

Facultad de Ingeniería



# Lenguaje de consulta de datos

## Tema VII

Semestre 2020-2

**El alumno comprenderá los conceptos teóricos y prácticos que le permitan realizar el acceso y consulta de datos a través del uso de sentencias del lenguaje SQL, así como las diferentes estrategias de acceso a datos.**

# Select

**La sentencia select nos permite obtener información de una tabla.**

# Select

```
SELECT (col1, col2, ...)  
FROM nombre_Tabla nt  
[WHERE ...]
```

## Consideraciones

- Permisos en la(s) tablas**
- Podemos hacer tan compleja la obtención de información como sea necesario**

# Select

```
SELECT DISTINCT nombre AS  
nombre_unico FROM cliente;
```

# Select

```
SELECT DISTINCT nombre AS  
nombre_unico FROM cliente  
ORDER BY nombre;
```

**Algunos DBMS las emplean para complementar consultas**



**El álgebra relacional define una serie de operaciones que podemos aplicar a una o más relaciones**

## Dos tipos:

- **Unarias**
- **Binarias**

Nos permite remover atributos que no es de nuestro interés visualizar.

$$\pi_{\textit{nombres-atributos}}(R)$$

*Empleado*

*SELECT sueldo,  
nombre*

*FROM empleado;*

$\pi_{sueldo,nombre}(Empleado)$

num. empleado	nombre	departamento	sueldo
2342	Juan	Contabilidad	8000
5236	Fernando	Computacion	12000
7643	Lorena	Marketing	10000
1232	Francisco	Computacion	8000
4356	Jimena	Computacion	13500

Sueldo	Nombre
8000	Juan
12000	Fernando
10000	Lorena
8000	Francisco
13500	Jimena

Permite seleccionar registros que cumplen una determinada condición, que puede evaluarse con los operadores:

<, >, <=, >=, =, !=, and, or

$$\sigma_{condiciones}(R)$$

## *Empleado*

num. empleado	nombre	departamento	sueldo
2342	Juan	Contabilidad	8000
5236	Fernando	Computacion	12000
7643	Lorena	Marketing	10000
1232	Francisco	Computacion	8000
4356	Jimena	Computacion	13500

$\sigma_{\text{departamento}='computacion' \text{ AND } \text{sueldo}>9000}(\text{Empleado})$

num. empleado	nombre	departamento	sueldo
5236	Fernando	Computacion	12000
4356	Jimena	Computacion	13500

```
SELECT *  
FROM empleado  
WHERE departamento = 'computacion'  
AND sueldo > 9000;
```

Para que dos tablas sean compatibles, deben cumplir lo siguiente:

1. Deben ser del mismo grado
2. Los atributos deben tener el mismo nombre en ambas relaciones
3. El  $i$ -ésimo atributo de la primer relación debe ser del mismo dominio del  $i$ -ésimo atributo de la segunda relación, para toda  $i$



# Unión

Permite obtener una nueva relación,  
compuesta por todos los registros de la  
primera y segunda relación

$$R1 \cup R2$$

# Unión

*Empleado*

nombre	edad
Juan	25
Fernando	26
Lorena	23
Francisco	22
Jimena	24

*Gerente*

nombre	edad
Francisco	22
Laura	29
Xavier	26

*Empleado U Gerente*

nombre	edad
Juan	25
Fernando	26
Lorena	23
Francisco	22
Jimena	24
Laura	29
Xavier	26

Permite obtener los registros que se encuentran en ambas relaciones

$$R1 \cap R2$$

# Intersección



*Empleado*

nombre	edad
Juan	25
Fernando	26
Lorena	23
Francisco	22
Jimena	24

*Gerente*

nombre	edad
Francisco	22
Laura	29
Xavier	26

*Empleado*  $\cap$  *Gerente*

nombre	edad
Francisco	22

Permite obtener los registros que se encuentran sólo en la primera relación

$$R1 - R2$$

# Diferencia



*Empleado*

nombre	edad
Juan	25
Fernando	26
Lorena	23
Francisco	22
Jimena	24

*Gerente*

nombre	edad
Francisco	22
Laura	29
Xavier	26

*Empleado — Gerente*

nombre	edad
Juan	25
Fernando	26
Lorena	23
Jimena	24

Genera las combinaciones entre los registros de ambas relaciones

$$R1 \times R2$$

como resultado una nueva relación de grado  $n + m$  y cardinalidad  $a*b$

# Producto cartesiano



*R1*

nombre	edad
Juan	25
Fernando	26

*R2*

departamento	sueldo
Contabilidad	12000
Sistemas	13900
Marketing	10000

*R1 X R2*

nombre	edad	departamento	sueldo
Juan	25	Contabilidad	12000
Juan	25	Sistemas	13900
Juan	25	Marketing	10000
Fernando	26	Contabilidad	12000
Fernando	26	Sistemas	13900
Fernando	26	Marketing	10000



Permite combinar registros de dos relaciones a través de una condición sobre los atributos

$$R1 \bowtie_{condicion} R2$$

$$\sigma_{condicion}(R1 \times R2)$$

# Join



$R1$

A	B	C
1	2	3
6	7	8
9	7	8

$R2$

B	C	D
2	3	4
2	3	5
7	8	10

$R1 \bowtie_{A < D} R2$

A	R1.B	R1.C	R2.B	R2.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Genera las combinaciones entre los atributos que se llaman igual en las dos relaciones

$$R1 \bowtie R2$$

¿Y si no hay?

