Bases de datos relacionales

Consultando información relacionada en varias tablas

{desafío} latam_

Identificar las características, rol y elementos fundamentales de una base de datos relacional para la gestión de la información en una organización y utilizar el lenguaje estructurado de consultas SQL para la obtención de información que satisface los requerimientos planteados a partir de un modelo de datos dado

- Unidad 1:
  Bases de datos relacionales
- Unidad 2:
  Manipulación de datos y transaccionalidad en las operaciones
- Unidad 3:
  Definición de tablas
- Unidad 4:
  Modelos Entidad-Relación y Relacional

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Utiliza sentencias SQL que requieren la consulta a varias tablas relacionadas a partir de un modelo de datos dado para resolver un problema planteado de selección.*

/* Qué es un modelo de datos y cómo leerlo */

Modelos de datos

Introducción a modelos de datos

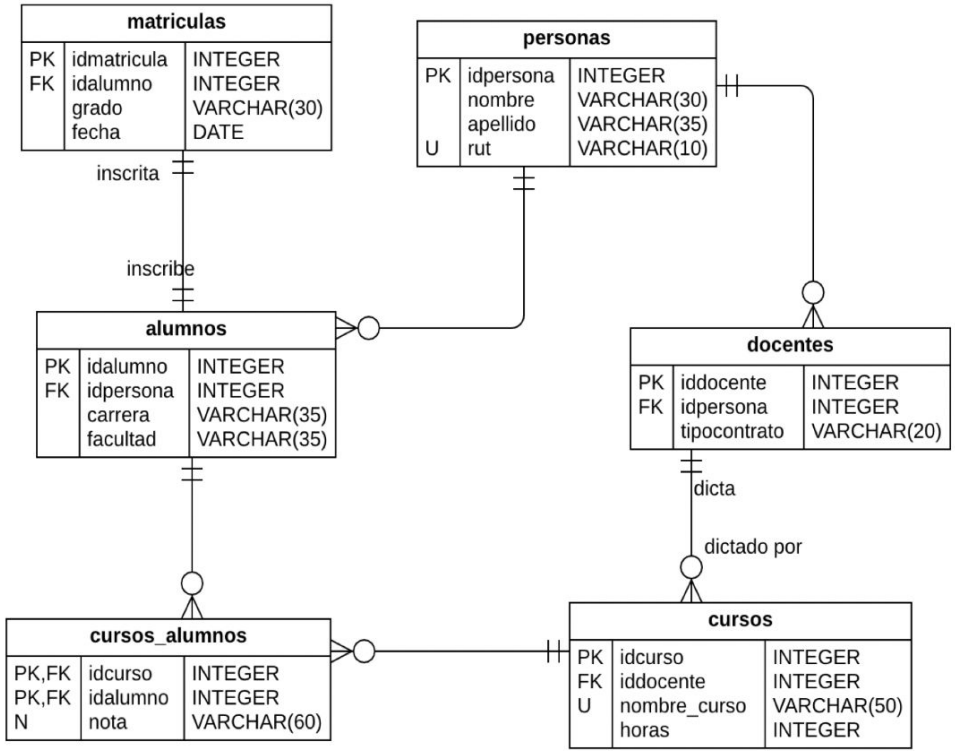
- Un modelo de datos se refiere a una representación estructurada y organizada de los datos que se almacenan en una base de datos relacional.
- El modelo de datos define la estructura lógica de la base de datos, incluyendo la forma en que se organizan los datos.
- Los modelos de datos en SQL suelen estar compuestos por **Tablas**:
  - Son la estructura básica en un modelo de datos relacional.
  - Representan entidades específicas o tipos de datos y están organizadas en filas y columnas.
  - Cada fila de una tabla representa una instancia individual de la entidad, y cada columna representa un atributo o una característica de esa entidad.

En próximas sesiones revisaremos en detalle otros aspectos de los modelos de datos.

Modelos de datos

Modelo físico

- El modelo de datos físicos representa cómo se construirá el modelo de la base de datos.
- Se especifican las tablas, columnas, claves primarias y foráneas así como otras restricciones.



{desafío} latam_

/* Queries anidadas o subconsultas */

¿Qué son las subconsultas?

