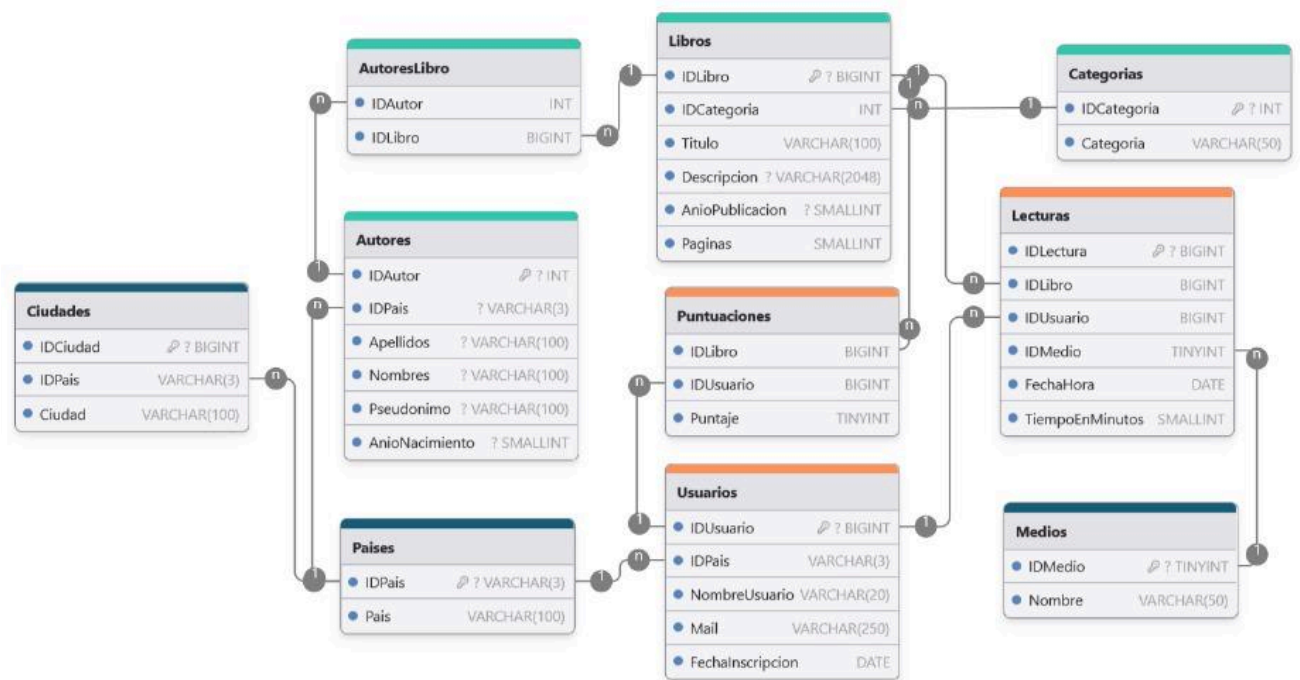


Consultas de selección - JOINS y UNION

NOTA: en este apunte trabajaremos nuevamente con la base de datos Librería, cuyo DER es el siguiente:



Hasta el momento, en nuestras prácticas, sólo obtuvimos datos provenientes de una sola tabla de nuestra base de datos. Sin embargo, lo más habitual es necesitar datos de dos o más tablas para elaborar listados más descriptivos. Por ejemplo, es usual querer obtener la descripción principal de un campo a través de su ID. Observemos la tabla Categorías:

	IDCategoria ↕	Categoria ↕
1	1	Novela
2	2	Terror
3	3	Fantasía
4	4	Historia
5	5	Ciencia Ficción

Si, a partir de estos datos, tuviéramos que elaborar un reporte de libros que incluya sus títulos, años de publicación, cantidad de páginas y categorías, la aparición de un registro con valor de categoría igual a 1 (ID de categoría “Novela”) no le va a representar nada a la persona que lo lea, y hasta es probable que aporte confusión. Pero, si en cambio, mostramos la descripción de la categoría, el reporte será mucho más legible y provechoso.

Dejemos por un momento este ejemplo y tratemos de obtener un listado con apellidos, nombres y país de origen de cada uno de los autores (con país de origen nos referimos al nombre del país y no al ID que posee en nuestra tabla Países).

Como podemos visualizar en el DER y en la siguiente consulta, los apellidos y nombres son datos que podemos extraer directamente de la tabla Autores:

```
SELECT * FROM Autores;
```

	IDAutor ↕	IDPais ↕	Apellidos ↕	Nombres ↕	Pseudonimo ↕	AñoNacimiento ↕
1	1	JPN	Murakami	Haruki	NULL	1949
2	2	GBR	NULL	NULL	Bitmap Books	NULL
3	3	NULL	King	Stephen	Richard Bachman	1947
4	4	USA	Crichton	Michael	NULL	1942
5	5	JPN	Kawamura	Genki	NULL	1979

Pero, para obtener el nombre del país, necesitamos incorporar la tabla Países a nuestra consulta, ya que Autores contiene (si no es nulo) el ID del país de origen del autor pero no su nombre. Además, también por el DER sabemos que existe integridad referencial entre las tablas Autores y Países. En el tercer registro vemos que IDPais es NULL, y esto es correcto ya que en este esquema de base de datos hemos definido que dicho campo puede aceptar valores nulos. Pero, en caso de tener un valor, obligatoriamente debe existir en la tabla Países.

Volviendo al listado que necesitamos, recordemos que debe tener la misma cantidad de filas que la tabla Autores, pero en lugar de IDPais necesitamos los valores del campo Pais.

Si la tabla Paises tiene los siguientes registros:

	IDPais ↑↓🔍	Pais ↑↓🔍
1	GBR	Reino Unido
2	JPN	Japón
3	USA	Estados Unidos

Veamos lo que sucede si incluimos a la tabla Paises en la consulta:

```
SELECT Autores.Apellidos, Autores.Nombres, Paises.Pais
FROM Autores, Paises;
```

	Apellidos ↑↓🔍	Nombres ↑↓🔍	Pais ↑↓🔍
1	Murakami	Haruki	Reino Unido
2	NULL	NULL	Reino Unido
3	King	Stephen	Reino Unido
4	Crichton	Michael	Reino Unido
5	Kawamura	Genki	Reino Unido
6	Murakami	Haruki	Japón
7	NULL	NULL	Japón
8	King	Stephen	Japón
9	Crichton	Michael	Japón
10	Kawamura	Genki	Japón
11	Murakami	Haruki	Estados Unidos
12	NULL	NULL	Estados Unidos
13	King	Stephen	Estados Unidos
14	Crichton	Michael	Estados Unidos
15	Kawamura	Genki	Estados Unidos

En lugar de obtener el listado de apellidos y nombres de autores con el nombre de su país de origen, obtuvimos el producto entre los registros de ambas tablas. Es decir, cada registro




de autores aparece combinado con cada uno de los registros de países. Es así que la cantidad de registros resultante es la cantidad de registros de autores multiplicado por la cantidad de filas de países. Este conjunto de resultados claramente es incorrecto respecto de lo que queríamos obtener.

El problema ocurre por el uso de más de una tabla dentro de la cláusula FROM, ya que no especificamos que queremos todos los registros de autores pero solamente un registro de países (el que se relaciona con cada registro de autor en particular).

Para lograr el resultado deseado sería necesario incluir una condición dentro del WHERE, basada en la relación existente entre la clave foránea de Autores (columna IDPais) y la clave primaria de la tabla Países (columna IDPais).

