r x s = {t, q | t ∈ r **y** q

prestatario

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

préstamo

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

- Combina la información de 2 re
  - Si r tiene n atributos y s, m→ r x s tiene n+m atributos cuyo nombre es el de los atributos originales
- Ejemplo:

  r = prestatario x préstamo

  = (prestatario.nombre-cliente, prestatario.númeropréstamo,préstamo.nombre-sucursal, préstamo.número-préstamo, préstamo.importe)

  - Si no hay ambigüedad → simplificar:

  r = (nombre-cliente, prestatario.número-préstamo, nombre-sucursal, préstamo.número-préstamo, importe)

Operación producto cartesiano

préstamo x prestatario

n*m tuplas (una tupla de cada relación)

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-11	Collado Mediano	900
Santos	P-17	P-14	Centro	1.500
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Santos	P-17	P-17	Centro	1.000
Santos	P-17	P-23	Moralzarzal	2.000
Santos	P-17	P-93	Becerril	500
Gómez	P-23	P-11	Collado Mediano	900
Gómez	P-23	P-14	Centro	1.500
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
Gómez	P-23	P-17	Centro	1.000
Gómez	P-23	P-23	Moralzarzal	2.000
Gómez	P-23	P-93	Becerril	500
López	P-15	P-11	Collado Mediano	900
López	P-15	P-14	Centro	1.500
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
López	P-15	P-17	Centro	1.000
López	P-15	P-23	Moralzarzal	2.000
López	P-15	P-93	Becerril	500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Valdivieso	P-17	P-11	Collado Mediano	900
Valdivieso	P-17	P-14	Centro	1.500
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Valdivieso	P-17	P-17	Centro	1.000
Valdivieso	P-17	P-23	Moralzarzal	2.000
Valdivieso	P-17	P-93	Becerril	500
Fernández	P-16	P-11	Collado Mediano	900
Fernández	P-16	P-14	Centro	1.500
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300
Fernández	P-16	P-17	Centro	1.000
Fernández	P-16	P-23	Moralzarzal	2.000
Fernández	P-16	P-93	Becerril	500

Operación producto cartesiano: Ejemplo

- Averiguar los nombres de todos los clientes con un préstamo en Navacerrada.
  - 1º - Producto cartesiano de prestatario y préstamo

$$prestatario \times préstamo$$

  - 2º -Seleccionar las tuplas de nombre-sucursal "Navacerrada".

$$\sigma_{nombre\_sucursal = \text{"Navacerrada"}} (prestatario \times préstamo).$$

  - 3º - Eliminar tuplas que no corresponden al mismo préstamo.

$$\sigma_{prestatario.número\_préstamo = préstamo.número\_préstamo} (\sigma_{nombre\_sucursal = \text{"Navacerrada"}} (prestatario \times préstamo))$$

  - 4º - Proyección para eliminar los atributos no necesarios.

$$\Pi_{nombre\_cliente} (\sigma_{prestatario.número\_préstamo = préstamo.número\_préstamo} (\sigma_{nombre\_sucursal = \text{"Navacerrada"}} (prestatario \times préstamo)))$$

prestatario

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

préstamo

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

$\sigma_{nombre\_sucursal = \text{"Navacerrada"}} (prestatario \times préstamo)$.

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
Sotoca	P-14	P-15	Navacerrada	1.500
Sotoca	P-14	P-16	Navacerrada	1.300
Pérez	P-93	P-15	Navacerrada	1.500
Pérez	P-93	P-16	Navacerrada	1.300
Gómez	P-11	P-15	Navacerrada	1.500
Gómez	P-11	P-16	Navacerrada	1.300
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300

nombre-cliente
Férnandez
López

Operación renombramiento

- Notación: $\rho_x(E)$
- Devuelve resultado de la expresión E bajo el nombre X
- Permite:
  - Poner nombres a los resultados de las expresiones del álgebra relacional
  - Referir a una relación por más de un nombre
- Si la expresión E tiene n atributos, entonces:
  - $\rho_{x(A1, A2, \ldots, An)}(E)$ (o X(A1,A2, An) ←E), devuelve el resultado de la expresión E bajo el nombre X y con los atributos renombrados como A1, A2, …., An.

Operación renombramiento

- Ejemplo: Buscar el máximo saldo de cuenta del banco
  - 1° - Calcular una relación intermedia *d* que contiene todos los saldos que no son el máximo.
    - Como vamos a comparar cada tupla de cuenta con otra tupl de la misma relación, renombramos como d.

$$\Pi_{cuenta.saldo}\left(\sigma_{cuenta.saldo < d.saldo}\left(cuenta \times \rho_d\left(cuenta\right)\right)\right)$$

    - En esta relación obtenemos los saldos para los que hay otro saldo mayor en cuenta.
  - 2° - Realizar la diferencia entre la proyección del saldo de las cuentas y esta relación intermedia.