Una subconsulta consiste en hacer una consulta interior dentro de una consulta exterior. Es por esto mismo que a veces reciben el nombre de consultas anidadas.

Las subconsultas nos permiten seleccionar registros en función a los resultados de otra consulta.

¿Qué son las subconsultas?

Veamos un ejemplo donde son útiles

NOMBRE	APELLIDO	SUELDO	DEPARTAMENTO
Juan	Pérez	3000	Ventas
María	González	3500	Marketing
Carlos	Rodríguez	4000	Tecnología
Ana	Martínez	2800	Recursos Humanos
Luis	García	3200	Finanzas

Nos piden seleccionar a todas las personas que ganan sobre el promedio.

Para esto tenemos que seleccionar el promedio y luego seleccionar a todas las personas que ganen sobre esto.

Subconsultas en el from

Partimos resolviendo el problema desde dentro.

1. Calculamos el promedio

```
SELECT avg(sueldo) FROM empleados
```

2. Seleccionamos todos los empleados cuyo sueldo es mayor a la consulta anterior.

```
SELECT *

FROM empleados

WHERE sueldo >

(SELECT AVG(sueldo) FROM empleados);
```

Consulta exterior vs consulta interior

```
SELECT *
FROM empleados
WHERE sueldo >
(SELECT AVG(sueldo) FROM empleados);
```

consulta exterior

consulta interior

Ejercicio guiado

"Aplicando subconsultas en dos tablas "

{desafío} latam_

Ejercicio guiado

Tabla Clientes

CLIENTE_ID	NOMBRE
1	Juan
2	María
3	Carlos
4	Ana
5	Luis

Tabla Pedidos

ID	MONTO	CLIENTE_ID
10	3000	1
20	800	1
30	1500	2
40	2800	3
50	3200	5

Tenemos dos tablas: una llamada clientes que contiene información sobre los **clientes**, y otra llamada **pedidos** que contiene información sobre los pedidos realizados por esos clientes. Queremos obtener una lista de clientes que hayan realizado al menos un pedido con un monto total superior a 1000. La subconsulta se utilizará en la cláusula WHERE para filtrar los clientes que cumplen con este criterio.

Ejercicio guiado

Primero realizamos la subconsulta

En la tabla de pedidos existe la columna cliente_id que indica que cliente está asociado a cada producto. Partiremos seleccionando los cliente_id que estén asociados a pedidos con un monto superior a 1000:

```
SELECT cliente_id FROM pedidos WHERE monto > 1000
```

Ejercicio guiado

Luego, realizamos la consulta exterior

La consulta principal seleccionará todos los campos de la tabla **`Clientes`** donde el `cliente_id` está presente en la lista de resultados de la subconsulta, lo que significa que estos clientes han realizado al menos un pedido con un monto total superior a 1000:

```
SELECT *
FROM clientes
WHERE cliente_id IN (SELECT cliente_id
                     FROM pedidos
                     WHERE monto > 1000);
```

La cláusula **`IN`** en SQL se utiliza para filtrar los resultados de una consulta basándose en si los valores de una columna están presentes en el conjunto de resultados de otra consulta

Subconsultas en el FROM

Así como las subconsultas pueden estar en el where también pueden ser parte de otras cláusulas como FROM.

Utilizarlas nos abre un abanico de oportunidades para resolver otros tipos de problemas.

Subconsultas en el FROM

EMPLEADO_ID	MONTO
1	100
1	150
2	200
2	250
3	300
3	350
4	400

Nos piden calcular el monto promedio vendido por vendedor. Sin embargo en esta tabla un vendedor puede tener registrada varias ventas por separado.

Para resolverlo necesitamos sumar los montos por vendedor y luego sobre estos resultados sacamos el promedio de las ventas.

Subconsultas en el FROM

EMPLEADO_ID	MONTO
1	100
1	150
2	200
2	250
3	300
3	350
4	400

```
SELECT AVG(total_venta) as promedio_ventas
FROM (
  SELECT empleado_id, SUM(monto) as total_venta
  FROM ventas
  GROUP BY empleado_id
)
```

La consulta interior nos permite obtener el total de ventas por empleado. Esta consulta interior la consideraremos como una nueva tabla con las columnas empleado_id y total_venta. Luego, aplicaremos la consulta exterior en la cual calcularemos el promedio de total_venta de nuestra consulta interna.