La consulta queda formada de la siguiente manera:

```
SELECT A.Apellidos, A.Nombres, P.Pais
FROM Autores AS A, Países AS P
WHERE A.IDPais = P.IDPais;
```

	Apellidos 	Nombres 	Pais 
1	Murakami	Haruki	Japón
2	NULL	NULL	Reino Unido
3	Crichton	Michael	Estados Unidos
4	Kawamura	Genki	Japón

Aquí podemos notar un par de cuestiones. La primera, es el uso de **alias** para las tablas, como habíamos adelantado en el tema anterior. En este ejemplo la tabla Autores se representa con el alias **A** mientras que la tabla de países se representa con el alias **P**.

Lo segundo es que, a diferencia de la consulta anterior (donde no existía la condición en el WHERE y aparecían resultados indeseados) obtuvimos los registros tal y como los necesitamos. Esto se debe a que, a cada fila, se le aplica la condición del WHERE: esta condición indica que el campo IDPais de la tabla Autores debe tener el mismo valor que en el campo IDPais de la tabla Países.

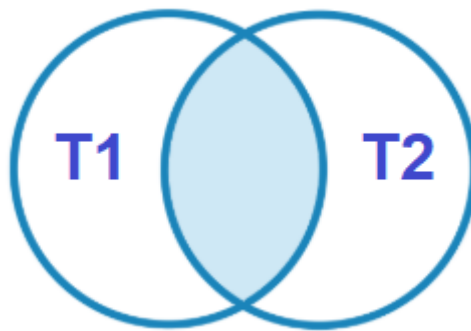
INNER JOIN

En el ejemplo anterior logramos obtener los resultados esperados pero, para ello, tuvimos que utilizar un WHERE obligatoriamente. Si bien el resultado es correcto (en términos de obtener los registros correctos), por cuestiones de eficiencia se recomienda utilizar la cláusula **INNER JOIN** en su lugar, cuya función específica es relacionar dos tablas mediante un campo que sirve como nexo, buscando la **coincidencia exacta** entre los valores de este campo en ambas tablas. ¿Qué significa la *coincidencia exacta*? Significa que INNER JOIN devuelve únicamente aquellos registros donde coincida (en las dos tablas) el valor de este campo que las une. Es decir, si en una de las tablas el valor no se encuentra, el registro no será devuelto.

La sintaxis de INNER JOIN es la siguiente:

```
SELECT T1.Columna1, T2.Columna2
FROM Tabla1 AS T1
INNER JOIN Tabla2 AS T2 ON T1.ID = T2.ID;
```

Y, si lo pensamos “gráficamente”, como en la teoría de conjuntos, INNER JOIN representa la **intersección** entre las tablas:






Ahora, escribamos la consulta anterior reemplazando la cláusula WHERE por INNER JOIN:

```
SELECT A.Apellidos, A.Nombres, P.Pais
FROM Autores AS A INNER JOIN Países AS P
ON A.IDPais = P.IDPais;
```

A diferencia de la consulta anterior, donde incluimos más de una tabla en la cláusula FROM, aquí sólo tenemos una (Autores). Luego, indicamos que la tabla Autores tendrá una

relación con la tabla Paises mediante la cláusula INNER JOIN, y finalmente su nexo (ON) que será el campo IDPais de Autores y IDPais de Paises. El resultado de esta consulta es:

	Apellidos 	Nombres 	Pais 
1	Murakami	Haruki	Japón
2	NULL	NULL	Reino Unido
3	Crichton	Michael	Estados Unidos
4	Kawamura	Genki	Japón

A esta altura quizá se estén preguntando qué sucedió con el registro del autor de apellido *King* y nombre *Stephen*, que tenía NULL en el campo IDPais. Bueno, la respuesta es que, gracias al funcionamiento de INNER JOIN, este registro no aparecerá. Esta cláusula exige una coincidencia exacta en ambas tablas, es decir, que el valor de las columnas que se relacionan exista y sea el mismo en las dos tablas. En este caso, obtendrá todos los registros de autores que se puedan relacionar con un país, y caso contrario, los excluirá. Por esta razón, si tuviera un valor en IDPais de la tabla Autores que no exista en la tabla Paises, o bien tuviese NULL (como en este caso), el registro no será incluido en la selección.

¿Es posible obtener el registro de la tabla Autores cuyo IDPais tiene el valor NULL? Lo veremos más adelante en este apunte. Antes, retomamos el ejemplo del inicio (listado de libros y categorías) y, ya conociendo la utilidad del INNER JOIN, escribiremos la consulta:

```
SELECT L.Titulo, L.AñoPublicacion, L.Paginas, C.Categoria
FROM Libros AS L INNER JOIN Categorías AS C
ON L.IDCategoria = C.IDCategoria;
```

	Titulo	AñoPublicacion	Paginas	Categoria
1	Holly	2023	432	Terror
2	El visitante	2018	592	Terror
3	Fin de guardia	2016	448	Terror
4	Quien pierde paga	2015	434	Terror
5	Mr Mercedes	2014	437	Terror
6	La sangre manda	2020	448	Terror
7	Micro	2011	416	Ciencia Ficción
8	El Instituto	2019	561	Terror
9	Salem's Lot	1975	653	Terror
10	Apocalipsis	1978	1152	Terror
11	Jurassic Park	1990	448	Ciencia Ficción
12	If Cats Disappeared from the World	2012	208	Novela
13	El lector	1995	224	Novela
14	Después	2021	272	Terror
15	Cementerio de animales	1983	416	Terror
16	El piloto del Danubio	1908	320	Novela
17	Drácula	1897	418	Terror
18	Viaje al centro de la Tierra	1864	312	Novela
19	Las Nieves del Kilimanjaro y otros cuentos	1936	240	Novela
20	Needful things	1991	736	Terror

Anidamiento de JOINS

Es probable que uno haya normalizado lo suficiente una base de datos como para que, a la hora de obtener datos, sea necesario recorrer diferentes niveles de relaciones entre más de dos tablas. Esto es posible, y lo veremos muy frecuentemente. Pero, para ello, debemos utilizar los JOINS de manera correcta.

Por ejemplo, queremos obtener un listado que tenga los siguientes datos de libros: título, cantidad de páginas y, además, nombres, apellidos y seudónimos de sus autores.

Si analizamos el DER observamos que los datos de título y cantidad de páginas los obtenemos de la tabla Libros, mientras que apellidos, nombres y seudónimos salen de la tabla Autores. Sin embargo, para que el resultado sea correcto, es preciso que los datos de libros y autores estén vinculados entre sí, es decir, que los datos de autores informados se correspondan con los libros listados.

Las tablas Libros y Autores tienen una relación de **muchos a muchos**, ya que un autor puede haber escrito más de un libro (o ninguno), y un libro puede haber sido escrito por más de un autor (o no registrar autor, por ser el mismo desconocido). Por este motivo es que la tabla Libros no tiene, por ejemplo, un campo IDAutor, ni la tabla Autores tiene un campo

IDLibro. Para relacionar ambas entidades entra en juego la tabla de unión AutoresLibro. Es así que, para este listado, necesitamos datos de tres tablas distintas. Y para relacionarlas, entonces, serán necesarios dos INNER JOIN:

```
SELECT L.Titulo, L.Paginas, A.Nombres, A.Apellidos, A.Pseudonimo
FROM Libros L INNER JOIN AutoresLibros AL ON L.IDLibro = AL.IDLibro
INNER JOIN Autores A ON AL.IDAutor = A.IDAutor;
```

El resultado de esta consulta es el siguiente (por cuestiones de espacio sólo mostramos algunos registros):

