Facultad de Ingeniería

Lenguaje de consulta de datos

Tema VII

Semestre 2026-1

UNAM
UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

Acceso al servidor

- Pwd: numCuenta##
- User: gpo1t_2026_1_numCuenta
- Puerto: 5432
- Host: 132.248.59.8
- BD: gpo1t_2026_1_numCuenta

Objetivo

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
[CONDICIONES|AGREGADOS|
ORDENAMIENTO …]
```

Consideraciones

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

- **Literales**

```
SELECT ‘Fernando’;
```

- **Expresiones**

```
SELECT 5*4;
```

- **Alias**

```
SELECT columna AS alias FROM tabla;
```

Algunos DBMS las emplean para complementar consultas

En postgres no es necesario

Select - Distinct

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

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

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

Definición

Dos tipos:

- Unarias
- Binarias

Proyección

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

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

Proyección

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

Mostrar sólo el sueldo y nombre

Proyección

AR:

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

SQL:

```
SELECT sueldo,
nombre

FROM empleado;
```

Resultado

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

Selección

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

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

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

Selección

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

Datos de los empleados que trabajen en el departamento de computación y sueldo mayor a 9000

Selección

AR:

R1 =

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

$$R = \pi_{nombre}(R1)$$

Resultado

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

SQL:

```
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_Jefe	edad
Francisco	22
Laura	29
Xavier	26

¿Empleado U *Gerente?*

Unión

Empleado

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

Gerente

nombre_Jefe	edad
Francisco	22
Laura	29
Xavier	26

AR:

Empleado ∪ *Gerente*

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

Unión

Empleado

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

Gerente

nombre_Jefe	edad
Francisco	22
Laura	29
Xavier	26

SQL:

```
SELECT *
FROM EMPLEADO
UNION
SELECT nombre_Jefe
AS nombre, edad
FROM GERENTE
```

Empleado ∪ *Gerente*

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

Intersección

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 ∩ Gerente?

Intersección

Empleado

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

Gerente

nombre	edad
Francisco	22
Laura	29
Xavier	26

AR:

Empleado $\cap$ *Gerente*

nombre	edad
Francisco	22

Intersección

Empleado

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

Gerente

nombre_Jefe	edad
Francisco	22
Laura	29
Xavier	26

SQL:

```
SELECT *
FROM EMPLEADO
INTERSECT
SELECT nombre_Jefe
AS nombre, edad
FROM GERENTE
```

Empleado $\cap$ *Gerente*

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

Diferencia

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?

Diferencia

Empleado

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

Gerente

nombre	edad
Francisco	22
Laura	29
Xavier	26

AR:

Empleado — Gerente

nombre	edad
Juan	25
Fernando	26
Lorena	23
Jimena	24

Diferencia

SQL:

*SELECT **

FROM EMPLEADO

EXCEPT

SELECT nombre_Jefe AS nombre, edad

FROM GERENTE

Empleado

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

Gerente

nombre	edad
Francisco	22
Laura	29
Xavier	26

nombre	edad
Juan	25
Fernando	26
Lorena	23
Jimena	24

Genera las combinaciones entre los registros de ambas relaciones

$$R1 \; X \; R2$$

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?

Producto cartesiano

R1

nombre	edad
Juan	25
Fernando	26

R2

departamento	sueldo
Contabilidad	12000
Sistemas	13900
Marketing	10000

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

Join

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

$$R1 \bowtie_{condicion} R2$$

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

Join

R1

nombre	edad	dept_id
Juan	25	1
Fernando	26	2
Lucia	27	1

R2

departamento	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$¿R1 \bowtie_{R1.dept\_id=R2.dept\_id} R2?$$

Join

R1

nombre	edad	dept_id
Juan	25	1
Fernando	26	2
Lucia	27	1

R2

departamento	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$R1 \bowtie_{R1.dept\_id=R2.dept\_id} R2$$

nombre	edad	R1.dept_id	R2.dept_id	departamento	sueldo
Juan	25	1	1	Contabilidad	12000
Fernando	26	2	2	Sistemas	13900
Lucia	27	1	1	Contabilidad	12000

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

$$¿R1 \bowtie_{A<D} R2?$$

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

Join natural

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

$$R1 \bowtie R2$$

¿Y si no hay?