$$\Pi_{saldo}\left(cuenta\right) - \Pi_{cuenta.saldo}\left(\sigma_{cuenta.saldo < d.saldo}\left(cuenta \times \rho_d\left(cuenta\right)\right)\right)$$

número_cuenta	*nombre_sucursal*	*saldo*
C-101	Centro	500
C-102	Navacerrada	400
C-201	Galapagar	900
C-215	Becerril	700
C-217	Galapagar	750
C-222	Moralzarzal	700
C-305	Collado Mediano	350

Relación cuenta

saldo
500
400
700
750
350

número_cuenta	*nombre_sucursal*	*saldo*
C-101	Centro	500
C-102	Navacerrada	400
C-201	Galapagar	900
C-215	Becerril	700
C-217	Galapagar	750
C-222	Moralzarzal	700
C-305	Collado Mediano	350

Relación d

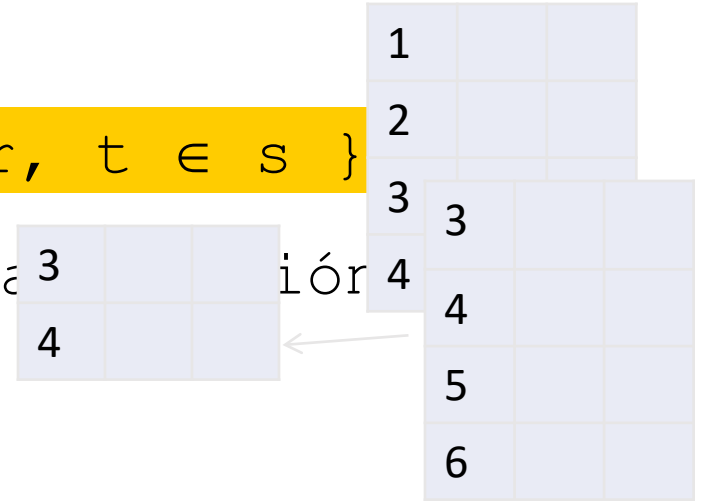
saldo
900

Expresiones del álgebra relacional

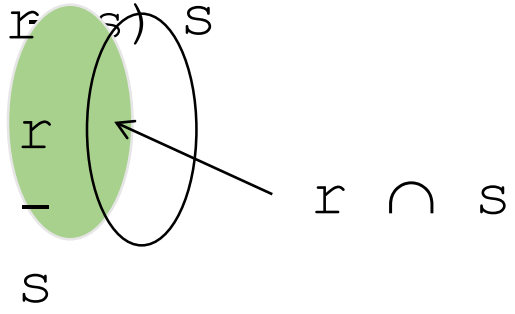
- Sean E_1 y E_2 expresiones del álgebra relacional. Las siguientes son también expresiones del álgebra relacional:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, donde P un predicado sobre atributos de E_1
  - $\prod_s (E_1)$, donde S es una lista de atributos de E_1
  - $\rho_{x\ E_1} (E_1)$, donde x es el nuevo nombre para el resultado

Operación intersección

- Notación: r ∩ s
- Definido como: r ∩ s ={ t | t ∈ r, t ∈ s }
- r y s han de ser compatibles (para[illegible]ión
  - Mismo número de atributos
  - Atributos en el mismo dominio



- Notad que:
  - r ∩ s = r - (r - s)



Operación intersección

Ejemplo: Nombre de clientes que tienen préstamos y cuentas

prestatario

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

impositor

nombre cliente	*número cuenta*
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

$$\Pi_{nombre\text{-}cliente}(prestatario) \cap \Pi_{nombre\text{-}cliente}(impositor)$$

nombre-cliente
Gómez
Pérez
Santos

NATURAL JOIN (concatenación o reunión natural)

- Notación: r⋈s
- Sean dos relaciones r(R) y s(S), r ⋈ s es una relación del esquema R U S definida por:

$$r \bowtie s = \Pi_{R \cup S} \left(\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n} (r \times s) \right)$$

donde $R \cap S = \{A_1, A_2, \ldots, A_n\}$.

- Relación binaria que permite combinar varias selecciones y un producto cartesiano en una sola operación de la siguiente manera:
  - 1°. Producto cartesiano de las dos relaciones: r x s
  - 2°. Selección forzando la igualdad de los atributos que aparecen en ambos esquemas de relación (R ∩ S)
- Propiedade
  - Es asociat

$$(cliente \bowtie cuenta) \bowtie impositor$$
$$cliente \bowtie (cuenta \bowtie impositor)$$

  - Si R ∩ S = ∅ $r \bowtie s = r \times s$

Natural join Ejemplo

$$r \bowtie s = \Pi_{R \cup S} \left(\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n} (r \times s) \right)$$

$$R \cap S = \{A_1, A_2, \ldots, A_n\}.$$

- Hallar el nombre de los clientes que tienen concedido un préstamo y averiguar el número de préstamo e importe.

