Relaciones y operaciones transaccionales (Parte I)

Operaciones de unión entre tablas

Competencias

- Realizar consultas usando el comando JOIN para el consumo de datos entre tablas relacionadas.
- Realizar subconsultas para la anidación de datos entre tablas.

Introducción

Hasta el momento todas las consultas que hemos realizado las hemos hecho sólo sobre una tabla. Previamente explicamos cómo las tablas se relacionan a través de las claves primarias y claves foráneas, pero debemos destacar que el objetivo principal de implementar estas claves en nuestras tablas, es la posibilidad de generar operaciones entre estas entidades y extraer conjuntos de datos concatenados para consultas más personalizadas.

En un plano real, los casos de persistencia de datos requieren de la creación de diferentes entidades (tablas), en las que distribuimos la información correspondiente a una operación ejecutada desde el software y consumir esos datos tabla por tabla sería muy engorroso, conlleva a un costo de computo considerable, para esto tenemos el comando JOIN qué nos servirá para navegar en los registros de tablas relacionadas y de esta manera lograr con una sola consulta obtener todos los datos relacionados en una situación determinada.

Unión de tablas

Como ya has aprendido en la unidad anterior, podemos relacionar dos tablas por medio de las claves primarias y foráneas, no obstante aún no le hemos sacado provecho a esto y es lo que te enseñaré a continuación.

Veamos el siguiente caso:

“Un sistema con diferentes secciones restringidas por niveles de usuarios, desea almacenar los datos de su personal e identificarlos según su rol en el proyecto, teniendo los roles de administrador, marketing, editor y subscriptor, cada uno con diferentes funciones.”

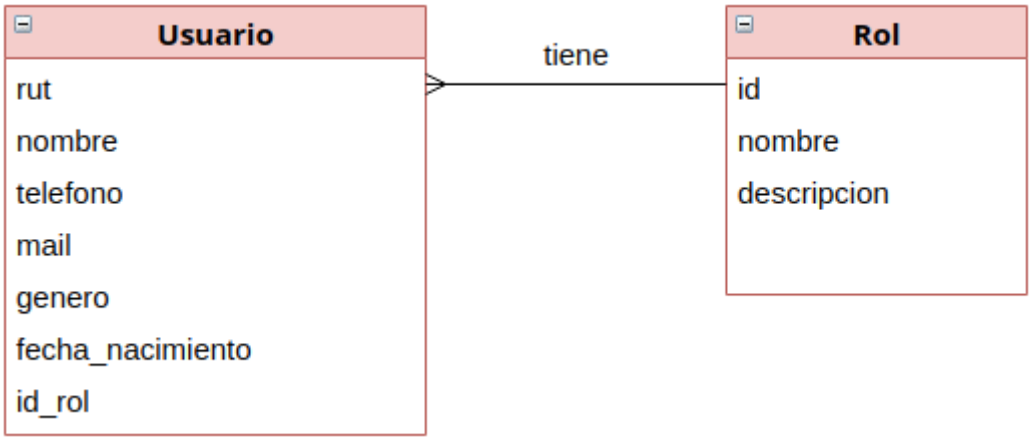

Imagen 1. Tabla Usuario y tabla Rol con llaves primarias y foráneas.

Como puedes ver en la imagen 1 tenemos una tabla de usuarios y una tabla de roles, como podrás intuir la primera tabla tiene un “id_rol” y estos campos dan puntapié a que funcionan como punto de enlace de más información referente a este “rol”, por lo que debe representarse como una clave foránea, por otro lado tenemos la propia tabla “Rol” que por supuesto registra la información referente a cada rol y servirá como fuente informativa para la tabla Usuario por medio de su campo “id” representando la clave primaria. El caso de uso de este ejemplo será: Un usuario tiene ciertos atributos y puede poseer un rol en nuestro sitio.

La tabla Usuario, posee los siguientes registros:

rut	nombre	telefono	mail	género	fecha_ nacimiento	id_rol
1-9	Juan Soto	9999999	juan.soto@gmail.com	M	24-06-2000	1
2-7	Jorge Perez	8888888	jorge.perez@hotmail.com	M	12-03-1998	4
3-5	Sara Morales	7777777	sara.morales@gmail.com	F	11-02-1990	

Tabla 1. Estructura y registros de la tabla Usuario.

Mientras que la tabla Rol posee las siguientes filas:

id	nombre	descripcion
1	administrador	control total
2	marketing	editar y crear anuncios
3	editor	editar artículos
4	subscriptor	leer artículos

Tabla 2. Estructura y registros de la tabla Rol.

Anteriormente, hemos visto cómo consultar datos de una tabla, pero en lo cotidiano necesitaremos saber datos de más de una entidad a la vez. ¿Cómo hacemos esto?.

Pensemos, por ejemplo, queremos saber a qué se refiere el rol que posee el usuario con rut 1-9. En la práctica le pediríamos a nuestro modelo algo así como "muéstrame el nombre y la descripción del rol que tenga relación con el usuario de rut 1-9".

