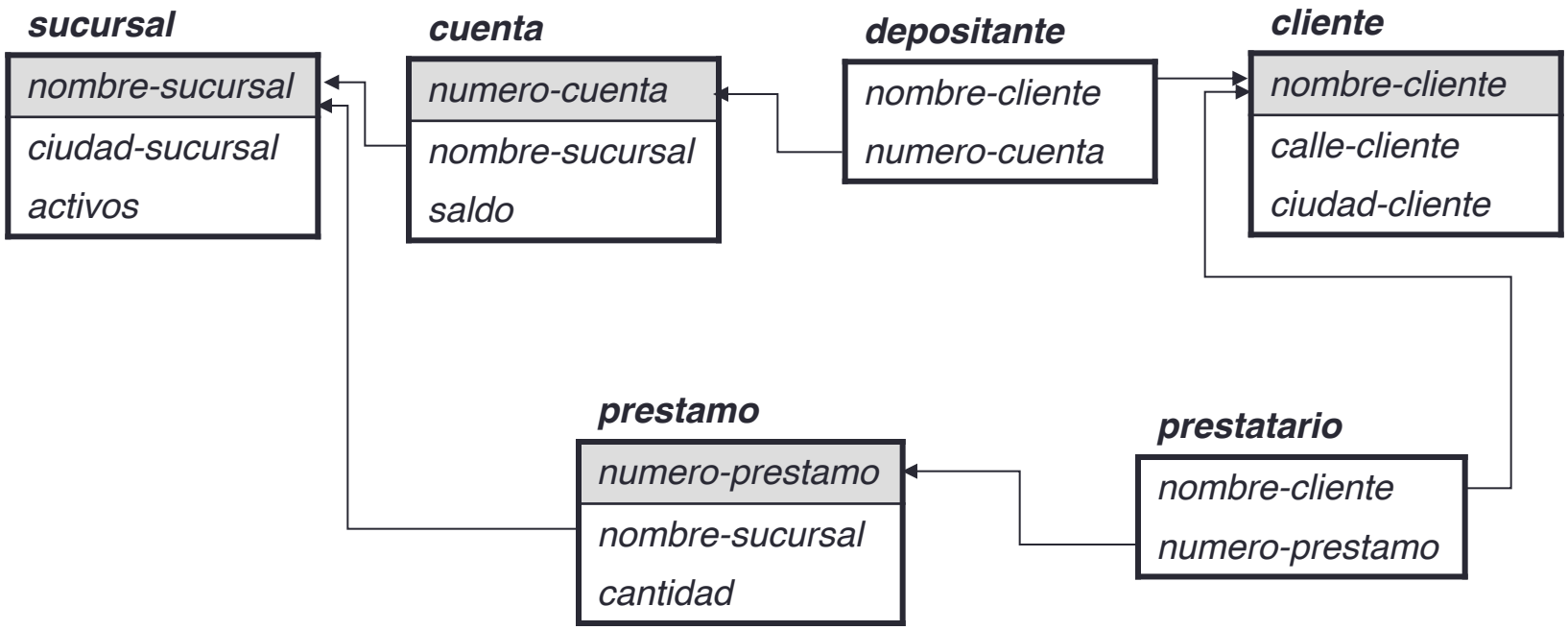
SQL COMO DML: CONSULTA DE DATOS

Prácticas

Esquema utilizado en los ejemplos



Consultas: Estructura básica

- SQL está basado en operaciones sobre relaciones y sobre conjuntos con algunas modificaciones y mejoras
- Una consulta típica en SQL tiene la siguiente forma:
-

select A_1, A_2, ..., A_n
from r_1, r_2, ..., r_m
where P

  - A_i*s* repesentan atributos
  - r_i*s* representan relaciones
  - P es un predicado.

- El resultado de una consulta SQL es siempre una relación.

La cláusula select

- La cláusula **select** lista los atributos que queremos en el resultado de la consulta

- P.e. encontrar los nombres de todas las sucursales de la relación *prestamo*

- **select** *nombre-sucursal*
  from *prestamo*

La clausula select (Cont.)

- SQL permite duplicados tanto en relaciones (tablas) como en los resultados de las consultas.
- Para forzar la eliminación de duplicados en los resultados utilizamos **distinct** después de **select.**
- Encontrar los nombres de todas las sucursales en la relación *prestamo* eliminando duplicados

  select distinct *nombre-sucursal*
  from *prestamo*

- La palabra reservada **all** indica que no se eliminen los duplicados (comportamiento por defecto).

  select all *nombre-sucursal*
  from *prestamo*

La clausula select (Cont.)