$$\Pi_{\text{nombre-cliente, número-préstamo, importe}} (\text{prestatario} \bowtie \text{préstamo})$$

prestatario

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

préstamo

número-préstamo	*importe*
P-11	900
P-14	1.500
P-15	1.500
P-16	1.300
P-17	1.000
P-23	2.000
P-93	500

nombre-cliente	*número-préstamo*	*importe*
Fernández	P-16	1.300
Gómez	P-23	2.000
Gómez	P-11	900
López	P-15	1.500
Pérez	P-93	500
Santos	P-17	1.000
Sotoca	P-14	1.500
Valdivieso	P-17	1.000

Notad que es equivalente a:

$$\Pi_{nombre\_cliente, préstamo.número\_préstamo, importe} \left(\sigma_{prestatario.número\_préstamo = préstamo.número\_préstamo} (prestatario \times préstamo) \right)$$

Operación θ-Join o reunión theta/zeta

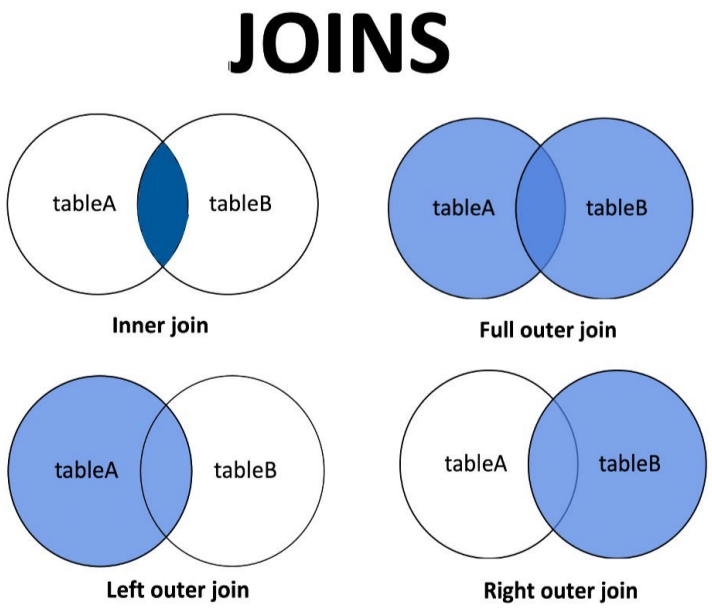
- Notación: $r \bowtie_\theta s = \sigma_\theta (r \times s)$
- Sean las relaciones r(R) y s(S) y sea θ un predicado de los atributos del esquema R ∪ S. La operación θ-join se define como:

$$r \bowtie_\theta s = \sigma_\theta (r \times s)$$

- θ-join es una extensión de la reunión natural
- La reunión zeta combina producto cartesiano y selección en una operación.
- Si θ es una comparación de igualdad, se denomina **EQUIJOIN.**

OUTER join o reunión externa

- El resultado de un natural JOIN no incluye información que no esté presente en ambas relaciones.
- Outer join es una extensión que evita la pérdida de información añadiendo valores null:
  - null significa valor desconocido o que no existe
  - todas las comparaciones en que interviene null son FALSE por definición
- Realiza el join y añade las tuplas de una relación que no coincide con el atributo de la reunión
- Puede hacerse la reunión externa por la:
  - Izquierda: left outer join (⟕)
  - Derecha: rigth outer join (⟖)
  - Completa: full outer join (⟗)



OUTER join o reunión externa Ejemplo

empleado

nombre-empleado	*calle*	*ciudad*
Segura	Tebeo	La Loma
Domínguez	Viaducto	Villaconejos
Gómez	Bailén	Alcorcón
Valdivieso	Fuencarral	Móstoles

trabajo-a-tiempo-completo

nombre-empleado	*nombre-sucursal*	*sueldo*
Segura	Majadahonda	1.500
Domínguez	Majadahonda	1.300
Barea	Fuenlabrada	5.300
Valdivieso	Fuenlabrada	1.500

- Natural join: pierdo info sobre Gómez (calle y ciudad) y sobre Barea (nombre de sucursal y sueldo)

empleado ⋈ *trabajo-a-tiempo-completo*

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500

- left outer join o reunión externa por la izquierda: pierdo información de Barea

empleado ⟕ *trabajo-a-tiempo-completo*

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Gómez	Bailén	Alcorcón	*nulo*	*nulo*

OUTER join o reunión externa Ejemplo

empleado

nombre-empleado	calle	ciudad
Segura	Tebeo	La Loma
Domínguez	Viaducto	Villaconejos
Gómez	Bailén	Alcorcón
Valdivieso	Fuencarral	Móstoles

trabajo-a-tiempo-completo

nombre-empleado	nombre-sucursal	sueldo
Segura	Majadahonda	1.500
Domínguez	Majadahonda	1.300
Barea	Fuenlabrada	5.300
Valdivieso	Fuenlabrada	1.500

