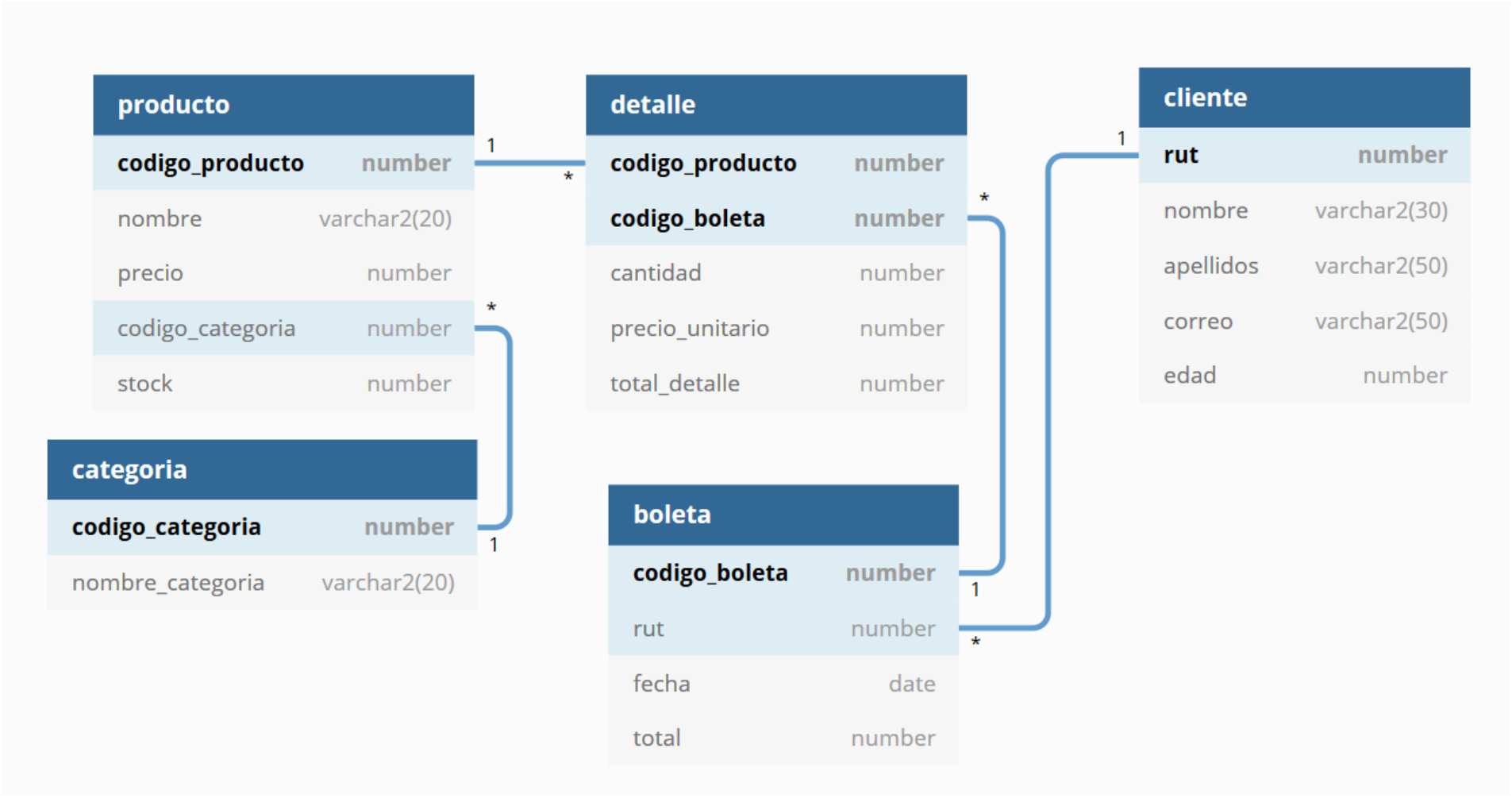
UNIVERSIDAD CATOLICA DEL MAULE

Consultas de agrupación y join

Programa de hoy

- Presentación del nuevo modelo
- Funciones de fechas
- Consultas de múltiples tablas (join)
- Consultas de agrupación

Modelo de trabajo de hoy



Funciones de fechas

Fechas	
SELECT SYSDATE FROM DUAL;	Muestra la fecha del sistema
SELECT EXTRACT(DAY FROM SYSDATE) FROM DUAL;	La función extract permite extraer elementos de una fecha, en este caso, el día.
SELECT TO_CHAR(SYSDATE,'DAY') FROM DUAL;	La función permite pasar a texto el "día" de la fecha actual.
SELECT EXTRACT(MONTH FROM SYSDATE) FROM DUAL;	La función extract permite extraer el mes de la fecha.
SELECT TO_CHAR(SYSDATE,'MONTH') FROM DUAL;	La función permite pasar a texto el "mes" de la fecha actual.
SELECT EXTRACT(YEAR FROM SYSDATE) FROM DUAL;	La función extract permite extraer el año de la fecha.
SELECT TO_CHAR(SYSDATE,'YEAR') FROM DUAL;	La función permite pasar a texto el "año" de la fecha actual.
SELECT ADD_MONTHS(SYSDATE, 2) FROM DUAL;	La función add_months, permite agregar meses a una fecha.
SELECT LAST_DAY(SYSDATE) FROM DUAL;	La función last_day, entrega la fecha del último día del mes actual.
SELECT NEXT_DAY(SYSDATE,'LUNES') FROM DUAL;	La función next_day, entrega la fecha del próximo día que queramos consultar.

Es posible realizar una operación de suma o resta sobre una fecha. Por ejemplo, sumarle a SYSDATE + 2. Esto quiere decir, que a la fecha actual, se le sumarán 2 días. Oracle interpreta las constantes numéricas como el **número de días**.

Sentencia SELECT

La sentencia SELECT permite consultar la información de una o más tablas pertenecientes a la base de datos.

Es la sentencia que posee la mayor cantidad de operadores, funciones e instrucciones para realizar las consultas.

Hoy abordaremos las consultas a múltiples tablas y consultas de agrupación.

Consulta a múltiples tablas

Las consultas que se realizan sobre 2 o más tablas son conocidas como consultas tipo “JOIN”.

Consulta a múltiples tablas

Las consultas que se realizan sobre 2 o más tablas son conocidas como consultas tipo “JOIN”.

El objetivo es crear un camino que relacione 2 o más tablas para mostrar información.

Consulta a múltiples tablas

Las consultas que se realizan sobre 2 o más tablas son conocidas como consultas tipo “JOIN”.

El objetivo es crear un camino que relacione 2 o más tablas para mostrar información.

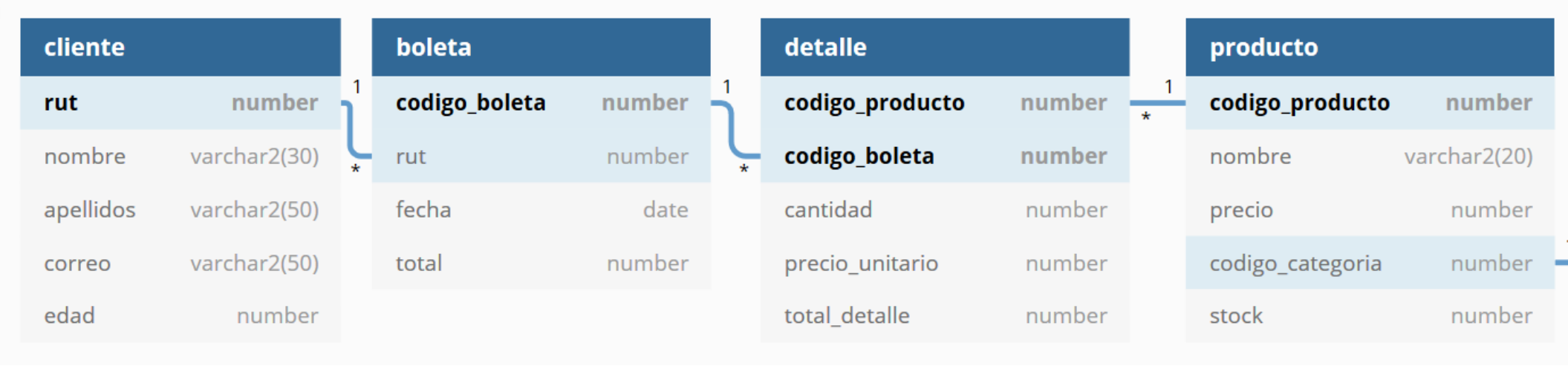
Aquí es cuando las claves foráneas tienen otro papel fundamental sobre una base de datos.

