UNIDAD 5. Consultas (de SQL) que envuelven más de una Relación.

La potencia de un lenguaje de consulta estándar como SQL proviene de la habilidad de combinar más de dos relación a través de operaciones de juntar (join), unir, intersectar, calcular la diferencia o el producto cartesiano. Podemos usar cualquiera de estas cinco operaciones en SQL.

5.1 Productos y Joins en SQL

SQL tiene una manera simple de acoplar relaciones en una consulta: se listan cada una de las relaciones en la clásula FROM. Luego, las cláusulas SELECT y WHERE pueden referirse a los atributos de cualquiera de las relaciones especificadas en la cláusula FROM.

Ejemplo:
Supongamos que queremos obtener el nombre del (de los) Autor(es) del libro de "A First Course in Database Systems". Las tablas involucradas son:

> LIBROS (ISBN, Nombre, Genero, Estado_de_cons, Sig_Top, Nro_Inventario, Editorial, Año_de_Edición, cant_de_pag)
> ESCRITO_POR (Libro_Nombre, Nombre_Autor, Apellido_Autor)

Aunque podríamos consultar directamente la tabla ESCRITO_POR de la siguiente forma:

```
SELECT *
FROM escrito_por
WHERE Libro_Nombre = 'A First Course in Database Systems'
```

Solo tendríamos como resultado el nombre del libro y sus autores. Esto es:

Libro_Nombre	Nombre_Autor	Apellido_Autor
A First Course in Database Systems'	Jennifer	Widom
A First Course in Database Systems'	Jeffrey D.	Ullman

Como podemos dar un resultado similar pero agregando información sobre el libro? Tal como ISBN, Genero, etc.? Necesitamos juntar ambas tablas, entonces consultamos de la siguiente forma:

```
SELECT Libro_Nombre, Nombre_Autor, Apellido_Autor, ISBN, Genero
FROM escrito_por , Libro
WHERE Libro_Nombre = 'A First Course in Database Systems'
    AND  Nombre = Libro_Nombre
```

Esta consulta nos permite considerar todos los pares de tuplas, una de LIBROS y otra de ESCRITO_POR. Las condiciones que deben reunir estos pares se especifica en la cláusula WHERE:

1. El atributo Libro_Nombre de la tupla de ESCRITO_POR debe tener el valor 'A First Course in Database Systems'.
2. El atributo Libro de la tabla LIBROS debe tener el mismo nombre de libro que el atributo Libro_Nombre de la tupla Escrito_Por. Esto es, estas dos tuplas deben referirse al mismo Libro.

Siempre que encontremos un par de tuplas satisfaciendo ambas condiciones, producimos los atributos Libro_Nombre, Nombre_Autor, Apellido_Autor de la tupla de ESCRITO_POR, y los atributos ISBN, Genero de la tupla de LIBROS. El resultado es:

Libro_Nombre	Nombre_Autor	Apellido_Autor	ISBN	Género
A First Course in Database Systems'	Jennifer	Widom	0-13-861337-0	Técnico
A First Course in Database Systems'	Jeffrey D.	Ullman	0-13-861337-0	Técnico

Relación con el Modelo Relacional

La sentencia JOIN en SQL permite combinar registros de dos o más tablas en una base de datos relacional. Siempre es importante tener en cuenta que relaciones existen entre las tablas con las cuales efectuamos la operación de JOIN. Ya que recordemos que las relaciones entre las tablas pueden ser 1-1, 1-muchos ó muchos-muchos.

Relación Uno a Uno

Supongamos que contamos con las siguientes tablas en nuestra Base de Datos

Tabla Empleado

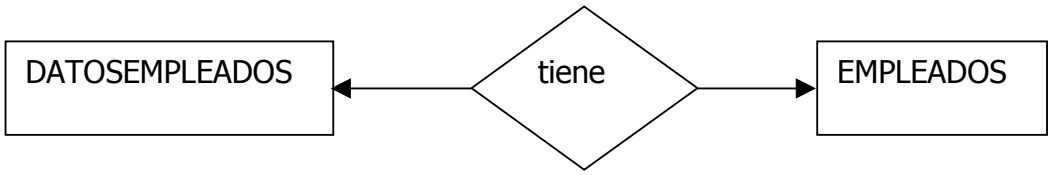
IDEmpleado		Apellido
1		Rafferty
2		Jordán
3		Steinberg
4		Róbinson

Tabla Datos Empleado

IDEmpleado	NombreCompleto	Teléfono	Dirección
1	Rafferty, Analia	4491236	Ba. As 123
2	Jordán, Luis	3446785	Santa Fe 56
3	Steinberg, Pablo	4431547	Av. Argentina 899
4	Róbinson, Sandra	4431547	San Luis 45

Las flechas de colores resaltan la relación entre las tablas, claramente se ve que cada hace referencia a una, y solo una tupla en la tabla de datos de los empleados.