	Título	Paginas	Nombres	Apellidos	Pseudonimo
32	Corazones en la Atlántida	672	Stephen	King	Richard Bachman
33	Micro	416	Michael	Crichton	NULL
34	Jurassic Park	448	Michael	Crichton	NULL
35	Prey	502	Michael	Crichton	NULL
36	If Cats Disappeared from the World	208	Genki	Kawamura	NULL
37	Buscando a Papá Noel	128	Genki	Kawamura	NULL
38	El lector	224	Bernhard	Schlink	NULL
39	Las Nieves del Kilimanjaro y otros cuentos	240	Ernest	Hemingway	NULL
40	Whales: Giants Of The Seas And Oceans	304	Paul	Greenberg	NULL
41	Armada	384	Ernest	Cline	NULL
42	Ready Player One	384	Ernest	Cline	NULL
43	1984	328	Ernest	Cline	NULL
44	La Granja	144	Ernest	Cline	NULL
45	El piloto del Danubio	320	Julio	Verne	NULL
46	Viaje al centro de la Tierra	312	Julio	Verne	NULL
47	El camino a Francia	308	Julio	Verne	NULL
48	20.000 Leguas de viaje submarino	416	Julio	Verne	NULL
49	El viejo y el mar	127	Julio	Verne	NULL
50	Drácula	418	Bram	Stoker	NULL
51	El diablo en la botella	112	Robert Louis	Stevenson	NULL
52	Un escándalo en Bohemia	40	Robert Louis	Stevenson	NULL

Necesitamos un listado que contenga algunos datos acerca de las lecturas efectuadas por los usuarios, tales como: nombre del usuario, nombre de su país de origen, título del libro que leyó, fecha en que realizó la lectura, nombre del medio de lectura que utilizó y la cantidad de minutos que duró su sesión de lectura. Además, el listado debe estar ordenado por nombre de usuario, título del libro y fecha de lectura, todo de forma ascendente.

¿Por dónde empezamos? Antes de escribir una consulta compleja, es recomendable dividir la tarea en partes más pequeñas. En primer lugar, analizar el DER y evaluar qué tablas necesitamos para extraer los datos. Posteriormente, las relaciones que existen entre esas tablas, cuál es el “camino” para llegar a una tabla a partir de otra, y cuáles son los campos que tienen en común para poder unirlos (a medida que elaboremos consultas más

complejas veremos que por lo general se utilizan varias tablas que no siempre tienen una relación directa entre sí). Luego, de existir condiciones que deben cumplir los registros a listar, escribir la cláusula WHERE con la proposición lógica (o conjunto de ellas) necesaria para filtrar los datos. Finalmente, quedarnos sólo con las columnas que nos interesan y realizar el ordenamiento según el criterio especificado.

Entonces, pongamos manos a la obra y analicemos el requerimiento:

1. El **nombre de usuario** se obtiene de la tabla **Usuarios** (campo **NombreUsuario**). El **nombre del país de origen del usuario**, se encuentra en la tabla **Países** (campo **Pais**). La manera de obtener el país a través del usuario es con la columna **IDPais** que relaciona ambas tablas. El **título del libro** se obtiene de la tabla **Libros** (**Titulo**). Pero... ¿es correcto relacionar Libros con Usuarios? La realidad es que no pueden relacionarse en forma directa ya que la relación entre libros y usuarios es de muchos a muchos (un usuario puede leer más de un libro, o ninguno, y un libro puede ser leído por más de un usuario, o por ninguno). Para ello contamos con la tabla **Lecturas**, que será el nexo entre los lectores (usuarios) y los libros. Mediante el campo **IDUsuario** podremos averiguar de qué usuario se trata, y con el campo **IDLibro** establecemos la relación con Libros para obtener el título. Tanto la **fecha de lectura** como la **duración en minutos** las obtenemos de **Lecturas**, mediante las columnas **FechaHora** y **TiempoEnMinutos** respectivamente. Finalmente, necesitamos conocer el nombre del medio de lectura: en la tabla Lecturas contamos con el ID del medio de lectura (**IDMedio**), pero no con su descripción. Por lo tanto, a partir de dicho campo debemos establecer una relación más con la tabla **Medios** para dar con el campo **Nombre**.
2. Escribamos una primera versión de la consulta que incluye solamente: las tablas intervinientes y sus alias, el tipo de relación y los campos mediante los cuales se relacionan:

```
SELECT * FROM Usuarios U
INNER JOIN Países P ON U.IDPais = P.IDPais
INNER JOIN Lecturas LE ON U.IDUsuario = LE.IDUsuario
INNER JOIN Libros LI ON LE.IDLibro = LI.IDLibro
INNER JOIN Medios M ON LE.IDMedio = M.IDMedio
```

3. Ahora que unimos convenientemente todas las tablas necesarias, debemos restringir las columnas seleccionadas a las que realmente necesitamos, ya que con la

consulta actual obtenemos todos los registros de las cinco tablas:

	IDUsuario	IDPais	NombreUsuario	Mail	FechaInscripcion	IDPais	Pais	IDLectura	IDUsuario
1	50	ARG	SusWolf	susana@gmail.com	2010-10-12	ARG	Argentina	1	50
2	50	ARG	SusWolf	susana@gmail.com	2010-10-12	ARG	Argentina	2	50
3	50	ARG	SusWolf	susana@gmail.com	2010-10-12	ARG	Argentina	3	50
4	50	ARG	SusWolf	susana@gmail.com	2010-10-12	ARG	Argentina	4	50
5	50	ARG	SusWolf	susana@gmail.com	2010-10-12	ARG	Argentina	5	50

Como se puede apreciar en este fragmento, obtenemos todas las columnas de Usuarios (entre IDUsuario y FechaInscripcion), a continuación se agregan las columnas de Paises (IDPais y Pais), luego las de Lecturas (IDLectura, IDUsuario...) y así sucesivamente. Entonces, seleccionemos las que realmente debemos listar, y agreguemos alias en los casos que sean necesarios:

SELECT

U.NombreUsuario, P.Pais, LI.Titulo, LE.FechaHora AS 'FechaLectura',
M.Nombre AS 'MedioLectura', LE.TiempoEnMinutos

FROM Usuarios U

INNER JOIN Paises P **ON** U.IDPais = P.IDPais

INNER JOIN Lecturas LE **ON** U.IDUsuario = LE.IDUsuario

INNER JOIN Libros LI **ON** LE.IDLibro = LI.IDLibro

INNER JOIN Medios M **ON** LE.IDMedio = M.IDMedio

- Finalmente, como no tenemos condiciones para filtrar los resultados, aplicaremos el ordenamiento. Nuestra consulta en su versión definitiva queda de la siguiente manera:

SELECT

U.NombreUsuario, P.Pais, LI.Titulo, LE.FechaHora AS 'FechaLectura',
M.Nombre AS 'MedioLectura', LE.TiempoEnMinutos

FROM Usuarios U

INNER JOIN Paises P **ON** U.IDPais = P.IDPais

INNER JOIN Lecturas LE **ON** U.IDUsuario = LE.IDUsuario

INNER JOIN Libros LI **ON** LE.IDLibro = LI.IDLibro

INNER JOIN Medios M **ON** LE.IDMedio = M.IDMedio

ORDER BY U.NombreUsuario **ASC**, LI.Titulo **ASC**, LE.FechaHora **ASC**;

	NombreUsuario	Pais	Titulo	FechaLectura	MedioLectura	TiempoEnMinutos
1	Ana	Argentina	20.000 Leguas de viaje submarino	2017-07-17	Ebook	94
2	Ana	Argentina	20.000 Leguas de viaje submarino	2017-07-18	Tablet	277
3	Ana	Argentina	20.000 Leguas de viaje submarino	2017-07-20	PC	33
4	Ana	Argentina	20.000 Leguas de viaje submarino	2017-07-22	Tablet con E-Ink	163
5	Ana	Argentina	20.000 Leguas de viaje submarino	2017-07-23	Audiolibro	28

Si lo desean, pueden corroborar estos resultados (de los cuales sólo copiamos un fragmento con los cinco primeros) practicando la consulta en sus PCs.