- Un asterisco en la clausula **select** indica “todos los atributos”

  select *
  from *prestamo*

- La clausula **select** puede contener expresiones aritméticas con las operaciones +, –, ∗, y /, y operar sobre constantes o atributos de tuplas.
- La consulta:

  select *numero-prestamo, nombre-sucursal, cantidad* ∗ 100
  from *prestamo*

  devolverá una relación igual a la relación *prestamo*, excepto que el atributo *cantidad* estará multiplicado por 100.

La clausula where

- La clausula **where** especifica condiciones que debe satisfacer el resultado.

- Encontrar todos los números de préstamo hechos en la sucursal de Vigo con cantidades prestadas mayores de 1200 euros.

  select *numero-prestamo*
  from *prestamo*
  where *nombre-sucursal* = *'Vigo'* **and** *cantidad* > 1200

- Los resultados lógicos se pueden combinar con las conectivas lógicas **and, or y not.**
- Las comparaciones se pueden aplicar al resultado de expresiones aritméticas.

La clausula where (Cont.)

- SQL incluye un operador de comparación **between**
- P.e. Encontrar los números de prétamo de aquellos préstamos cuya cantidad esté entre 90,000 y 100,000 euros (es decir, ≥ 90,000 y ≤100,000)

select *numero-prestamo*
from *prestamo*
where *cantidad* **between** 90000 **and** 100000

La clausula from

- La clausula **from** lista las relaciones involucradas en la consulta
- Encontrar el producto cartesiano *prestatario x prestamo*

select ∗
from *prestatario, prestamo*

Encontrar el nombre, número de préstamo y cantidad prestada de todos los clientes con un préstamo en la sucursal de Vigo.

select *nombre-cliente, prestatario.numero-prestamo, cantidad*
from *prestatario, prestamo*
where *prestatario.numero-prestamo = prestamo.numero-prestamo*
and *nombre-sucursal =* ‘Vigo’

La operación de renombrado

- SQL permite renombrar relaciones y atributos mediante la cláusula **as** :

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

- Encontrar el nombre, número de préstamo y cantidad prestada de todos los clientes; renombrar la columna *numero-prestamo* como *id-prestamo.*

  select *nombre-cliente, prestatario.numero-prestamo* **as** *id-prestamo, cantidad*
  from *prestatario, prestamo*
  where *prestatario.numero-prestamo = prestamo.numero-prestamo*

Variables de tupla

- Las variables de tupla se definen en la cláusula **from** mediante el uso de la cláusula **as**.
- Encontrar los nombres de cliente y sus números de préstamo para todos los clientes que tengan un préstamo en alguna sucursal.

select *nombre-cliente, T.numero-prestamo, S.cantidad*
from *prestatario* **as** *T, prestamo* **as** *S*
where *T.numero-prestamo = S.numero-prestamo*

- Encontrar los nombres de todas las oficinas que tienen unos activos mayores que alguna sucursal de Barcelona.

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

Operaciones sobre cadenas de caracteres

- SQL incluye un operador de coincidencia para comparar cadenas de caracteres. Los patrones se describen usando dos caracteres especiales:
  - porcentaje (%). El % representa cualquier subcadena de caracteres.
  - subrayado (_). El _ representa cualquier caractér.
- Encontrar los nombres de los clientes cuya calle incluya la subcadena "Mayor".

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

- Para encontrar "Mayor%"

```
like 'Mayor\%' escape '\'
```

- SQL soporta diversas operaciones sobre cadenas de caracteres, como
  - concatenación (utilizando "||")
  - convertir de mayúsculas a minúsculas (y viceversa)
  - calcular la longitud, extraer subcadenas, etc.

Ordenar las tuplas obtenidas

- Listar en orden alfabético los nombres de todos los clientes que tenganun préstamo en la sucursal de Vigo

```
select distinct nombre-cliente
from    prestatario, prestamo
where prestatario.numero-prestamo =
            prestamo.numero-prestamo and
          nombre-sucursal = 'Vigo'
order by nombre-cliente
```

- Podemos especificar **desc** para orden descendente o **asc** para orden ascendente para cada atributo; el orden por defecto es el ascendente.
  - P.e. **order by** *nombre-cliente* **desc**

Operaciones de conjuntos

- Las operaciones de conjuntos unión (**union**), intersección (**intersect**), y diferencia (**except**) se pueden aplicar sobre relaciones.
- Cada una de las operaciones anteriores elimina dupicados automáticamente; para conservar los duplicados se debe utilizar **union all, intersect all** and **except all.**