En SQL, haríamos lo siguiente:

```
SELECT rol.nombre, rol.descripcion
FROM rol
JOIN usuario ON rol.id=usuario.id_rol
WHERE usuario.id = '1-9';
```

Es decir estaríamos cumpliendo la siguiente sintaxis:

```
-- Seleccionamos las columnas desde la tabla1
SELECT columnas FROM tabla1
-- Posterior a la selección de la columna, indicamos que vamos a generar la unión
-- con la columna de la tabla2
JOIN tabla2 ON tabla1.columna=tabla2.columna
[WHERE condicion];
```

Esto unirá las columnas indicadas, pero sólo se mostrarán las filas que cumplan la condición dada. Existen distintas formas de combinar filas y se ve traducido en distintos tipos de JOIN que te presento a continuación:

- `INNER JOIN`: Une sólo las columnas comunes entre ambas tablas.
- `LEFT JOIN`: Une todas las columnas de la primera tabla con las columnas en común de la segunda tabla.
- `FULL OUTER JOIN`: Une todas las columnas de ambas tablas.

Veamos ejemplos de consultas usando los distintos tipos de JOIN.

INNER JOIN

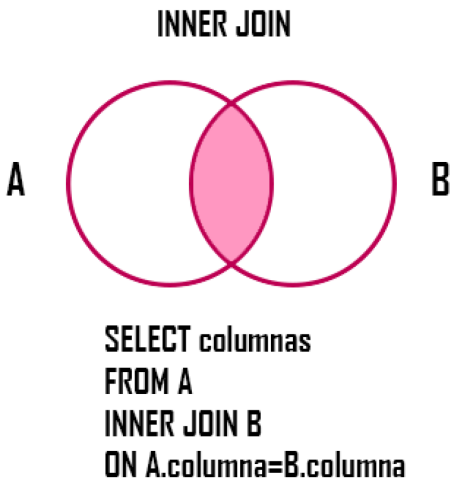

Imagen 2. Inner join.

Observa detenidamente la imagen 2, pues usaré este tipo de gráfico para los siguientes ejemplos. En este caso se hace referencia a este tipo de relación, que nos permite conocer aquellos elementos que tienen en conjunto una o más tablas, como vimos anteriormente en el caso del rol de nuestro usuario Juan Soto.

En nuestro ejemplo si hiciéramos un JOIN entre las tablas usuario y rol obtendremos sólo dos registros. ¿Te imaginas por qué? Ya lo sabrás, por mientras comprobémoslo con el siguiente código:

```
SELECT *
FROM usuario
INNER JOIN rol ON usuario.id_rol = rol.id;
```

El resultado sería la unión de los datos de ambas tablas, mostrando sólo aquellos datos que tienen en común la columna que utilizamos para relacionarlas y como solo existen 2 usuarios con roles asignados obtendremos la siguiente tabla.

rut	nombre	telefono	mail	genero	fecha_nacimiento	id_rol	id	nombre	descripcion
1-9	Juan Soto	9999999	juan.soto@gmail.com	M	24-06-2000	1	1	administrador	control total
2-7	Jorge Perez	8888888	jorge.perez@hotmail.com	M	12-03-1998	4	4	subscriptor	leer articulos

Tabla 3. Inner Join Usuario y Rol.

Tómate unos minutos para volver al pasado y ver cómo encajaría esta nueva herramienta, pensemos en los datos que cargamos en sesiones anteriores sobre los pokemones. Si quisiéramos saber cuáles son los pokemones que ya poseemos y la información detallada, podríamos consultar primero nuestra tabla `mis_pokemones` y luego consultar el detalle en la tabla `pokemones`. Para dar contexto, en la siguiente imagen te muestro las tablas tal y como fueron creadas en la unidad pasada e identificando la clave primaria con la siglas “PK” y la clave foránea con “FK”.

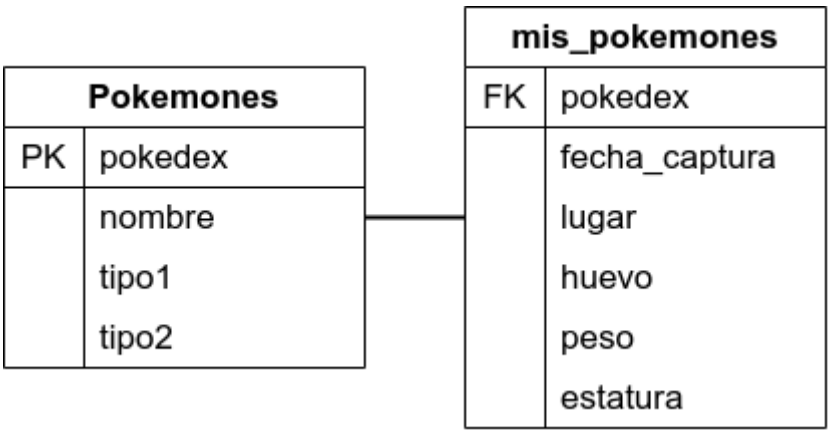

Imagen 3.Tablas creadas en la unidad anterior sobre pokemones.