Gráficamente se ve en el modelo ER así:



Relación Muchos a Uno

Supongamos que contamos con las siguientes tablas en nuestra Base de Datos

Tabla Departamento

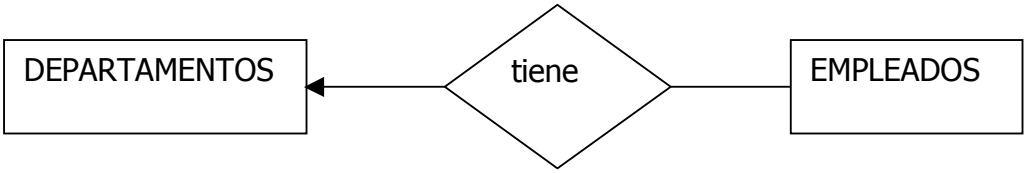
NombreDepartamento	IDDepartamento
Ventas	31
Ingenireria	33
Producción	34
Marketing	35

Tabla Empleado

IDEmpleado	Apellido	IDDepartamento
1	Rafferty	31
2	Jordán	33
3	Steinberg	33
4	Róbinson	33
5	Smith	34
6	Gaspar	36

Las flechas de colores resaltan la relación entre las tablas, claramente se ve que cada empleado pertenece a un departamento y solo a uno, mientras que un departamento puede tener varios empleados.

Gráficamente se ve en el modelo ER así:



Relación Unos a Muchos

Supongamos que contamos con las siguientes tablas en nuestra Base de Datos

Tabla DUENOS

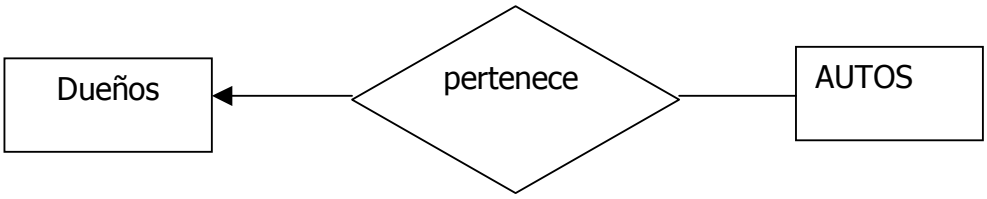
IDDueño	NombreDueño
1	Lopez, Juan
2	Sanchez, Sofia
3	Martinez, Sonia
4	Ruiz, Pablo

Tabla AUTOS

IDDueño	IdAuto	Marca	Patente
1	1	Mercedez	CDL 456
1	2	Fiat	AWS 785
2	3	Fiat	CVB 457
3	4	Renault	JKI 496
3	5	Renault	MJR 346
4	6	Peugeot	KSW 486

Las flechas de colores resaltan la relación entre las tablas, claramente se ve que cada dueño tiene uno o mas autos, y que un auto pertenece a un único dueño

Gráficamente se ve en el modelos ER así:



Relación Muchos a Muchos

Supongamos que contamos con las siguientes tablas en nuestra Base de Datos

Tabla Persona

IDPersona	Nombre
1	Lopez, Juan
2	Sanchez, Sofia
3	Martinez, Sonia
4	Ruiz, Pablo
5	Lopez, Susana
6	Sosa, Raul

Tabla PersonaDirección

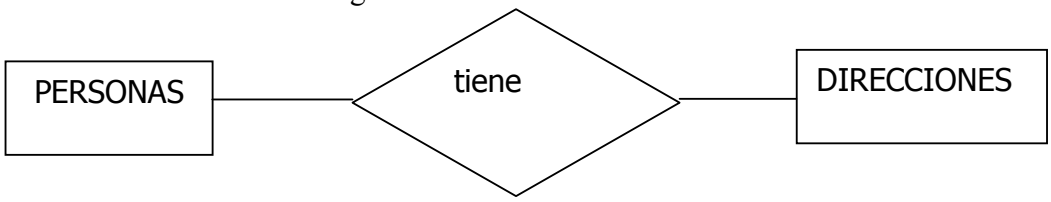
IDPersona	IdDireccion
1	1
1	3
2	2
2	5
3	4
4	4
5	6
6	7