  Supongamos que una tupla aparece *m* veces en *r* y *n* veces en *s*, entonces aparece:
  - *m* + *n* veces en *r* **union all** *s*
  - min(*m,n)* veces en *r* **intersect all** *s*
  - max(0, *m* – *n)* veces en *r* **except all** *s*

Operaciones de conjuntos

Encontrar todos los clientes que tengan un préstamo, una cuenta o ambas cosas:

(select *nombre-cliente* **from** *depositante*)
union
(select *nombre-cliente* **from** *prestatario)*

Encontrar todos los clientes que tienen tanto una cuenta como un préstamo

(**select** *nombre-cliente* **from** *depositante*)
intersect
(select *nombre-cliente* **from** *borrower)*

Encontrar todos los clientes que tengan una cuenta pero no un préstamo.

(**select** *nombre-cliente* **from** *depositante*)
except
(select *nombre-cliente* **from** *prestatario)*

Funciones agregadas

- Estas funciones operan sobre un conjunto de valores de una columna de una relación y devuelven un valor

avg: valor medio
min: valor mínimo
max: valor máximo
sum: suma de valores
count: número de valores

Funciones agregadas (Cont.)

- Encontrar el saldo medio de las cuentas de la sucursal de Vigo.

```
select avg (saldo)
        from cuenta
        where nombre-sucursal = 'Vigo'
```

Encontrar el número de tuplas de la relación *cliente*.

```
select count (*)
        from cliente
```

Encontrar el número de depositantes del banco.

```
select count (distinct nombre-cliente)
        from depositante
```

Funciones agregadas – Group By

- Encontrar el número de depositantes de cada sucursal.

select *nombre-sucursal,* **count (distinct** *nombre-cliente)*
from *depositante, cuenta*
where *depositante.numero-cuenta = cuenta.numero-cuenta*
group by *nombre-sucursal*

Nota: Los atributos en la clausula **select** fuera de las funciones agregadas deben aparecer en la lista **group by**

Funciones agregadas – cláusula Having

- Encontrar los nombres de todas las sucursales donde el saldo medio de las cuentas sea de más de 1.200 euros.

```
select nombre-sucursal, avg (saldo)
        from cuenta
        group by nombre-sucursal
        having avg (saldo) > 1200
```

Valores nulos (*null)*

- Las tuplas pueden tener valores nulos, indicado por *null*, para algunos de sus atributos.
- *null* significa "valor desconocido" o que ese valor no existe.
- El predicado **is null** se utiliza para comprobar valores nulos.
  - P.e. Encontrar todos los números de préstamo que aparecen en la relación *prestamo* con un valor nulo en *cantidad.*

    select *numero-prestamo*
    from *prestamo*
    where *cantidad* **is null**
- El resultado de cualquier expresión aritmética en la que participa *null* es *null*
  - P.e. 5 + null devuelve null
- Sin embargo, las funciones agregadas simplemente ignoran los nulos

Valores nulos y lógica tri-valorada

- Cualquier comparación con *null* devuelve *desconocido*
  - *P.e. 5 < null o null <> null o null = null*
- Lógica tri-valorada utilizando el valor de verdad *desconocido*:
  - OR: (*desconocido* **or** *true*) = *true*, (*desconocido* **or** *false*) = *desconocido*
    (*desconocido* **or** *desconocido*) = *desconocido*
  - AND: *(desconocido* **and** *true) = desconocido, (desconocido* **and** *false) = false,*
    (desconocido **and** *desconocido) = desconocido*
  - NOT*: (***not** *desconocido) = desconocido*
- Los resultados de los predicados de la cláusula **where** se tratan como *false* si toman el valor *desconocido*

Valores nulos y agregados

- Total de cantidades de todos los préstamos

  select sum (*cantidad)*
  from *prestamo*

  - Esta sentencia ignora las cantidades nulas
  - El resultado es nulo si no hay cantidades no nulas
- Todas las funciones agregadas excepto **count(*)** ignoran las tuplas con valores nulos en los atributos agregados.

Subconsultas anidadas

- SQL proporciona un mecanismo para anidar subconsultas.
- Una subconsulta es una expresión **select-from-where** que está anidada en otra consulta.
- Un uso habitual de las subconsultas es realizar comprobaciones de pertenencia a un conjunto, comparaciones de conjuntos y de cardinalidades de conjuntos.

