



# ANÁLISIS Y DISEÑO LÓGICO DE SISTEMAS

Noviembre 2023

Luis E.Canales C.  
lcanales@utalca.cl

# Ejemplo Self- Join

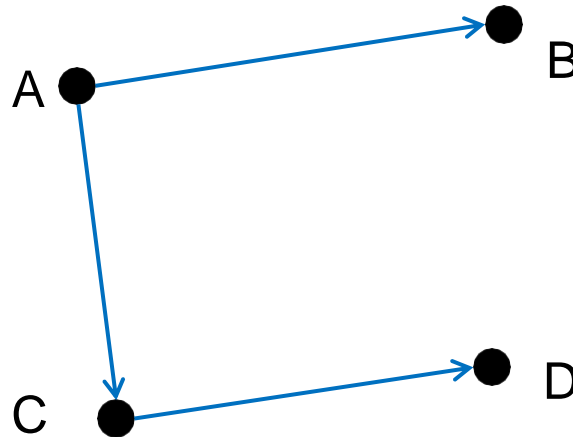
---

¡Self Joins se puede usar para encontrar rutas en un grafo!

# Ejemplo Self- Join

---

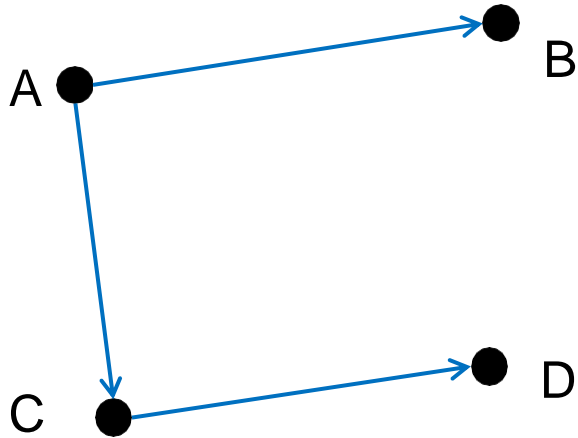
Aristas(start, end)



start	end
A	B
A	C
C	D

# Ejemplo Self- Join

Aristas(start, end)



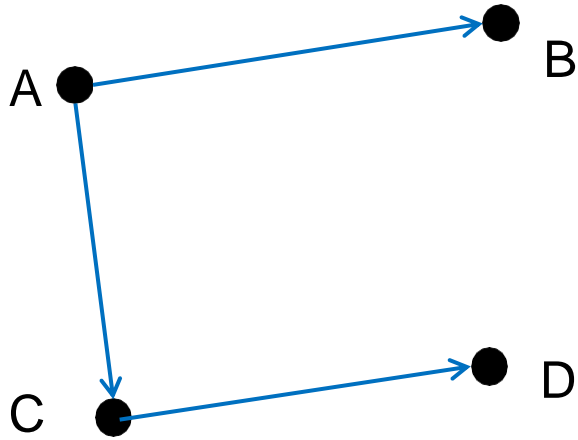
```
SELECT *  
FROM   Aristas e1, Aristas e2
```

start	end
A	B
A	C
C	D

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C
C	D	C	D

# Ejemplo Self- Join

Aristas(start, end)



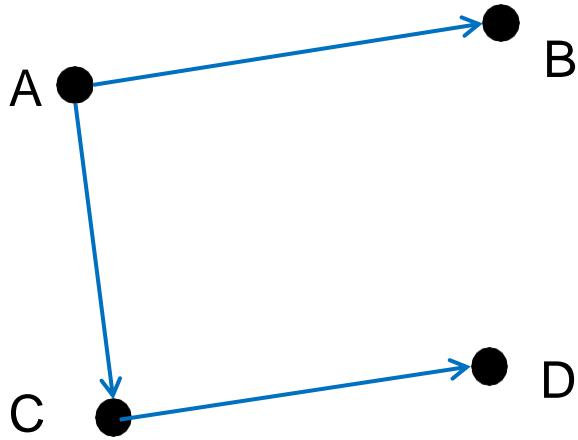
```
SELECT *  
FROM   Aristas e1, Aristas e2  
WHERE  e1.end = e2.start
```

start	end
A	B
A	C
C	D

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C
C	D	C	D

# Ejemplo Self- Join

Aristas(start, end)



