Arquitectura y Administración de bases de datos con SQL 2021

Héctor Manuel Garduño Castañeda

Diciembre, 2021

scidata

Contenido

Motivación del problema

Para resolver el problema anterior, observemos la información que tenemos en cada una de nuestras tablas:

Tabla de clientes (customer)

Data Output | Explain | Messages | Notifications

	customer_id character varying	customer_name character varying	segment character varying	age integer	country character varying	city character varying	state character varying	postal_code integer	region character varying
1	CG-12520	Claire Gute	Consumer	67	United States	Henderson	Kentucky	42420	South

Tabla de ventas (sales)

Data Output | Explain | Messages | Notifications

	order_line [PK] integer	order_id character varying	order_date date	ship_date date	ship_mode character varying	customer_id character varying	product_id character varying	sales numeric	quantity integer	discount numeric	profit numeric
1	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	FUR-BO-10001798	261.96	2	0	41.9136

Tabla de productos (product)

Data Output | Explain | Messages | Notifications

	product_id [PK] character varying	category character varying	sub_category character varying	product_name character varying
1	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase

Como puede observarse, en la tabla de ventas no aparece la información por región. Las regiones las tenemos en la tabla de clientes.

Supongamos que las tablas contienen únicamente los siguientes registros.

Order Line	Order ID	Order Date	Customer ID	Product ID	Sales
1	CA-2016-152156	08-11-2016	CG-12520	FUR-BO-10001798	261.96
88	CA-2017-155558	26-10-2017	PG-18895	OFF-LA-10000134	6.16
3	CA-2016-138688	12-06-2016	DV-13045	OFF-LA-10000240	14.62
5	US-2015-108966	11-10-2015	SO-20335	OFF-ST-10000760	22.368

Tabla de ventas (sales)

Customer ID	Customer Name	State	Region
CG-12520	Claire Gute	Kentucky	South
DV-13045	Darrin Van Huff	California	West
SO-20335	Sean O'Donnell	Florida	South
BH-11710	Brosina Hoffman	California	West

Tabla de clientes (customer)

Note que:

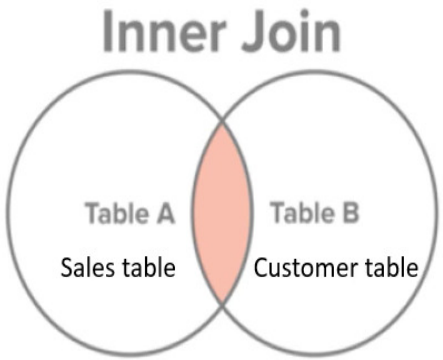
1. CG-12520, DV-13045 y SO-20335 están presentes en las dos tablas.
2. PG-18895 está en ventas pero no en clientes.
3. BH-11710 está en clientes pero no en ventes.

Inner Join

+ El INNER JOIN puede que reduzca el número de registros

Una unión interna de tablas se refiere a intersectar las tablas. Por lo tanto tendremos como resultado una tabla con **tal vez menos registros**.

Order Line	Order ID	Order Date	Customer ID	Product ID	Sales	Customer Name	State	Region
1	CA-2016-152156	08-11-2016	CG-12520	FUR-BO-10001798	261.96	Claire Gute	Kentucky	South
3	CA-2016-138688	12-06-2016	DV-13045	OFF-LA-10000240	14.62	Darrin Van Huff	California	West
5	US-2015-108966	11-10-2015	SO-20335	OFF-ST-10000760	22.368	Sean O'Donnell	Florida	South



Left Join

+ El LEFT JOIN no va a reducir el número de renglones

En el LEFT JOIN nos quedamos originalmente con los registros de la tabla de la izquierda, y solo llenamos las nuevas columnas con los datos de la tabla derecha

Una unión izquierda de tablas se refiere a unir los registros de la tabla derecha a la de la izquierda. Por lo tanto tendremos como resultado una tabla con **con los mismos registros que la tabla izquierda y valores nulos**.