Ejemplo

- Encontrar todos los clientes que tengan tanto una cuenta como un préstamo en el banco.

```
select distinct nombre-cliente
        from prestatario
        where nombre-cliente in (select nombre-cliente
                                 from depositante)
```

Encontrar todos los clientes que tienen un préstamo pero no una cuenta en el banco

```
select distinct nombre-cliente
        from prestatario
        where nombre-cliente not in (select nombre-cliente
                                     from depositante)
```

Ejemplo

- Encontrar todos los clientes que tiene tanto una cuenta como un préstamo en la sucursal de Vigo

```
select distinct nombre-cliente
        from prestatario, prestamo
        where prestatario.numero-prestamo = prestamo.numero-prestamo
          and nombre-sucursal = "Vigo"
          and (nombre-sucursal, nombre-cliente) in
                        (select nombre-sucursal, nombre-cliente
                        from depositante, cuenta
                        where depositante.numero-cuenta =
                                cuenta.numero-cuenta)
```

Comparación de conjuntos

- Encontrar todas las sucursales que tienen unos activos mayores que alguna sucursal de Madrid.

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

La misma consulta utilizando la cláusula **> some**

```
select nombre-sucursal
        from sucursal
        where activos > some
                (select activos
                 from sucursal
                        where ciudad-sucursal = 'Madrid' )
```

Definición de la cláusula Some

- F <comp> **some** *r* ⇔∃ *t* ∈ *r* que cumple *(*F <comp> *t*)
  Donde <comp> puede ser: <, ≤, >, ≥. =, ≠

(5< **some** | 0 / 5 / 6 |) = true
(leer: 5 < *some* tupla de la relación)

(5< **some** | 0 / 5 |) = false

(5 = **some** | 0 / 5 |) = true

(5 ≠ **some** | 0 / 5 |) = true (dado que 0 ≠ 5)

(= some) ≡ in

Sin embargo, **(≠ some) ≢ not in**

Definición de la cláusula All

- F <comp> **all** *r* ⇔∀*t* ∈ *r* que cumple (F <comp> *t)*

(5< **all** | 0 / 5 / 6 |) = false

(5< **all** | 6 / 10 |) = true

(5 = **all** | 4 / 5 |) = false

(5 ≠ **all** | 4 / 6 |) = true (dado que 5 ≠ 4 y 5 ≠ 6)

(≠ **all**) ≡ **not in**

Sin embargo, (= **all**) ≢ **in**

Consulta de ejemplo

- Encontrar los nombres de todas las sucursales que tengan unos activos mayores que todas las sucursales de Madrid.

```
select nombre-sucursal
        from sucursales
        where activos > all
                (select activos
                from sucursal
                where ciudad-sucursal = 'Madrid' )
```

Comprobación de relaciones vacías

- La construcción **exists** devuelve el valor **true** si la subconsulta argumento no está vacía.
- **exists** $r \Leftrightarrow r \neq \varnothing$
- **not exists** $r \Leftrightarrow r = \varnothing$

Consulta de ejemplo

- Encontrar todos los clientes que tengan una cuenta en todas las sucursales de Madrid.

```
select distinct S.nombre-cliente
        from depositante as S
        where not exists (
                (select nombre-sucursal
                from sucursal
                where ciudad-sucursal = 'Madrid' )
        except
                (select R.nombre-sucursal
                from depositante as T, cuenta as R
                where T.numero-cuenta = R.numero-cuenta and
                        S.nombre-cliente = T.nombre-cliente))
```

Notar que $X - Y = \emptyset \Leftrightarrow X \subseteq Y$

Nota: Esta consulta no se puede escribir con = **all** y sus variantes

Comprobación de ausencia de tuplas duplicadas

- La construcción **unique** comprueba si el resultado de una subconsulta tiene tuplas duplicadas.
- Encontrar todos los clientes que tienen como mucho una cuenta en la sucursal de Vigo.

```
select T.nombre-cliente
from depositante as T
where unique (
    select R.nombre-cliente
    from cuenta, depositante as R
    where T.nombre-cliente = R.nombre-cliente and
          R.numero-cuenta = cuenta.numero-cuenta
and
          cuenta.nombre-sucursal = 'Vigo'
```

Consulta de ejemplo

- Encontrar todos los clientes que tengan al menos dos cuentas en la sucursal de Vigo.

```
select distinct T.nombre-cliente
from depositante T
where not unique (
        select R.nombre-cliente
        from cuenta, depositante as R
        where T.nombre-cliente = R.nombre-cliente
and
        R.numero-cuenta = cuenta.numero-cuenta
and
        cuenta.nombre-sucursal = 'Vigo' )
```