Explicación de JOIN

Repasemos el modelo inicial de esta presentación sin considerar la tabla categoría:

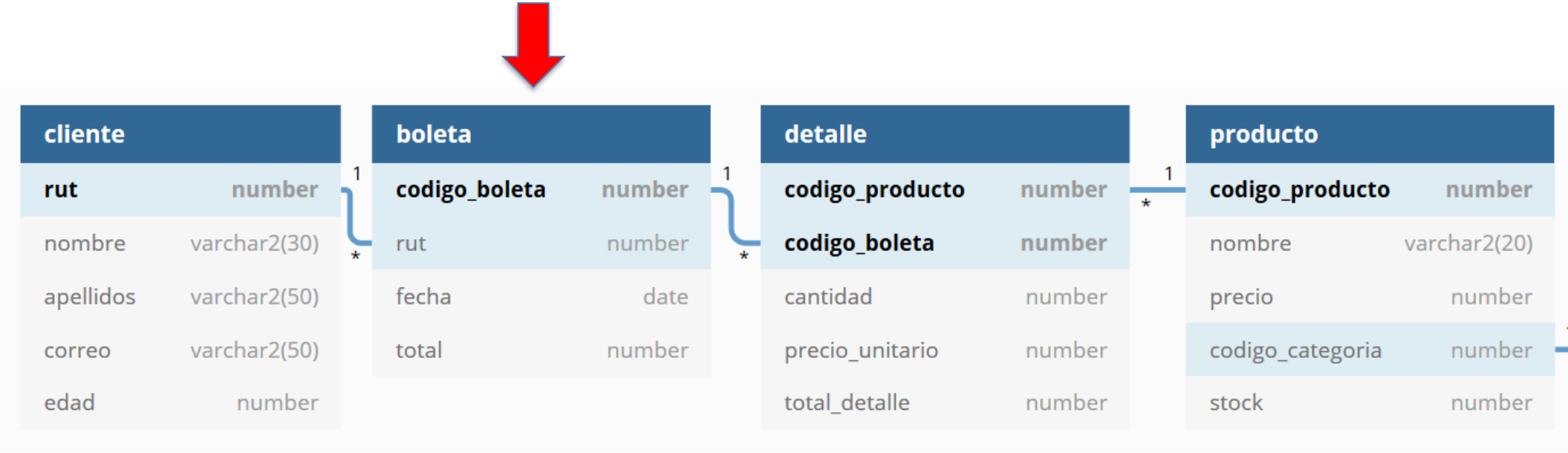
Explicación de JOIN

Repasemos el modelo inicial de esta presentación sin considerar la tabla categoría:



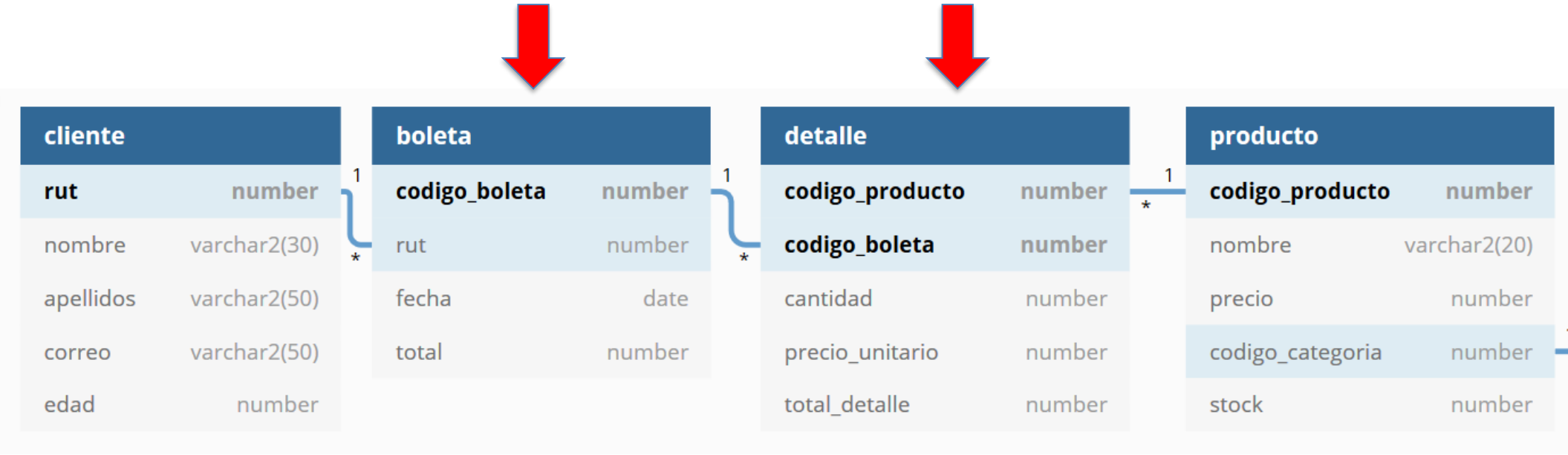
Explicación de JOIN

Cuando un cliente realiza una compra, esta se asocia a un código de boleta.



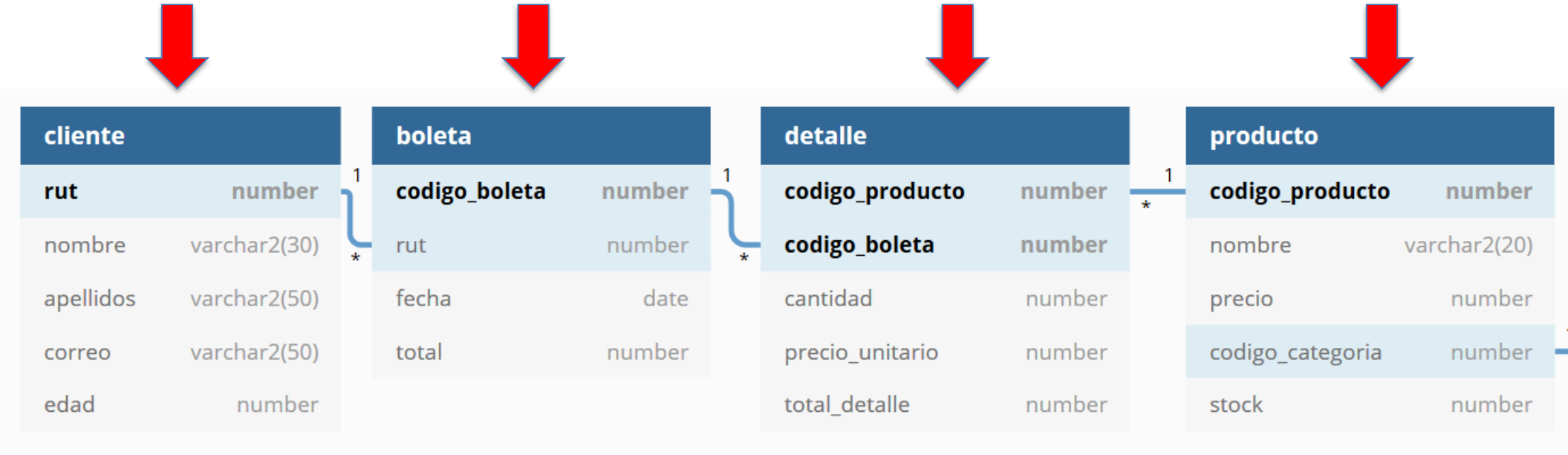
Explicación de JOIN

Cuando un cliente realiza una compra, esta se asocia a un código de boleta. Que a su vez, se asocia con un código de producto.



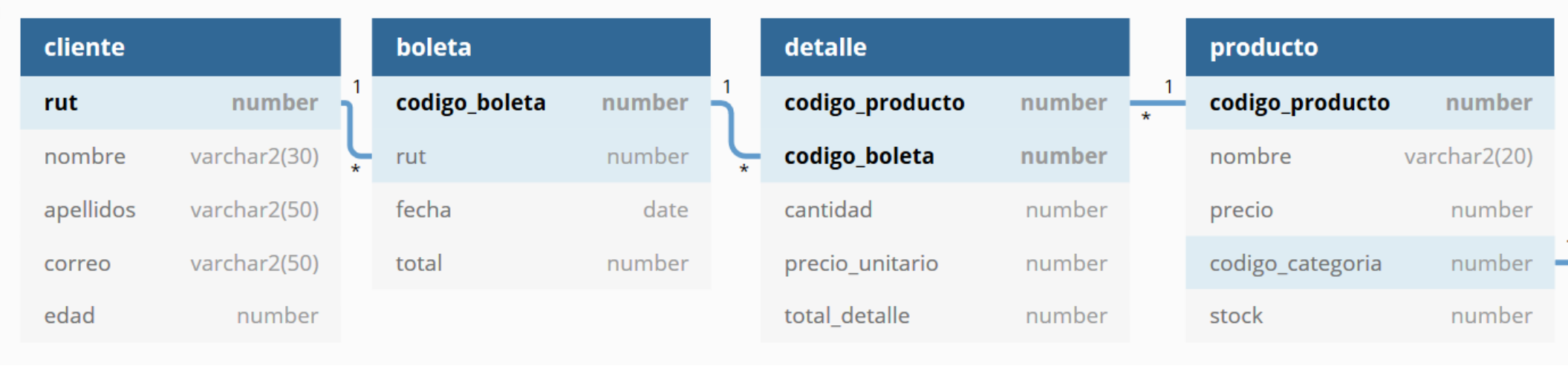
Explicación de JOIN

Todo este proceso permite unir al cliente con un producto que desea comprar.