Entonces `INNER JOIN` nos permite relacionar varias tablas y obtener aquellos elementos en común a través de una columna que las relacione (generalmente una llave primaria y una foránea), en el caso de los pokemones sería la columna pokedex.

Abre tu terminal de PostgreSQL y conéctate a la base de datos en donde tienes la tabla de los pokemones para realizar la siguiente consulta y ver qué información obtenemos al ejecutar lo siguiente:

```
SELECT *
-- selecciona todos los registros
FROM pokemones
-- de la tabla pokemones
INNER JOIN mis_pokemones
-- relacionada con la tabla mis_pokemones
ON pokemones.pokedex=mis_pokemones.pokedex
-- donde el campo pokedex de la tabla A, sea igual a de la tabla B
ORDER BY nombre;
-- ordenado por nombre
```

En este caso, se unen ambas tablas cuando se cumple el criterio que establecimos: que tengan el mismo número de pokedex. Es por eso que se muestran los 199 registros que tenemos en la tabla `mis_pokemones` y se añade como información los datos de nombre, tipo1 y tipo2 como te muestro en la tabla 4.

pokedex	nombre	tipo1	tipo2	pokedex	fecha_captura	lugar	huevo	peso	estatura
63	Abra	psiquico		63	2019-07-19	Huechuraba	f	13.4	0.7
63	Abra	psiquico		63	2019-06-01	Huechuraba	f	10.1	0.6
144	Articuno	hielo	volador	144	2019-07-13	Independencia	f	55.4	1.7
9	Blastoise	agua		9	2019-01-26	Independencia	f	180.9	1.8
9	Blastoise	agua		9	2019-02-16	Independencia	f	151	1.6
1	Bulbasaur	planta	veneno	1	2019-01-30	Estación Central	f	28.3	0.8
1	Bulbasaur	planta	veneno	1	2019-01-31	Independencia	f	20.9	0.5
12	Butterfree	bicho	volador	12	2019-03-04	Recoleta	f	27.8	1.1
10	Caterpie	bicho		10	2019-01-26	Independencia	f	7.1	0.2
10	Caterpie	bicho		10	2019-02-13	Cerro Navia	f	10.1	0.5
...									
41	Zubat	veneno	volador	41	2019-02-22	Santiago	f	7.6	0.5

(199 rows)

Tabla 4. Inner Join pokemones y mis_pokemones.

Ejercicio propuesto (1)

El programador de la base de datos de la empresa Mawashi Cars Spa ha logrado avanzar bastante en los requisitos levantados del software, pero luego de mostrarle los avances al cliente surgieron nuevos requerimientos que deben ser atendidos. En el apoyo lectura de esta sesión encontraras 2 archivos llamados autos.csv y ventas.csv para la solución de este ejercicio propuesto.

Para empezar con las soluciones debes crear las tablas correspondientes a los archivos .csv deduciendo el tipo de dato de cada columna, definiendo la llave primaria y foránea correspondiente en las tablas.

El dueño de la empresa ha solicitado que diariamente le hagan llegar reportes sobre sus clientes, marca y modelo del auto que compró, por lo que se deberá realizar una consulta que obtenga el nombre de los clientes registrados en la tabla venta junto con la marca y el modelo del auto asociado a la venta realizada.

LEFT JOIN

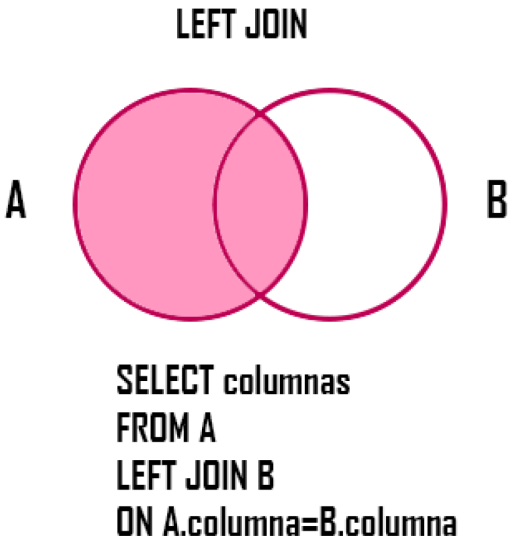

Imagen 4. Left Join.

En la imagen 4 se hace referencia a este tipo de unión, que nos permite obtener toda la información de una tabla y el conjunto de la relación con otra. La diferencia en el código radica en agregarle la palabra “INNER” por “LEFT” antes de “JOIN”.