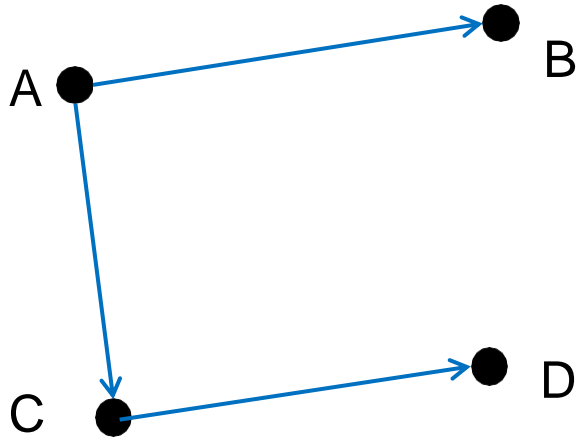
```
SELECT *  
FROM   Aristas e1, Aristas e2  
WHERE  e1.end = e2.start
```

start	end
A	B
A	C
C	D

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C
C	D	C	D

# Ejemplo Self- Join

Aristas(start, end)



```
SELECT e1.start, e2.end
FROM   Aristas As e1, Aristas AS e2
WHERE  e1.end = e2.start
```

e1.start	e2.end
A	D

start	end
A	B
A	C
C	D

# El problema del testimonio simplificado

---

- Queremos un join a los máximos de cada trabajo
  - La técnica GROUP BY fue interesante
  - Anteriormente sugerimos que podemos
    - **Primero, calcular el máximo y luego el join**

UserID	Nombre	Trabajo	Salario	Máximo
123	Juan	Contador	500000	600000
345	Aline	Contador	600000	600000
567	Magda	Profesora	900000	1000000
789	Diana	Profesora	1000000	1000000

Retorna a la persona (o personas) con el salario más alto para cada tipo de trabajo



# El problema del testimonio - previamente

---

UserID	Nombre	Trabajo	Salario	Máximo
123	Juan	Contador	500000	600000
345	Aline	Contador	600000	600000
567	Magda	Profesora	900000	1000000
789	Diana	Profesora	1000000	1000000

Retorna a la persona (o personas) con el salario más alto para cada tipo de trabajo

```
SELECT S1.Nombre, MAX(S2.Salario)
FROM Sueldos AS S1, Sueldos AS S2
WHERE S1.Trabajo = S2.Trabajo
GROUP BY S2.Trabajo, S1.Salario, S1.Nombre
HAVING S1.Salario = MAX(S2.Salario)
```

# El problema del testimonio - previamente

```
SELECT S1.Nombre, MAX(S2.Salario)
  FROM Sueldos AS S1, Sueldos AS S2
 WHERE S1.Trabajo = S2.Trabajo
 GROUP BY S2.Trabajo, S1.Salario, S1.Nombre
 HAVING S1.Salario = MAX(S2.Salario)
```

Join sobre grupo de  
atributos

S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
123	Juan	Contador	500000
345	Aline	Contador	600000
345	Aline	Contador	600000
567	Magda	Profesora	900000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
789	Diana	Profesora	1000000

S2

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
345	Aline	Contador	600000
123	Juan	Contador	500000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
789	Diana	Profesora	1000000
567	Magda	Profesora	900000

# El problema del testimonio - previamente

```
SELECT S1.Nombre, MAX(S2.Salario)
  FROM Sueldos AS S1, Sueldos AS S2
 WHERE S1.Trabajo = S2.Trabajo
 GROUP BY S2.Trabajo, S1.Salario, S1.Nombre
 HAVING S1.Salario = MAX(S2.Salario)
```