# Join natural



*R1*

A	B
1	2
3	4

*R2*

B	C	D
2	5	6
4	7	8
9	10	11

*R1* ⋈ *R2*

A	B	C	D
1	2	5	6
3	4	7	8

# Ejemplo



cuenta(nombreSucursal,numCta,saldo)

sucursal(nombreSucursal,ciudad,activos)

cliente(nombreCliente,calle,ciudad)

ctaCliente(nombreCliente,numCta)

prestamo(nombreSucursal,numPrestamo,importe)

prestatario(nombreCliente,numPrestamo)

Encontrar la información de todos los préstamos realizados en la sucursal “copilco”

# Ejemplo



Determinar el nombre de los clientes que viven en Guanajuato

# Ejemplo



Nombre de los clientes del banco que tienen una cuenta, un préstamo o ambas cosas



# Ejemplo



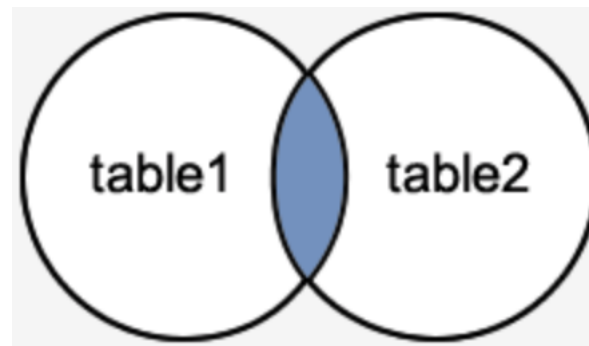
Relación de clientes que tienen abierta una cuenta pero no tienen ninguna de préstamo

# Ejemplo



Nombre de los clientes con préstamo mayor a 5000 pesos

Regresa todas las columnas de múltiples tablas donde se cumple la condición del join



**inner**

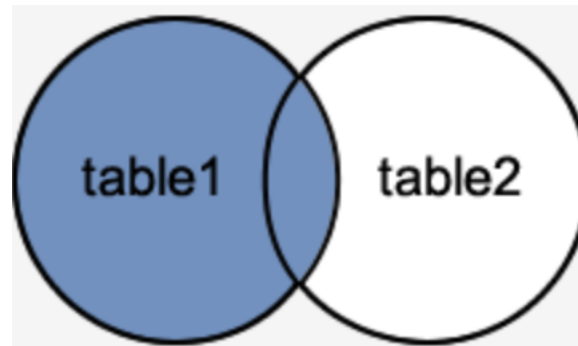
SELECT columns

FROM table1

INNER JOIN table2

ON table1.column = table2.column;

Regresa los registros de la tabla del lado izquierdo y los registros del lado derecho que hagan match en la condición



*left outer*

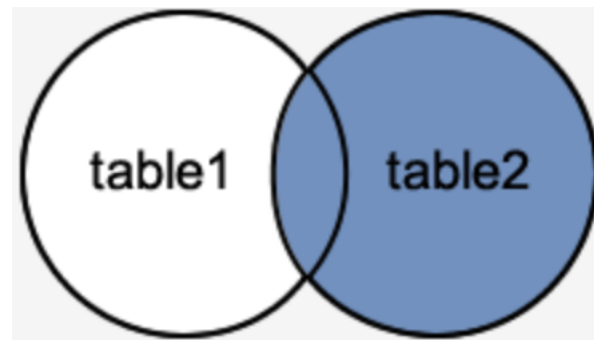
SELECT columns

FROM table1

LEFT OUTER JOIN table2

ON table1.column = table2.column;

Regresa los registros de la tabla del lado derecho y los registros del lado izquierdo que hagan match en la condición



*right outer*

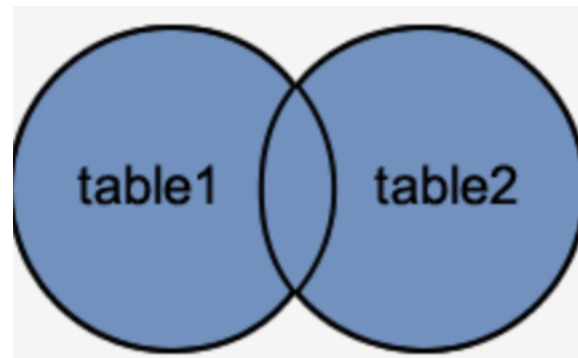
SELECT columns

FROM table1

RIGHT OUTER JOIN table2

ON table1.column = table2.column;

Regresa los registros de la tabla izquierda y los registros de la tabla derecha, asignando un valor nulo donde la condición no hace match