Volviendo con nuestro ejemplo, aplica la siguiente consulta en tu terminal de postgreSQL.

```
SELECT *
FROM usuario
LEFT JOIN rol ON usuario.id_rol = rol.id;
```

Y deberás obtener la siguiente tabla.

rut	nombre	telefono	mail	gener	fecha_nacimiento	id_rol	id	nombre	descripcion
1-9	Juan Soto	9999999	juan.soto@gmail.com	M	24-06-2000	1	1	administrador	control total
2-7	Jorge Perez	8888888	jorge.perez@hotmail.com	M	12-03-1998	4	4	subscriptor	leer articulos
3-5	Sara Morales	7777777	sara.morales@gmail.com	F	11-02-1990				

Tabla 5. Left join usuario y rol.

Observando el resultado de los registros obtenidos en la tabla 5, ¿Por qué esta vez nos trajo los datos de Sara?. Eso es porque el `LEFT JOIN` muestra todos los registros de la tabla 1 (usuario) sin importar que no posean una relación con la segunda tabla y Sara no tenía

especificado el “id_rol” en su registro. Por el contrario, aquellos registros de la tabla 2 (rol) que no tengan relación, no los desplegamos.
Veamos otro ejemplo:

¿Qué ocurre si quiero obtener toda la información disponible de los pokemones, sin importar si lo he capturado o no?.

`LEFT JOIN` nos permite relacionar una o más tablas a través de una columna en común y obtener todos los elementos de la tabla `pokemones`, y los campos donde existe información de la segunda tabla. Si los registros no existen en la segunda tabla, se mostrará el campo vacío.

Obtengamos todos los pokemones de la siguiente manera:

```
SELECT *
-- selecciona todos los registros
FROM pokemones
-- de la tabla pokemones (tabla izquierda)
LEFT JOIN mis_pokemones
-- en relación a la tabla mis_pokemones (tabla derecha)
on pokemones.pokedex=mis_pokemones.pokedex
-- relacionados a través de la columna pokedex
ORDER BY nombre;
-- ordenados por nombre
```

Y obtendrás la siguiente tabla.

pokedex	nombre	tipo1	tipo2	pokedex	fecha_captura	lugar	huevo	peso	estatura
63	Abra	psiquico		63	2019-07-19	Huechuraba	f	13.4	0.7
63	Abra	psiquico		63	2019-06-01	Huechuraba	f	10.1	0.6
142	Aerodactyl	roca	volador						
65	Alakazam	psiquico							
24	Arbok	veneno							
59	Arcanine	fuego							
144	Articuno	hielo	volador	144	2019-07-13	Independencia	f	55.4	1.7
15	Beedrill	bicho	veneno						
69	Bellsprout	planta	veneno						
9	Blastoise	agua		9	2019-01-26	Independencia	f	180.9	1.8
9	Blastoise	agua		9	2019-02-16	Independencia	f	151	1.6
1	Bulbasaur	planta	veneno	1	2019-01-31	Independencia	f	20.9	0.5
1	Bulbasaur	planta	veneno	1	2019-01-30	Estación Central	f	28.3	0.8
12	Butterfree	bicho	volador	12	2019-03-04	Recoleta	f	27.8	1.1
...									

(242 rows)

Tabla 6. Left join pokemones y mis_pokemones.

En este caso podemos ver que a pesar que en `mis_pokemones` (tabla derecha) no existan Alakazam, Arbok, Arcanite, por ejemplo, nos muestra sus registros de la tabla `pokemones` (tabla izquierda).

Ejercicio propuesto (2)

Parece que hubo un problema en la creación del software de la empresa Mawashi Cars y está registrando ventas sin el dato del auto que se está vendiendo, por lo que necesitará obtener todas las filas de la tabla ventas indiferentemente si tienen o no un auto asociado para proceder con una auditoría.

Realizar la consulta SQL que obtenga todos los registros de la tabla Ventas con o sin autos asociados.

LEFT JOIN Donde una columna en la tabla “B” es null

Este tipo de unión nos permite obtener la información de los registros que no se encuentran en la relación, en la siguiente imagen te muestro esto gráficamente.

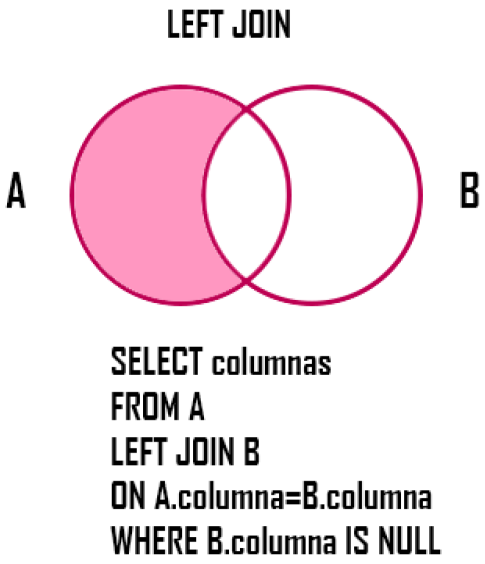

Imagen 5. Left join con columna NULL

Por ejemplo, nos gustaría saber aquellos roles que no han sido asignados a ningún usuario, entonces ejecutamos la siguiente consulta:

```
SELECT rol.*
FROM rol
LEFT JOIN usuario ON rol.id = usuario.id_rol
WHERE usuario.id_rol IS NULL;
```

Y obtendremos la siguiente tabla.

id	nombre	descripcion
2	marketing	editar y crear anuncios
3	editor	editar articulos

Tabla 7. Left join entre rol y usuario con columna NULL.