Grupo de atributos  
adicionales

S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
123	Juan	Contador	500000
345	Aline	Contador	600000
345	Aline	Contador	600000
567	Magda	Profesora	900000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
789	Diana	Profesora	1000000

S2

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
345	Aline	Contador	600000
123	Juan	Contador	500000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000
789	Diana	Profesora	1000000
567	Magda	Profesora	900000

# El problema del testimonio - previamente

```
SELECT S1.Nombre, MAX(S2.Salario)
  FROM Sueldos AS S1, Sueldos AS S2
 WHERE S1.Trabajo = S2.Trabajo
 GROUP BY S2.Trabajo, S1.Salario, S1.Nombre
 HAVING S1.Salario = MAX(S2.Salario)
```

Grupo de atributos  
adicionales

S1

S2

UserID	Nombre	Trabajo	Salario	UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000	123	Juan	Contador	500000
123	Juan	Contador	500000	345	Aline	Contador	600000
345	Aline	Contador	600000	345	Aline	Contador	600000
345	Aline	Contador	600000	123	Juan	Contador	500000
567	Magda	Profesora	900000	567	Magda	Profesora	900000
567	Magda	Profesora	900000	789	Diana	Profesora	1000000
789	Diana	Profesora	1000000	789	Diana	Profesora	1000000
789	Diana	Profesora	1000000	567	Magda	Profesora	900000

# El problema del testimonio - previamente

```
SELECT S1.Nombre, MAX(S2.Salario)
  FROM Sueldos AS S1, Sueldos AS S2
 WHERE S1.Trabajo = S2.Trabajo
 GROUP BY S2.Trabajo, S1.Salario, S1.Nombre
 HAVING S1.Salario = MAX(S2.Salario)
```

S1

S2

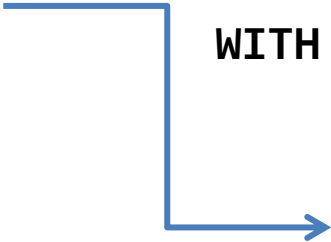
UserID	Nombre	Trabajo	Salario	UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000	123	Juan	Contador	500000
123	Juan	Contador	500000	345	Aline	Contador	600000
345	Aline	Contador	600000	345	Aline	Contador	600000
345	Aline	Contador	600000	123	Juan	Contador	500000
567	Magda	Profesora	900000	567	Magda	Profesora	900000
567	Magda	Profesora	900000	789	Diana	Profesora	1000000
789	Diana	Profesora	1000000	789	Diana	Profesora	1000000
789	Diana	Profesora	1000000	567	Magda	Profesora	900000

# El problema del testimonio simplificado

---

```
SELECT S1.Nombre, MAX(S2.Salario)
  FROM Sueldos AS S1, Sueldos AS S2
 WHERE S1.Trabajo = S2.Trabajo
 GROUP BY S2.Trabajo, S1.Trabajo, S1.Nombre
 HAVING S1.Salario = MAX(S2.Salario)
```

Podemos calcular  
la misma pregunta!



```
WITH MaxSalario AS
  (SELECT S1.Trabajo AS Trabajo,
         MAX(S1.Salario) AS Salario
   FROM Sueldos AS S1
  GROUP BY S1.Trabajo)
SELECT S.Nombre, S.Salario
  FROM Sueldos AS S, MaxSalario AS MS
 WHERE S.Trabajo = MS.Trabajo AND
       S.Salario = MS.Salario
```

# El problema del testimonio simplificado

---

## MaxSalario

Trabajo	Salario
Contador	600000
Profesora	1000000

Resultado intermedio útil!

```
WITH MaxSalario AS
    (SELECT S1.Trabajo AS Trabajo,
        MAX(S1.Salario) AS Salario
    FROM Sueldos AS S1
    GROUP BY S1.Trabajo)
SELECT S.Nombre, S.Salario
    FROM Sueldos AS S, MaxSalario AS MS
    WHERE S.Trabajo = MS.Trabajo AND
        S.Salario = MS.Salario
```