Tabla Direcciones

Calle	Número	IDDireccion
Santa Fe	256	1
Entre Rios	96	2
San Luis	365	3
Av. Del Trabajador	123	4
Jujuy	47	5
Bouquet Roldan	667	6
Salta	456	7

Las flechas de colores resaltan la relación entre las tablas, claramente se ve que cada Persona puede tener mas de una dirección (■) , y que en una dirección pueden vivir mas de una persona (■)

Gráficamente se ve en el diagrama ER así:



5.2 Ambigüedad en el nombre de Atributos.

A veces realizamos consultas envolviendo diversas relaciones, y entre estas relaciones existen dos o más atributos con el mismo nombre. Si es así, es necesaria una manera de indicar a cuales de estos atributos nos referimos. SQL nos permite solucionar este problema permitiéndonos ubicar el nombre de la relación y un punto en frente de un atributo. Así R.A refiere al atributo A de la relación R.

Ejemplo:
Supongamos ahora, que queremos obtener el nombre, el apellido y la fecha de Nacimiento de los autores del Libro 'A First Course in Database Systems'. Las tablas involucradas son:

AUTORES (Nombre_Autor, Apellido_Autor, Fecha_de_Nacimiento)
ESCRITO_POR (Libro_Nombre, Nombre_Autor, Apellido_Autor)

La consulta siguiente:

```
SELECT Nombre_Autor, Apellido_Autor, Fecha_de_Nacimiento
FROM Autores, Escrito_Por
WHERE Libro_Nombre = 'A First Course in Database Systems' AND
Nombre_Autor = Nombre_Autor AND Apellido_Autor = Apellido_Autor;
```

Esta consulta presenta problemas de ambigüedad, pues no sabemos si Nombre_Autor refiere al atributo de Autores o de Escrito_Por. Una consulta correcta utilizando la notación R.A es:

```
SELECT Autores.Nombre_Autor, Autores.Apellido_Autor, Fecha_de_Nacimiento
FROM Autores, Escrito_Por
WHERE Libro_Nombre = 'A First Course in Database Systems' AND
        Escrito_Por.Nombre_Autor = Autores.Nombre_Autor AND
        Escrito_Por.Apellido_Autor = Autores.Apellido_Autor;
```

Variables de Tuplas.
Desambiguar los atributos agregando el prefijo de la relación es útil, tal vez resulta un tanto largo escribir el nombre de la relación antes de cada atributo que pueda ser ambiguo. Sin embargo, a veces necesitamos realizar una consulta que envuelve dos o

más tuplas de la misma relación. Podemos listar una relación R tantas veces como lo necesitemos en la cláusula FROM, pero necesitamos una manera de referirnos a cada ocurrencia de R. SQL nos permite definir, para cada ocurrencia de R en la cláusula FROM, un "alias" a la cual nos referiremos como *variable de tupla*. Cada uso de R en la cláusula FROM es seguida por la palabra reservada (AS) y el nombre de la tupla.

En las cláusulas SELECT y WHERE, podemos desambiguar atributos de R precediéndolos por la variable de tupla apropiada y un punto. De este modo, la variable de tupla sirve como otro nombre para la relación R y puede ser usado este nuevo nombre de R en los lugares donde R puede ser usada.

Ejercicio:
Es posible rescribir la última consulta del ejercicio 5.2.2. utilizando variables de tuplas.

```
SELECT A.Nombre_Autor, A.Apellido_Autor, A.Fecha_de_Nacimiento
FROM Autores AS A, Escrito_Por As E
WHERE E.Libro_Nombre = 'A First Course in Database Systems' AND
        E.Nombre_Autor = A.Nombre_Autor
        AND E.Apellido_Autor = A.Apellido_Autor;
```

Un ejemplo de que a veces es necesario referirnos a dos ocurrencias de una misma tabla esta mostrado en el siguiente ejemplo:

Ejercicio:
Supongamos que queremos recordar el nombre de dos hermanos (un varón y una mujer) que son socios de la biblioteca. ¨Podemos aproximarnos a la respuesta listando aquellos pares de socios que tienen el mismo apellido, que viven en la misma casa, y que tienen distinto sexo.
Utilizaremos para esta consulta la tabla SOCIOS:
SOCIOS (DNI, Nombre, Apellido, Nro_Socio, Profesión, Nombre_Calle, Nro_Calle, Sexo)

```
SELECT  S1.Apellido, S1.Nombre, S2.Apellido, S2.Nombre
FROM  Socios S1, Socios S2
WHERE S1.Apellido = S2.Apellido AND
    S1.Nombre_Calle = S2.Nombre_Calle AND
    S1.Nro_Calle = S2.Nro_Calle AND
    S1.Sexo <> S2.Sexo
```

5.3 Unión, Intersección y Diferencias de Consultas.

SQL provee estos operadores que se pueden aplicar a los resultados de consultas, siempre y cuando los resultados de las consultas tengan el mismo conjunto de atributos. Las palabras claves que usaremos son UNION, INTERSECT, y EXCEPT para unión, intersección y diferencia, respectivamente.