Nota que en la consulta luego de “SELECT” está escrito “rol.*” es para obtener solamente las columnas correspondientes a la tabla “Rol”.

Esto tiene mucha utilidad para obtener información y discriminar registros relevantes para nuestros análisis. Por ejemplo, es posible que nos interese saber qué pokemones aún no tenemos. Para eso usamos un `LEFT JOIN` con una condición `WHERE`, de la siguiente manera:

```
SELECT *
-- selecciona todos los registros
FROM pokemones
-- de la tabla pokemones
LEFT JOIN mis_pokemones
-- en relación a la tabla mis_pokemones
on pokemones.pokedex=mis_pokemones.pokedex
-- a través de la columna pokedex
WHERE mis_pokemones IS NULL
-- donde el registro no exista en la tabla mis_pokemones
ORDER BY nombre;
-- ordenados por nombre
```

Y deberás recibir la siguiente tabla:

pokedex	nombre	tipo1	tipo2	pokedex	fecha_captura	lugar	huevo	peso	estatura
142	Aerodactyl	roca	volador						
65	Alakazam	psiquico							
24	Arbok	veneno							
59	Arcanine	fuego							
15	Beedrill	bicho	veneno						
69	Bellsprout	planta	veneno						
91	Cloyster	agua	hielo						
104	Cubone	tierra							
132	Ditto	normal							
133	Eevee	normal							
83	Farfetchd	normal	volador						
74	Geodude	roca	tierra						
42	Golbat	veneno	volador						
118	Goldeen	agua							
...									
110	Weezing	veneno							

(43 rows)

Tabla 8. Left join entre pokemones y mis_pokemones con columna NULL.

Aca podemos ver que nos muestra los registros que corresponden solo a aquellos `pokemones` de la tabla pokemones que no existen en `mis_pokemones`.

Ten cuidado con la aplicación de estas consultas porque es común cuando se aprende esto confundirlo con el símbolo “=” al momento de definir la condición, no obstante no aplica en este tipo de consulta por lo que debes usar por sintaxis la palabra “**IS**”.

Ejercicio propuesto (3)

La empresa Mawashi Cars ha notado que aproximadamente un 30% de sus autos no se han vendido y está considerando la posibilidad de negociar un convenio con una compañía aliada con estos vehículos, pero necesitará una tabla que muestre los autos que no han sido vendidos.

Realizar la consulta necesaria para obtener todos los autos cuyos id no se encuentran en la tabla Ventas.

FULL (OUTER) JOIN

Este tipo de unión nos permite obtener todos los registros disponibles en las tablas que estamos relacionando, sin filtrar la información así como lo puedes inferir al ver la imagen 6.

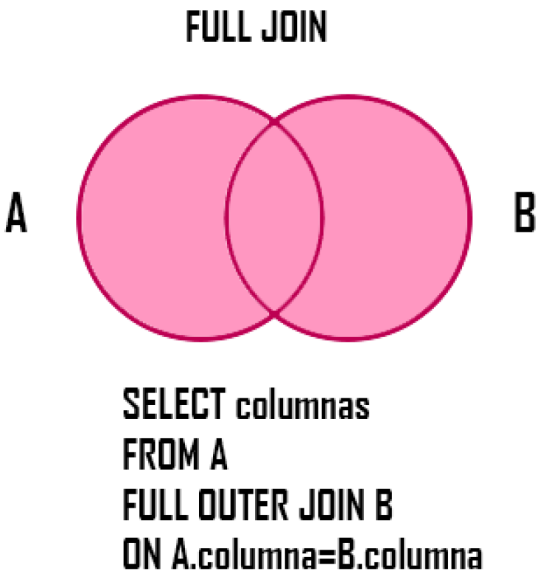

Imagen 6. Full outer join.

Aplicado en el ejemplo de usuarios y roles usamos la siguiente consulta.

```
SELECT *
FROM usuario
FULL OUTER JOIN rol
ON usuario.id_rol=rol.id;
```

Y obtendremos la tabla 9.

rut	nombre	telefono	mail	genero	fecha_nacimiento	id_rol	id	nombre	descripcion
1-9	Juan Soto	9999999	juan.soto@gmail.com	M	24-06-2000	1	1	administrador	control total
2-7	Jorge Perez	8888888	jorge.perez@hotmail.com	M	12-03-1998	4	4	subscriptor	leer articulos
3-5	Sara Morales	7777777	sara.morales@gmail.com	F	11-02-1990				
							2	marketing	editar y crear anuncios
							3	editor	editar articulos

Tabla 9. Full join entre usuario y rol.