Utilizando lo aprendido:

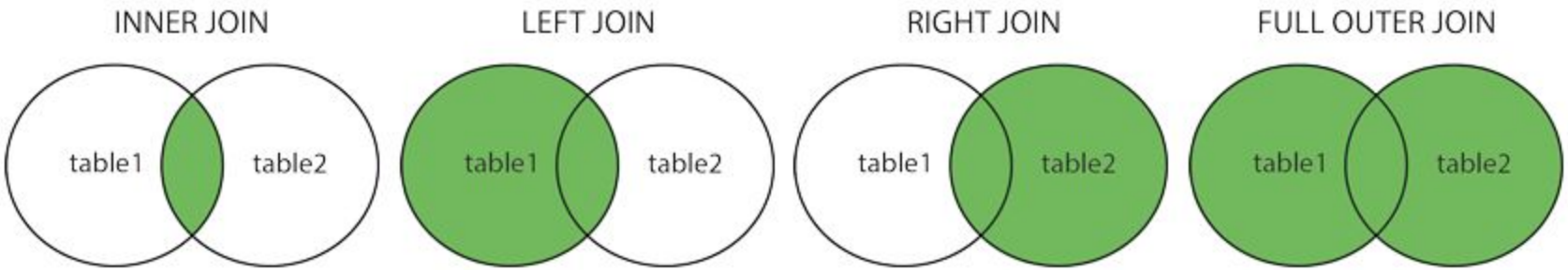
Tienes una tabla llamada productos que registra información sobre los productos disponibles, y otra tabla llamada pedidos_detalle que contiene detalles específicos de los productos incluidos en cada pedido. Encuentra los productos que se han pedido más de una vez.

{desafío} latam_

Operaciones de unión entre tablas

- **Inner Join:** Se usa para unificar filas de dos o más tablas siempre que exista una columna que las relacione.
- **Left outer Join:** Entrega todos los registros de la tabla de la izquierda y las coincidencias de la tabla derecha.
- **Right outer Join:** Muestra todos los registros de la tabla de la derecha y las coincidencias de la tabla izquierda.
- **Full Outer Join:** Retorna todos los registros que coincidan en la tabla derecha e izquierda.

Ejemplificación gráfica



En la guía de ejercicios aplicaremos estos conceptos.

Ejercicio guiado

"Utilizando los tipos de join "

{desafío}
latam_

Ejercitando el uso de tipos de JOIN

Para visualizar la diferencia de resultados al utilizar los distintos tipos de join, realizaremos algunos ejercicios utilizando las siguientes tablas

Tabla Clientes

CLIENTE_ID	NOMBRE
1	Juan
2	María
3	Carlos
4	Ana

Tabla Productos

ID	MONTO	CLIENTE_ID
10	3000	1
20	800	1
30	1500	2
40	2800	3
50	3200	3
60	400	1

Ejercitando el uso de INNER JOIN

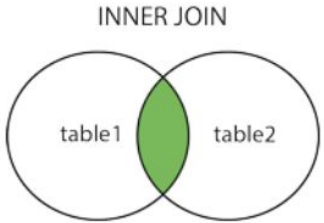
Tabla Clientes

CLIENTE_ID	NOMBRE
1	Juan
2	María
3	Carlos
4	Ana

Tabla Productos

ID	MONTO	CLIENTE_ID
10	3000	1
20	800	1
30	1500	2
40	2800	3
50	3200	3
60	400	1

Realizaremos una consulta SQL utilizando INNER JOIN para obtener una lista que nos muestre el `CLIENTE_ID` y el `NOMBRE` de los clientes junto con el `ID` y `MONTO` de los productos que han comprado. Asegúrate de incluir **solo** las filas donde haya coincidencia en la columna `CLIENTE_ID` entre ambas tablas.



{desafío} latam_

Ejercitando el uso de INNER JOIN

```
SELECT clientes.cliente_id, clientes.nombre, productos.id, productos.monto
FROM clientes
INNER JOIN productos ON clientes.cliente_id = productos.cliente_id;
```

Esta consulta seleccionará las columnas `CLIENTE_ID` y `NOMBRE` de la tabla `clientes`, junto con las columnas `ID` y `MONTO` de la tabla `productos`. La condición **`ON clientes.CLIENTE_ID = productos.CLIENTE_ID`** especifica que las filas deben coincidir en la columna `CLIENTE_ID` para que sean incluidas en el resultado.