Explicación de JOIN

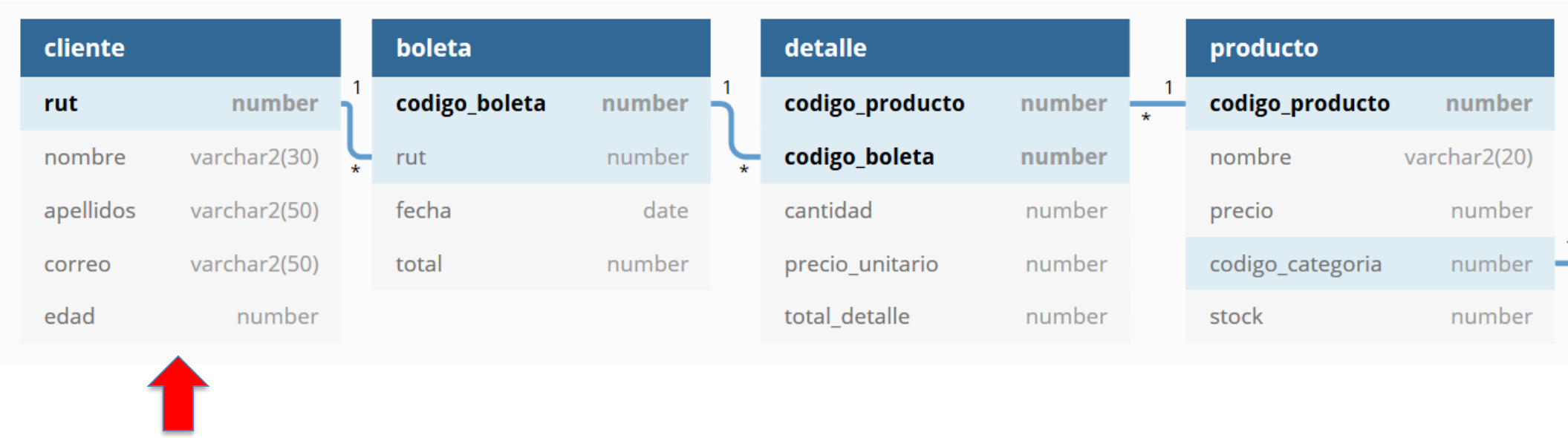
Para consultar los productos que compró un cliente, debemos comenzar uniendo tabla por tabla. No importa por cual tabla se comienza la unión. Si el JOIN se hizo correctamente, no debería existir error.



Explicación de JOIN

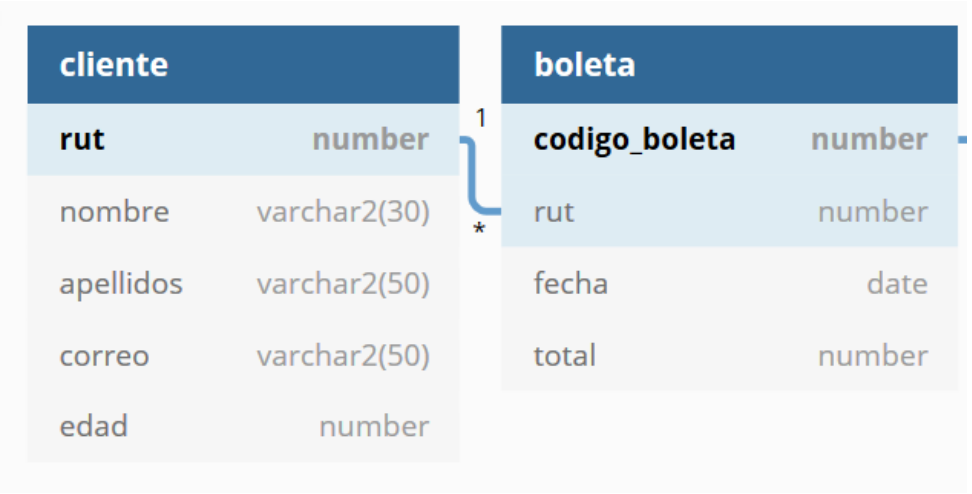
Para consultar los productos que compró un cliente, debemos comenzar uniendo tabla por tabla. No importa por cual tabla se comienza la unión. Si el JOIN se hizo correctamente, no debería existir error.

Para el ejemplo comenzaremos por la tabla cliente.



Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

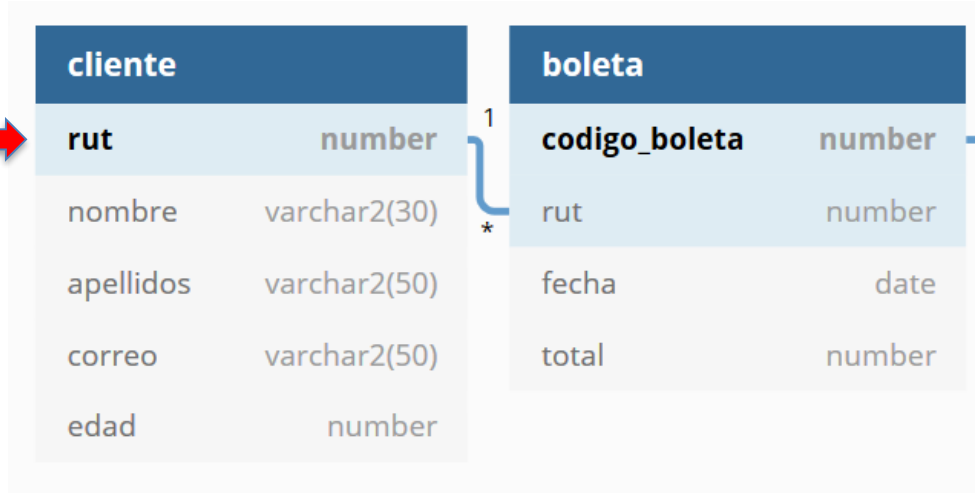


PK: Primary Key (Clave primaria) - FK: Foreign Key (Clave foránea)

Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

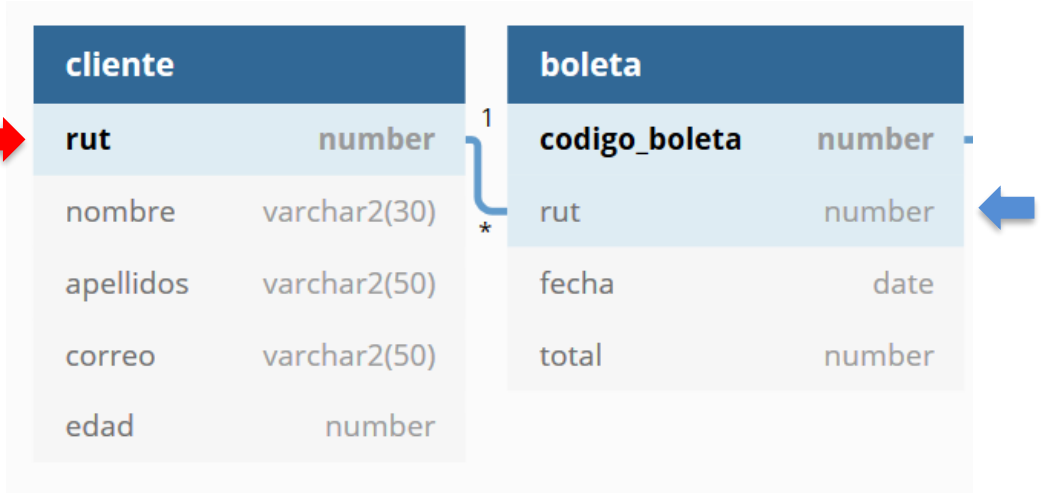
Debemos unir el rut de la tabla cliente



Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

Debemos unir el rut de la tabla cliente, con el rut de la tabla boleta.

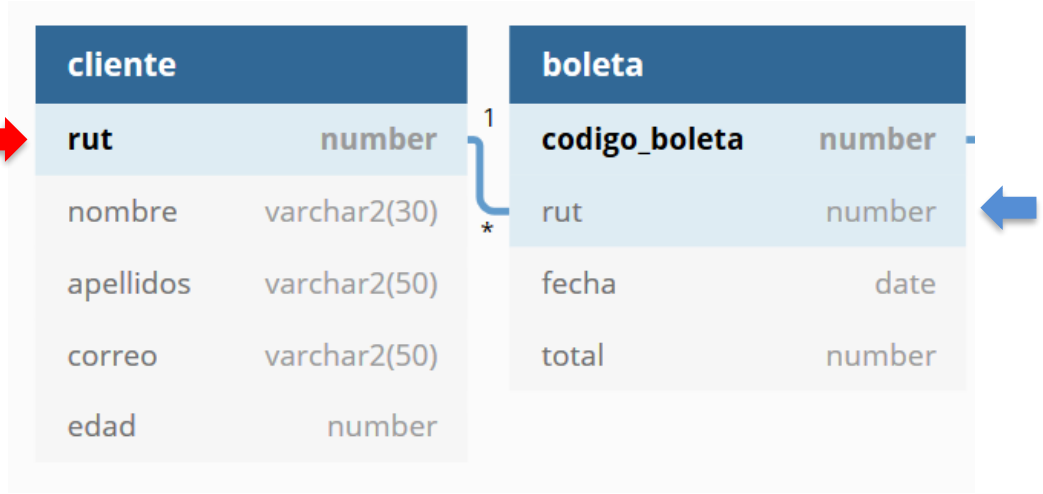


Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

Debemos unir el rut de la tabla cliente, con el rut de la tabla boleta.

Para esto, utilizamos un select con la instrucción JOIN.