En el caso de los pokemones, podríamos querer saber toda la información disponible, sin importar si poseemos o no el pokemon. Para eso usamos el `FULL OUTER JOIN` de la siguiente manera:

```
SELECT *
-- selecciona todos los registros
FROM pokemones
-- de la tabla pokemones
FULL OUTER JOIN  mis_pokemones
-- en relación con la tabla mis_pokemones
ON pokemones.pokedex=mis_pokemones.pokedex
-- a través de la columna pokedex
ORDER BY nombre;
-- ordenados por nombre
```

Y obtendremos la siguiente tabla.

pokedex	nombre	tipo1	tipo2	pokedex	fecha_captura	lugar	huevo	peso	estatura
63	Abra	psiquico		63	2019-07-19	Huechuraba	f	13.4	0.7
63	Abra	psiquico		63	2019-06-01	Huechuraba	f	10.1	0.6
142	Aerodactyl	roca	volador						
65	Alakazam	psiquico							
24	Arbok	veneno							
59	Arcanine	fuego							
144	Articuno	hielo	volador	144	2019-07-13	Independencia	f	55.4	1.7
15	Beedrill	bicho	veneno						
69	Bellsprout	planta	veneno						
9	Blastoise	agua		9	2019-01-26	Independencia	f	180.9	1.8
9	Blastoise	agua		9	2019-02-16	Independencia	f	151	1.6
1	Bulbasaur	planta	veneno	1	2019-01-31	Independencia	f	20.9	0.5
...									
41	Zubat	veneno	volador	41	2019-06-26	Peñalolén	f	15.8	1.1

(242 rows)

Tabla 10. Full join entre pokemones y mis_pokemones.

Para este caso, nos muestra todos los registros independientemente que no se encuentren en una tabla o la otra.

Ejercicio propuesto (4)

Las tablas entregadas al dueño de Mawashi Cars relacionadas con las ventas que no tienen el dato del id de los autos y los autos que aún no han traído nuevas curiosidades, ahora requiere que se le entregue todos los registros en una misma tabla para realizar la auditoría más cómodamente, es decir, se necesitan todos los registros de las tablas “ventas” y “autos”.

Realizar la sentencia SQL necesaria para satisfacer este requerimiento.

FULL OUTER JOIN Donde una columna en la tabla “A” o “B” es null

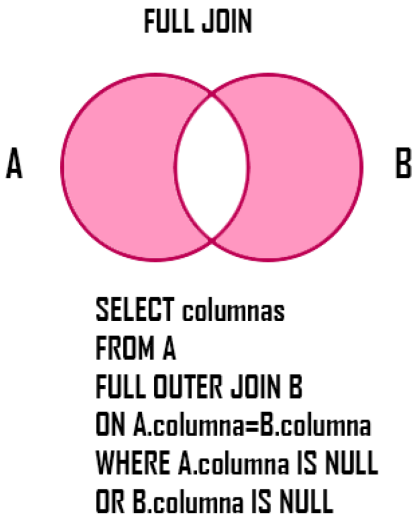

Imagen 7. Full outer join.

Se puede inferir por lo que se muestra en la imagen 7 que `FULL OUTER JOIN` nos permite conocer aquellos registros que no tienen en común las tablas que estamos relacionando.

Aplicado en el ejemplo de usuarios y roles haríamos la siguiente consulta.

```
SELECT *
FROM usuario
FULL OUTER JOIN rol
ON usuario.id_rol=rol.id
WHERE usuario.id_rol is null OR rol.id is null;
```

Y obtendremos la siguiente tabla.

rut	nombre	telefono	mail	genero	fecha_nacimiento	id_rol	id	nombre	descripcion
3-5	Sara Morales	7777777	sara.morales@gmail.com	F	11-02-1990				
							2	marketing	Editar y crear anuncios
							3	editor	editar articulos

Tabla 11. Full outer join entre usuario y rol.

Y en el ejemplo de los pokemones, la consulta se realizaría de la siguiente manera:

```
SELECT *
-- selecciona todos los registros
FROM pokemones
-- de la tabla pokemones
FULL OUTER JOIN  mis_pokemones on
pokemones.pokedex=mis_pokemones.pokedex
-- en relación con la tabla mis_pokemones a través de la columna pokedex
WHERE pokemones.pokedex is null or mis_pokemones.pokedex is null
-- donde la columna tipo2 es una o la columna pokedex es nula
ORDER BY nombre;
-- ordenados por nombre
```

Para obtener la tabla 12.

pokedex	nombre	tipo1	tipo2	pokedex	fecha_captura	lugar	huevo	peso	estatura
142	Aerodactyl	roca	volador						
65	Alakazam	psiquico							
24	Arbok	veneno							
59	Arcanine	fuego							
15	Beedrill	bicho	veneno						
69	Bellsprout	planta	veneno						
....									
110	Weezing	veneno							

(43 rows)

Tabla 12. Full outer join entre pokemones y mis_pokemones.

En este caso, nos muestra los 43 registros que no tienen en común ambas tablas.