- Reunión externa por la derecha: pierdo la información de Gómez

empleado ⟖ *trabajo-a-tiempo-completo*

nombre-empleado	calle	ciudad	nombre-sucursal	sueldo
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Barea	*nulo*	*nulo*	Fuenlabrada	5.300

- full outer join o reunión externa complete: no pierdo información

empleado ⟗ *trabajo-a-tiempo-completo*

nombre-empleado	calle	ciudad	nombre-sucursal	sueldo
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Gómez	Bailén	Alcorcón	*nulo*	*nulo*
Barea	*nulo*	*nulo*	Fuenlabrada	5.300

Operación de asignación ←

- Operación de asignación (←) permite escribir una expresión de álgebra relacional mediante la asignación de partes de esa expresión a variables temporales.
- Ejemplo: escribir r ÷ s como

$$temp1 \leftarrow \Pi_{R-S}\ (r)$$

$$temp2 \leftarrow \Pi_{R-S}\ ((temp1 \times s) - \Pi_{R-S,S}\ (r))$$

$$result = temp1 - temp2$$

- El resultado de la derecha de ← se asigna a la variable relación temporal de la izquierda que puede usarse en expresiones posteriores.

Borrado, inserción y modificación

- Borrado se expresa en el álgebra relacional como:

  r ← r − E, donde r es la relación y E una expresión

- Sólo se pueden borrar tuplas enteras. No valores de atributos

  Ejemplo: Borrar todas las cuentas de Gómez

$$impositor \leftarrow impositor - \sigma_{nombre\_cliente = \text{“Gómez”}}(impositor)$$

- Inserción:

  r ← r ∪ E

  Ejemplo: Insertar 1.200€ en la cuenta de Gómez C-973 de la sucursal de Navacerrada:

$$cuenta \leftarrow cuenta \cup \{(\text{C-973, “Navacerrada”, 1200})\}$$
$$impositor \leftarrow impositor \cup \{(\text{“Gómez”, C-973})\}$$

Borrado, inserción y modificación

- Inserción:
  - Ejemplo: Se desea ofrecer una nueva cuenta de ahorro con 200€ como regalo a todos los clientes con préstamos concedidos en la sucursal de Navacerrada:

$$r_1 \leftarrow (\sigma_{nombre\_sucursal = \text{“Navacerrada”}} (prestatario \bowtie préstamo))$$
$$r_2 \leftarrow \Pi_{número\_préstamo, nombre\_sucursal} (r_1)$$
$$cuenta \leftarrow cuenta \cup (r_2 \times \{(200)\})$$
$$impositor \leftarrow impositor \cup \Pi_{nombre\_cliente, número\_préstamo} (r_1)$$

- Modificación:

$$r \leftarrow \Pi_{F_1, F_2, \ldots, F_n} (r)$$

donde F_i son expresiones que involucran constantes y atributos de r.

Para varias tuplas de r: $r \leftarrow \Pi_{F_1, F_2, \ldots, F_n} (\sigma_P (r)) \cup (r - \sigma_P (r))$

Ejemplo: Las cuentas con saldos superiores a 10.000€ reciben un interés del 6% y el resto del 5%

$$cuenta \leftarrow \Pi_{número\_cuenta, nombre\_sucursal,\ saldo * 1.06} (\sigma_{saldo > 10000} (cuenta))$$
$$\cup\ \Pi_{número\_cuenta, nombre\_sucursal,\ saldo * 1.05} (\sigma_{saldo \leq 10000} (cuenta))$$

Introducción SQL

- Basado en algebra relacional + cálculo relacional
- Versión original ⇒ IBM (Sequel dentro de System R)
- Evolucionó ⇒ Structured Query Language
- Estandarización ⇒ ANSI e ISO
  - Normas 1986: SQL-86
  - Normas: SQL 89, SQL 92 y SQL 99
- Componentes:
  - DDL – Lenguaje de Definición de Datos
  - DML – Lenguaje de Manipulación de Datos
  - Vistas
  - Transacciones
  - SQL incorporado y dinámico
  - Integridad
  - Autorización.

Lenguaje de definición de datos

- El LDD permite especificar:
  - Esquema de cada relación
  - El dominio de valores asociado a cada atributo
  - Restricciones de integridad
  - Índices asociados a cada relación
  - Información de seguridad y autorización
  - Estructura de almacenamiento físico en disco.
- Tipos de dominio base
  - Char(n), varchar(n),text
  - int, integer, smallint, float, real, double precission,numeric(p,d)
  - Date, time, timestamp, función extract(campo from d), interval
  - Valor null , pertenece a todos los dominios base
  - Se puede especificar que un dominio es not null

número-cuenta **char**(10) **not null**

Lenguaje de definición de datos
Definición de esquemas

- create table

create table r $(A_1 D_1, A_2 D_2, \ldots A_n D_n,$
$\langle$restricción-integridad$_1\rangle$.
...
$\langle$restricción-integridad$_k\rangle$)

  - Donde A_i es el nombre atributo y D_i el dominio
- Restricciones de integridad:
  - Primary key (A_1,A_2,…,A_n), valores no nulos y únicos (Opcional)
  - Check (P) , predicado P que debe de satisfacer el atributo de la tupla

```
create table cuenta
    (número-cuenta    char (10),
     nombre-sucursal  char (15),
     saldo            integer,
     primary key (número-cuenta),
     check (saldo >= 0))
```

  - Unique (A_1,A_2,…,A_n), especificar clave candidata. Permite nulos.