LEFT JOIN

Retomando el ejemplo de página 6:

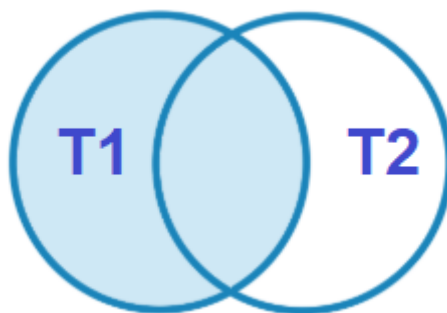
```
SELECT A.Apellidos, A.Nombres, P.Pais
FROM Autores AS A INNER JOIN Países AS P
ON A.IDPais = P.IDPais;
```

Y la pregunta que nos quedó pendiente: ¿es posible obtener el registro de Autores cuyo valor en la columna IDPais es NULL?

Sí, es posible y para ello utilizamos la cláusula **LEFT JOIN**, que tiene la siguiente estructura:




```
SELECT T1.Columna1, T2.Columna2
FROM Tabla1 AS T1
LEFT JOIN Tabla2 AS T2 ON T1.ID = T2.ID;
```

¿Cómo funciona? Al igual que INNER JOIN, la cláusula LEFT JOIN se utiliza para indicar la relación existente entre dos tablas, mediante las columnas que sirven como nexo. Pero, a diferencia de INNER, LEFT JOIN no exige que exista el valor de la columna de la tabla de la izquierda en la tabla de la derecha para poder obtener los datos. Dicho de otro modo, LEFT JOIN devuelve todos los registros de la tabla izquierda (T1), incluyendo las coincidencias con la tabla derecha (T2) pero también los registros que no son coincidentes en T2, ya sea por contener NULL en T1.ID o por no existir un ID en T2 que sea igual a T1.ID. Los campos de T2 que no pudieron ser relacionados con T1, se completan con NULL:



Veamos qué resultados arroja nuestra consulta modificando el INNER JOIN original por un LEFT JOIN:

```
SELECT A.Apellidos, A.Nombres, P.Pais
FROM Autores AS A LEFT JOIN Países AS P
ON A.IDPais = P.IDPais;
```

	Apellidos 	Nombres 	País 
1	Murakami	Haruki	Japón
2	NULL	NULL	Reino Unido
3	King	Stephen	NULL
4	Crichton	Michael	Estados Unidos
5	Kawamura	Genki	Japón

Aquí podemos ver que logramos recuperar el tercer registro de la tabla de autores, que tiene NULL en el campo IDPaís. En este ejemplo, al utilizar LEFT JOIN, estamos pidiendo que la consulta devuelva todos los registros de la tabla izquierda (Autores), incluyendo las coincidencias con la tabla derecha (Países). Si además existe un registro en Autores que no tiene su correspondiente en Países (por tener NULL en el campo IDPaís, que es la clave foránea con Países, o porque no existe el ID de país en Países), indicamos que rellene el campo País (obtenido de la tabla Países) con NULL.

Veremos algunos ejemplos más como refuerzo:

Ejemplo 1: obtener apellidos y nombres de los autores, junto a los títulos y años de publicación de sus libros. Si hay autores sin libros asociados, se deberán listar también.

Realizando un análisis inicial de la situación, necesitamos apellidos y nombres de los autores (que se obtienen de la tabla Autores) y títulos y años de publicación de los libros (que surgen de la tabla Libros). Sin embargo, como antes vimos que autores y libros tienen una relación de varios a varios, sabemos que ambas tablas no se relacionan directamente sino que debemos pasar a través de la tabla de unión AutoresLibro, que las vincula con los campos IDAutor y IDLibro respectivamente. Escribamos la consulta con lo que sabemos hasta aquí:

```
SELECT * FROM Autores AS A
(???) JOIN AutoresLibro AS AL ON A.IDAutor = AL.IDAutor
(???) JOIN Libros AS L ON AL.IDLibro = L.IDLibro;
```

Notar que aún no he definido el tipo de JOIN a utilizar. Como se nos indica que los autores sin libros asociados también deben ser listados, eso significa que hay que traer

todos los registros de Autores sin importar si tienen o no correspondencia con los registros de Libros. Eso nos da una idea de que tenemos que utilizar LEFT JOIN, y de que Autores será la tabla izquierda. Ahora bien, ¿cuál de los dos JOINS será LEFT? ¿Con AutoresLibro (y Libros queda como INNER), con Libros (y AutoresLibro queda como INNER), o con ambas?

Una forma de averiguarlo es conocer previamente cuáles son los autores que no tienen libros, si es que los hay. Por ejemplo, los siguientes autores no tienen libros (es decir, no hay registros en AutoresLibro donde aparezca su ID):

	IDAutor	IDPais	Apellidos	Nombres
1	1	JPN	Murakami	Haruki
2	22	FRA	Saint-Exupéry	Antoine de
3	25	FRA	Yves	Cohat

Luego, nuestra consulta debe devolver, entre otros, los registros con IDAutor 1, 22 y 25, con valores nulos en las tablas de la derecha.

Veamos qué registros se obtienen en cada caso (modificaré ligeramente la consulta para simplificar):

```
SELECT A.IDAutor, A.Apellidos, A.Nombres, AL.IDAutor, AL.IDLibro,
L.IDLibro, L.Titulo
FROM Autores AS A
LEFT JOIN AutoresLibro AS AL ON A.IDAutor = AL.IDAutor
INNER JOIN Libros AS L ON AL.IDLibro = L.IDLibro
WHERE A.IDAutor IN (1, 2, 3, 22, 25)
ORDER BY A.IDAutor;
```

Esta consulta, si bien trae algunos resultados, no incluye a los autores 1, 22 y 25 que vimos que no tenían libros:

	IDAutor	Apellidos	Nombres	IDAutor	IDLibro	IDLibro	Titulo
1	2	NULL	NULL	2	40	40	La historia de Nintendo
2	2	NULL	NULL	2	41	41	Visual Studio Code Succintly
3	3	King	Stephen	3	1	1	Holly
4	3	King	Stephen	3	2	2	El visitante
5	3	King	Stephen	3	3	3	Fin de guardia

Ahora probemos invertir LEFT e INNER:

```
SELECT A.IDAutor, A.Apellidos, A.Nombres, AL.IDAutor, AL.IDLibro,
L.IDLibro, L.Titulo
FROM Autores AS A
INNER JOIN AutoresLibro AS AL ON A.IDAutor = AL.IDAutor
LEFT JOIN Libros AS L ON AL.IDLibro = L.IDLibro
WHERE A.IDAutor IN (1, 2, 3, 22, 25)
```

ORDER BY A.IDAutor;

	IAutor	Apellidos	Nombres	IAutor	IDLibro	IDLibro	Titulo
1	2	NULL	NULL	2	40	40	La historia de Nintendo
2	2	NULL	NULL	2	41	41	Visual Studio Code Succintly
3	3	King	Stephen	3	1	1	Holly
4	3	King	Stephen	3	2	2	El visitante
5	3	King	Stephen	3	3	3	Fin de guardia

El resultado fue el mismo, no vemos los registros de los autores con ID 1, 22 y 25. ¿Y si usamos LEFT JOIN para ambas relaciones?

```
SELECT A.IDAutor, A.Apellidos, A.Nombres, AL.IDAutor, AL.IDLibro,
L.IDLibro, L.Titulo
FROM Autores AS A
LEFT JOIN AutoresLibro AS AL ON A.IDAutor = AL.IDAutor
LEFT JOIN Libros AS L ON AL.IDLibro = L.IDLibro
WHERE A.IDAutor IN (1, 2, 3, 22, 25)
ORDER BY A.IDAutor;
```

	IAutor	Apellidos	Nombres	IAutor	IDLibro	IDLibro	Titulo
1	1	Murakami	Haruki	NULL	NULL	NULL	NULL
2	2	NULL	NULL	2	40	40	La historia de Nintendo
3	2	NULL	NULL	2	41	41	Visual Studio Code Succintly
4	3	King	Stephen	3	1	1	Holly
33	3	King	Stephen	3	77	77	Corazones en la Atlántida
34	22	Saint-Exupéry	Antoine de	NULL	NULL	NULL	NULL
35	25	Yves	Cohat	NULL	NULL	NULL	NULL

Ahora sí, los resultados son los que esperábamos: vemos a todos los autores, incluso los que no escribieron libros, y en esos casos, se rellenan con NULL todos los campos de las tablas que se encuentran a la derecha, es decir, AutoresLibro y Libros. La explicación de por qué sucede esto, es decir, tener que usar LEFT JOIN para unir a Autores con AutoresLibro y Libros, es porque:

Autores A INNER JOIN AutoresLibro AL

Esta sentencia busca la coincidencia exacta, es decir, que el ID de autor encuentre su correspondiente en la tabla de autores por libro (o libros por autor). Los ID de autores 1, 22 y 25 no existen en AutoresLibro, con lo cual, un INNER los excluye de la selección.