**full outer**

SELECT columns

FROM table1

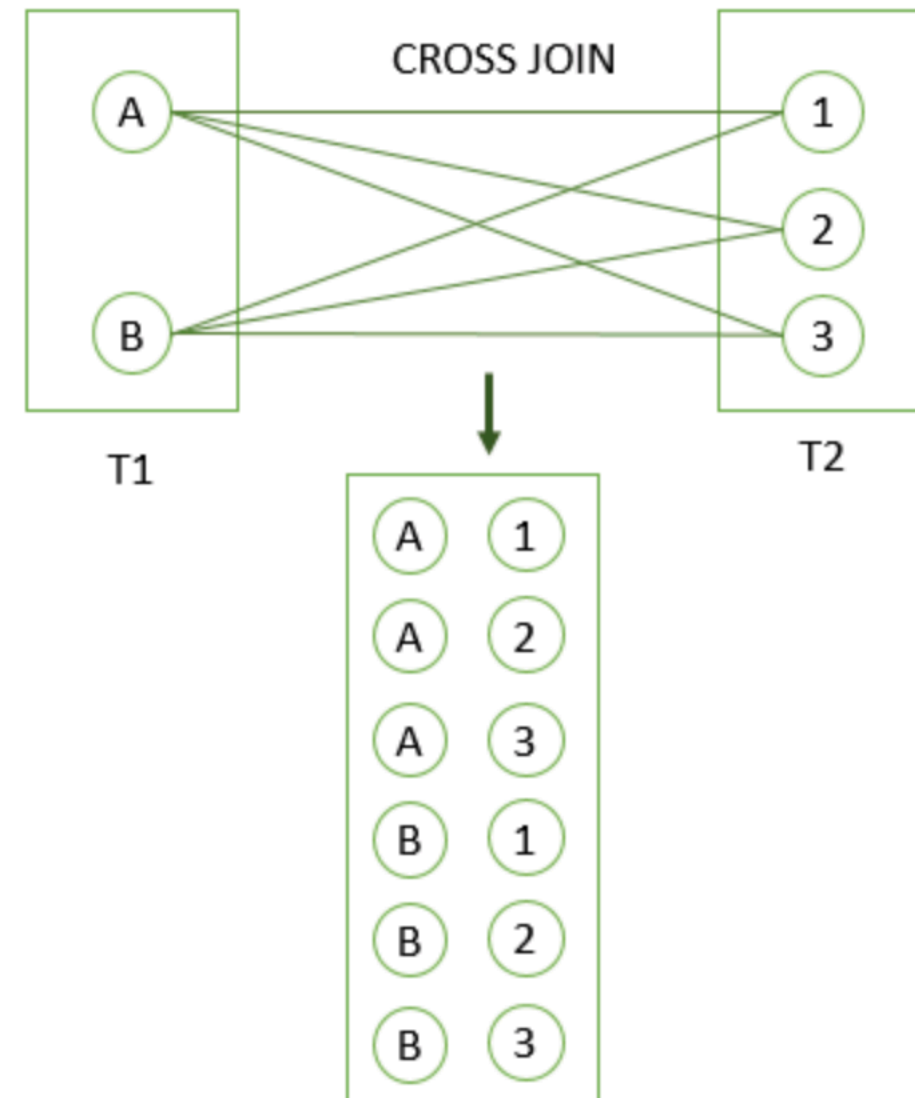
FULL OUTER JOIN table2

ON table1.column = table2.column;

Producto cartesiano entre dos tablas. No requiere condición alguna.

**cross**

SELECT columns  
FROM table1,  
CROSS JOIN table2;



Crea un join implícito basado en las columnas con el mismo nombre sobre las tablas que estamos operando.

*natural*

SELECT columns

FROM table1

NATURAL [INNER, LEFT, RIGHT] JOIN table2;



Sintaxis anterior

SELECT columns

FROM table1, table2

WHERE table1.column = table2.column;

SELECT columns

FROM table1, table2 ;

# Operadores



+

-

\*

/

%

aritméticos

AND

OR

NOT

*lógicos*

**lógicos**

**ALL: TRUE** si todos los resultados de una subconsulta cumplen la condición.

**ANY/SOME: TRUE** si alguno de los resultados de una subconsulta cumplen la condición.

**EXISTS: TRUE** si la subconsulta regresa al menos, un registro.

**lógicos**

**BETWEEN:** TRUE si el operando se encuentra en el rango de comparación.

**LIKE:** TRUE si el operando cumple un patrón.

**IN:** TRUE si el operando es igual a algún valor dentro de una lista de expresiones.

**GROUP BY:** La cláusula GROUP BY se usa en colaboración con la instrucción SELECT para agrupar esas filas en una tabla que tiene datos idénticos. Esto se hace para eliminar la redundancia en la salida y / o los agregados de cómputo que se aplican a estos grupos.

La cláusula GROUP BY sigue la cláusula WHERE en una instrucción SELECT y precede a la cláusula ORDER BY.