UNIVERSIDAD COOPERATIVA DE COLOMBIA

Gestión de la Información

Gustavo Adolfo Gómez Gómez
MSc. Gestión, aplicación y desarrollo de software
2024

www.ucc.edu.co

Gestión de la Información

Unidad 1: Asociar diseños, plataformas y soportes informáticos a los modelos de gestión de información de la organización.

Tema: Lenguaje Estructurado de Consultas: SQL

UNIVERSIDAD COOPERATIVA DE COLOMBIA

Structured Query Language - SQL

Avanzado

Bases de datos relacionales

Datos Relación Sucursal - Cuenta

Sucursal		
nombre_sucursal	ciudad_sucursal	activos
SedeA	Bucaramanga	4000000
SedeB	Floridablanca	520000
SedeC	Lebrija	45000000
SedeD	Girón	95020000

Cuenta			
numero_cuenta	id_cliente	nombre_sucursal	saldo
c101	100026	SedeB	1560000
c102	100028	SedeA	10000
c104	100030	SedeA	35000
c105	100025	SedeC	300000
c106	100028	SedeD	200000

Bases de datos relacionales

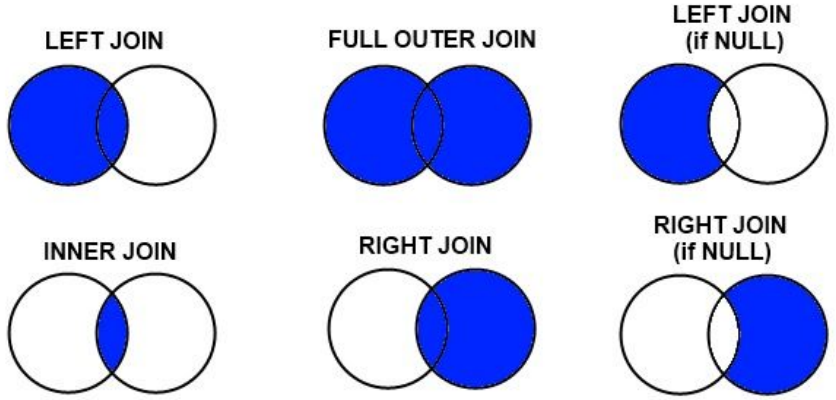
Datos Relación Prestamo - Cliente

Prestamo			
numero_prestamo	id_cliente	nombre_sucursal	valor_prestado
p10006	100026	SedeB	10000
p10007	100028	SedeA	2510000
p10008	100027	SedeA	2611000
p10009	100029	SedeC	21540000
p10010	100028	SedeD	2541000
p10011	100026	SedeA	620000

Cliente			
id_cliente	nombre_cliente	calle_cliente	ciudad_cliente
100025	Pedro Perez	Americas	Bucaramanga
100026	Maria Suarez	Bulevard	Piedecuesta
100027	Antonio Alvarez	Bolivar	Floridablanca
100028	Marcela Gutierrez	Los angeles	Lebrija
100029	Adriana Martinez	Curazao	Girón
100030	Jose Perez	Americas	Bucaramanga

Uniones entre relaciones (avanzado)

- Inner Join
- Left Outer Join
- Left Join
- Right Join

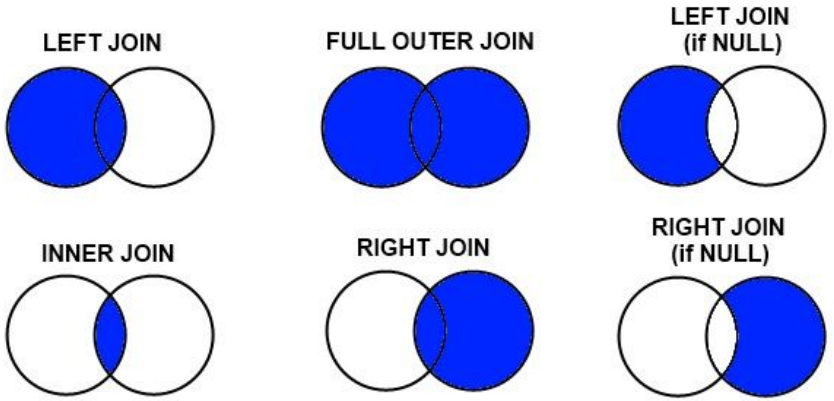


Consultas SQL

Uniones (avanzado)