Ejemplo:
Supongamos que queremos obtener aquellas autores de libros de genero femenino que también son socias de la biblioteca.
Utilizaremos dos tablas:

AUTORES (Nombre_Autor, Apellido_Autor, Fecha_de_Nacimiento, Sexo)
SOCIOS (DNI, Nombre, Apellido, Nro_Socio, Profesión, Nombre_Calle, Nro_Calle, Sexo)

```
(SELECT Nombre_Autor,  Apellido_Autor
FROM Autores
WHERE  Sexo = "F")
            INTERSECT
(SELECT Nombre,  Apellido
FROM Socios
WHERE  Sexo = "F")
```

Ejemplo:
Similarmente, podemos obtener la diferencia de estos dos conjuntos de personas, que arroja como resultado los autores de libros de genero femenino que no son socias de la biblioteca.

```
(SELECT Nombre_Autor,  Apellido_Autor
FROM Autores
WHERE  Sexo = "F")
            EXCEPT
(SELECT Nombre,  Apellido
FROM Socios
WHERE  Sexo = "F")
```

5.4 Subconsultas.

En esta sección ampliaremos las expresiones que pueden aparecer en las clásulas WHERE. Previamente, diremos que condiciones pueden comparar valores escalares (valores simples como enteros, reales, strings, o fechas, o expresiones que representan estos valores) .
Aprenderemos como usar una *subconsulta* (subquery) en una condición. Una subconsulta es una expresión que evalúa a una relación. Por ejemplo, una expresión Select-From-Where puede ser una subconsulta. También se analizan en esta sección algunos operadores de SQL para comparar tuplas y relaciones dentro de una clásula WHERE.

Subconsultas que producen un valor escalar.

Es posible usar un consulta select-from-where como si parte de una expresion, encerrarla entre paréntesis y utilizarla como si fuera una constante. En particular, esta expresión puede aparecer en una cláusula WHERE en cualquier lugar donde podemos esperar una constante o un atributo representando una componente de una tupla. Por ejemplo, podriamos comparar el resultado de tal subconsulta con una constante.

Ejemplo 5.3.1.1.
Una consulta similar al ejercicio 5.2.1. en el cual obtuvimos el nombre y apellido de los autores del libro 'A First Course in Database Systems', y en el resultado devolvíamos también ademas del nombre del libro, el isbn y el género, es la siguiente consulta.

```
SELECT Nombre_Autor, Apellido_Autor
FROM escrito_por , Libro
WHERE Libro_Nombre = 'A First Course in Database Systems'
    AND  Nombre = Libro_Nombre
```

En la cual solamente damos como resultado el nombre y apellido de los autores.

Es posible escribir esta consulta utilizando una subconsulta:

```
SELECT Nombre_Autor, Apellido_Autor
FROM    Escrito_por
WHERE Libro_Nombre =  (SELECT Nombre
                       FROM LIBROS
                       WHERE Nombre = 'A First Course in Database Systems');
```

Compare ambas consultas, los resultados de ambas son los mismos pero una utiliza un join entre las dos tables y la segunda compara el resultado de una subconsulta (la subconsulta es SELECT Nombre FROM LIBROS WHERE Nombre = 'A First Course in Database Systems') con un atributo de la tabla Escrito_Por.

Ejemplo.
Podemos usar una subconsulta para obtener los teléfonos del Socio "Juan Perez". Las tablas que utilizaremos en la consulta son:

SOCIOS (DNI, Nombre, Apellido, Nro_Socio, Profesión, Nombre_Calle, Nro_Calle, Sexo)
TELEFONOS_SOCIOS (Nro_Socio, teléfono)

```
SELECT Telefono
FROM Telefonos_Socios
WHERE Nro_Socio = (SELECT Nro_Socio
                   WHERE  Apellido = "Perez" AND Nombre = "Juan");
```