Lenguaje de definición de datos

- Borrar esquema de tabla ⇒ drop table
  - Borra las tuplas y la relación

drop table *r*

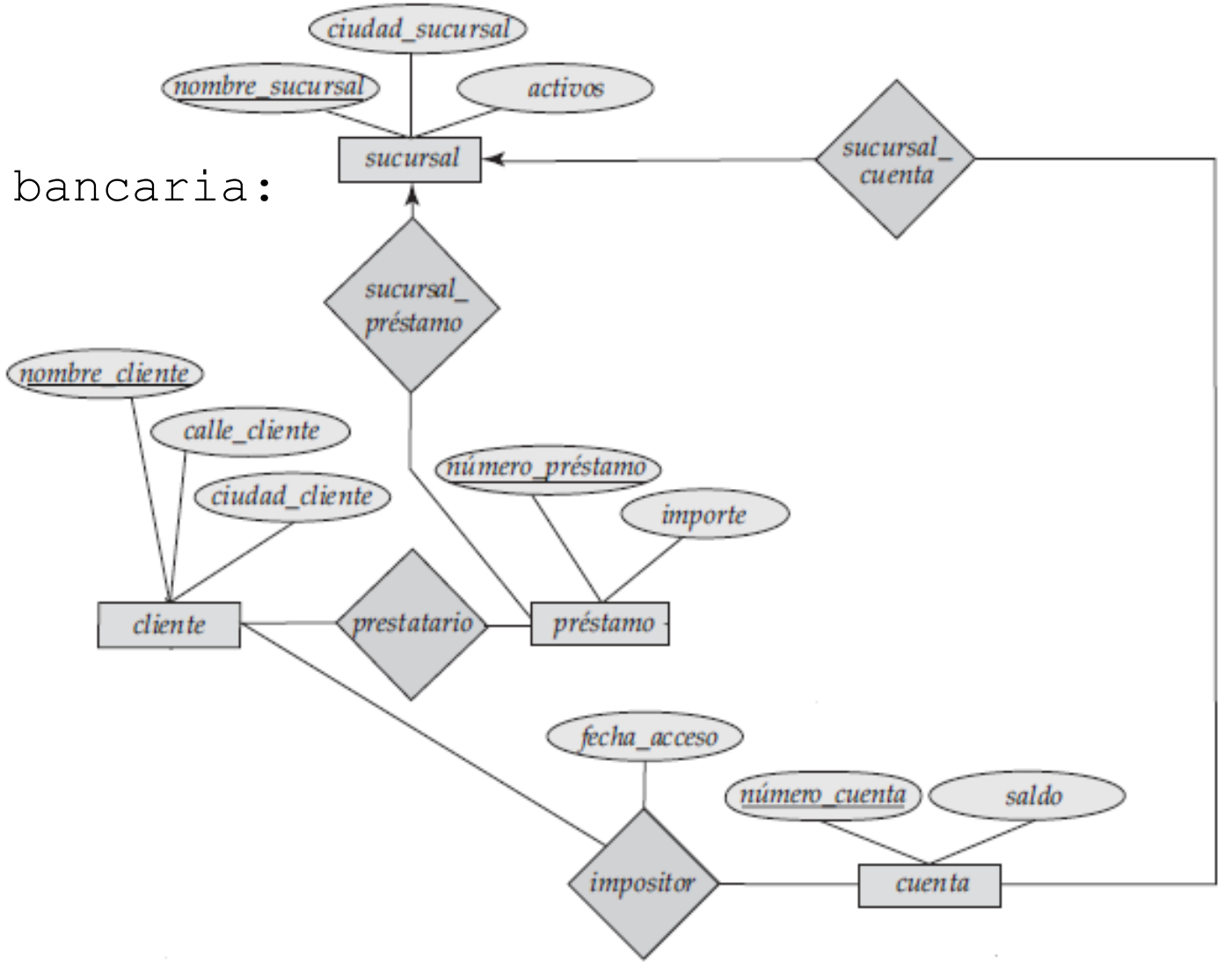
- SQL-92 ⇒ alter table. Permite modificar un esquema de tabla
  - Añadir atributo: **alter table** *r* **add** *A D*
  - Eliminar atributo: **alter table** *r* **drop** *A*

SQL DML
Caso de estudio

- Ejemplos basados en empresa bancaria:

sucursal(*nombre_sucursal*, *ciudad_sucursal*, *activos*)
cliente (*nombre_cliente*, *calle_cliente*, *ciudad_cliente*)
préstamo (*número_préstamo*, *nombre_sucursal*, *importe*)
prestatario (*nombre_cliente*, *número_préstamo*)
cuenta (*número_cuenta*, *nombre_sucursal*, *saldo*)
impositor (*nombre_cliente*, *número_cuenta*)

Figura 3.1 Esquema de la entidad bancaria.