# El problema del testimonio simplificado

```
WITH MaxSalario AS
    (SELECT S1.Trabajo AS Trabajo,
             MAX(S1.Salario) AS Salario
     FROM Sueldos AS S1
     GROUP BY S1.Trabajo)
SELECT S.Nombre, S.Salario
FROM   Sueldos AS S, MaxSalario AS MS
WHERE  S.Trabajo = MS.Trabajo AND
       S.Salario = MS.Salario
```

## Sueldos

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

## MaxSalario

Trabajo	Salario
Contador	600000
Profesora	1000000



# El problema del testimonio simplificado

```
WITH MaxSalario AS
  (SELECT S1.Trabajo AS Trabajo,
          MAX(S1.Salario) AS Salario
   FROM Sueldos AS S1
   GROUP BY S1.Trabajo)
SELECT S.Nombre, S.Salario
   FROM   Sueldos AS S, MaxSalario AS MS
  WHERE  S.Trabajo = MS.Trabajo AND
         S.Salario = MS.Salario
```

Join Predicado

## Sueldos

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

## MaxSalario

Trabajo	Salario
Contador	600000
Profesora	1000000

# El problema del testimonio simplificado

```
WITH MaxSalario AS
  (SELECT S1.Trabajo AS Trabajo,
          MAX(S1.Salario) AS Salario
   FROM Sueldos AS S1
   GROUP BY S1.Trabajo)
SELECT S.Nombre, S.Salario
   FROM Sueldos AS S, MaxSalario AS MS
  WHERE S.Trabajo = MS.Trabajo AND
        S.Salario = MS.Salario
```

Selección Predicado

## Sueldos

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

## MaxSalario

Trabajo	Salario
Contador	600000
Profesora	1000000

# El problema del testimonio simplificado

```
WITH MaxSalario AS
  (SELECT S1.Trabajo AS Trabajo,
          MAX(S1.Salario) AS Salario
   FROM Sueldos AS S1
   GROUP BY S1.Trabajo)
SELECT S.Nombre, S.Salario
  FROM Sueldos AS S, MaxSalario AS MS
 WHERE S.Trabajo = MS.Trabajo AND
       S.Salario = MS.Salario
```

Resolver un subproblema puede hacerle la vida más fácil

## Sueldos

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

## MaxSalario

Trabajo	Salario
Contador	600000
Profesora	1000000

# Algo más sobre Subqueries

---

- Las subconsultas pueden interpretarse como **valores únicos** o como **relaciones enteras**
  - Un solo valor (una relación 1x1) se puede retornar como parte de una tupla
  - Una relación puede ser:
    - Se utiliza como entrada para otra consulta.
    - Se verificó la contención de un valor

# Operaciones de Conjuntos

---

- **SQL imita** la teoría de conjuntos de muchas maneras
- **Bag** = duplicados permitidos
- **UNION** (TODOS) → conjunto union (bag union)
- **INTERSECT** (ALL) → conjunto intersección (bag intersección)
- **EXCEPT** (TODOS) → conjunto diferencia (bag diferencia)

# Operaciones de Conjuntos

---

- Los operadores de SQL, similares a los de teoría de conjuntos, básicamente combinan dos consultas (no es realmente una subconsulta ...)

```
SELECT * FROM T1  
UNION  
SELECT * FROM T2;
```

# Subqueries en un **SELECT**

---

- Debe devolver un valor único
- Usos:
  - Calcular un valor asociado

# Subqueries en un **SELECT**

---

- Debe devolver un valor único
- Usos:
  - Calcular un valor asociado
- **Ejemplo:** para cada empleado, devuelva su nombre y el salario promedio del trabajo.

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```



# Subqueries en un **SELECT**

---

- Debe devolver un valor único
- Usos:
  - Calcular un valor asociado
- **Ejemplo:** para cada empleado, devuelva su nombre y el salario promedio del trabajo.