Ejercicio propuesto (5)

El dueño de la empresa Mawashi Cars se dio cuenta que para facilitar el proceso de auditoría, sería beneficioso saber los registros que no tienen relación entre ambas tablas para hacer el cruce con la información cedida anteriormente y terminar la auditoría.

Realizar la sentencia SQL necesaria para satisfacer este requerimiento.

Subquery

Una subquery (o consulta interna) es una query implementada dentro de otra query principal de SQL, la cual debe enmarcarse dentro de la cláusula `WHERE`.

Una de las principales aplicaciones de las subqueries, es para retornar datos que serán utilizados posteriormente en una consulta principal, de esta manera funcionan como una condición para restringir los datos. Es decir, las subconsultas son consultas temporales que sólo estarán para poder complementar el resultado de nuestra consulta principal.

Operador WHERE

Por ejemplo, queremos obtener como resultado los nombres de los pokemones que fueron obtenidos por huevos.

Con la siguiente consulta, podemos obtener el número de la pokedex de los pokemones que están en la tabla `mis_pokemones` pero no sus nombres.

```
SELECT pokedex
FROM mis_pokemones
WHERE huevo=true;
```

Y obtendremos la siguiente tabla.

pokedex
7
11
13
16
19
23
33
48
58
62
70
93
99
105

(14 rows)

Tabla 13. Pokemones que fueron obtenidos por huevo.

Para obtener los nombres, debemos utilizar el operador `WHERE` de la siguiente manera:

```
SELECT columna1,columna2,...
FROM nombre_tabla
WHERE columna1 IN
   (SELECT columna1
   FROM nombre_tabla2
   WHERE condicion);
```

Por lo que en nuestro ejemplo la consulta quedaría de la siguiente forma:

```
SELECT pokedex, nombre
FROM pokemones
WHERE pokedex IN
   (SELECT pokedex
   FROM mis_pokemones
   WHERE huevo= true
  );
```

Donde la subquery es equivalente a la que habíamos realizado inicialmente. Así, obtenemos el número de la pokedex y el nombre de los pokemones que fueron eclosionados.

Obtendremos entonces la siguiente tabla.

pokedex	nombre
7	Squirtle
11	Metapod
13	Weedle
16	Pidgey
19	Rattata
23	Ekans
33	Nidorino
48	Venonat
58	Growlithe
62	Poliwrath
70	Weepinbell
93	Haunter
99	Kingler
105	Marowak

(14 rows)

Tabla 14. Subquery.

Ejercicio propuesto (6)

Se ha reconocido de parte de la empresa Mawashi Cars la buena labor del programador de base de datos y ahora solicitan información aún más personalizada. En esta ocasión requieren una tabla que muestre solo los autos que han sido comprados por los clientes con débito como método de pago.

Realizar una consulta que devuelva los registros de la tabla Autos que tengan una relación con la tabla Ventas y cuyo método de pago fue débito.

Operador FROM

En este caso, la subquery obtiene como resultado una subtabla temporal, sobre la cual se hará una nueva consulta. La sintaxis está dada de la siguiente manera.

```
SELECT x.columa1, xcoluma2, ...
FROM (
   SELECT columa1, columna2, ...
   FROM nombre_tabla2
   WHERE condicion
 ) as x
INNER JOIN nombre_tabla1 as y on x.columna1 = y.columna1;
```

Si queremos obtener los nombres de los pokemones capturados que tienen un peso mayor a 200 kilos, debemos hacer la siguiente consulta:

```
SELECT y.nombre, x.pokedex, x.peso
FROM (
   SELECT peso, pokedex
   FROM mis_pokemones
   WHERE peso>200
) AS x
INNER JOIN pokemones AS y ON x.pokedex = y.pokedex;
```

Y obtendremos la siguiente tabla.

nombre	pokedex	peso
Golem	76	305.8
Golem	76	320.1
Onix	95	240.8
Lapras	131	252.8
Snorlax	143	460.1
Dragonite	149	288.9
Dragonite	149	279.5
Dragonite	149	290.1

Tabla 15. Subquery de los pokemones con peso mayor a 200 kilos.

Donde la subquery:

```
SELECT peso, pokedex
FROM mis_pokemones
WHERE peso>200
```

Nos entregará el peso y el número de la pokedex de aquellos pokemones que cumplen la condición de superar los 200 kilos.

Ejercicio propuesto (7)

La empresa Mawashi Cars ha notado que sus ventas han mejorado y le gustaría saber el detalle de los autos que se han vendido a un precio mayor de $1.500.000 para hacerles próximamente una promoción de descuentos y mejorar aún más sus ventas.

Realizar la sentencia SQL que resuelva esta necesidad.

Reglas de las subqueries:

La siguiente lista muestra algunas reglas que te recomiendo sigas para realizar subquerys en tus consultas:

- Las consultas internas deben estar encapsuladas entre paréntesis.
- Una subquery puede tener sólo una columna especificada en SELECT, con la excepción de múltiples columnas definidas en la consulta principal.
- El comando ORDER BY no se puede utilizar en una consulta interna. La excepción es que esta instrucción sí puede ser incluida en la consulta principal.
- Para obtener un resultado similar a ORDER BY dentro de una consulta interna, se puede implementar el comando GROUP BY.
- Aquellas consultas internas que retornen más de una fila sólo pueden ser utilizadas con operadores de múltiples valores como IN.