Estructura básica consultas SQL
Select

- Operación selección. En álgebra relacional se usa la letra griega sigma minúscula para denotarla

select … **from** …

  - Select ⇒ hacer una consulta de A1…An atributos
  - From ⇒ tablas en la que se solicita información r1, …, rn
  - Where ⇒ selección sobre lo descrito en *from a través de la condición P*

select $A_1, A_2, \ldots, A_n$

from $r_1, r_2, \ldots, r_m$

where P

- Si se omite **where**, P es cierto y se devuelve toda la información.

Estructura básica consultas SQL
Select

- Siempre el resultado es una nueva tabla

```
select nombre-sucursal
from préstamo
```

Mostrará el valor de *nombre-sucursal* una vez por cada tupla de la relación *préstamo* en la que aparezca → Navacerrada 2 veces

- Permite duplicados en las relaciones y en el resultado de las expresiones SQL por defecto, ya que su eliminación consume tiempo
- Usar **distinct** para evitarlos:

```
select distinct nombre-sucursal
from préstamo
```

número_préstamo	*nombre_sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

Figura 2.6 La relación *préstamo*.

- Permite usar **all** ⇒ indicar que no se eliminan duplicados

```
select all nombre-sucursal
from préstamo
```

Estructura básica consultas SQL
Select

- * ⇒ indica todos los atributos
  - Select * , indica todos los atributos de todas las relaciones
  - Select prestamo.* , indica todos los atributos de la relación prestamo
- Puede haber expresiones aritméticas +,-,*,/

```
select nombre-sucursal, número-préstamo,
       importe * 100
from préstamo
```

- Cláusula where

```
select número-préstamo
from préstamo
where nombre-sucursal = 'Navacerrada' and
      importe > 1200
```

- Operadores lógicos AND,OR, NOT $\sigma_{nombre\_sucursal = \text{“Navacerrada”} \wedge importe > 1200}(préstamo)$
- Operadores de comparación: >,<,=,>=,<=, <>

Estructura básica consultas SQL

Conectores lógicos y operadores de comparación

- En la claúsula where se usan los conectores lógicos and, or y not y los operadores de comparación <, <=, etc.. También se puede usar between,

```
select número-préstamo
from préstamo
where importe between 90000 and 100000
```

- En la cláusula from realiza el producto cartesiano de las relaciones

```
select nombre-cliente, prestatario.número-préstamo,
       importe
from prestatario, préstamo
where prestatario.número-préstamo
      = préstamo.número-préstamo
```

Estructura básica consultas SQL

Operación de renombramiento: as

- Útil cuando 2 relaciones tienen atributos con el mismo nombre (el resultado saldrá duplicado), para expresiones aritméticas (el resultado no tiene nombre) o simplemente cuando queremos renombrar uno de los atributos de la relación resultado:
- Cláusula **as** tanto en select como en from:

nombre-antiguo **as** *nombre-nuevo*

```
select nombre_cliente, prestatario.número_préstamo as id_préstamo, importe
from prestatario, préstamo
where prestatario.número_préstamo = préstamo.número_préstamo
```

- También se usa cuando se quiere comparar variables tupla de la misma relación.
  - Ejemplo: Determinar el nombre de todas las sucursales que tienen activos mayores que, al menos, una sucursal de Arganzuela

```
select distinct T.nombre-sucursal
from sucursal as T, sucursal as S
where T.activo > S.activo and S.ciudad-sucursal
      = 'Barcelona'
```

Estructura básica consultas SQL

Cadenas de caracteres

- Cadenas entre comillas simples: 'Navacerrada'
- Comparación con patrones: **like**
  - %, encaja con cualquier cadena
  - _ , encaja con cualquier carácter
  - Ejemplo: 'Nava%', '%cer%', '_ _ _', '_ _ _ %'

```
select nombre-cliente
from cliente
where calle-cliente like '%Mayor%'
```

- Carácter de escape para que las cadenas puedan incluir caracteres espaciales: escape
  - (**like** 'ab\%cd%' **escape** '\'en por ab%cd:

- Otras funciones:
  - Not like, similar to, concatenación de caracteres (||), upper, lower

Estructura básica consultas SQL
Ordenación de tuplas: order by

- order by en cláusula where:
  - Asc, indica ascendente (defecto)
  - Desc, indica descendente

```
select distinct nombre_cliente
from prestatario, préstamo
where prestatario.número-préstamo
        = préstamo.número-préstamo and
        nombre-sucursal = 'Navacerrada'
order by nombre-cliente
```