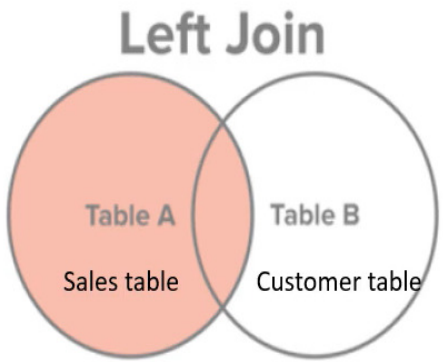
tabla de ventas

Order Line	Order ID	Order Date	Customer ID	Product ID	Sales	Customer Name	State	Region
1	CA-2016-152156	08-11-2016	CG-12520	FUR-BO-10001798	261.96	Claire Gute	Kentucky	South
88	CA-2017-155558	26-10-2017	PG-18895	OFF-LA-10000134	6.16	Null	Null	Null
3	CA-2016-138688	12-06-2016	DV-13045	OFF-LA-10000240	14.62	Darrin Van Huff	California	West
5	US-2015-108966	11-10-2015	SO-20335	OFF-ST-10000760	22.368	Sean O'Donnell	Florida	South

Elementos en común

tabla clientes

Si la tablaA tiene N registros, entonces nos quedamos con N registros después de hacer un LEFT JOIN



+ De aquellos datos que teniamos en A, si hay columnas en comun en ambas tablas, va a mostrar la tabla de A junto con otras columnas que vienen de la tabla B. Si hay un elemento de A que no está en B pondrá valores nulos en las nuevas columnas que trajimos de B

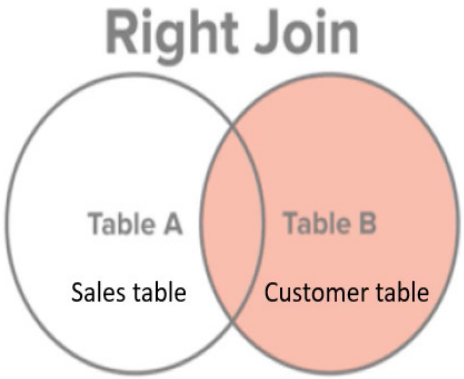
Right Join

+ El RIGHT JOIN no reduce el número de registros

En el RIGHT JOIN nos quedamos originalmente con los registros de la tabla derecha y añadimos la información de las columnas de la tabla de la izquierda

Una unión izquierda de tablas se refiere a unir los registros de la tabla izquierda a la de la derecha. Por lo tanto tendremos como resultado una tabla con **con los mismos registros que la tabla derecha y valores nulos**.

Order Line	Order ID	Order Date	Customer ID	Product ID	Sales	Customer Name	State	Region
1	CA-2016-152156	08-11-2016	CG-12520	FUR-BO-10001798	261.96	Claire Gute	Kentucky	South
3	CA-2016-138688	12-06-2016	DV-13045	OFF-LA-10000240	14.62	Darrin Van Huff	California	West
5	US-2015-108966	11-10-2015	SO-20335	OFF-ST-10000760	22.368	Sean O'Donnell	Florida	South
Null	Null	Null	BH-11710	Null	Null	Brosina Hoffman	California	West



Si la tablaB tiene M registros, nos quedamos con M registros después de hacer un RIGHT JOIN

scidata

+ Mismo razonamiento que el INNER JOIN

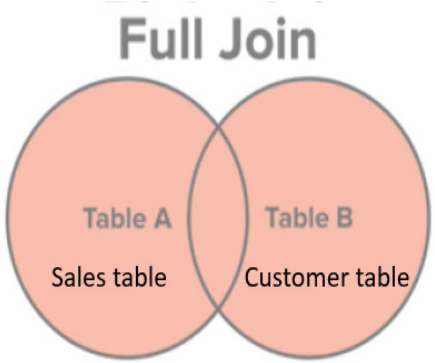
Full Join

+ El FULL JOIN probablemente tenga más registros

En el FULL JOIN nos junta los registros de ambas tablas.

Una unión completa de tablas se refiere a unir todos los registros de ambas tablas. Por lo tanto tendremos como resultado una tabla con **con probablemente mas registros que las originales y valores nulos**.

Order Line	Order ID	Order Date	Customer ID	Product ID	Sales	Customer Name	State	Region
1	CA-2016-152156	08-11-2016	CG-12520	FUR-BO-10001798	261.96	Claire Gute	Kentucky	South
88	CA-2017-155558	26-10-2017	PG-18895	OFF-LA-10000134	6.16	Null	Null	Null
3	CA-2016-138688	12-06-2016	DV-13045	OFF-LA-10000240	14.62	Darrin Van Huff	California	West
5	US-2015-108966	11-10-2015	SO-20335	OFF-ST-10000760	22.368	Sean O'Donnell	Florida	South
Null	Null	Null	BH-11710	Null	Null	Brosina Hoffman	California	West



scidata

+ El INNER y el LEFT JOIN son los más importantes

El comando **INNER JOIN** *compara* cada renglón de la tabla 1 (izquierda) con cada renglón de la tabla 2 (derecha) y viceversa, para encontrar todos los renglones que tienen en *común*. En realidad, lo que hace es tomar la columna *join* de cada tabla y buscar los valores comunes. Luego, arma una sola tabla uniendo en un solo renglón los correspondientes renglones de 1 y 2 compartiendo las columnas join.