(Autores A LEFT JOIN AutoresLibro AL) INNER JOIN Libros L

De la misma manera, aquí se buscan los registros donde coincidan exactamente los valores del campo IDLibro en las tablas AutoresLibro y Libros. AutoresLibro no contiene, por ejemplo, un registro con IDAutor = 1. Por ende, no puede haber coincidencia exacta entre un registro no existente (NULL) en AutoresLibro y un registro existente en Libros.

Una vez analizado esto, y cuando armamos correctamente los JOINS, escribimos la consulta definitiva:

```
SELECT A.Apellidos, A.Nombres, L.Titulo, L.AñoPublicacion
FROM Autores AS A
LEFT JOIN AutoresLibro AS AL ON A.IDAutor = AL.IDAutor
LEFT JOIN Libros AS L ON AL.IDLibro = L.IDLibro;
```

Ejemplo 2: necesitamos un listado con nombre de usuario, título de libro leído y fecha de lectura. El listado debe estar ordenado por fecha de lectura y, si hay usuarios que no registraron lecturas, deben visualizarse de todos modos.

Para obtener nombre de usuario debemos acudir a la tabla **Usuarios** (**NombreUsuario** es el campo), para el título del libro que leyó consultamos la tabla **Libros** (**Título**) y para la fecha de lectura necesitamos la tabla **Lecturas** (**FechaHora**). Pero no solamente para este último campo debemos utilizar la tabla de lecturas. Los usuarios y los libros tienen una relación de muchos a muchos, como vimos en otro ejemplo anteriormente. Por lo cual, la tabla Lecturas es la tabla de unión entre Usuarios y Libros.

Ahora bien, los usuarios que no leyeron también deben ser listados, es decir, debemos obtener todos los registros de usuarios. Esto incluye a los que hayan leído y a los que no, es decir, se incluyen las filas con existencia de IDUsuario en Lecturas y, de no suceder esto último, se incluyen de todas formas las filas de los usuarios, completando los campos de Usuarios con los datos que posee, y con NULL todos los campos de las tablas Lecturas y Libros.

La consulta, finalmente, queda así:

```
SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora
FROM Usuarios AS U
LEFT JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
LEFT JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
ORDER BY LE.FechaHora ASC;
```

Ejemplo 3: necesitamos obtener el nombre de usuario y nombre de su país de origen, de todos aquellos usuarios que no hayan registrado lecturas.

Comencemos evaluando las tablas necesarias, que son **Usuarios** (para obtener **NombreUsuario**), **Países** (relacionada con Usuarios a través del campo **IDPaís**, para obtener el campo **País**), y **Lecturas**, para recuperar solamente los registros que cumplan con la condición. Si armamos el “esqueleto” de la consulta tenemos algo como lo que sigue:

```
SELECT U.NombreUsuario, P.Pais
FROM Usuarios AS U
(???) JOIN Paises AS P ON U.IDPais = P.IDPais
(???) JOIN Lecturas AS L ON U.IDUsuario = L.IDUsuario;
```

En la tabla Usuarios, el campo IDPais no puede ser nulo, y es clave foránea (referencia a Paises). Por todo esto, inferimos que, para que exista un registro de usuario, el campo IDPais debe estar completo con un valor que exista en la tabla Paises. Luego, el tipo de unión que vincula a Usuarios con Paises es de tipo INNER:

```
SELECT U.NombreUsuario, P.Pais
FROM Usuarios AS U
INNER JOIN Paises AS P ON U.IDPais = P.IDPais
(???) JOIN Lecturas AS L ON U.IDUsuario = L.IDUsuario;
```

Respecto de la tabla Lecturas, sabemos que no todos los registros de Usuarios deben vincularse con ella, ya que puede haber usuarios que no hayan registrado lecturas. Entonces:

```
SELECT U.NombreUsuario, P.Pais
FROM Usuarios AS U
INNER JOIN Paises AS P ON U.IDPais = P.IDPais
LEFT JOIN Lecturas AS L ON U.IDUsuario = L.IDUsuario;
```

Finalmente, debemos incorporar a la consulta una cláusula WHERE que contenga la condición para que obtenga solamente los registros de usuarios que no registraron lecturas. Es decir, debemos indicar de alguna manera que, para listar un registro de usuario, su ID no debe aparecer en ningún registro de la tabla Lecturas, (si existiese tal registro, significaría que ese usuario registró al menos una lectura). ¿Y con qué valor completa LEFT JOIN las columnas de la tabla derecha en caso de no poder establecer la relación? Con NULL:

```
SELECT U.NombreUsuario, P.Pais
FROM Usuarios AS U
INNER JOIN Paises AS P ON U.IDPais = P.IDPais
LEFT JOIN Lecturas AS L ON U.IDUsuario = L.IDUsuario
WHERE L.IDLectura IS NULL;
```


	NombreUsuario ↑↓🔍	País ↑↓🔍
1	PabloFernandez	España
2	AlejandroGarcia	España
3	CarmenLopez	España
4	LuciaMartinez	España

¿Cómo se entiende esta relación y su condición? Al establecer una relación de tipo LEFT JOIN entre Usuarios y Lecturas, en principio solicitamos todos los registros de Usuarios, se encuentren o no en Lecturas. Al pedir que IDLectura en Lecturas sea NULL, como dicho campo es la clave primaria de esa tabla, lo que queremos decir es que no debe existir un registro en Lecturas que satisfaga la restricción de clave foránea con Usuarios (teniendo presente que IDUsuario es el campo que determina si hay o no relación entre ambas tablas). Esto equivale a decir que el IDUsuario de Usuarios no debe estar presente en Lecturas o, en otras palabras, que el usuario no está registrado en Lecturas.

Para saber si el resultado es correcto, lo contrastaremos con una búsqueda visual: supongamos que la tabla Usuarios tiene 60 registros, cuyos IDs son numerados del 1 al 60 inclusive. Si ejecutamos esta consulta obtendremos todos los ID de usuarios que registraron lecturas:

```
SELECT DISTINCT IDUsuario FROM Lecturas ORDER BY IDUsuario;
```

IDUsuario ↑↓🔍
51
52
53
54
55
60

La consulta arroja 56 filas de resultados. Por cuestiones de espacio no copiaremos todos los registros obtenidos, pero, analizando los últimos, podemos ver que los IDs de usuario 56,

57, 58 y 59 no figuran en la tabla Lecturas. Luego, estos son los registros de usuarios que deberíamos obtener en nuestra consulta original:

	IDUsuario ↑↓🔍	IDPaís ↑↓🔍	NombreUsuario ↑↓🔍
1	56	ESP	AlejandroGarcia
2	57	ESP	LuciaMartinez
3	58	ESP	PabloFernandez
4	59	ESP	CarmenLopez

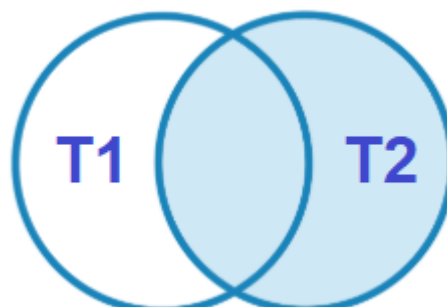
Que, en efecto, son los que obtuvimos.