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

**Subconsulta correlacionada!**

La semántica es que la subconsulta completa se vuelve a calcular para cada tupla de S

# Subqueries en un **SELECT**

---

- Debe devolver un valor único
- Usos:
  - Calcular un valor asociado
- **Ejemplo:** para cada empleado, devuelva su nombre y el salario promedio del trabajo.

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

# Subqueries en un SELECT

---

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S



UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S



UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

Sueldos S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S



UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

Sueldos S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S



UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

Sueldos S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

55000



# Subqueries en un SELECT

---

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

55000



# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

55000

Sueldos S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000



# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

Sueldos S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

55000



# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

Sueldos S1

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

55000

55000



# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S

UserID	Nombre	Trabajo	Salario	
123	Juan	Contador	500000	55000
345	Aline	Contador	600000	55000
567	Magda	Profesora	900000	95000
789	Diana	Profesora	1000000	

# Subqueries en un SELECT

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```

Sueldos S

UserID	Nombre	Trabajo	Salario	
123	Juan	Contador	500000	55000
345	Aline	Contador	600000	55000
567	Magda	Profesora	900000	95000
789	Diana	Profesora	1000000	95000



# Subqueries en un SELECT

---

Para cada persona, encuentre el salario promedio de su trabajo

```
SELECT S.Nombre, (SELECT AVG(S1.Salario)
                  FROM Sueldos AS S1
                  WHERE S.Trabajo = S1.Trabajo)
FROM Sueldos AS S
```



Igual (pero no anidado)

```
SELECT S1.Nombre, AVG(S2.Salario)
FROM Sueldos AS S1, Sueldos AS S2
WHERE S1.Trabajo = S2.Trabajo
GROUP BY S1.Nombre
```

# Subqueries en un SELECT

---

Para cada persona, encuentre la cantidad de autos que conduce

# Subqueries en un SELECT

---

Para cada persona, encuentre la cantidad de autos que conduce

```
SELECT S.Nombre, (SELECT COUNT(R.Car)
                    FROM Registros AS R
                    WHERE S.UserID = R.UserID)
FROM Sueños AS S
```



¿Lo Mismo? ¡Discutir!

```
SELECT S.Nombre, COUNT(R.Car)
FROM Sueños AS S, Registros AS R
WHERE S.UserID = R.UserID
GROUP BY S.Nombre
```

# Subqueries en un SELECT

---

Para cada persona, encuentre la cantidad de autos que conduce

```
SELECT S.Nombre, (SELECT COUNT(R.Car)
                   FROM Registros AS R
                   WHERE S.UserID = R.UserID)
FROM Sueños AS S
```

¡El caso de 0  
recuentos no está  
cubierto!

```
SELECT S.Nombre, COUNT(R.Car)
FROM Sueños AS S, Registros AS R
WHERE S.UserID = R.UserID
GROUP BY S.Nombre
```



# Subqueries en un SELECT

---

Para cada persona, encuentre la cantidad de autos que conduce

```
SELECT S.Nombre, (SELECT COUNT(R.Car)
                    FROM Registros AS R
                    WHERE S.UserID = R.UserID)
FROM Sueldos AS S
```



Todavía es posible descifrar y desanidar

# Subqueries en un SELECT

---

Para cada persona, encuentre la cantidad de autos que conduce

```
SELECT S.Nombre, (SELECT COUNT(R.Car)
                    FROM Registros AS R
                    WHERE S.UserID = R.UserID)
FROM Sueldos AS S
```



Todavía es posible descifrar y desanidar

```
SELECT S.Nombre, COUNT(R.Car)
FROM Sueldos AS S LEFT OUTER JOIN
Registros AS R ON S.UserID = R.UserID
GROUP BY S.Nombre
```