- Inner Join
- Left Outer Join
- Left Join
- Right Join

Unión de conjuntos

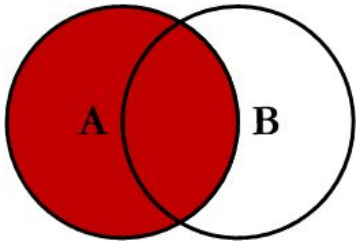


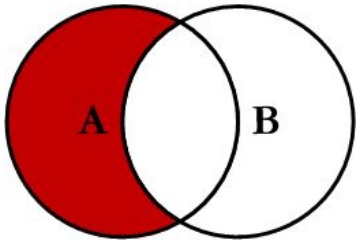
Select *
From relacionA
____ **Join** relacionB **on** atrA=atrB

Consultas SQL

Uniones (avanzado)

Ejemplo Left Join: Consulta de los clientes con o sin cuentas



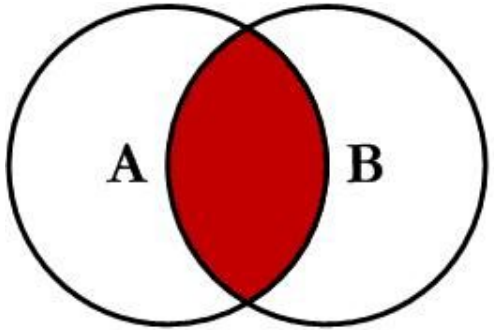



```
Select c.*
From clientes c
Left Join cuentas cu on c.cliente_id=cu.cliente_id
```

Modificación de base de datos

Uniones (avanzado)

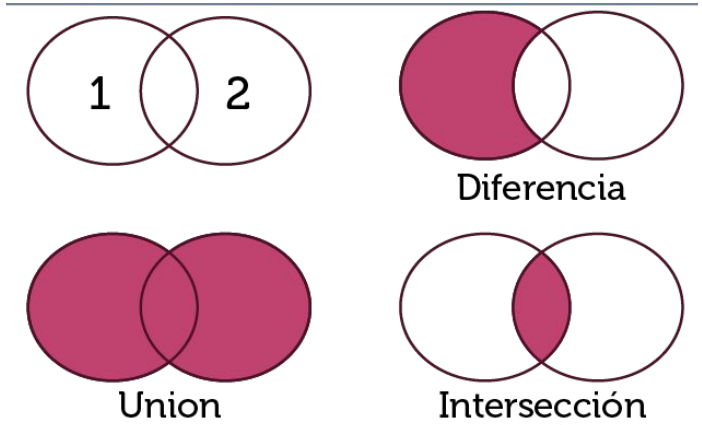
Ejemplo inner Join: seleccionar solo los clientes con préstamo




```
Select c.*
From clientes c
Inner Join prestamos p on c c.cliente_id=p.cliente_id
```

SQL

Operaciones sobre conjuntos



Ejercicio:

Teniendo en cuenta las siguientes relaciones:

Cliente(id,nombre,direccion,telefono)
compraCarro(id_carro, id_cliente, id_cliente_v, modelo,marca,valor, fecha)
rentaCarro(id_carro,id_cliente,dias,fecha)

1. Consulta los clientes (con duplicados) que han comprado carros marca “Mazda”y han rentado carros por más de 1 dia
2. Consulta los clientes que han comprado carros de modelos mayor al 2000 pero nunca han rentado

Ejercicio

Ejercicio Clase parte 3:

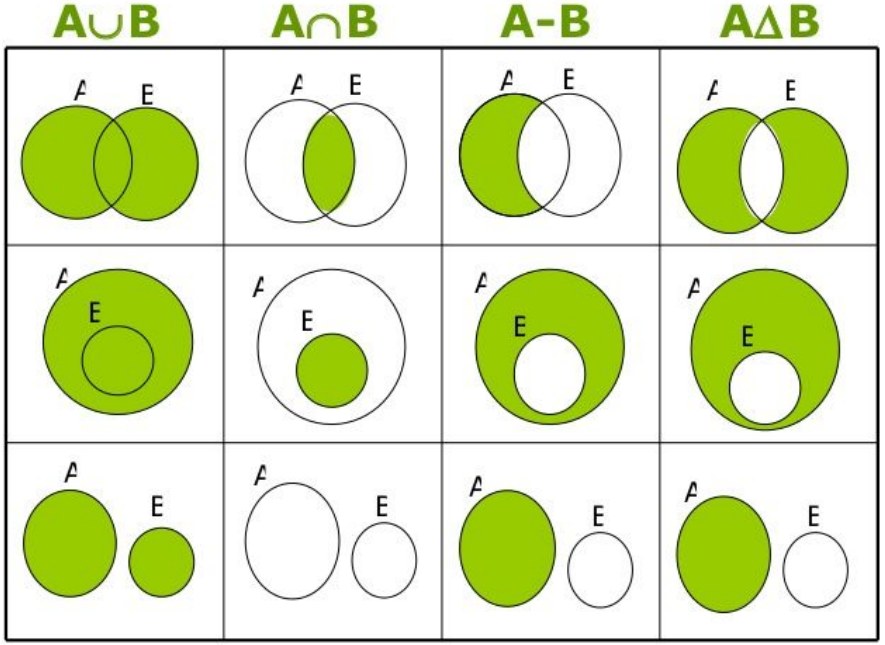
- Recuperar el nombre y la ciudad de los clientes que tienen préstamos con el banco agregando además el valor del préstamo
- Recuperar el nombre, ciudad y activos de las sucursales que han abierto cuentas con saldos mayores a 35000
- Consultar el nombre de los clientes que han abierto cuenta de una sucursal ubicada en la misma ciudad donde residen.
- Recuperar el número de préstamo y el nombre del cliente a quien se le aprobó préstamos mayores a 450000