- Ordenar por varios atributos

```
select *
from préstamo
order by importe desc, número-préstamo asc
```

Operaciones sobre conjuntos
union, intersect, except

- Las relaciones deben de ser compatibles (mismo conjunto de atributos)
- Por defecto se eliminan los duplicados. Para que no lo haga, hay que incluir all.
- Operación unión (U): igual que la operación de unión de conjuntos
  - Determinar los clientes del banco que tienen una cuenta, un préstamo o ambas cosas:

```
(select nombre-cliente
 from impositor)
 union
(select nombre-cliente
 from prestatario)
```

- Se eliminan duplicados.
- **Union all** para no eliminar duplicados:

```
(select nombre-cliente
 from impositor)
union all
(select nombre-cliente
 from prestatario)
```

Operaciones sobre conjuntos
union, intersect, except

- Operación intersección (∩)
  - Clientes que tienen tanto un préstamo, como una cuenta bancaria

```
(select distinct nombre-cliente
 from impositor)
intersect
(select distinct nombre-cliente
 from prestatario)
```

- Operación diferencia
  - Clientes que tienen cuenta, pero no préstamo

```
(select distinct nombre-cliente
 from impositor)
except
(select distinct nombre-cliente
 from prestatario)
```

Reunión de Relaciones: JOIN

- Mecanismo que proporciona SQL para enlazar dos tablas relacionadas a través de campo común.
- Suele usarse en la claúsula **from**
- Hay diferentes tipos de JOIN: inner, outer, natural, using
- Tipos y condiciones de reunión

Tipos de reunión
inner join **left outer join** **right outer join** **full outer join**

Condiciones de reunión
natural **on** <predicado> **using** (A_1, A_2, …, A_n)

Reunión de Relaciones: inner join

número_préstamo	*nombre_sucursal*	*importe*
P-170	Centro	3.000
P-230	Moralzarzal	4.000
P-260	Navacerrada	1.700

préstamo

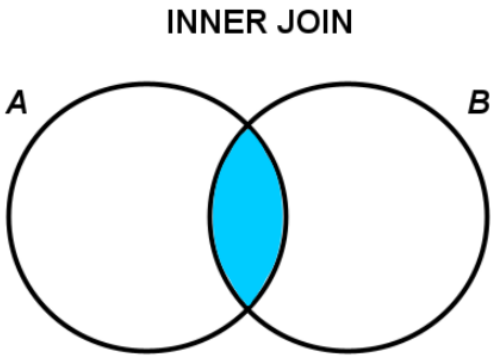
nombre_cliente	*número_préstamo*
Santos	P-170
Gómez	P-230
López	P-155

prestatario

- **INNER JOIN** o reunión interna:
  - Las tuplas resultantes son solo las comunes.
  - Resultado: atributos del lado izdo. seguidos de los del lado drcho.

préstamo **inner join** *prestatario* **on** *préstamo.número_préstamo* = *prestatario.número_préstamo*

número-préstamo	*nombre-sucursal*	*importe*	*nombre-cliente*	*número-préstamo*
P-170	Centro	3.000	Santos	P-170
P-230	Moralzarzal	4.000	Gómez	P-230



  - SQL no exige que los atributos en estos resultados sean únicos, aunque podemos renombrar para que lo sean:

préstamo **inner join** *prestatario* **on** *préstamo.número_préstamo* = *prestatario.número_préstamo*
as *pp(número_préstamo, sucursal, importe, cliente, número_préstamo_cliente)*

Reunión de Relaciones outer join

número_préstamo	*nombre_sucursal*	*importe*
P-170	Centro	3.000
P-230	Moralzarzal	4.000
P-260	Navacerrada	1.700

préstamo

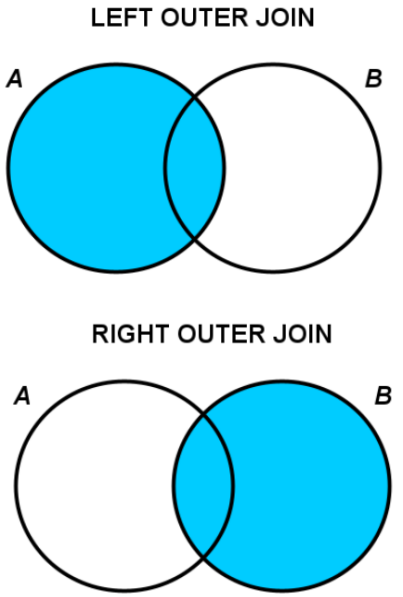
nombre_cliente	*número_préstamo*
Santos	P-170
Gómez	P-230
López	P-155

prestatario

- **OUTER JOIN** o reunión externa:
  - Se puede usar a la izda. (left) o a la derecha (right)
  - SQL primero calcula el inner join como antes y, luego, para cada tupla t de la relación del lado izdo. (préstamo) (drcho. para right) que no coincide con ninguna del lado derecho (prestatario) (izdo. para right), se añade al resultado la tupla t rellenando con null el resto de atributos.