RIGHT JOIN

Esta cláusula tiene un funcionamiento similar al de LEFT JOIN, pero podría decirse que “al revés”. Es decir, para obtener los datos, no exige que el valor de los campos (que actúan como nexos) de la tabla de la derecha existan en la tabla de la izquierda.

```
SELECT T1.Columna1, T2.Columna2
FROM Tabla1 AS T1
RIGHT JOIN Tabla2 AS T2 ON T1.ID = T2.ID;
```

Lo que hará una consulta como esta es devolver todos los registros de T2. En los casos que haya coincidencia con T1, los registros de T1 serán incluidos. Caso contrario, se completarán los valores de los campos de T1 con NULL. El conjunto de resultados se ve así:



Veremos su funcionamiento con un ejemplo: haremos un listado que contenga el nombre de usuario, título del libro leído, fecha de lectura y medio de lectura. Además, incluiremos en el listado a los medios de lectura que no han sido utilizados.

Como siempre, empezamos relevando las tablas necesarias: **Usuarios** (campo NombreUsuario), **Libros** (Titulo), **Lecturas** (FechaHora), **Medios** (Nombre). La tabla Medios se relaciona con Lecturas a través del campo **IDMedio**.

Luego, armamos un bosquejo de la consulta:

```
SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
(???) JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
(???) JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
(???) JOIN Medios AS M ON LE.IDMedio = M.IDMedio;
```

Y ahora pensemos cómo se relacionan las tablas: **Usuarios** y **Lecturas** deben tener coincidencia exacta, ya que de lo contrario se indicaría que se incluyan los usuarios que no registren lecturas, o que no hayan leído:

```
SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
INNER JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
(???) JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
(???) JOIN Medios AS M ON LE.IDMedio = M.IDMedio;
```

Entre Lecturas y Libros también necesitamos una coincidencia exacta, porque tampoco se especifica que incluyamos los registros de libros que no hayan sido leídos:

```
SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
INNER JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
INNER JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
(???) JOIN Medios AS M ON LE.IDMedio = M.IDMedio;
```

En el caso de Medios y Lecturas, sí nos piden que se incluyan los medios que no hayan sido utilizados. Entonces, ahora sí, tenemos que usar RIGHT JOIN:

```

SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
INNER JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
INNER JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
RIGHT JOIN Medios AS M ON LE.IDMedio = M.IDMedio;

```

Notemos que no tenemos necesidad de incluir una condición ya que, a diferencia del ejemplo anterior, no se pide excluir un conjunto de resultados en particular, sino que obtendremos todos los registros de Medios, sin importar que hayan sido utilizados o no. O sea, esta vez no preguntaremos por el cumplimiento de la restricción de clave foránea que existe entre ambas tablas.

Ejecutamos la consulta y obtenemos el siguiente resultado:

	NombreUsuario ↕	Titulo ↕	FechaHora ↕	NombreMedio ↕
9060	JavierRuiz	Cementerio de animales	2024-05-30	Braille Ebook
9061	NULL	NULL	NULL	Lector de pantalla
9062	NULL	NULL	NULL	Libro de papel en Braille

Se ha listado el nombre de los dos medios que no fueron utilizados, que son “Lector de pantalla” y “Libro de papel en Braille”. Los campos de las tablas Usuarios, Libros y Lecturas que no han podido ser relacionados, se completaron con NULL. Ahora bien, ¿existe una forma rápida de corroborar que estos dos medios nunca fueron utilizados, sin tener que chequear uno por uno? Sí: haremos una consulta similar a la que vimos anteriormente con LEFT JOIN, y obtendremos los nombres de los medios cuyo ID no aparezca en ningún registro de la tabla Lecturas:

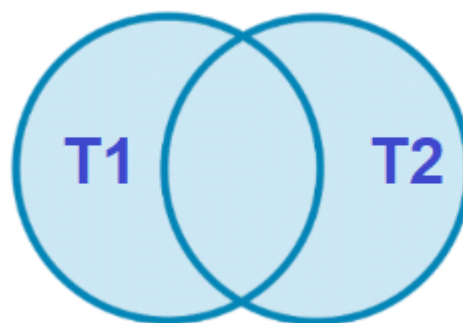
```
SELECT M.Nombre AS NombreMedio
FROM Medios AS M
LEFT JOIN Lecturas AS L ON M.IDMedio = L.IDMedio
WHERE L.IDLectura IS NULL;
```

He aquí el resultado:

	NombreMedio	↕
1	Lector de pantalla	
2	Libro de papel en Braille	

FULL JOIN

Como su nombre lo indica, FULL JOIN permite obtener el listado completo de registros de dos tablas relacionadas, conformados por la combinación de LEFT JOIN y RIGHT JOIN. Esto es, FULL JOIN devolverá todos los registros de la tabla izquierda, más todos los registros de la tabla derecha, con sus respectivas coincidencias. Los campos que no coinciden serán rellenados con NULL.



```
SELECT T1.Columna1, T2.Columna2
FROM Tabla1 AS T1
FULL JOIN Tabla2 AS T2 ON T1.ID = T2.ID;
```

Siguiendo en la línea de los casos anteriores, confeccionaremos un listado que contenga nombre de usuario, título de libro leído, fecha de lectura, nombre del medio utilizado, y la condición de que, además de mostrar todos los medios (utilizados o no) también se listen todos los usuarios, hayan leído o no.

Escribiremos la consulta partiendo de la anterior:

```

SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
INNER JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
INNER JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
RIGHT JOIN Medios AS M ON LE.IDMedio = M.IDMedio;

```

La relación entre Medios y Lecturas en principio queda igual, y nos detendremos un momento en la de Usuarios y Lecturas, ya que para obtener los registros de Usuarios, no necesariamente tienen que tener registrada una lectura.

```

SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
LEFT JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
INNER JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
RIGHT JOIN Medios AS M ON LE.IDMedio = M.IDMedio;

```

Con este cambio desligamos la obligatoriedad de los usuarios respecto de las lecturas, pero si dejamos INNER JOIN en la relación Lecturas-Libros, estamos exigiendo que obligatoriamente exista un registro que contenga la vinculación entre IDLibro de Lecturas y IDLibro de Libros, y a su vez, por transitividad, ese registro de Lecturas debe estar relacionado con Usuarios a través de IDUsuario. Esto contradice lo anterior: no corresponde pedir la coincidencia exacta entre Lecturas y Libros. Entonces:

```

SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
LEFT JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
LEFT JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
RIGHT JOIN Medios AS M ON LE.IDMedio = M.IDMedio;

```

Finalmente, dado que necesitamos todos los registros de Usuarios (tabla izquierda) pero también todos los de Medios (tabla derecha), deducimos que el tipo de unión que corresponde a Medios con las tablas que se encuentran a su izquierda, es FULL:

```

SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
LEFT JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
LEFT JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro

```

```
FULL JOIN Medios AS M ON LE.IDMedio = M.IDMedio;
```

	NombreUsuario ↕	Titulo ↕	FechaHora ↕	NombreMedio ↕
90...	PabloFernandez	NULL	NULL	NULL
90...	AlejandroGarcia	NULL	NULL	NULL
90...	CarmenLopez	NULL	NULL	NULL
90...	LuciaMartinez	NULL	NULL	NULL
90...	NULL	NULL	NULL	Lector de pantalla
90...	NULL	NULL	NULL	Libro de papel en Braille

Si observamos las primeras cuatro filas de este fragmento de resultados, vemos a los cuatro usuarios que no registraron lecturas, y los campos de las tablas de la derecha (Libros, Lecturas y Medios) rellenos con NULL. Por el otro lado, mirando los últimos dos registros, aparecen los nombres de los medios que nunca se han utilizado, y los campos de las tablas de la izquierda (Usuarios, Libros y Lecturas) se han completado con NULL. Esto nos da la pauta de que resolvimos correctamente la consulta.