# Subqueries en un FROM

---

- Equivalente a una subquery WITH
- Usos:
  - Resolver subproblemas que luego se pueden unir / evaluar

```
WITH MaxPago AS
    (SELECT S1.Trabajo AS Trabajo, MAX(S1.Salario) AS Salario
     FROM Sueldos AS S1
     GROUP BY S1.Trabajo)
```

```
SELECT S.Nombre, S.Salario
FROM Sueldos AS S, MaxPago AS MP
WHERE S.Trabajo = MP.Trabajo AND S.Salario = MP.Salario
```

```
SELECT S.Nombre, S.Salario
FROM Sueldos AS S, (SELECT S1.Trabajo AS Trabajo, MAX(S1.Salario) AS Salario
                    FROM Sueldos AS S1
                    GROUP BY S1.Trabajo) AS MP
WHERE S.Trabajo = MP.Trabajo AND S.Salario = MP.Salario
```

# Subqueries en WHERE/HAVING

- Usos:

- ANY  $\rightarrow \exists$
- ALL  $\rightarrow \forall$
- (NOT) IN  $\rightarrow (\notin) \in$
- (NOT) EXISTS  $\rightarrow (\emptyset = \dots) \emptyset \neq \dots$

UserID	Nombre	Trabajo	Salario
123	Juan	Contador	500000
345	Aline	Contador	600000
567	Magda	Profesora	900000
789	Diana	Profesora	1000000

UserID	Car
123	Charger
567	Civic
567	Pinto

Encuentra el nombre y el salario de las personas que no conducen automóviles

```
SELECT S.Nombre, S.Salario
FROM Sueldos AS S
WHERE S.UserID NOT IN (SELECT UserID
                        FROM Registros)
```

# Subqueries en WHERE

---

- SELECT ..... WHERE EXISTS (subquery);
- SELECT ..... WHERE NOT EXISTS (subquery);
- SELECT ..... WHERE attribute IN (subquery);
- SELECT ..... WHERE attribute NOT IN (subquery);
- SELECT ..... WHERE constant > ANY (subquery );
- SELECT ..... WHERE constant > ALL (subquery);

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañia (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio < 200

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio <200

Cuantificadores existenciales

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)  
Compañia (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio <200

Cuantificadores existenciales

Uso de **EXISTS**: EXISTS (subquery) **retorna verdadero iff la cardinalidad de la subquery > 0**

```
SELECT DISTINCT C.cnombre
FROM Compañia AS C
WHERE EXISTS (SELECT *
              FROM Producto AS P
              WHERE C.cid = P.cid and P.precio < 200)
```



# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio <200

Cuantificadores existenciales

Uso de **IN**: Attr IN (subquery) retorna verdadero iff el valor del attr está contenido en la subquery

```
SELECT DISTINCT C.cnombre
FROM Compañía C
WHERE C.cid IN (SELECT P.cid
                FROM Producto P
                WHERE P.precio < 200)
```

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio <200

Cuantificadores existenciales

Uso de **ANY**:  $\text{const} > \text{ANY}(\text{sub})$  retorna verdadero si  $\text{const} > \text{valor}$  para al menos un valor en sub

```
SELECT DISTINCT C.cnombre
FROM Compañía C
WHERE 200 > ANY (SELECT precio
                  FROM Producto P
                  WHERE P.cid = C.cid)
```

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio <200

Cuantificadores existenciales

Ahora vamos a desanidarlo

```
SELECT DISTINCT C.cnombre
FROM Compañía C
WHERE 200 > ANY (SELECT precio
                  FROM Producto P
                  WHERE P.cid = C.cid)
```

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)  
Compañia (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio <200

Cuantificadores existenciales

Ahora vamos a desanidarlo

```
SELECT DISTINCT C.cnombre  
FROM Compañia C, Producto P  
WHERE C.cid = P.cid and P.precio < 200
```

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)  
Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas que fabrican algunos productos con un precio <200

Cuantificadores existenciales

Ahora vamos a desanidarlo

```
SELECT DISTINCT C.cnombre  
FROM Compañía C, Producto P  
WHERE C.cid = P.cid and P.precio < 200
```

¡Los cuantificadores existenciales son fáciles! 😊

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañia (cid, cnombre, ciudad)