Consultas SQL

Operaciones sobre consultas

También denominadas:

operaciones de conjuntos a nivel de resultados

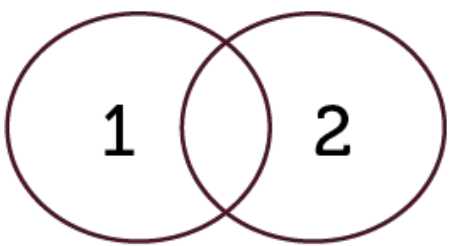


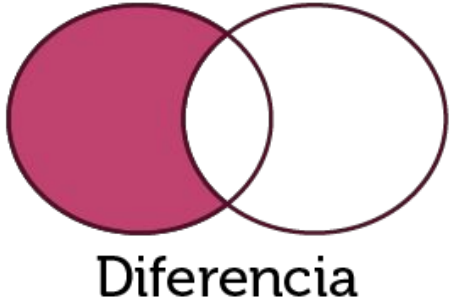
SQL

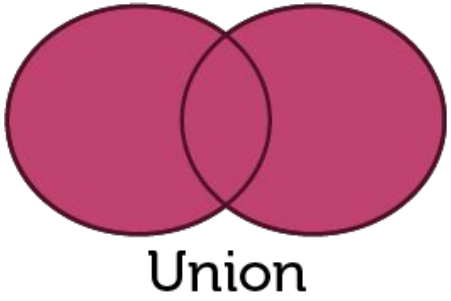
Operaciones sobre conjuntos

Cláusulas:

- **Union (U):** Unión
- **Intersect (∩):** Intersección
- **Except (-):** Diferencia







SQL

Consideremos

1. Los clientes con cuentas = CA
2. Los clientes con préstamos = CB

CA:

```
Select *
From clientes cc, cuentas cu
Where cc.cliente_id=cu.cliente_id
```

CB:

```
Select *
From clientes cc, prestamo pr
Where cc.cliente_id=pr.cliente_id
```

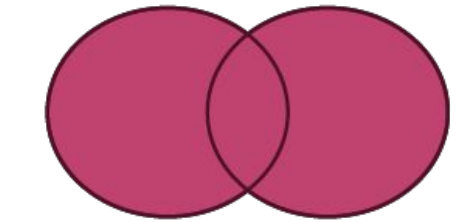
SQL

Operaciones sobre conjuntos

Operación **UNION** (Sin duplicados)

- Consultar los clientes que tienen cuenta **Ó** préstamos

```
(
        Select cc.nombre, cc.telefono
        From clientes cc, cuentas cu
        Where cc.cliente_id=cu.cliente_id
) union (
        Select  cc.nombre, cc.telefono
        From clientes cc, prestamo pr
        Where cc.cliente_id=pr.cliente_id
)
```



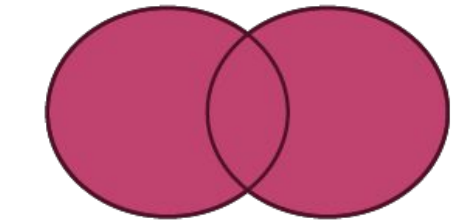
SQL

Operaciones sobre conjuntos

Operación **UNION ALL** (incluye duplicados)

- Consultar los clientes que tienen cuenta **Ó** préstamos

```
(
        Select cc.nombre, cc.telefono
        From clientes cc, cuentas cu
        Where cc.cliente_id=cu.cliente_id
) union all (
        Select cc.nombre, cc.telefono
        From clientes cc, prestamo pr
        Where cc.cliente_id=pr.cliente_id
)
```



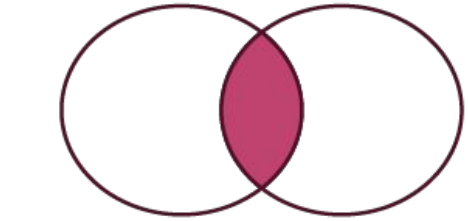
SQL

Operaciones sobre conjuntos

Operación **INTERSECCIÓN** (sin duplicados)

- Consultar los clientes que tienen cuenta **Y** préstamos

```
(
        Select cc.nombre, cc.telefono
        From clientes cc, cuentas cu
        Where cc.cliente_id=cu.cliente_id
) intersect (
        Select cc.nombre, cc.telefono
        From clientes cc, prestamo pr
        Where cc.cliente_id=pr.cliente_id
)
```



SQL

Operaciones sobre conjuntos

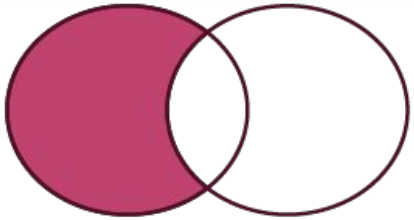
Operación **INTERSECCIÓN ALL**(incluye duplicados)

- Consultar los clientes que tienen cuenta **Y** préstamos

```
(
        Select cc.nombre, cc.telefono
        From clientes cc, cuentas cu
        Where cc.cliente_id=cu.cliente_id
) intersect all (
        Select cc.nombre, cc.telefono
        From clientes cc, prestamo pr
        Where cc.cliente_id=pr.cliente_id
)
```

SQL

Operaciones sobre conjuntos



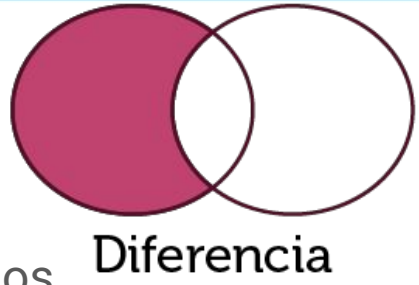
Operación **EXCEPTO (sin duplicados)**

- Consultar los clientes que tienen cuenta **PERO NO** préstamos

```
(
        Select cc.nombre, cc.telefono
        From clientes cc, cuentas cu
        Where cc.cliente_id=cu.cliente_id
) except (
        Select cc.nombre, cc.telefono
        From clientes cc, prestamo pr
        Where cc.cliente_id=pr.cliente_id
)
```

SQL

Operaciones sobre conjuntos



Operación **EXCEPTO (incluye duplicados)**

- Consultar los clientes que tienen cuenta **PERO NO** préstamos

```
(
        Select cc.nombre, cc.telefono
        From clientes cc, cuentas cu
        Where cc.cliente_id=cu.cliente_id
) except all (
        Select cc.nombre, cc.telefono
        From clientes cc, prestamo pr
        Where cc.cliente_id=pr.cliente_id
)
```

SQL

Valores nulos

SQL permite el uso de valores nulos para indicar la ausencia de información sobre el valor de un atributo.

Palabra reservada: **NULL** o **NOT NULL**

```
select numero_prestamo
from prestamo
where valor_prestado IS NULL
```

Nota: Minúscula o mayúscula es igual

SQL

Valores nulos

SQL permite el uso de valores nulos para indicar la ausencia de información sobre el valor de un atributo.

Palabra reservada: **NULL** o **NOT NULL**

```
select numero_prestamo
from prestamo
where valor_prestado IS NOT NULL
```

Nota: Minúscula o mayúscula es igual

UNIVERSIDAD COOPERATIVA DE COLOMBIA

Valores nulos

La existencia de valores nulos también complica el procesamiento de los operadores de agregación

Qué pasa si se hace un promedio con valores nulos?

valor_prestado
5000
4000
(null)
5000
4000
avg=**3600** ó **4500 ???**

```
select avg(valor_prestado)
from prestamo
```

Los valores nulos se ignoran

UNIVERSIDAD COOPERATIVA DE COLOMBIA

Subconsultas anidadas

SQL proporciona un mecanismo para anidar subconsultas. Las subconsultas son expresiones *select-from-where* que están anidadas dentro de otra consulta.

Se utilizan para determinar **pertenencia** del resultado de una consulta en otra.

Comando: **IN** / **NOT IN**

Ejp: se requiere saber los clientes con préstamos que no tienen cuenta

```
select c.nombre_cliente
from cliente c
Join prestamo p ON c.id_cliente=p.id_cliente
```

```
select c.nombre_cliente
from cliente c
Join cuenta cu ON c.id_cliente=cu.id_cliente
```

SQL

Subconsultas anidadas

SQL proporciona un mecanismo para anidar subconsultas. Las subconsultas son expresiones *select-from-where* que están anidadas dentro de otra consulta.

- Escalar o simple
- Pertenencia o de fila: Se compara fila a fila
- ALL: Verdadero sólo si **todos** cumplen la condición.
- ANY: Verdadero sólo si **algunos** cumplen la condición. Al menos uno.
- EXISTS: Devuelve resultado si existe al menos un dato en la subconsulta
- FROM: Subconsulta en forma de Tabla (relación)

SQL

Subconsultas anidadas

Escalar ó simples: Comando: **=**

Una subconsulta escalar devuelve un valor único.

Ejp: consultar préstamo que coincida con el valor más alto del saldo de las cuentas

```
select *
from prestamo p
where p.valor_prestado = (

        select max(c.saldo)
        from cuenta c

)
```

SQL

Subconsultas anidadas

PERTENENCIA ó FILA: Comando: **IN / NOT IN**

Ejp: se requiere saber los clientes con préstamos que no tienen cuenta

```
select c.nombre_cliente
from cliente c
Join prestamo p ON c.id_cliente=p.id_cliente
where c.nombre_cliente NOT IN (

        select c.nombre_cliente
        from cliente c
        Join cuenta cu ON c.id_cliente=cu.id_cliente

)
```

SQL

Subconsultas anidadas

ALL: Comando: **ALL** y operadores de comparación**: =, >, <, >=, <=, <> or !=**
devolverá verdadero si la comparación devuelve verdadero para cada fila devuelta por la subconsulta, o si la subconsulta no devuelve ninguna fila.

Ejp: seleccionar sucursales sólo si los activos son mayores a todos los saldos de cuentas

```
select *
from sucursal s
where s.activos > ALL (
        select cu.saldo
        from cuenta cu
)
```

Subconsultas anidadas

ANY: Comando: **ANY** y operadores de comparación**: =, >, <, >=, <=, <> or !=**
devolverá verdadero si la comparación devuelve verdadero para al menos una fila devuelta por la subconsulta.

Ejp: seleccionar cuentas si al menos una de las cuentas tiene saldos mayores a los activos de cualquier sucursal

```
select *
from cuenta cu
where cu.saldo > ANY (
        select s.activos
        from sucursal s
)
```

Subconsultas anidadas

EXITS: Comando: **EXISTS / NOT EXISTS**
devolverá verdadero si la subconsulta devuelve alguna fila. Por el contrario, las subconsultas que utilizan NOT EXISTS devolverán verdadero solo si la subconsulta no devuelve filas de la tabla.

Ejp: devolver las cuentas solo si existen préstamos mayores a 100

```
select *
from cuenta cu
where  EXISTS (
        select p.*
        from prestamo p
        where p.valor_prestado > 100
)
```

Subconsultas anidadas

FROM: Comando: **FROM ()**
Aunque las subconsultas se usan con cláusula WHERE, también se puede desde la cláusula **FROM**. Estas subconsultas se denominan comúnmente **tablas derivadas**.

Nota: Se debe usar una cláusula AS de renombramiento

Ejp: Calcular el promedio de la sumatoria de los saldos (cuentas) que tiene cada cliente. (agrupar por cada cliente)

Id_cuenta / Id_cliente / Saldo

1	/ 101	/ 100
2	/ 101	/ 200
3	/ 102	/ 100
4	/ 102	/ 150

```
select AVG(sum_cliente)
FROM (
        select SUM(cu.saldo) AS sum_cliente
        from cuenta cu
        Group By cu.id_cliente
)
```

Vistas

Las vistas en SQL se definen mediante la instrucción *create view*. Para definir una vista hay que darle un nombre e indicar la consulta que la va a poblar (o definir). La forma de la instrucción ***create view*** es

```
create view v as <expresión de consulta>
```

Ejp: Crear vista de clientes con cuentas

```
create view clientes_cuentas as (
        Select c.nombre, cu.saldo
        from cliente c
        join cuenta cu ON c.id_cliente = cu.id_cliente
)
```

SQL

Taller

Averiguar de las vistas:

- Cómo se crean en MariaDB

Crear:

***create view** v **as** <expresión de consulta>*

- 2 Vista ejemplo de base de datos por cada taller
- Insertar registros en bases de dato