Por último, si ahora queremos obtener el mismo listado anterior, pero con la condición de que se listen todos los registros de Libros (hayan sido leídos o no), tenemos que modificar levemente la consulta, quedando de esta manera:

```
SELECT U.NombreUsuario, LI.Titulo, LE.FechaHora, M.Nombre AS NombreMedio
FROM Usuarios AS U
LEFT JOIN Lecturas AS LE ON U.IDUsuario = LE.IDUsuario
FULL JOIN Libros AS LI ON LE.IDLibro = LI.IDLibro
FULL JOIN Medios AS M ON LE.IDMedio = M.IDMedio;
```

	NombreUsuario ↕	Titulo ↕	FechaHora ↕	NombreMedio ↕
90...	PabloFernandez	NULL	NULL	NULL
90...	AlejandroGarcia	NULL	NULL	NULL
90...	CarmenLopez	NULL	NULL	NULL
90...	LuciaMartinez	NULL	NULL	NULL
90...	NULL	Cantar de Mio Cid	NULL	NULL
90...	NULL	NULL	NULL	Lector de pantalla
90...	NULL	NULL	NULL	Libro de papel en Braille

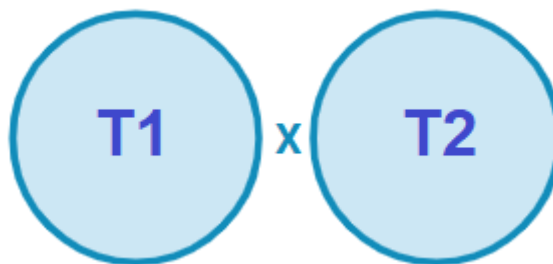
Los resultados obtenidos son los mismos que en la consulta anterior, pero se ha incorporado un registro con Título = 'Cantar de Mio Cid' que corresponde al único libro que no ha sido leído. En resumen, tenemos LEFT JOIN en la relación de Libros y sus tablas de la izquierda (Usuarios y Lecturas) pues queremos recuperar todos los registros de usuarios, aunque no tengan correspondencia. Luego incorporamos un RIGHT JOIN en la relación de

Libros y sus tablas de la izquierda porque necesitamos obtener todos los registros de libros, leídos o no. Esto convierte a la combinación de LEFT y RIGHT en FULL.

Lo mismo sucede con Medios, que se relaciona con LEFT JOIN con sus tablas de la izquierda (Usuarios, Lecturas y Libros) porque queremos obtener todos los usuarios, hayan leído o no, y todos los libros, hayan sido leídos o no. Pero a su vez también queremos todos los medios, utilizados o no, con lo cual también hay una unión de tipo RIGHT JOIN. Nuevamente, la combinación de LEFT y RIGHT convierte a la vinculación con Medios en un FULL JOIN.

CROSS JOIN

Este tipo de unión no suele ser frecuente, pero es interesante conocerla ya que permite unir tablas sin restricciones, es decir, las tablas pueden estar relacionadas (con una clave foránea) o no. Consiste en el **producto cartesiano** entre dos tablas: si aplicamos CROSS JOIN a las tablas T1 y T2 de la imagen, obtendremos todos los registros que se pueden formar combinando cada registro de T1 con cada registro de T2:



La sintaxis de CROSS JOIN es:

```
SELECT T1.Columna1, T2.Columna2
FROM Tabla1 AS T1
CROSS JOIN Tabla2 AS T2;
```

Un ejemplo de CROSS JOIN lo vimos en la página 4, donde se combinaba cada registro de Autores con cada uno de los registros de Países. Sin embargo, fue un resultado accidental: habíamos advertido entonces que la consulta elaborada en ese ejemplo no era correcta para resolver la consigna. En cambio, ahora sí queremos obtener la combinación de todos los autores y todos los países. La cláusula correcta para obtener este resultado es CROSS JOIN:


```
SELECT A.IDAutor, A.Apellidos, A.Nombres, P.Pais
FROM Autores AS A
CROSS JOIN Paises AS P;
```

Veamos algunos resultados:

	IDAutor	Apellidos	Nombres	Pais
1	1	Murakami	Haruki	Argentina
2	2	NULL	NULL	Argentina
3	3	King	Stephen	Argentina
4	4	Crichton	Michael	Argentina
5	5	Kawamura	Genki	Argentina

	IDAutor	Apellidos	Nombres	Pais
27	1	Murakami	Haruki	Alemania
28	2	NULL	NULL	Alemania
29	3	King	Stephen	Alemania
30	4	Crichton	Michael	Alemania
31	5	Kawamura	Genki	Alemania

	IDAutor	Apellidos	Nombres	Pais
67	15	Amicis	Edmundo de	España
68	16	Druon	Maurice	España
69	17	Eco	Umberto	España
70	18	Pérez Rodríguez	Uxio	España
71	19	Alighieri	Dante	España
72	20	Bourdain	Anthony	España
73	21	Balmaceda	Daniel	España
74	22	Saint-Exupéry	Antoine de	España

Notar que en este tipo de JOIN no se debe indicar la columna que actúa como nexo entre las dos tablas, ya que podría no existir tal columna: CROSS JOIN une a las tablas por

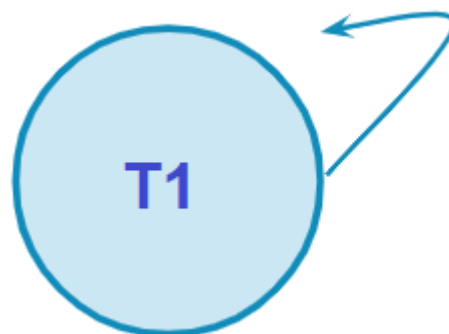
defecto, sin exigir que tengan una relación establecida con una clave foránea. Por caso, también podríamos unir con CROSS a las tablas Categorías y Países, que, según podemos ver en el DER, no tienen ninguna relación entre sí:

```
SELECT C.*, P.*
FROM Categorías AS C
CROSS JOIN Países AS P;
```

	IDCategoría	Categoría	IDPaís	País
1	1	Novela	ARG	Argentina
2	2	Terror	ARG	Argentina
3	3	Fantasía	ARG	Argentina
4	4	Historia	ARG	Argentina
5	5	Ciencia Ficción	ARG	Argentina

SELF JOIN

Si bien SELF JOIN no es considerado un JOIN propiamente dicho, y no se utiliza explícitamente la palabra SELF para hacer referencia a él, se conoce con este término a un tipo de JOIN en el que una tabla se une consigo misma. Aunque en principio parece carecer de sentido, suele ser muy frecuente la necesidad de recurrir al SELF JOIN.



Se podría representar la sintaxis básica de SELF JOIN de la siguiente manera:

```
SELECT T1.ID, T2.Nombre
FROM Tabla1 AS T1
LEFT JOIN Tabla1 AS T2 ON ...;
```

Como podemos observar, se seleccionan campos de la misma tabla (T1), pero, para distinguir el rol que cumple cada uno dentro de la entidad, se utilizan alias. En el ejemplo de la sintaxis básica no se indicó la condición del ON porque dependerá de lo que necesitemos resolver.

Veamos un ejemplo puntual:



Pensemos en esta tabla como una entidad que representa, de manera simplificada, los usuarios de una aplicación web, por ejemplo, una red social. La misma solamente puede accederse si alguien te invita. Es decir, necesitás un usuario que te refiera para poder ser un usuario. En estos casos tenemos una relación de Usuarios a Usuarios.

Cabe mencionar que IDUsuarioReferencia es una Foreign Key que acepta valores nulos porque claramente al menos un usuario debe poder ser creado sin tener IDUsuarioReferencia.