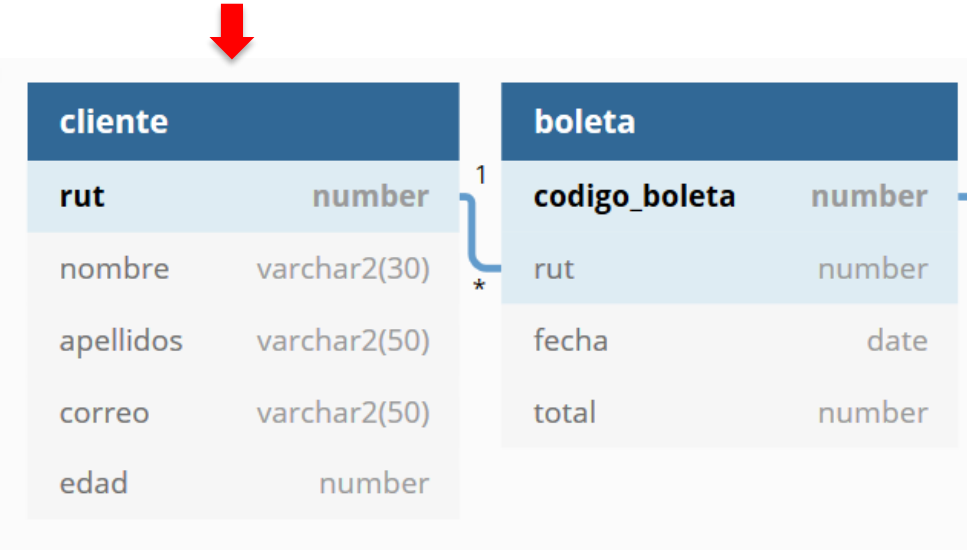


Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

Debemos unir el rut de la tabla cliente, con el rut de la tabla boleta.

Para esto, utilizamos un select con la instrucción JOIN.



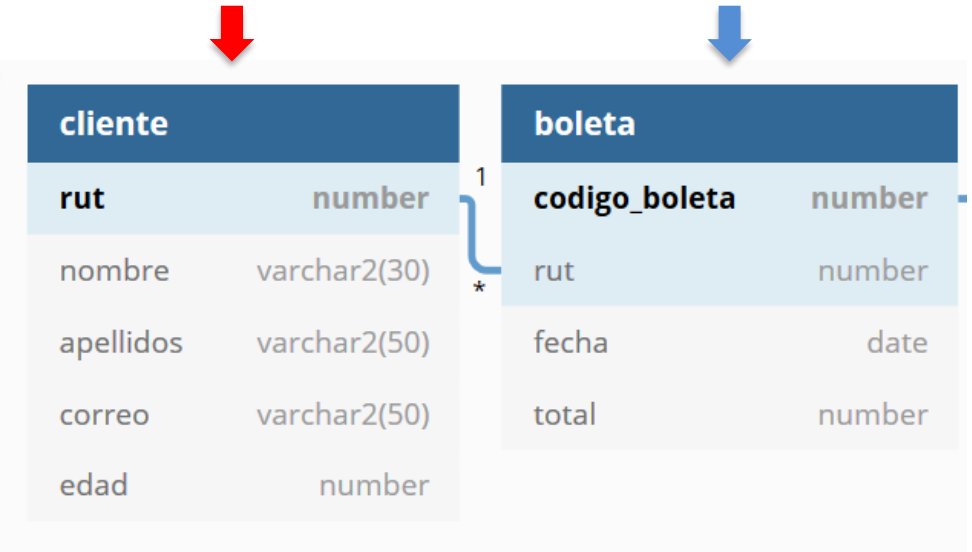

```
SELECT *
FROM CLIENTE
```

Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

Debemos unir el rut de la tabla cliente, con el rut de la tabla boleta.

Para esto, utilizamos un select con la instrucción JOIN.




```
SELECT *
FROM CLIENTE
JOIN BOLETA
```

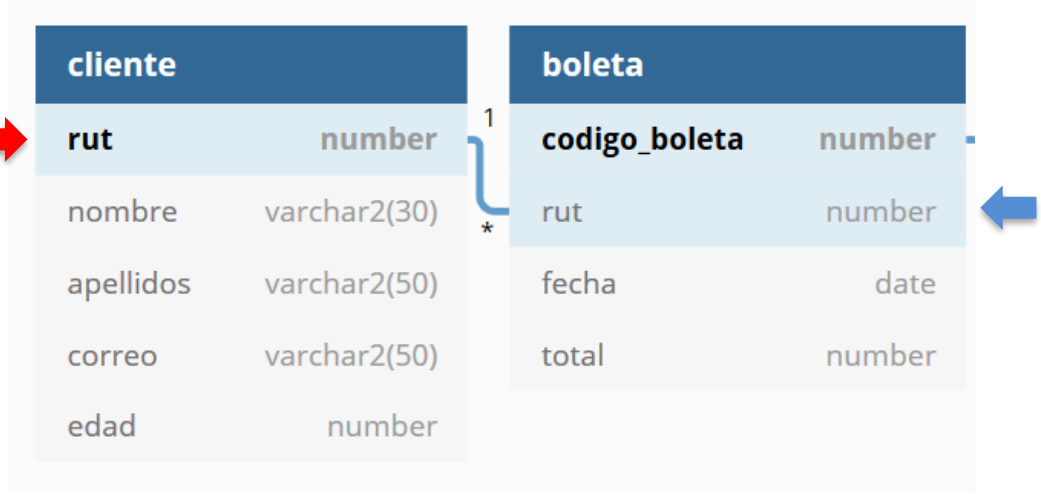
Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

Debemos unir el rut de la tabla cliente, con el rut de la tabla boleta.

Para esto, utilizamos un select con la instrucción JOIN.

La instrucción ON le dice a la base de datos las columnas que se conectarán.




```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
```

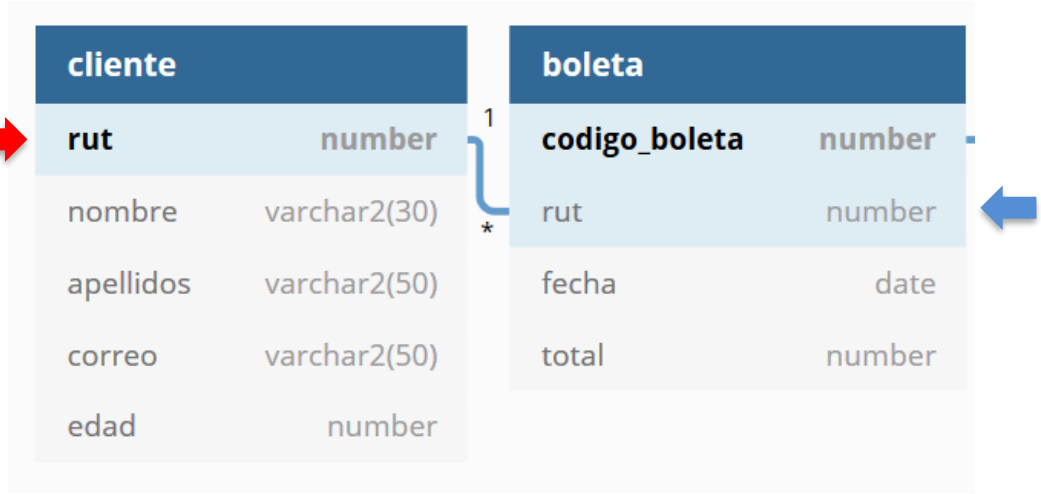
Explicación de JOIN

La conexión entre ambas tablas de hace por medio de la relación entre PK y FK.

Debemos unir el rut de la tabla cliente, con el rut de la tabla boleta.

Para esto, utilizamos un select con la instrucción JOIN.

La instrucción ON le dice a la base de datos las columnas que se conectarán.




```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
```

Esta instrucción, muestra todos los datos entre cliente y boleta donde ambos rut coinciden.

Resultado del primer JOIN

Tomemos la siguiente instrucción y veamos qué sucede con el siguiente conjunto de datos para un rut en particular.

```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
```

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Resultado del primer JOIN

Tomemos la siguiente instrucción y veamos qué sucede con el siguiente conjunto de datos para un rut en particular. El rut: 204821232

```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
WHERE CLIENTE.RUT = 204821232
```

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Resultado del primer JOIN

Primero ejecutamos el select habitual que ya conocemos.

```
SELECT *
FROM CLIENTE
```

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Resultado del primer JOIN

Luego la unión con la tabla boleta indicando las columnas que permitirán la conexión. Los datos que se muestran corresponden a los que existen en ambas tablas.

```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
```

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Resultado del primer JOIN

Luego solo indicamos los registros asociados al rut 204821232.

```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
WHERE CLIENTE.RUT = 204821232
```

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Resultado del primer JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	RUT_1	FECHA	TOTAL
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	204821232	12-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	204821232	16-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	204821232	11-10-2019	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	204821232	12-12-2019	130000

```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
WHERE CLIENTE.RUT = 204821232
```

Resultado del primer JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	RUT_1	FECHA	TOTAL
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	204821232	12-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	204821232	16-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	204821232	11-10-2019	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	204821232	12-12-2019	130000

```
SELECT *
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
WHERE CLIENTE.RUT = 204821232
```

Podemos notar que el campo rut se repite, ya que saca el rut de la tabla cliente y el rut de la tabla boleta. Podemos modificar la consulta indicando los campos que deseamos mostrar.

Resultado del primer JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	FECHA	TOTAL
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	12-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	16-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	11-10-2019	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	12-12-2019	130000

```
SELECT CLIENTE.RUT, CLIENTE.NOMBRE, CLIENTE.APELLIDOS,
CLIENTE.CORREO, CLIENTE.EDAD, BOLETA.CODIGO_BOLETA,
BOLETA.FECHA, BOLETA.TOTAL
FROM CLIENTE
JOIN BOLETA
ON CLIENTE.RUT = BOLETA.RUT
WHERE CLIENTE.RUT = 204821232
```

También podemos notar que los atributos correspondientes a la tabla “cliente” y “boleta” deben llevar antes de su nombre, el nombre de la tabla seguido por un punto. Esto siempre es necesario en los JOIN’s, para distinguir los atributos de cada tabla.