Join natural

R1

nombre	edad	dept_id
Juan	25	1
Fernando	26	2
Lucia	27	1

R2

departamento	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$¿R1 \bowtie R2?$$

Join natural

R1

nombre	edad	dept_id
Juan	25	1
Fernando	26	2
Lucia	27	1

R2

departamento	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$R1 \bowtie R2$$

nombre	edad	dept_id	departamento	sueldo
Juan	25	1	Contabilidad	12000
Fernando	26	2	Sistemas	13900
Lucia	25	1	Contabilidad	12000

Join natural

R1

nombre	edad	emp_dept_id
Juan	25	1
Fernando	26	2
Lucia	27	1

R2

nombre	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$¿R1 \bowtie R2?$$

Join natural

R1

nombre	edad	emp_dept_id
Juan	25	1
Fernando	26	2
Lucia	27	1

R2

nombre	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$R1 \bowtie R2$$

nombre	edad	emp_dept_id	dept_id	sueldo

Join natural

R1

nombre	edad	dept_id
Juan	25	1
Fernando	26	2
Marketing	27	1
Sistemas	30	2

R2

nombre	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$¿R1 \bowtie R2?$$

Join natural

R1

nombre	edad	dept_id
Juan	25	1
Fernando	26	2
Marketing	27	1
Sistemas	30	2

R2

nombre	sueldo	dept_id
Contabilidad	12000	1
Sistemas	13900	2
Marketing	10000	3

$$R1 \bowtie R2$$

nombre	edad	dept_id	sueldo
Sistemas	**30**	**2**	**13900**

Ejercicio 1_7

Partiendo de las siguientes relaciones:

cuenta(nombreSucursal,numCta,saldo)

sucursal(nombreSucursal,ciudad,activos)

cliente(nombreCliente,calle,ciudad)

ctaCliente(nombreCliente,numCta)

prestamo(nombreSucursal,numPrestamo,importe)

prestatario(nombreCliente,numPrestamo)

Ejercicio 1_7

- Generar el MER correspondiente
- Encontrar la información de todos los préstamos realizados en la sucursal “copilco”
- Determinar el nombre de los clientes que viven en Guanajuato

Ejercicio 1_7

- Nombre de los clientes del banco que tienen una cuenta, un préstamo o ambas cosas
- Relación de clientes que tienen abierta una cuenta pero no tienen ninguna de préstamo
- Nombre de los clientes con préstamo mayor a 5000 pesos

Ejercicio

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

$$\sigma_{\text{nombreSucursal = 'copilco'}}(\text{prestamo})$$

Ejercicio

Determinar el nombre de los clientes que viven en Guanajuato

$$\pi_{\text{nombreCliente}} \left(\sigma_{\text{ciudad = 'Guanajuato'}} (\text{cliente}) \right)$$

Ejercicio

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

$$R1 = \pi_{nombreCliente}(prestatario)$$

$$R2 = \pi_{nombreCliente}(ctaCliente)$$

$$R = R1 \cup R2$$

Ejercicio

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

$$R1 = \pi_{nombreCliente}(ctaCliente)$$

$$R2 = \pi_{nombreCliente}(prestatario)$$

$$R = R1 - R2$$

Ejercicio

Nombre de los clientes con préstamo mayor a 5000 pesos

$$R1 = prestamo \bowtie prestatario$$

$$R2 = \sigma_{importe>5000}(R1)$$

$$R = \pi_{nombreCliente}(R2)$$

Tarea 13

R1

A	X	B	Y
7	2	6	11
3	4	9	15
10	7	2	4
1	12	2	11

R2

B	W	D	Y	A	Z
2	5	6	11	1	30
4	7	8	4	7	8
9	10	11	28	5	12

Tabla que incluya:

- *atributos*
- *registros*

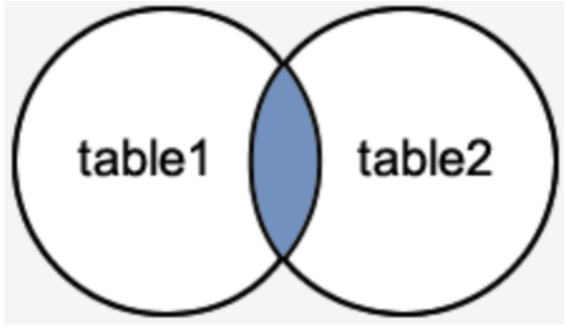
$$R1 X R2$$

$$R2 \bowtie R1$$

$$R1 \bowtie_{((R1.A > R2.Z \; or \; R1.A >= R2.W) \; and \; R1.Y = R2.Y)} R2$$

Joins

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

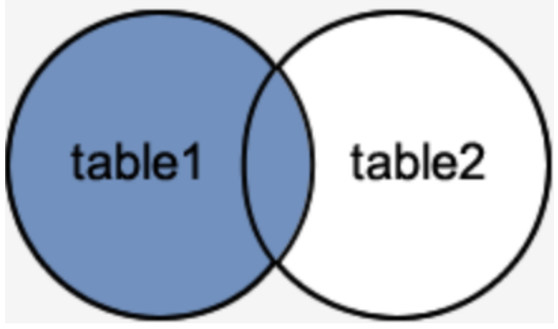


inner

```
SELECT columns
FROM table
INNER JOIN table2
ON table1.column = table2.column;
```

Joins

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

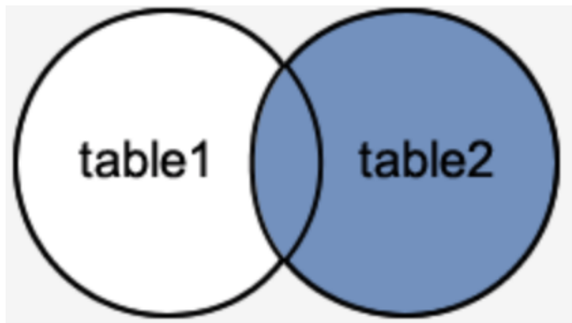


left

```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.column = table2.column;
```

Joins

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

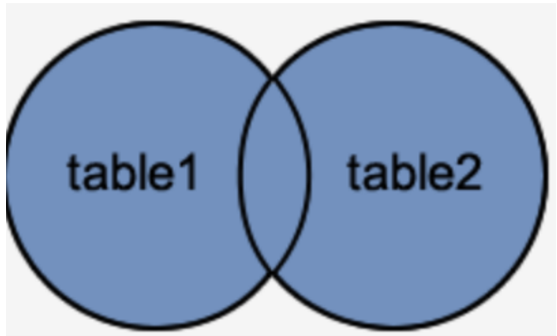


right

```
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.column = table2.column;
```

Joins

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



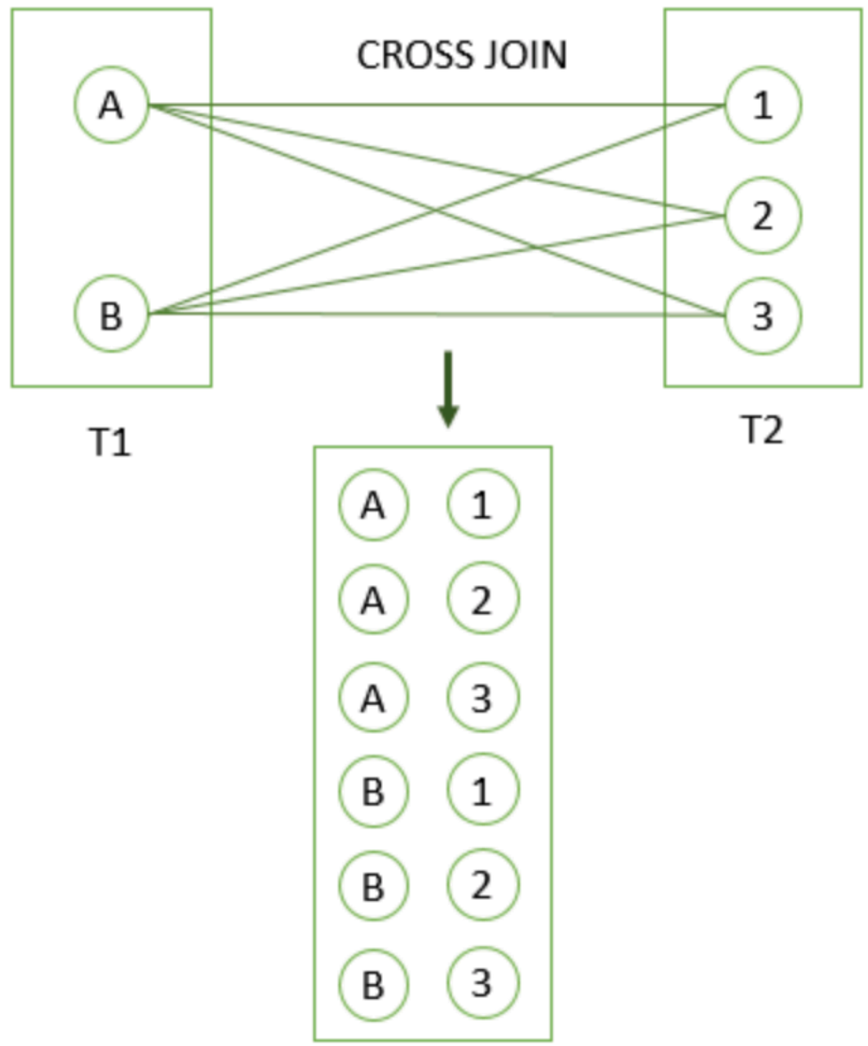
full

```
SELECT columns
FROM table1
FULL JOIN table2
ON table1.column = table2.column;
```

Joins

Producto cartesiano entre dos tablas. No requiere condición alguna.

cross



SELECT columns

FROM table1,

CROSS JOIN table2;

Joins

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;

Joins

Sintaxis anterior

SELECT columns

FROM table1, table2

WHERE table1.column = table2.column;

SELECT columns

FROM table1, table2 ;

Operadores

+

-

*

/

%

aritméticos

Operadores

AND

OR

NOT

lógicos

Operadores

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.

Funciones de agregación

Funciones especiales en las que la información de varios registros es agrupada para generar un único valor resultante.

- MIN
- MAX
- AVG
- SUM
- COUNT

Funciones de agregación

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.

Funciones de agregación

HAVING: Cláusula que permite filtrar grupos de observaciones que no cumplan una condición dada.

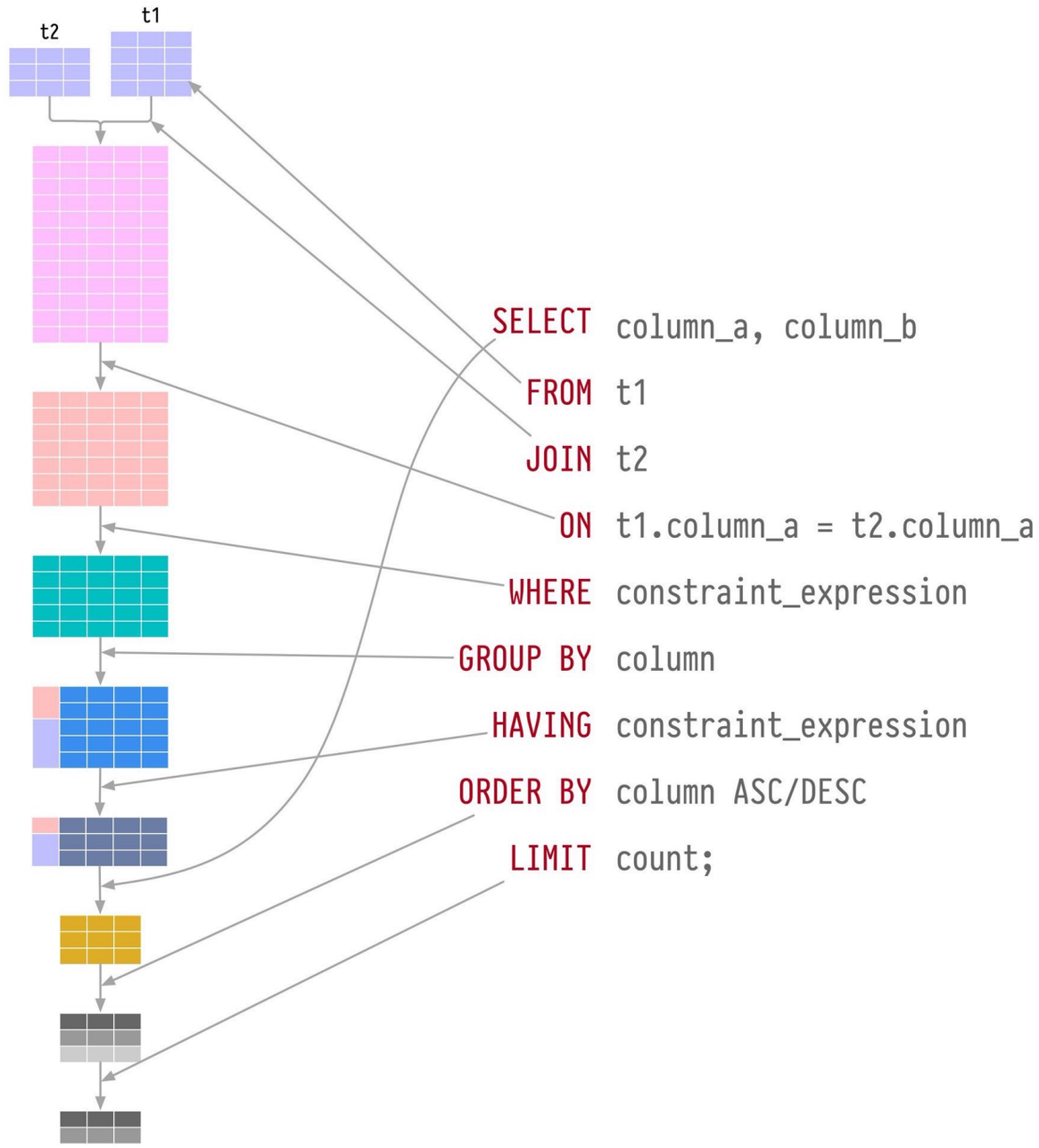
Se emplea después de la cláusula GROUP BY.