Ahora veamos la siguiente tabla:

IDUsuario	Nombre	IDUsuarioReferencia
1	Administrador	NULL
2	Fernanda	1
3	Mariano	1
4	Abel	2
5	Gladys	4

Podemos observar que al usuario Administrador no lo refirió nadie. Por eso tiene NULL en IDUsuarioReferencia. Luego, a Fernanda y Mariano los refirió el usuario Administrador. De allí que su IDUsuarioReferencia es 1. Y Abel fue referido por Fernanda y Gladys por Abel.

La idea es poder realizar una consulta que muestre el IDUsuario, el nombre del usuario y el nombre del usuario que lo refirió. Es decir, poder ver en el resultado de la query lo que recién escribimos con palabras. La consulta sería como la siguiente:

```
Select U.IDUsuario, U.Nombre, UsuarioReferencia.Nombre as  
'NombreUsuarioReferencia'  
From Usuarios AS U  
Left Join Usuarios As UsuarioReferencia  
ON U.IDUsuarioReferencia = UsuarioReferencia.IDUsuario;
```

El resultado de esta consulta es el siguiente:

IDUsuario	Nombre	NombreUsuarioReferencia
1	Administrador	NULL
2	Fernanda	Administrador
3	Mariano	Administrador
4	Abel	Fernanda
5	Gladys	Abel

Veamos un caso interesante que se puede realizar con la tabla de Libros de nuestra base de datos de Libreria.

Quisiéramos un listado que por cada libro, muestre su título y la cantidad de páginas, junto con el título y la cantidad de páginas de todos los libros que tengan más páginas que él.

Es decir que por cada libro L quiero el título y las páginas. Y quiero el título y las páginas de cada libro Lx que tenga más páginas que L.

Voy a obtener en la consulta entonces L.Titulo, T.Paginas, Lx.Titulo, Lx.Paginas siempre y cuando Lx tenga más páginas que L.

Este es otro caso de un Join sobre la misma tabla. En este caso, la de libros. Con la variante de que la condición del Join no va a preguntar una igualdad como solemos usar sino que utilizará el operador mayor.

```

Select
  L.Titulo,
  L.Paginas,
  LibroMasLargo.Titulo as 'TituloDeLibroMasLargo',
  LibroMasLargo.Paginas As 'PaginasDeLibroMasLargo'
From Libros L
Inner Join Libros as LibroMasLargo ON LibroMasLargo.Paginas > L.Paginas
Order By L.Paginas DESC, LibroMasLargo.Paginas DESC;

```

Para mejorar la lectura, en la consulta llamamos LibroMasLargo a la tabla en lugar de Lx como se hizo en el ejemplo.

Titulo	Paginas	TituloDeLibroMasLargo	PaginasDeLibroMasLargo
It	1138	Apocalipsis	1152
La cúpula	1074	Apocalipsis	1152
La cúpula	1074	It	1138
Needful things	736	Apocalipsis	1152
Needful things	736	It	1138
Needful things	736	La cúpula	1074
History of food	728	Apocalipsis	1152
History of food	728	It	1138
History of food	728	La cúpula	1074
History of food	728	Needful things	736
Corazones en la Atlántida	672	Apocalipsis	1152
Corazones en la Atlántida	672	It	1138
Corazones en la Atlántida	672	La cúpula	1074

Podemos inferir que Apocalipsis es el libro más largo con 1152 páginas. Luego le sigue IT con 1138 páginas. Como IT tiene un libro más largo que él entonces debe aparecer y junto a su información la del libro que lo supera en longitud. Podemos ver eso en el primer registro. Luego, La cúpula tiene dos libros más largos. IT y, por transitividad, Apocalipsis. Entonces La Cúpula tiene dos registros que dan verdadero en la condición LibroMasLargo.Paginas > L.Paginas. Es por eso que aparece dos veces. Luego le sigue Needful things, etc.

Dato adicional: Los Joins que no utilizan el operador de igualdad se denominan Non Equi Join.

UNION

El operador UNION en SQL se utiliza para combinar los resultados de dos o más consultas SELECT en un solo conjunto de resultados. Es especialmente útil cuando deseamos juntar información de diferentes tablas o consultas que tienen la misma cantidad y tipo de columnas en su salida. Al ejecutar un UNION, las filas duplicadas se eliminan automáticamente, mostrando sólo valores únicos. Si queremos incluir todos los registros, incluso los duplicados, se utiliza UNION ALL.

Vamos a suponer que en una base de datos con la información de un instituto disponemos de las tablas de Estudiantes y Docentes. Por supuesto, cada tabla contiene información distinta que representa a cada entidad. Ahora, la institución quiere hacer una reunión e invitar a todos los estudiantes que se hayan inscripto en el año 2025 y además a todos los docentes. Para ello necesita un listado con Apellidos y Nombres y Mails de todos los docentes y estudiantes que correspondan para hacer las invitaciones a la reunión.

El problema es que queremos un solo listado. Y tenemos dos tablas que no tienen relación entre sí. No hay join posible aquí. Lo que necesitamos es la unión de las tablas. Es decir, unir horizontalmente dos consultas de selección distintas.

Estudiantes	Docentes
● Legajo	● ID
● Apellidos	● Apellidos
● Nombres	● Nombres
● Email	● Email
● FechaInscripcion	● Sueldo
	● IDCategoria

Hipoteticamente tenemos 5 estudiantes y 3 docentes. Para facilitar el ejemplo:

Estudiantes

Legajo	Apellidos	Nombres	Email	FechaInscripcion
1	González	María	maria.gonzalez@email.com	2023-02-15
2	Pérez	Carlos	carlos.perez@email.com	2025-03-10
3	Rodríguez	Ana	ana.rodriguez@email.com	2025-01-20
4	López	Jorge	lopez.jorge@mail.com	2025-04-01
5	Martínez	Lucía	lucia.martinez@email.com	2023-02-28

Docentes

ID	Apellidos	Nombres	Email	Sueldo	IDCategoria
1	Fernández	Laura	laura.fernandez@email.com	50000.50	1
2	García	Santiago	santiago.garcia@email.com	45000.00	2
3	Romero	Marta	romero.marta@email.com	40000.00	3

Si ejecutamos la consulta siguiente que realiza la union de ambas tablas, obtendremos el listado que necesitamos:

```
SELECT Apellidos, Nombres, Email From Estudiantes Where  
Year(FechaInscripcion) = 2025  
UNION  
SELECT Apellidos, Nombres, Email from Docentes;
```

El resultado será:

Apellidos	Nombres	Email
Fernández	Laura	laura.fernandez@email.com
García	Santiago	santiago.garcia@email.com
López	Jorge	lopez.jorge@mail.com
Pérez	Carlos	carlos.perez@email.com
Rodríguez	Ana	ana.rodriguez@email.com
Romero	Marta	romero.marta@email.com

El resultado es el esperado. En una sola consulta hemos unido la información de dos consultas de selección.

Si quisiéramos indicar el rol de cada persona "Estudiante" o "Docente" es tan simple como agregar literalmente esa información en sus respectivos selects.

```
SELECT Apellidos, Nombres, Email, 'Estudiante' as Rol From Estudiantes
Where Year(FechaInscripcion) = 2025
UNION
SELECT Apellidos, Nombres, Email, 'Docente' as Rol from Docentes;
```

Obtendremos como resultado la siguiente información:

Apellidos	Nombres	Email	Rol
Fernández	Laura	laura.fernandez@email.com	Docente
García	Santiago	santiago.garcia@email.com	Docente
López	Jorge	lopez.jorge@mail.com	Estudiante
Pérez	Carlos	carlos.perez@email.com	Estudiante
Rodríguez	Ana	ana.rodriguez@email.com	Estudiante
Romero	Marta	romero.marta@email.com	Docente