El resultado mostrará sólo las combinaciones de clientes y productos donde el `CLIENTE_ID` coincide en ambas tablas, por lo tanto, la fila correspondiente a Ana de clientes no será mostrada

CLIENTE_ID	NOMBRE	ID	MONTO
1	Juan	10	3000
1	Juan	20	800
2	María	30	1500
3	Carlos	40	2800
3	Carlos	50	3200
1	Juan	60	400

Ejercitando el uso de LEFT JOIN

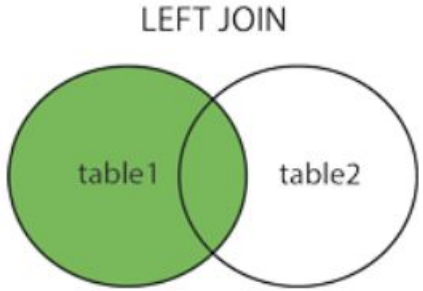
Tabla Clientes

CLIENTE_ID	NOMBRE
1	Juan
2	María
3	Carlos
4	Ana

Tabla Productos

ID	MONTO	CLIENTE_ID
10	3000	1
20	800	1
30	1500	2
40	2800	3
50	3200	3
60	400	1

Ahora, utilizando las mismas tablas, realizaremos una consulta SQL utilizando LEFT JOIN para obtener una lista que muestre todos los nombres de los clientes junto con el `ID` y `MONTO` de los productos que han comprado (si es que han comprado alguno)



Ejercitando el uso de LEFT JOIN

```
SELECT clientes.nombre, productos.id, productos.monto
FROM clientes
LEFT JOIN productos ON clientes.cliente_id = productos.cliente_id;
```

Esta consulta seleccionará la columna `NOMBRE` de la tabla `clientes`, junto con las columnas `ID` y `MONTO` de la tabla `productos`. La condición **`ON clientes.CLIENTE_ID = productos.CLIENTE_ID`** especifica que las filas deben coincidir en la columna `CLIENTE_ID` para que sean incluidas en el resultado.

En la tabla resultante, puedes observar que se incluyen todas las filas de la tabla `clientes`, incluso aquellos clientes que no tienen ninguna compra asociada en la tabla `productos`. Para esos clientes, los campos relacionados con productos (`ID` y `MONTO`) muestran valores nulos.

NOMBRE	ID	MONTO
Juan	10	3000
Juan	20	800
Juan	60	400
Maria	30	1500
Carlos	40	2800
Carlos	50	3200
Ana	NULL	NULL

Ejercitando el uso de RIGHT JOIN

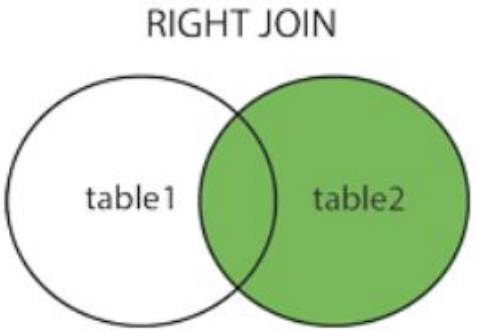
Tabla Clientes

CLIENTE_ID	NOMBRE
1	Juan
2	María
3	Carlos
4	Ana

Tabla Productos

ID	MONTO	CLIENTE_ID
10	3000	1
20	800	1
30	1500	2
40	2800	3
50	3200	3
60	400	1

Realiza una consulta SQL utilizando RIGHT JOIN para obtener una lista que muestre todos los productos, junto con la información del cliente que realizó cada compra (si hay alguna). Incluye el `NOMBRE` de los clientes, así como el `ID` y `MONTO del producto`



{desafío} latam_

Ejercitando el uso de RIGHT JOIN

```
SELECT  clientes.NOMBRE, productos.ID, productos.MONTO
FROM clientes
RIGHT JOIN productos ON clientes.CLIENTE_ID = productos.CLIENTE_ID;
```

En este caso, la tabla resultante muestra todas las filas de la tabla `productos`, y solo las filas de la tabla `clientes` que tienen coincidencias en la columna `CLIENTE_ID`.