columna

SELECT ⟨⟨columnas⟩⟩
FROM ⟨⟨tabla_1⟩⟩
INNER JOIN ⟨⟨tabla_2⟩⟩
ON tabla_1.columna_join = tabla_2.columna_join;

+ Columnas que quiero mostrar en el Inner Join

[nombre de la tabla].[nombre de la columna] (sintaxis nueva)

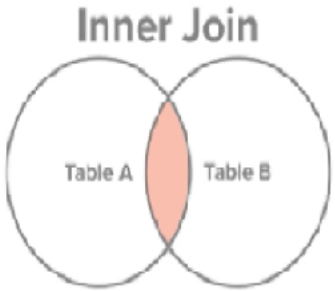
col1A	col2A	col3A
A	dato A	o
B	dato B	p
C	dato C	q
D	dato D	r

col1B	col2B
C	s
E	t
F	u

=

tabla 1 | tabla 2

col1A	col2A	col3A	col1B	col2B
C	dato C	q	C	s



+ La "columna join" es la que vamos a usar como "pegamento" en cada una de las tablas, generalmente es la columna id

+ col1A es la columna join de la primera
col1B es la columna join de la segunda

Práctica

Mostrar todos los números de orden, identificador de productos, identificador de clientes, ingresos, nombre del cliente y edad de todos los clientes entre 20 y 60 años que realizaron compras en 2015, ordenados por identificador de cliente.

```
SELECT
  a.order_line,
  a.product_id,
  a.customer_id,
  a.sales,
  b.customer_name,
  b.age
FROM sales_2015 AS a
INNER JOIN customer_20_60 AS b
ON a.customer_id = b.customer_id
ORDER BY customer_id;
```

+ Las columnas que quiero van separándose con comas

+ A las tablas les asignamos un nombre, A y B ahí mismo en la query

+ Las líneas anteriores ya me crearon una tabla donde una columna se llama customer_id, por eso ya no es necesario poner "A." o "B."

El comando **LEFT JOIN** nos regresa **todas las filas de la tabla izquierda**, aun cuando no exista coincidencia con las de la derecha.

SELECT ⟨⟨columnas⟩⟩
FROM ⟨⟨tabla_1⟩⟩
LEFT JOIN ⟨⟨tabla_2⟩⟩
ON tabla_1.columna_join = tabla_2.columna_join**;**

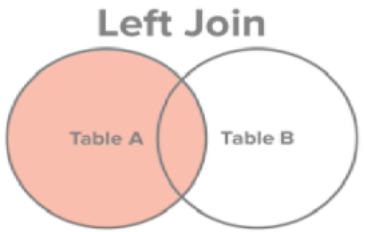
col1A	col2A	col3A
A	dato A	o
B	dato B	p
C	dato C	q
D	dato D	r

col1B	col2B
C	s
E	t
F	u

=

col1A	col2A	col3A	col1B	col2B
A	dato A	o	NULL	NULL
B	dato B	p	NULL	NULL
C	dato C	q	C	s
D	dato D	r	NULL	NULL

+ Misma sintaxis, solo cmabia INNER JOIN por LEFT JOIN



Práctica

Mostrar todos los números de orden, identificador de productos, identificador de clientes, ingresos y también, si se dispone, nombre del cliente y edad de todos los clientes entre 20 y 60 años, ordenados por identificador de cliente.

"Si lo encuentras, jálalo, si no, déjalo como null"

```
SELECT
  a.order_line,
  a.product_id,
  a.customer_id,
  a.sales,
  b.customer_name,
  b.age
FROM sales_2015 AS a
LEFT JOIN customer_20_60 AS b
ON a.customer_id = b.customer_id
ORDER BY customer_id;
```

El comando **RIGHT JOIN** nos regresa **todas las filas de la tabla derecha**, aun cuando no exista coincidencia con las de la izquierda.

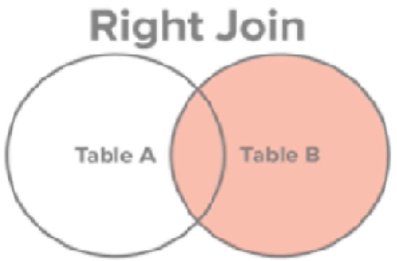
SELECT ⟨⟨columnas⟩⟩
FROM ⟨⟨tabla_1⟩⟩
RIGHT JOIN ⟨⟨tabla_2⟩⟩
ON tabla_1.columna_join = tabla_2.columna_join;

col1A	col2A	col3A
A	dato A	o
B	dato B	p
C	dato C	q
D	dato D	r

col1B	col2B
C	s
E	t
F	u

=

col1A	col2A	col3A	col1B	col2B
C	dato C	q	C	s
NULL	NULL	NULL	E	t
NULL	NULL	NULL	F	u



+ Podemos intercambiar el orden de las tablas y hacer un LEFT JOIN en lugar de un RIGHT JOIN

+ No necesariamente se pegan a la izquierda, se van a mostrar los datos dependiendo de cómo hayamos definido el SELECT

Práctica

Mostrar todos los nombre de clientes y edad de todos los clientes entre 20 y 60 años y, si se dispone, números de orden, identificador de productos, identificador de clientes, ingresos de quienes realizaron compras en 2015, ordenados por identificador de cliente.

```
SELECT
  a.order_line,
  a.product_id,
  a.customer_id,
  a.sales,
  b.customer_name,
  b.age
FROM sales_2015 AS a
RIGHT JOIN customer_20_60 AS b
ON a.customer_id = b.customer_id
ORDER BY customer_id;
```

El comando **FULL JOIN** nos regresa **todas las filas de ambas tablas**, aun cuando no exista coincidencia.

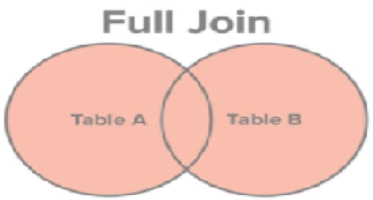
SELECT ⟨⟨columnas⟩⟩
FROM ⟨⟨tabla_1⟩⟩
FULL JOIN ⟨⟨tabla_2⟩⟩
ON tabla_1.columna_join = tabla_2.columna_join**;**

col1A	col2A	col3A
A	dato A	o
B	dato B	p
C	dato C	q
D	dato D	r

col1B	col2B
C	s
E	t
F	u

=

col1A	col2A	col3A	col1B	col2B
A	dato A	o	NULL	NULL
B	dato B	p	NULL	NULL
C	dato C	q	C	s
D	dato D	r	NULL	NULL
NULL	NULL	NULL	E	t
NULL	NULL	NULL	F	u



scidata

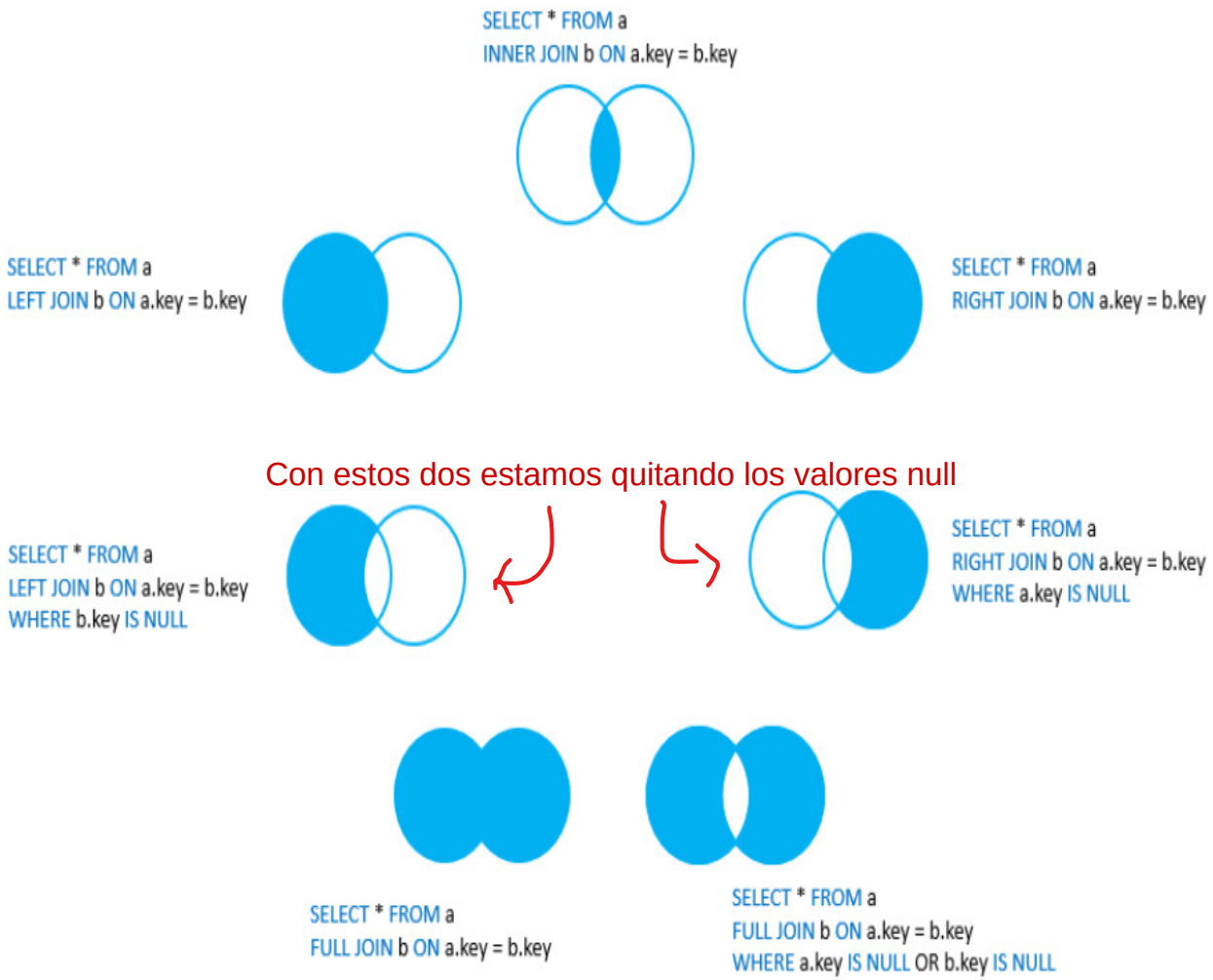
+ Misma sintaxis, funciona como la unión de conjuntos

Práctica

Mostrar todos los nombres de clientes y edad de todos los clientes entre 20 y 60 años y todos los números de orden, identificador de productos, identificador de clientes, ingresos de quienes realizaron compras en 2015, ordenados por identificador de cliente.

```
SELECT
  a.order_line,
  a.product_id,
  a.customer_id,
  a.sales,
  b.customer_name,
  b.age
FROM sales_2015 AS a
FULL JOIN customer_20_60 AS b
ON a.customer_id = b.customer_id
ORDER BY a.customer_id, b.customer_id;
```

La siguiente imagen muestra un resumen de los joins



Este filtro de WHERE es para devolver aquellos que sí encontró en ambos lados

El *producto cartesiano* entre dos tablas es una tabla resultado de combinar todas las filas de la tabla 1 con todas las filas de la tabla 2.

SELECT ⟨⟨columnas⟩⟩
FROM ⟨⟨tabla 1⟩⟩, ⟨⟨tabla 2⟩⟩,...

col1A	col2A	col3A
A	dato A	o
B	dato B	p
C	dato C	q
D	dato D	r

col1B	col2B
C	s
E	t
F	u

=

col1A	col2A	col3A	col1B	col2B
A	dato A	o	C	s
A	dato A	o	E	t
A	dato A	o	F	u
B	dato B	p	C	s
B	dato B	p	E	t
B	dato B	p	F	u
C	dato C	q	C	s
C	dato C	q	E	t
C	dato C	q	F	u
D	dato D	r	C	s
D	dato D	r	E	t
D	dato D	r	F	u

+ Toma el primer registro de la tabla A y la combina con el primer registro de la tabla B, luego con la segunda de la tabla B, y así sucesivamente.

Práctica

Crear una tabla con los años del 2011 al 2021 con cada uno de los meses.

CREATE TABLE month(MM int);
CREATE TABLE year(YYYY int);

INSERT INTO month **VALUES**
(1),(2),(3),(4),(5),(6),(7),(8),(9),(10),(11),(12);
INSERT INTO year **VALUES**
(2011),(2012),(2013),(2014),(2015),(2016),(2017),(2018),(2019),(2020),(2021);

SELECT a.YYYY, b.MM **FROM** year **AS** a, month **AS** b;

+ Combina cada uno de los años que están en la tabla a, con los valores de la tabla b
2011 1
2011 2
...
2012 1
2012 2