Lo bueno, es que podemos asignarle a cada tabla un “alias” para ahorrar espacio.

Resultado del primer JOIN

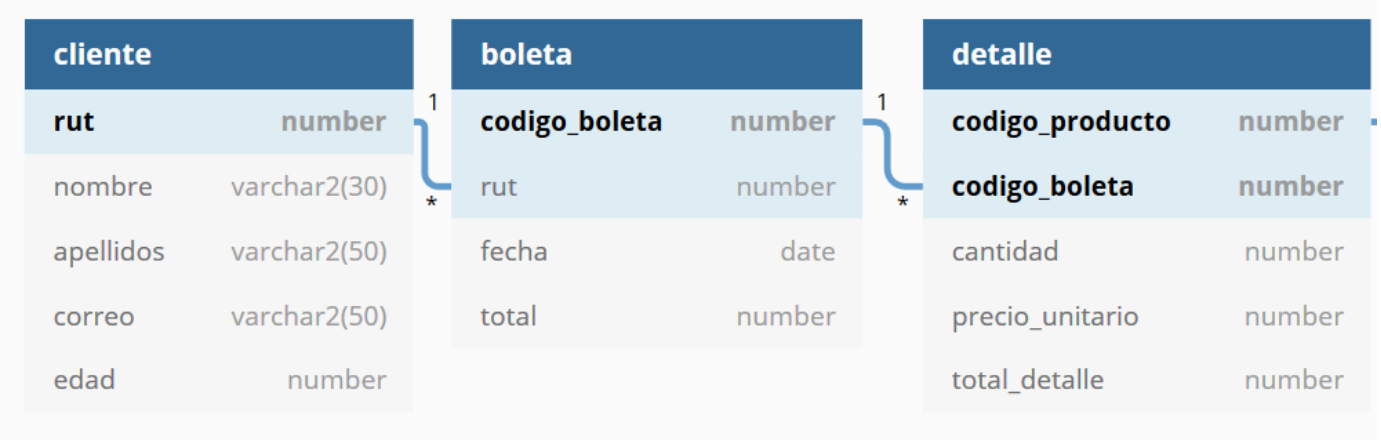
RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	FECHA	TOTAL
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	12-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	16-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	11-10-2019	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	12-12-2019	130000

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
WHERE C.RUT = 204821232
```

Lo que acaban de visualizar, es el proceso completo de una consulta tipo JOIN entre la tabla cliente y boleta. Aún no cumplimos el objetivo de mostrar los productos que compró el cliente con rut 204821232. Por lo que seguimos avanzando con el resto de las tablas. Desde ahora en adelante, este select solo continuará creciendo.

Explicación del segundo JOIN

Ahora que ya tenemos el join entre cliente y boleta, es necesario unir boleta con detalle. Esta unión se hace por medio del atributo codigo_boleta de lo obtenido anteriormente y codigo_boleta de “detalle”.



Explicación del segundo JOIN

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
WHERE C.RUT = 204821232
```

Explicación del segundo JOIN

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
WHERE C.RUT = 204821232
```

Explicación del segundo JOIN

Internamente, la base de datos asocia la consulta que ya realizamos.

Explicación del segundo JOIN

Internamente, la base de datos asocia la consulta que ya realizamos.

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
WHERE C.RUT = 204821232
```

Explicación del segundo JOIN

Internamente, la base de datos asocia la consulta que ya realizamos, con la tabla “detalle” realizando la unión por medio del atributo “codigo_boleta”.

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	FECHA	TOTAL
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	12-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	16-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	11-10-2019	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	12-12-2019	130000

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
WHERE C.RUT = 204821232
```

Explicación del segundo JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	FECHA	TOTAL
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	12-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	16-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	11-10-2019	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	12-12-2019	130000

DETALLE				
CODIGO_PRODUCTO	CODIGO_BOLETA	CANTIDAD	PRECIO_UNITARIO	TOTAL
1	1	1	380000	380000
21	1	1	8500	8500
2	2	1	110000	110000
6	3	1	81990	81990
3	4	1	3500	3500
17	5	1	370000	370000
17	6	1	370000	370000
19	7	1	260000	260000
21	8	2	8500	17000
2	9	1	110000	110000
8	10	1	560000	560000
3	11	2	3500	7000
7	12	1	280990	280990
20	13	1	850000	850000
21	13	1	8500	8500
6	13	1	81990	81990
2	14	3	110000	330000
5	14	1	67990	67990
10	15	1	28990	28990
9	15	1	450000	450000
10	16	1	28990	28990
15	17	1	130000	130000
20	18	1	850000	850000

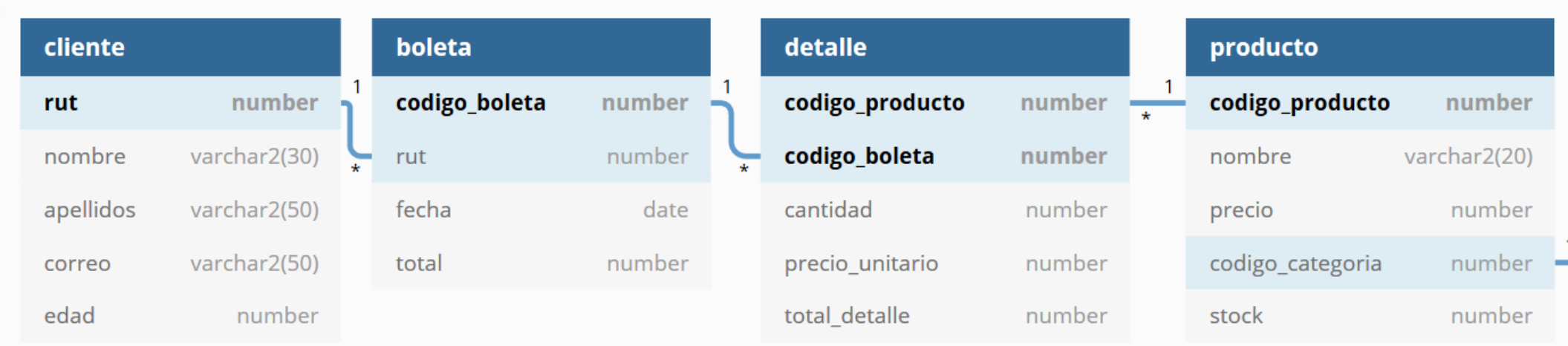
Resultado del segundo JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	FECHA	TOTAL	CODIGO_PRODUCTO	CANTIDAD	PRECIO_UNITARIO	TOTAL_1
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	12-08-2019	370000	17	1	370000	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	16-08-2019	370000	17	1	370000	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	11-10-2019	280990	7	1	280990	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	12-12-2019	130000	15	1	130000	130000

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
WHERE C.RUT = 204821232
```

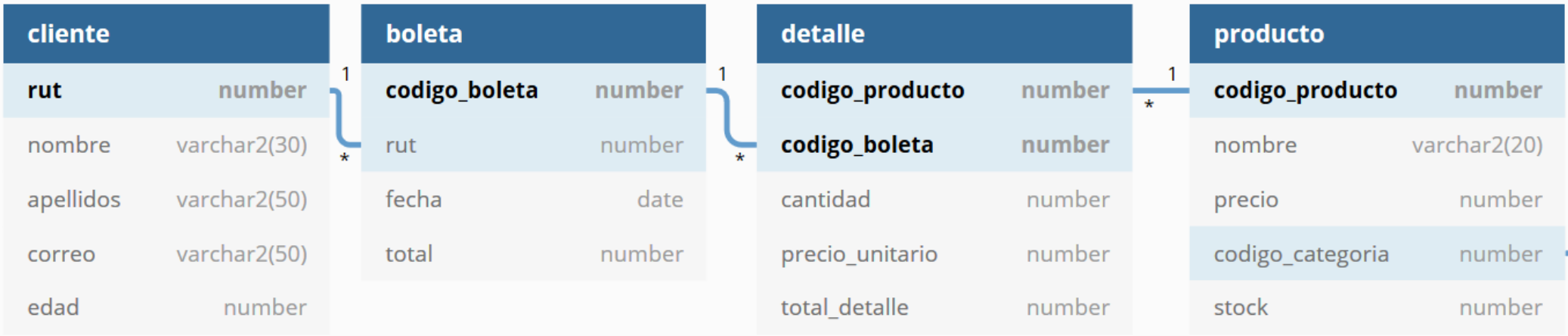
Explicación del tercer JOIN

Ahora que ya tenemos el join entre cliente, boleta y detalle, solo nos falta unir todo con la tabla producto. Esto se hace por medio del atributo codigo_producto de la tabla “detalle” con el codigo_producto de la tabla “producto”.