Funciones de agregación

Having vs. Where

La diferencia radica en que having aplica a grupos de registros, mientras where aplica sobre registros individuales.

Resumen


t2
t1
SELECT column_a, column_b
FROM t1
JOIN t2
ON t1.column_a = t2.column_a
WHERE constraint_expression
GROUP BY column
HAVING constraint_expression
ORDER BY column ASC/DESC
LIMIT count;

Tarea 14

Subconsultas

Investigar:

- definición
- ¿dónde puedo usarlas y bajo qué condiciones?
- subconsultas correlacionadas
- ejemplos

Subconsultas

Es una sentencia SELECT que aparece en la definición de otra sentencia SELECT, a la que denominamos consulta principal.

Subconsultas

En la cláusula SELECT:

Generalmente es para obtener la información agregada de algún dato dentro de una tabla

Restricción: La subconsulta no puede regresar más de un valor y un atributo

Subconsultas

En la cláusula FROM:

Permite obtener una serie de filas y columnas, basadas en el resultado de la subconsulta, que podemos tratar como una tabla.

Subconsultas

En un join:

Permite obtener una serie de filas y columnas, basadas en el resultado de la subconsulta, que podemos tratar como una tabla para emplearla como operando según sea el caso.

Subconsultas

En la cláusula WHERE/HAVING:

La subconsulta fungirá como operando para la condición. Puede aplicarse de dos formas:

- Directamente
- Dentro de algún operador lógico (IN, ANY, ALL, EXISTS)

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

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

CONDICION 18 > ALL(5,7,11,17,14,21)

FALSE

lógicos

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

lógicos

CONDICION 18 > ANY(19,33,23,21,17)

TRUE

lógicos

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

lógicos

CONDICION EXIST (select nombre from alumno where nombre_Alumno = ‘Juan’)

Subconsultas

Correlacionada:

La subconsulta hace referencia a valores de la consulta principal, por lo que el resultado de la consulta principal es un operando de la subconsulta

Subconsultas

```
SELECT nombre, direccion, email, salario
FROM empleado AS sq_principal
WHERE salario >
(SELECT AVG(salario)
FROM EMPLEADO AS sq_secundaria
WHERE depto_id = sq_principal.depto_id);
```

Subconsultas

Buena práctica:

Uso de cláusula WITH. Resultados temporales.

Subconsultas

```
WITH sq_1 AS (…),
[sq_n AS (…)]
SELECT …
FROM …
…;
```