Encuentre todas las empresas tal que todos sus productos tiene precio < 200

igual que:

Encuentre todas las empresas que fabrican solo productos con un precio <200

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañia (cid, cnombre, ciudad)

Encuentre todas las empresas tal que todos sus productos tiene precio  $< 200$

igual que:

Encuentre todas las empresas que fabrican solo productos con un precio  $< 200$

Cuantificadores universales

¡Los cuantificadores universales son difíciles! 😞

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañia (cid, cnombre, ciudad)

Encuentre todas las empresas tal que todos sus productos tiene precio < 200

1. Encuentra las otras compañías que hacen algún producto  $\geq 200$

```
SELECT DISTINCT C.cnombre
FROM Compañia C
WHERE C.cid IN (SELECT P.cid
                FROM Producto P
                WHERE P.precio >= 200)
```



# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas tal que todos sus productos tiene precio < 200

1. Encuentra las otras compañías que hacen algún producto  $\geq 200$

```
SELECT DISTINCT C.cnombre
FROM Compañía C
WHERE C.cid IN (SELECT P.cid
                FROM Producto P
                WHERE P.precio >= 200)
```

2. Encuentra todas las compañías tal que todos sus productos tiene precio < 200

```
SELECT DISTINCT C.cnombre
FROM Compañía C
WHERE C.cid NOT IN (SELECT P.cid
                    FROM Producto P
                    WHERE P.precio >= 200)
```

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)  
Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas tal que todos sus productos tiene precio < 200

Cuantificadores universales

Uso de **EXISTS**

```
SELECT DISTINCT C.cnombre
FROM Compañía C
WHERE NOT EXISTS (SELECT *
                  FROM Producto P
                  WHERE P.cid = C.cid and P.precio >= 200)
```

# Subqueries en WHERE

---

Producto (pnombre, precio, cid)

Compañía (cid, cnombre, ciudad)

Encuentre todas las empresas tal que todos sus productos tiene precio < 200

Cuantificadores universales

Uso de **ALL**

```
SELECT DISTINCT C.cnombre
FROM Compañía C
WHERE 200 >= ALL (SELECT precio
                  FROM Producto P
                  WHERE P.cid = C.cid)
```

# Subqueries en WHERE/HAVING

---

- Usos:

- ANY  $\rightarrow \exists$       WHERE P.UserID =  
R.UserID)
- ALL  $\rightarrow \forall$
- (NOT) IN  $\rightarrow (\notin) \in$
- (NOT) EXISTS  $\rightarrow (\emptyset = \dots) \emptyset \neq \dots$

Encuentra el nombre y el salario de las personas que no conducen automóviles

```
SELECT P.Nombre, P.Salario
FROM Sueldos AS S
WHERE S.UserID NOT IN (SELECT UserID
                        FROM Registro)
                        WHERE P.UserID = R.UserID
```

# Subqueries en WHERE/HAVING

---

- Usos:

- ANY  $\rightarrow \exists$       WHERE P.UserID =  
R.UserID)
- ALL  $\rightarrow \forall$
- (NOT) IN  $\rightarrow (\notin) \in$
- (NOT) EXISTS  $\rightarrow (\emptyset = \dots) \emptyset \neq \dots$

Encuentra el nombre y el salario de las personas que no conducen automóviles

```
SELECT P.Nombre, P.Salario
FROM Sueldos AS S
WHERE S.UserID NOT IN (SELECT UserID
                        FROM Registro)
```

# Codificación de cuantificadores universales

---

- ¿Podríamos codificar un cuantificador universal con una consulta **SELECT-FROM-WHERE** sin subconsultas o agregados?

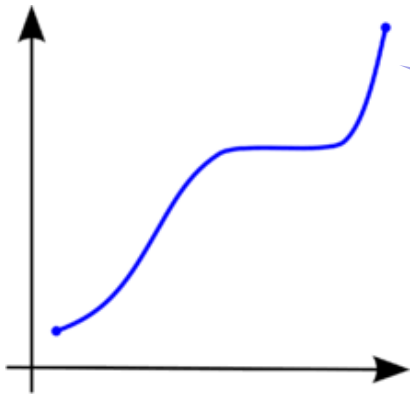
# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde  $I$  y  $J$  son instancias de datos y  $q$  es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de  $I$ , la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de  $I$ .



Las queries monótonas pueden ser similar a las funciones monótonicamente crecientes cuando consideramos la cardinalidad de los resultados

# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde  $I$  y  $J$  son instancias de datos y  $q$  es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de  $I$ , la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de  $I$ .

```
SELECT S.Nombre, S.Car  
FROM Sueldos AS S, Registro AS R  
WHERE S.UserID = R.UserID
```

¿Es esta consulta monótona?



# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde  $I$  y  $J$  son instancias de datos y  $q$  es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de  $I$ , la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de  $I$ .

```
SELECT S.Nombre, S.Car  
FROM Sueldos AS S, Registro AS R  
WHERE S.UserID = R.UserID
```

¿Es esta consulta monótona? **Sí!!!**

# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde  $I$  y  $J$  son instancias de datos y  $q$  es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de  $I$ , la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de  $I$ .

```
SELECT S.Nombre, S.Car  
FROM Sueldos AS S, Registro AS R  
WHERE S.UserID = R.UserID
```

¿Es esta consulta monótona? **Sí!!!**

No puedo agregar tuplas a la nómina o Registre eso  
"Eliminar" un resultado anterior

# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde I y J son instancias de datos y q es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de I, la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de I.

```
SELECT S.Nombre  
FROM Sueldos AS S  
WHERE S.Salario >= ALL (SELECT Salario  
                        FROM Sueldos)
```

¿Es esta consulta monótona?

# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde I y J son instancias de datos y q es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de I, la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de I.

```
SELECT S.Nombre  
FROM Sueldos AS S  
WHERE S.Salario >= ALL (SELECT Salario  
                        FROM Sueldos)
```

¿Es esta consulta monótona? **NO!!**

# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde I y J son instancias de datos y q es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de I, la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de I.

```
SELECT S.Nombre  
FROM Sueldos AS S  
WHERE S.Salario >= ALL (SELECT Salario  
                        FROM Sueldos)
```

Puedo agregar una tupla a Sueldos que tiene un valor de salario más alto que cualquier otro

¿Es esta consulta monótona? **NO!!**

# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde  $I$  y  $J$  son instancias de datos y  $q$  es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de  $I$ , la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de  $I$ .

```
SELECT S.Trabajo, COUNT (*)  
FROM Sueldos AS S  
GROUP BY S.Trabajo
```

¿Es esta consulta monótona?

# Monotonicidad

## Monótona

Una consulta monótona es aquella que obedece a la siguiente regla donde  $I$  y  $J$  son instancias de datos y  $q$  es una query:

$$I \subseteq J \rightarrow q(I) \subseteq q(J)$$

Eso es para cualquier superconjunto de  $I$ , la consulta sobre ese superconjunto debe contener al menos los resultados de la consulta de  $I$ .

```
SELECT S.Trabajo, COUNT (*)  
FROM Sueldos AS S  
GROUP BY S.Trabajo
```

¿Es esta consulta monótona? **NO!!**

Los agregados generalmente son sensible a cualquier nueva tupla

# Monotonicidad

---

- Todas las consultas SELECT-FROM-WHERE (sin agregados) son monótonos
- Las queries con cuantificadores universales no son generalmente monótonos
- Debe hacer algo "complejo" si necesita codificar un cuantificador universal



# A considerar

---

- SQL es capaz de reflejar la lógica sobre conjuntos más o menos directamente
- La interpretación interna de consultas anidadas puede estar bastante involucrado
  - Pero nuestro DBMS es capaz de derivar tales interpretaciones automáticamente
- Podemos razonar sobre el poder expresivo de ciertas consultas