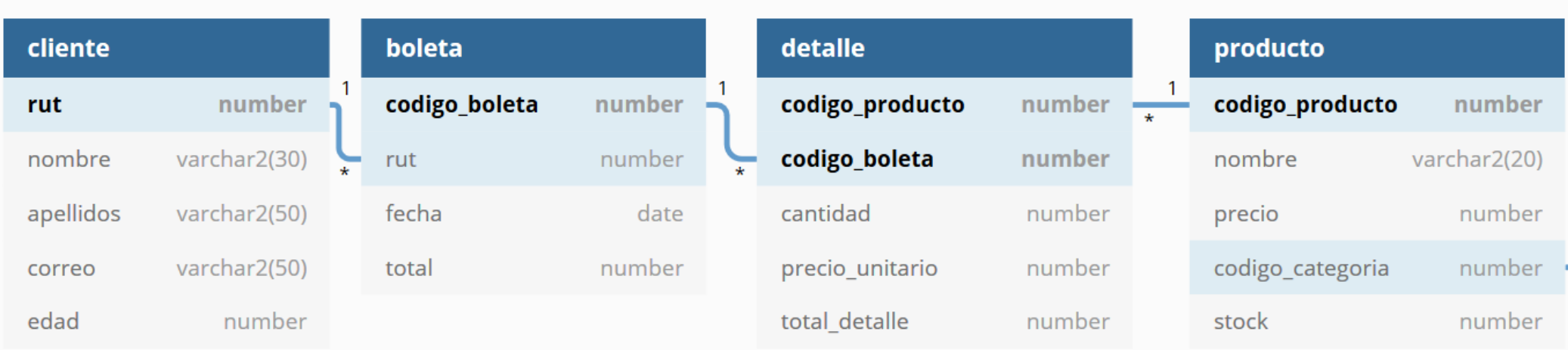
Explicación del tercer JOIN

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE,
P.CODIGO_PRODUCTO, P.NOMBRE,
P.PRECIO, P.CODIGO_CATEGORIA,
P.STOCK
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
WHERE C.RUT = 204821232
```



Explicación del tercer JOIN

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE,
P.CODIGO_PRODUCTO, P.NOMBRE,
P.PRECIO, P.CODIGO_CATEGORIA,
P.STOCK
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
WHERE C.RUT = 204821232
```



Explicación del tercer JOIN

Internamente, la base de datos asocia la consulta que ya realizamos.

Explicación del tercer JOIN

Internamente, la base de datos asocia la consulta que ya realizamos.

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE,
P.CODIGO_PRODUCTO, P.NOMBRE,
P.PRECIO, P.CODIGO_CATEGORIA,
P.STOCK
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
WHERE C.RUT = 204821232
```

Explicación del tercer JOIN

Internamente, la base de datos asocia la consulta que ya realizamos, con la tabla “producto” realizando la unión por medio del atributo “codigo_producto”.

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE,
P.CODIGO_PRODUCTO, P.NOMBRE,
P.PRECIO, P.CODIGO_CATEGORIA,
P.STOCK
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
WHERE C.RUT = 204821232
```

Explicación del tercer JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	FECHA	TOTAL	CODIGO_PRODUCTO	CANTIDAD	PRECIO_UNITARIO	TOTAL_1
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	12-08-2019	370000	17	1	370000	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	16-08-2019	370000	17	1	370000	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	11-10-2019	280990	7	1	280990	280990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	12-12-2019	130000	15	1	130000	130000

PRODUCTO				
CODIGO_PRODUCTO	NOMBRE	PRECIO	CODIGO_CATEGORIA	STOCK
1	NOTEBOOK LENOVO	$380.000	1	20
2	CELULAR MOTOROLA	$110.000	2	15
3	AUDIFONOS MACROTEL	$3.500	3	12
4	NOTEBOOK SAMSUNG	$500.000	1	3
5	MONITOR 17" AOC	$67.990	4	80
6	MONITOR 21" DELL	$81.990	4	20
7	NOTEBOOK HP	$280.990	1	1
8	CELULAR IPHONE 6S	$560.000	2	6
9	CELULAR LG	$450.000	2	7
10	MOUSE BLUETOOTH	$28.990	5	9
11	MONITOR 17" SAMSUNG	$150.000	4	7
12	MONITOR 17" LENOVO	$250.000	4	7
13	MONITOR 17" LG	$125.000	4	4
14	MONITOR 15" LENOVO	$200.000	4	3
15	MOTO G 2013	$130.000	2	45
16	MOTO G 2015	$180.000	2	32
17	MOTO X PLAY	$370.000	2	12
18	IPAD MINI 2	$200.000	6	4
19	IPAD AIR	$260.000	6	2
20	NOTEBOOK MSI	$850.000	1	4
21	MOUSE MICROSOFT	$8.500	5	5

El join anterior, utiliza el “codigo_producto” para hacer la unión con el “codigo_producto” de la tabla “producto”.

Resultado del tercer JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	FECHA	TOTAL	CODIGO_PRODUCTO
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	11-10-2019	280990	7
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	12-12-2019	130000	15
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	12-08-2019	370000	17
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	16-08-2019	370000	17

CANTIDAD	PRECIO_UNITARIO	TOTAL_1	CODIGO_PRODUCTO_1	NOMBRE_1	PRECIO	CATEGORIA	STOCK
1	280990	280990	7	NOTEBOOK HP	280990	1	1
1	130000	130000	15	MOTO G 2013	130000	2	45
1	370000	370000	17	MOTO X PLAY	370000	2	12
1	370000	370000	17	MOTO X PLAY	370000	2	12

```
SELECT C.RUT, C.NOMBRE, C.APELLIDOS,
C.CORREO, C.EDAD, B.CODIGO_BOLETA,
B.FECHA, B.TOTAL, D.CODIGO_PRODUCTO,
D.CANTIDAD, D.PRECIO_UNITARIO,
D.TOTAL AS TOTAL_DETALLE
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
WHERE C.RUT = 204821232
```

Resultado FINAL del JOIN

El objetivo inicial era solo mostrar los productos que el cliente ha comprado. Por lo que modificaremos el select para que cumpla ese objetivo.

Resultado FINAL del JOIN

El objetivo inicial era solo mostrar los productos que el cliente ha comprado. Por lo que modificaremos el select para que cumpla ese objetivo.

Para esto, mostraremos el rut y nombre de cliente, el código y nombre del producto y por último, la fecha de compra y el total de la boleta.

Resultado FINAL del JOIN

El objetivo inicial era solo mostrar los productos que el cliente ha comprado. Por lo que modificaremos el select para que cumpla ese objetivo.

Para esto, mostraremos el rut y nombre de cliente, el código y nombre del producto y por último, la fecha de compra y el total de la boleta.

```
SELECT C.RUT, C.NOMBRE, P.CODIGO_PRODUCTO,
P.NOMBRE AS NOMBRE_PRODUCTO, B.FECHA, B.TOTAL
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
WHERE C.RUT = 204821232
```

Resultado FINAL del JOIN

RUT	NOMBRE	CODIGO_PRODUCTO	NOMBRE_PRODUCTO	FECHA	TOTAL
204821232	Francisca	7	NOTEBOOK HP	11-10-2019	280990
204821232	Francisca	15	MOTO G 2013	12-12-2019	130000
204821232	Francisca	17	MOTO X PLAY	12-08-2019	370000
204821232	Francisca	17	MOTO X PLAY	16-08-2019	370000

```
SELECT C.RUT, C.NOMBRE, P.CODIGO_PRODUCTO,
P.NOMBRE AS NOMBRE_PRODUCTO, B.FECHA, B.TOTAL
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
WHERE C.RUT = 204821232
```

JOIN Explícito

Lo que acabamos de visualizar es un join explícito. Este JOIN es una mejora de la sintaxis antigua para realizar la unión de tablas.

```
SELECT COLUMNAS
FROM TABLA1 JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

JOIN Implícito

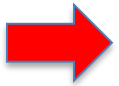
El join implícito permite obtener los mismos resultados, solo que la sintaxis es un poco distinta.

```
SELECT COLUMNAS
FROM TABLA1, TABLA2, TABLAX
WHERE COLUMNA_TABLA1 = COLUMNA_TABLA2
AND COLUMNA_TABLA2 = COLUMNA_TABLAX
```

JOIN Explícito e Implícito

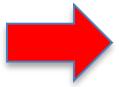
Podemos comparar la consulta realizada anteriormente.

```
SELECT C.RUT, C.NOMBRE, P.CODIGO_PRODUCTO,
P.NOMBRE AS NOMBRE_PRODUCTO, B.FECHA, B.TOTAL
FROM CLIENTE C
JOIN BOLETA B
ON C.RUT = B.RUT
JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA
JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
WHERE C.RUT = 204821232
```



Join explícito

```
SELECT C.RUT, C.NOMBRE, P.CODIGO_PRODUCTO,
P.NOMBRE AS NOMBRE_PRODUCTO, B.FECHA, B.TOTAL
FROM CLIENTE C, BOLETA B, DETALLE D, PRODUCTO P
WHERE C.RUT = B.RUT
AND B.CODIGO_BOLETA = D.CODIGO_BOLETA
AND D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO
AND C.RUT = 204821232;
```



Join implícito

NATURAL JOIN

Este JOIN actúa igual que los JOIN’s anteriores, la única diferencia, es que para este JOIN, no es necesario indicar las columnas que realizan la conexión entre tablas.

NATURAL JOIN

Este JOIN actúa igual que los JOIN’s anteriores, la única diferencia, es que para este JOIN, no es necesario indicar las columnas que realizan la conexión entre tablas.

Internamente, la base de datos identifica como se conectan las tablas.

NATURAL JOIN

Este JOIN actúa igual que los JOIN's anteriores, la única diferencia, es que para este JOIN, no es necesario indicar las columnas que realizan la conexión entre tablas.

Internamente, la base de datos identifica como se conectan las tablas.

```
SELECT COLUMNAS
FROM TABLA1 NATURAL JOIN TABLA2
WHERE COLUMNA OPERATOR VALOR
```

NATURAL JOIN

```
SELECT COLUMNAS
FROM TABLA1 NATURAL JOIN TABLA2
WHERE COLUMNA OPERATOR VALOR
```

Para que el NATURAL JOIN funcione, primero que todo, las columnas en ambas tablas deben tener el mismo nombre.

Realizar muchos NATURAL JOIN confunde a la base de datos cuando el modelo es muy grande.

Ejemplo NATURAL JOIN

Para que el NATURAL JOIN funcione, primero que todo, las columnas en ambas tablas deben tener el mismo nombre.

Realizar muchos NATURAL JOIN confunde a la base de datos cuando el modelo es muy grande.

Ejemplo NATURAL JOIN

Consideremos obtener el rut y nombre de la tabla “cliente” y el codigo_boleta y total de la tabla “boleta” utilizando un JOIN tradicional y un NATURAL JOIN.

Ejemplo NATURAL JOIN

Consideremos obtener el rut y nombre de la tabla “cliente” y el codigo_boleta y total de la tabla “boleta” utilizando un JOIN tradicional y un NATURAL JOIN.

```
SELECT C.RUT, C.NOMBRE, B.CODIGO_BOLETA,
B.TOTAL
FROM CLIENTE C JOIN BOLETA B
ON C.RUT = B.RUT
WHERE C.RUT = 204821232

SELECT RUT, NOMBRE, CODIGO_BOLETA, TOTAL
FROM CLIENTE NATURAL JOIN BOLETA
WHERE RUT = 204821232;
```

Inner Join

En algunas ocasiones podrían observar un select con la instrucción “INNER JOIN” en vez de JOIN.

Inner Join

En algunas ocasiones podrían observar un select con la instrucción “INNER JOIN” en vez de JOIN.

```
SELECT COLUMNAS
FROM TABLA1 INNER JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

Inner Join

En algunas ocasiones podrían observar un select con la instrucción “INNER JOIN” en vez de JOIN.

```
SELECT COLUMNAS
FROM TABLA1 INNER JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

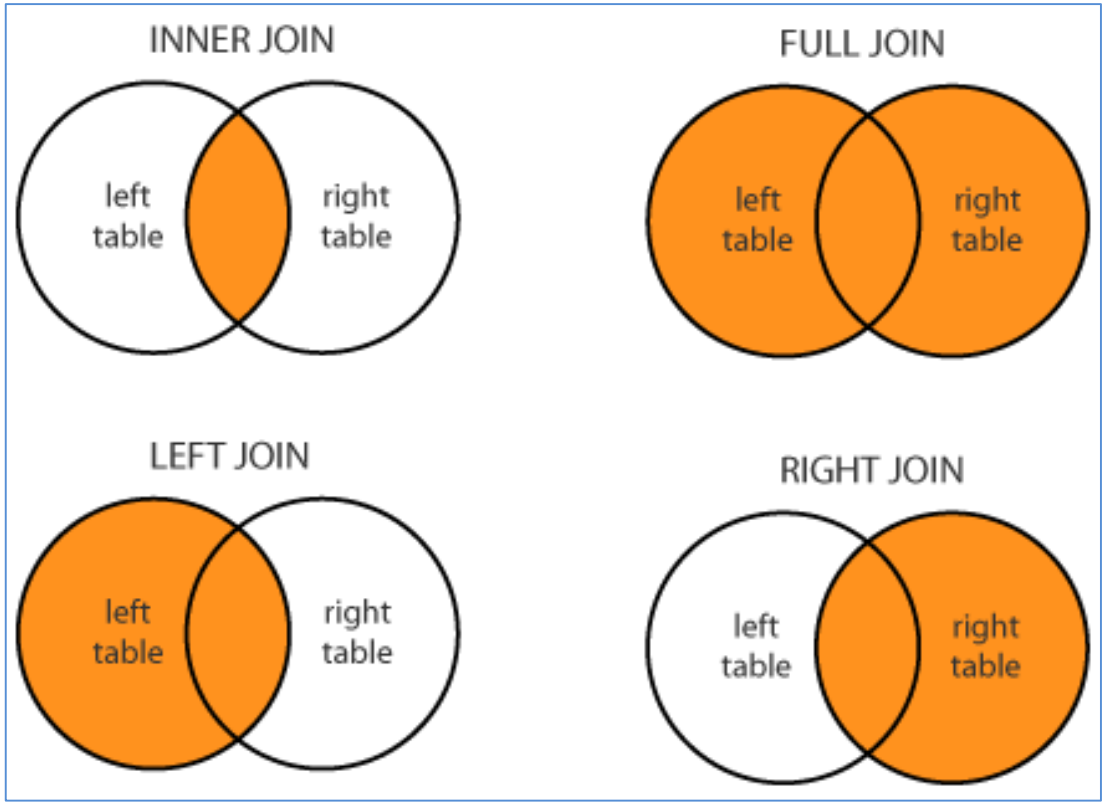
Este es el “JOIN” por defecto que se realiza internamente cuando un select lleva solamente la instrucción “JOIN” sin acompañamiento.

Tipos de JOIN

Cuando hablamos de “acompañamiento” en la instrucción JOIN, nos referimos a lo siguiente:

Tipos de JOIN

Cuando hablamos de “acompañamiento” en la instrucción JOIN, nos referimos a lo siguiente:



LEFT, FULL, RIGHT e INNER JOIN

```
SELECT COLUMNAS
FROM TABLA1 JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR

SELECT COLUMNAS
FROM TABLA1 LEFT OUTER JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR

SELECT COLUMNAS
FROM TABLA1 RIGHT OUTER JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR

SELECT COLUMNAS
FROM TABLA1 FULL OUTER JOIN JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

LEFT OUTER JOIN

Este JOIN retorna todos los registros de la tabla de la “izquierda”, aún cuando no exista un registro correspondiente en la tabla de la “derecha”.

```
SELECT COLUMNAS
FROM TABLA1 LEFT OUTER JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

Ejemplo LEFT OUTER JOIN

Consideremos la siguiente información entre cliente y boleta.

Ejemplo LEFT OUTER JOIN

Consideremos la siguiente información entre cliente y boleta.

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Ejemplo LEFT OUTER JOIN

Consideremos la siguiente información entre cliente y boleta.

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

De los 6 clientes que están en la tabla “cliente”, solo uno, no posee boletas asociadas en la tabla “boleta”.

Ejemplo LEFT OUTER JOIN

Consideremos la siguiente información entre cliente y boleta.

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

De los 6 clientes que están en la tabla “cliente”, solo uno, no posee boletas asociadas en la tabla “boleta”.

Ejemplo LEFT OUTER JOIN

Si realizamos un JOIN normal, en los resultados no aparecerá Carlos Hernandez, ya que no coincide en la tabla “boleta”.

Ejemplo LEFT OUTER JOIN

Si realizamos un JOIN normal, en los resultados no aparecerá Carlos Hernandez, ya que no hay coincidencias de su rut en la tabla “boleta”.

Si hacemos un LEFT OUTER JOIN, podemos obtener al cliente Carlos Hernandez aún cuando no tiene una boleta asociada a su rut.

Ejemplo LEFT OUTER JOIN

Si realizamos un JOIN normal, en los resultados no aparecerá Carlos Hernandez, ya que no coincide en la tabla “boleta”.

Si hacemos un LEFT OUTER JOIN, podemos obtener al cliente Carlos Hernandez aún cuando no tiene una boleta asociada a su rut.

```
SELECT *
FROM CLIENTE C LEFT OUTER JOIN BOLETA B
ON C.RUT = B.RUT
```

Ejemplo LEFT OUTER JOIN

```
SELECT *
FROM CLIENTE C LEFT OUTER JOIN BOLETA B
ON C.RUT = B.RUT
```

El select tomará todos los registros de la tabla “cliente” y asociará el rut de los clientes con el rut de la tabla “boleta”. Para el rut de cliente que no tenga coincidencia en la tabla “boleta”, le asignará valores nulos a esos campos.

Ejemplo LEFT OUTER JOIN

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Primero el JOIN asociará los ruts que coinciden.

Ejemplo LEFT OUTER JOIN

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Primero el JOIN asociará los ruts que coinciden.

Ejemplo LEFT OUTER JOIN

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Primero el JOIN asociará los ruts que coinciden.

Luego, para el rut que no tiene coincidencia en la tabla boleta, asignará valores nulos a los campos en la tabla “boleta”.

Ejemplo LEFT OUTER JOIN

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000
null	null	null	null

CLIENTE				
RUT	NOMBRE	APELLIDOS	CORREO	EDAD
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18
185756203	Felipe	Tapia	ftapia46@gmail.com	25
85762340	Camila	Fernandez	cfernandez@ucm.cl	58
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28

Primero el JOIN asociará los ruts que coinciden.

Luego, para el rut que no tiene coincidencia en la tabla boleta, asignará valores nulos a los campos en la tabla “boleta”.

Resultado LEFT OUTER JOIN

RUT	NOMBRE	APELLIDOS	CORREO	EDAD	CODIGO_BOLETA	RUT_1	FECHA	TOTAL
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39	1	145375682	12-07-2019	388500
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39	3	145375682	17-08-2019	81990
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39	4	145375682	24-08-2019	3500
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	5	204821232	12-08-2019	370000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	6	204821232	16-08-2019	370000
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24	7	183447872	02-09-2019	260000
85762340	Camila	Fernandez	cfernandez@ucm.cl	58	8	85762340	10-09-2019	17000
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24	9	183447872	16-09-2019	110000
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39	10	145375682	20-09-2019	560000
85762340	Camila	Fernandez	cfernandez@ucm.cl	58	11	85762340	01-10-2019	7000
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	12	204821232	11-10-2019	280990
185756203	Felipe	Tapia	ftapia46@gmail.com	25	13	185756203	12-10-2019	940490
183447872	Javiera	Faundez	javi_faundez94@gmail.com	24	14	183447872	05-11-2019	397990
145375682	Alejandro	Sepulveda	alesepulveda@hotmail.com	39	15	145375682	08-11-2019	478990
185756203	Felipe	Tapia	ftapia46@gmail.com	25	16	185756203	12-12-2019	28990
204821232	Francisca	Perez	fran__xdperez2000@gmail.com	18	17	204821232	12-12-2019	130000
85762340	Camila	Fernandez	cfernandez@ucm.cl	58	18	85762340	21-12-2019	850000
175682842	Carlos	Hernandez	carlos_herna91@gmail.com	28	null	null	null	null

RIGHT OUTER JOIN

Este JOIN retorna todos los registros de la tabla de la "derecha", aún cuando no exista un registro correspondiente en la tabla de la "izquierda".

```
SELECT COLUMNAS
FROM TABLA1 RIGHT OUTER JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

RIGHT OUTER JOIN

Este JOIN retorna todos los registros de la tabla de la "derecha", aún cuando no exista un registro correspondiente en la tabla de la "izquierda".

```
SELECT COLUMNAS
FROM TABLA1 RIGHT OUTER JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

Básicamente es lo mismo que el LEFT OUTER JOIN, pero visto desde la derecha.

RIGHT OUTER JOIN

De hecho, es posible realizar el mismo select anterior del LEFT OUTER JOIN con el RIGHT OUTER JOIN.

RIGHT OUTER JOIN

De hecho, es posible realizar el mismo select anterior del LEFT OUTER JOIN con el RIGHT OUTER JOIN.

Solo es necesario intercambiar las de lado tablas.

```
SELECT *
FROM BOLETA B RIGHT OUTER JOIN CLIENTE C
ON B.RUT = C.RUT

SELECT *
FROM CLIENTE C LEFT OUTER JOIN BOLETA B
ON C.RUT = B.RUT
```

FULL OUTER JOIN

Este JOIN retorna todos los registros de la tabla de la “derecha”, aún cuando no exista un registro correspondiente en la tabla de la “izquierda” y retorna todos los registros de la tabla “izquierda”, aún cuando no exista un registro correspondiente en la tabla de la “derecha”.

FULL OUTER JOIN

Es decir, muestra todo lo que coincide y no coincide de ambas tablas.

```
SELECT COLUMNAS
FROM TABLA1 FULL OUTER JOIN JOIN TABLA2
ON COLUMNA_TABLA1 = COLUMNA_TABLA2
WHERE COLUMNA OPERATOR VALOR
```

Consultas de Agrupación

Este tipo de consultas son bastante especiales, ya que solo trabajan con un conjunto de funciones.

Consultas de Agrupación

Este tipo de consultas son bastante especiales, ya que solo trabajan con un conjunto de funciones.

La agrupación permite realizar resúmenes sobre la información contenida en la base de datos.

Consultas de Agrupación

Este tipo de consultas son bastante especiales, ya que solo trabajan con un conjunto de funciones.

La agrupación permite realizar resúmenes sobre la información contenida en la base de datos.

Los resúmenes están enfocados a cantidades, sumas, totales, mínimos y máximos, por decir algo básico de su utilidad.

Consultas de Agrupación

Para realizar estos resúmenes las consultas de agrupación cuentan con las siguientes funciones:

Funciones de agrupación	
AVG	Calcula el promedio sobre una columna
COUNT	Realiza el conteo sobre una columna
MAX	Calcula el máximo valor sobre una columna
MIN	Calcula el mínimo valor sobre una columna
STDDEV	Calcula la desviación estándar sobre una columna
SUM	Realiza la suma sobre una columna
VARIANCE	Calcula la varianza sobre una columna

Ejemplo de agrupación

Consideremos la siguiente información:

Ejemplo de agrupación

Consideremos la siguiente información:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

Ejemplo de agrupación

Consideremos la siguiente información:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

Sería interesante saber cuál es el cliente que más compra en nuestra tienda. Esto tomando en cuenta el dinero que desembolsa.

Ejemplo de agrupación

Consideremos la siguiente información:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

Sería interesante saber cuál es el cliente que más compra en nuestra tienda. Esto tomando en cuenta el dinero que desembolsa.

Para realizar esto, deberíamos considerar cada cliente por separado.

Ejemplo de agrupación

Consideremos la siguiente información:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

Sería interesante saber cuál es el cliente que más compra en nuestra tienda. Esto tomando en cuenta el dinero que desembolsa.

Para realizar esto, deberíamos considerar cada cliente por separado.

Ejemplo de agrupación

Consideremos la siguiente información:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

Sería interesante saber cuál es el cliente que más compra en nuestra tienda. Esto tomando en cuenta el dinero que desembolsa.

Para realizar esto, deberíamos considerar cada cliente por separado.

Y luego sumar la columna “total” y asociar esa suma a cada usuario.

Ejemplo de agrupación

Consideremos la siguiente información:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

Sería interesante saber cuál es el cliente que más compra en nuestra tienda. Esto tomando en cuenta el dinero que desembolsa.

Para realizar esto, deberíamos considerar cada cliente por separado.

Y luego sumar la columna “total” y asociar esa suma a cada usuario.

Para realizar esto, hay columnas que no vamos a considerar (codigo_boleta y fecha).

Ejemplo de agrupación

Resultado:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

Ejemplo de agrupación

Resultado:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$940.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

Para realizar este tipo de consultas, SQL nos provee de una nueva instrucción.

Esta instrucción se llama GROUP BY, y permite agrupar columnas respecto a algunas de las funciones antes mencionadas.

¿Cómo hacer un GROUP BY?

Resultado esperado

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

```
SELECT COLUMNAS, FUNCION(COLUMNA)
FROM TABLA
WHERE COLUMNA OPERATOR VALOR
GROUP BY COLUMNAS
```

¿Cómo hacer un GROUP BY?

Resultado esperado

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

```
SELECT COLUMNAS, FUNCION(COLUMNA)
FROM TABLA
WHERE COLUMNA OPERATOR VALOR
GROUP BY COLUMNAS
```

Se añade la instrucción GROUP BY al final del select que ya conocemos.

¿Cómo hacer un GROUP BY?

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

Resultado esperado

```
SELECT COLUMNAS, FUNCION(COLUMNA)
FROM TABLA
WHERE COLUMNA OPERATOR VALOR
GROUP BY COLUMNAS
```

Se añade la instrucción GROUP BY al final del select que ya conocemos.

Para el ejemplo (sobre la tabla “boleta”), debemos indicar la columna sobre la cual agruparemos y la columna sobre la cual utilizaremos una función de agrupación.

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

¿Cómo hacer un GROUP BY?

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

```
SELECT COLUMNAS, FUNCION(COLUMNA)
FROM TABLA
WHERE COLUMNA OPERATOR VALOR
GROUP BY COLUMNAS
```

Se añade la instrucción GROUP BY al final del select que ya conocemos.

Para el ejemplo (sobre la tabla “boleta”), debemos indicar la columna sobre la cual agruparemos y la columna sobre la cual utilizaremos una función de agrupación.

En este ejemplo, agrupamos sobre “RUT”, el valor de la suma de la columna “TOTAL”.

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

¿Cómo ordenar la información?

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

La información de resúmenes debería siempre estar ordenada. SQL ofrece una instrucción para realizar ordenamiento llamada ORDER BY

Para el ejemplo (sobre la tabla “boleta”), debemos indicar la columna sobre la cual agruparemos y la columna sobre la cual utilizaremos una función de agrupación.

En este ejemplo, agrupamos sobre “RUT”, el valor de la suma de la columna “TOTAL”.

¿Cómo ordenar la información?

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

La información de resúmenes debería siempre estar ordenada. SQL ofrece una instrucción para realizar ordenamiento llamada ORDER BY

La instrucción ORDER BY puede ordenar en forma ascendente (ASC) o descendente (DESC) una o múltiples columnas.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

¿Cómo ordenar la información?

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$969.480

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

La información de resúmenes debería siempre estar ordenada. SQL ofrece una instrucción para realizar ordenamiento llamada ORDER BY

La instrucción ORDER BY puede ordenar en forma ascendente (ASC) o descendente (DESC) una o múltiples columnas.

Para el ejemplo, ordenaremos la consulta realizada de acuerdo al TOTAL, en forma descendente. Así mostrará primero el mayor TOTAL y luego irá descendiendo.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

¿Cómo ordenar la información?

RUT	TOTAL
145375682	$1.622.980
204821232	$1.150.990
185756203	$969.480
85762340	$874.000
183447872	$767.990

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

La información de resúmenes debería siempre estar ordenada. SQL ofrece una instrucción para realizar ordenamiento llamada ORDER BY

La instrucción ORDER BY puede ordenar en forma ascendente (ASC) o descendente (DESC) una o múltiples columnas.

Para el ejemplo, ordenaremos la consulta realizada de acuerdo al TOTAL, en forma descendente. Así mostrará primero el mayor TOTAL y luego irá descendiendo.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Ejercicios

- Descargue el archivo modelov2.sql
- Compile el nuevo modelo de datos
- Ejecute las instrucciones de inserción y siga las indicaciones
- Resuelva las consultas que se plantean

UNIVERSIDAD CATOLICA DEL MAULE