préstamo **left outer join** *prestatario* **on**
préstamo.número-préstamo
= prestatario.número-préstamo

número-préstamo	*nombre-sucursal*	*importe*	*nombre-cliente*	*número-préstamo*
P-170	Centro	3.000	Santos	P-170
P-230	Moralzarzal	4.000	Gómez	P-230
P-260	Navacerrada	1.700	*null*	*null*



Reunión de Relaciones: natural join

número_préstamo	*nombre_sucursal*	*importe*
P-170	Centro	3.000
P-230	Moralzarzal	4.000
P-260	Navacerrada	1.700

préstamo

nombre_cliente	*número_préstamo*
Santos	P-170
Gómez	P-230
López	P-155

prestatario

- **NATURAL JOIN** o reunión natural
  - Igual que inner join, pero:
    - el atributo común solo aparece 1 vez
    - Orden atributos: primero los atributos del join + atributos de la relación izquierda + atributos de relación derecha

préstamo **natural inner join** *prestatario*

número-préstamo	*nombre-sucursal*	*importe*	*nombre-cliente*
P-170	Centro	3.000	Santos
P-230	Moralzarzal	4.000	Gómez

- A diferencia de inner join:

préstamo **inner join** *prestatario* **on** $préstamo.número\_préstamo = prestatario.número\_préstamo$

número-préstamo	*nombre-sucursal*	*importe*	*nombre-cliente*	*número-préstamo*
P-170	Centro	3.000	Santos	P-170
P-230	Moralzarzal	4.000	Gómez	P-230

Reunión de Relaciones

- SQL-92 otros 2 tipos de reunión:
  - **Cross join** (producto cartesiano de las dos tablas)
    - Devuelve todas las tuplas cruzadas, es decir, para cada tupla izda. todas las de drcha.
  - **Union join**
    - Unión de todas las columnas de ambas tablas (usando null)

Modificación de la base de datos
Borrado: delete

- Se expresa como una consulta

delete from *r*
where *P*

- Donde r es una tabla y P el predicado
- Si se omite el predicado ⇒ se borran todas las tuplas
- Ejemplos:
  - Borrar todas las tuplas de la rela

delete from *préstamo*

  - Borrar todas las tuplas de la sucursa

delete from *cuenta*
where *nombre-sucursal* = 'Navacerrada'

  - Borrar todas las tuplas de la relación cuenta de todas las sucursales de Navacerrada

delete from *cuenta*
where *nombre-sucursal* **in** (**select** *nombre-sucursal*
from *sucursal*
where *ciudad-sucursal*
= 'Navacerrada')

Modificación de la base de datos
Inserción: insert

- Se inserta o bien la tupla deseada o el resultado de una consulta:
  - Debe de respetar el dominio de los atributos
  - Y el número de atributos

insert into *cuenta*
values ('C-9732', 'Navacerrada', 1200)

- Se usa la claúsula insert

- Se puede especificar el orden de los atributos

insert into *cuenta* (*nombre-sucursal*, *número-cuenta*, *saldo*)
values ('Navacerrada', 'C-9732', 1200)

Modificación de la base de datos
Inserción: insert

- Se pueden insertar tuplas provenientes de una consulta
  - Ejemplo: A todos los clientes con préstamos en la sucursal de Navacerrada

```
insert into cuenta
    select nombre-sucursal, número-préstamo, 200
    from préstamo
    where nombre-sucursal = 'Navacerrada'
```

- Importante ⇒ finalizar la evaluación de la sentencia select antes de insertar

```
insert into cuenta
    select *
    from cuenta
```

- Se pueden insertar valores nulos: null

```
insert into cuenta
    values ('C-401', null, 1200)
```

Modificación de la base de datos
Actualización: update

- Actualizar algunos campos de las tuplas ⇒ UPDATE
  - Ejemplo: Incrementar todos los saldos un 5%:

```
update cuenta
set saldo = saldo * 1.05
```

  - Ejemplo: si solo incrementamos los que tienen saldo superior a 1000€:

```
update cuenta
set saldo = saldo * 1.05
where saldo >= 1000
```

Modificación de la base de datos
Actualización: update

- El orden de actualización es muy importante:
  - Ejemplo: Si cambiamos el orden, una cuenta con un saldo igual o poco inferior a 10.000€ recibiría un 11,3%

```
update cuenta
set saldo = saldo * 1.06
where saldo > 10000

update cuenta
set saldo = saldo * 1.05
where saldo <= 10000
```

```
update cuenta
set saldo = saldo * 1.05
where saldo >= 1000
```

```
case
    when pred1 then result1
    when pred2 then result2
    ...
    when predn then resultn
    else result0
end
```

- Constructor CASE para mantener el orden de actualización
  - Ejemplo: El ejemplo anterior con CASE

```
update cuenta
set saldo = case
            when saldo <= 10000 then saldo * 1.05
            else saldo * 1.06
        end
```