Solución de los ejercicios propuestos

1. El programador de la base de datos de la empresa Mawashi Cars Spa ha logrado avanzar bastante en los requisitos levantados del software, pero luego de mostrarle los avances al cliente surgieron nuevos requerimientos que deben ser atendidos. En el apoyo lectura de esta sesión encontraras 2 archivos llamados autos.csv y ventas.csv para la solución de este ejercicio propuesto.

   Para empezar con las soluciones debes crear las tablas correspondientes a los archivos .csv deduciendo el tipo de dato de cada columna, definiendo la llave primaria y foránea correspondiente en las tablas.

   El dueño de la empresa ha solicitado que diariamente le hagan llegar reportes sobre sus clientes, marca y modelo del auto que compró, por lo que se deberá realizar una consulta que obtenga el nombre de los clientes registrados en la tabla venta junto con la marca y el modelo del auto asociado a la venta realizada.

```
CREATE TABLE autos(
id INT,
marca VARCHAR(25),
modelo VARCHAR(25),
año VARCHAR(4),
color VARCHAR(25),
precio FLOAT,
PRIMARY KEY(id)
);
```

```
CREATE TABLE ventas(
fecha VARCHAR(20),
id_auto INT,
cliente VARCHAR(25),
referencia INT,
cantidad FLOAT,
metodo_pago VARCHAR(10),
FOREIGN KEY (id_auto) REFERENCES
autos(id)
);
```

```
SELECT cliente, marca, modelo FROM ventas INNER JOIN autos ON
ventas.id_auto=autos.id;
```

2. Parece que hubo un problema en la creación del software de la empresa Mawashi Cars y está registrando ventas sin el dato del auto que se está vendiendo, por lo que necesitará obtener todas las filas de la tabla ventas indiferentemente si tienen o no un auto asociado para proceder con una auditoría.

   Realizar la consulta SQL que obtenga todos los registros de la tabla Ventas con o sin autos asociados.

```
SELECT cliente, marca, modelo FROM ventas LEFT JOIN autos ON
ventas.id_auto=autos.id;
```

3. La empresa Mawashi Cars ha notado que aproximadamente un 30% de sus autos no se han vendido y está considerando la posibilidad de negociar un convenio con una compañía aliada con estos vehículos, pero necesitará una tabla que muestre los autos que no han sido vendidos.

   Realizar la consulta necesaria para obtener todos los autos cuyos id no se encuentran en la tabla Ventas.

```
SELECT autos.* FROM autos LEFT JOIN ventas on autos.id=ventas.id_auto
WHERE ventas IS NULL;
```

4. Las tablas entregadas al dueño de Mawashi Cars relacionadas con las ventas que no tienen el dato del id de los autos y los autos que aún no han traído nuevas curiosidades, ahora requiere que se le entregue todos los registros en una misma tabla para realizar la auditoría más cómodamente, es decir, se necesitan todos los registros de las tablas “ventas” y “autos”.

   Realizar la sentencia SQL necesaria para satisfacer este requerimiento.

```
SELECT * FROM autos FULL OUTER JOIN ventas ON autos.id=ventas.id_auto;
```

5. El dueño de la empresa Mawashi Cars se dio cuenta que para facilitar el proceso de auditoría, sería beneficioso saber los registros que no tienen relación entre ambas tablas para hacer el cruce con la información cedida anteriormente y terminar la auditoría.

   Realizar la sentencia SQL necesaria para satisfacer este requerimiento.

```
SELECT * FROM autos FULL OUTER JOIN ventas on ventas.id_auto=autos.id
WHERE autos.id is null or ventas.id_auto is null;
```

6. Se ha reconocido de parte de la empresa Mawashi Cars la buena labor del programador de base de datos y ahora solicitan información aún más personalizada. En esta ocasión requieren una tabla que muestre solo los autos que han sido comprados por los clientes con débito como método de pago.

   Realizar una consulta que devuelva los registros de la tabla Autos que tengan una relación con la tabla Ventas y cuyo método de pago fue débito.

```
SELECT * FROM autos WHERE id IN (SELECT id_auto FROM ventas WHERE
metodo_pago='debito');
```

7. La empresa Mawashi Cars ha notado que sus ventas han mejorado y le gustaría saber el detalle de los autos que se han vendido a un precio mayor de $1.500.000 para hacerles próximamente una promoción de descuentos y mejorar aún más sus ventas.

   Realizar la sentencia SQL que resuelva esta necesidad.

```
SELECT y.*
FROM (
   SELECT *
   FROM ventas
   WHERE cantidad>1500000
) AS x
INNER JOIN autos AS y ON x.id_auto = y.id;
```