Aquí, cada fila representa un producto, y si el producto está asociado a un cliente, se muestran los detalles del cliente. Si un producto no está asociado a ningún cliente, los campos de cliente serán nulos en esa fila. En este caso en particular no hay productos con campos de cliente nulo, ya que todos los productos están asociados a un cliente.

NOMBRE	ID	MONTO
Juan	10	3000
Juan	20	800
Maria	30	1500
Carlos	40	2800
Carlos	50	3200
Juan	60	400

Ejercitando el uso de FULL OUTER JOIN

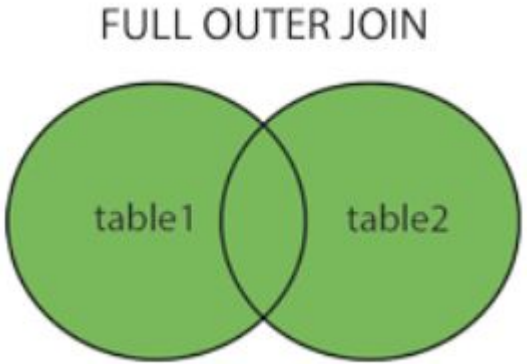
Tabla Clientes

CLIENTE_ID	NOMBRE
1	Juan
2	María
3	Carlos
4	Ana

Tabla Productos

ID	MONTO	CLIENTE_ID
10	3000	1
20	800	1
30	1500	2
40	2800	3
50	3200	3
60	400	1

Realizaremos una consulta SQL utilizando FULL OUTER JOIN para obtener una lista que muestre todos los nombres de los clientes y todos los id y montos de los productos



Ejercitando el uso de FULL OUTER JOIN

```
SELECT clientes.CLIENTE_ID, clientes.NOMBRE, productos.ID, productos.MONTO
FROM clientes
FULL OUTER JOIN productos ON clientes.CLIENTE_ID = productos.CLIENTE_ID;
```

En este caso, la tabla resultante muestra todas las filas de ambas tablas (clientes y productos). Los campos de cliente y producto serán nulos si no hay coincidencia en la columna CLIENTE_ID.

Es importante señalar que no todos los sistemas de bases de datos admiten la sintaxis FULL OUTER JOIN, en cuyo caso puedes utilizar LEFT JOIN y RIGHT JOIN combinados para lograr un resultado similar.

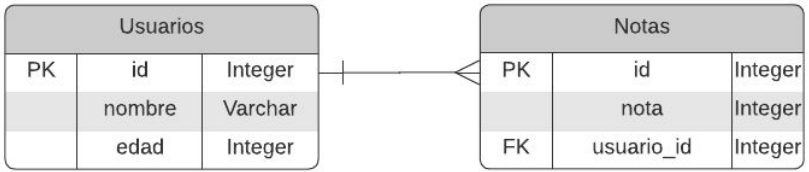
NOMBRE	ID	MONTO
Juan	10	3000
Juan	20	800
Maria	30	1500
Carlos	40	2800
Carlos	50	3200
Juan	60	400
Ana	null	null

¿Comprendieron la diferencia entre los tipos de joins?

{desafío}
latam_

Lectura de datos

Cuando nos encontremos en el ámbito laboral enfrentando problemas que demanden el análisis de datos, similares a los que abordamos con los tipos de join, estos datos se presentarán de la siguiente manera



En esta unidad, aunque no profundizaremos en la explicación detallada de las flechas o en los conceptos específicos de claves primarias (PK) y claves foráneas (FK), es esencial que adquieran la habilidad de analizar la presencia de un indicador común en ambas tablas, como el campo "id". Este indicador común actúa como punto de conexión que facilita el cruce de datos entre las tablas. Si no existe un id que permita el cruce, no será posible realizar joins como los vistos previamente.

Utilizando lo aprendido:

En una empresa, hay dos tablas relevantes en la base de datos: empleados y departamentos. Ambas tablas contienen la columna empleado_id.

Crea una consulta SQL para obtener una lista que muestre todos los empleados junto con sus respectivos departamentos .

¿Qué tipo de JOIN debes utilizar?

{desafío} latam_

Próxima sesión...

- *Guía de ejercicios*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales