Uniendo tablas MySQL (I). Introducción.

Iniciamos aquí una pequeña serie de artículos dedicados a las consultas MySQL que afectan a dos o más tablas. Cuando, en una consulta, necesitamos recurrir a dos tablas, significa que ambas están relacionadas entre sí, en una relación de 1-m o de m-1. Cuando se recurre a consultas que implican más de dos tablas, puede tratarse de que alguna de ellas esté relacionada con las demás en relaciones del tipo mencionado, o bien que dos tablas estén relacionadas entre sí, en una relación de tipo m-n, a través de una tercera tabla intermedia.
Además, podemos encontrarnos con la necesidad de construir consultas que impliquen relaciones "especiales", bajo determinadas circunstancias, como veremos en esta serie. Procuraremos analizar las sintaxis adecuadas en los casos que se nos presenten, de modo que, cuando tengas necesidad de consultas de este tipo, puedas hacerlas del modo más óptimo posible.
En estos artículos presentaremos la sintaxis MySQL. Esto significa que podría haber algunas diferencias si tu motor de base de datos SQL es otro (SQL Server, SQLite, etc). Optar por MySQL ha sido una decisión lógica, teniendo en cuenta que es el motor de bases de datos relacionales más empleado en Internet, con mucha diferencia y, desde luego, por méritos propios.

EL ESCENARIO
En este artículo vamos a empezar trabajando con consultas entre tablas que implican una relación m-1. Tendremos dos tablas. En una de ellas almacenamos usuarios y en la otra vamos a almacenar las provincias en las que residen dichos usuarios. Así pues, tenemos una relación en la que un usuario reside en una provincia, y en una provincia puede haber uno o más usuarios. Además, se puede dar el caso de que ciertos usuarios no tengan asignada ninguna provincia de residencia porque, en el momento de registrarlos, no sabíamos cuál es dicha provincia. Además, para poder filtrar los usuarios por un criterio (a fin de poder construir consultas más completas), les hemos asignado una categoría, llamada premium, que cada usuario puede o no tener activada. La estructura de nuestra base de datos, a la que llamamos consultas_1.sql, es la siguiente:

CAMPO (Tabla **usuarios**)
id (El id del usuario, auntoicrementable)
login (El nombre de usuario)
clave (La clave del usuario)
premium (Si el usuario es Premium o no)
id_provincia (El id de la provincia donde reside el usuario)

CAMPO (Tabla **provincias**)
id (El id de la provincia, autoincrementable)
provincia (El nombre de la provincia)

Como ves, son dos tablas muy sencillas que, en un escenario real se quedarían "cortas" de datos para cualquier aplicación mínima, pero con el valor didáctico necesario para este artículo.
RELACIONANDO LAS TABLAS DIRECTAMENTE
En este apartado vamos a ver como relacionar ambas tablas directamente, usando el campo id_provincia de la tabla usuarios y el campo id de la tabla provincias. El objetivo es obtener, mediante una consulta, aquellos usuarios que son premium, junto con el nombre de la provincia en la que residen. Así pues, los usuarios que no son premium no saldrán en los resultados. Tampoco

obtendremos en los resultados aquellos usuarios que, aún siendo premium, no tienen asignada una provincia. La consulta es la siguiente:

```
SELECT login, provincia
FROM usuarios, provincias
WHERE premium = 'S' AND provincias.id = id_provincia;
```

Conceptualmente, esta es la manera más simple de efectuar esta consulta.

CONSULTAS INNER JOIN

El mismo resultado que en el apartado anterior lo podemos obtener mediante una consulta con INNER JOIN (literalmente, "unión interna"). Al principio, cuesta un poco más familiarizarse con esta técnica, porque la sintaxis adecuada no es tan intuitiva como la anterior. Sin embargo, hay dos razones básicas para preferir la que vamos a ver aquí:

La primera es que ante la sintaxis anterior, el motor de MySQL efectúa "por debajo", una consulta INNER JOIN. Lo que ocurre es que, entre la consulta que hemos escrito y la que realmente se ejecuta, MySQL añade una capa de procesamiento adicional que "traduce" la consulta. Por esta razón, una consulta INNER JOIN se efectúa más rápido.

Como consecuencia de lo anterior, en la programación actual, y de cara a tener un código lo más limpio y eficiente, optamos por esta modalidad que vamos a ver aquí.

La consulta que nos interesa, en este ejemplo, y para obtener el mismo resultado que antes, es la siguiente:

```
SELECT login, provincia
FROM usuarios
INNER JOIN provincias
 ON provincias.id = id_provincia
WHERE premium = 'S';";
```

Vamos a ver lo que hacemos:

En primer lugar, indicamos los campos que vamos a recuperar. Se trata de login (en la tabla usuarios) y provincia (en la tabla provincias). Como se trata de campos que sólo aparecen cada uno en una de las tablas, no necesitamos desambigüarlos. En caso contrario los habríamos referido como usuarios.login y provincias.provincia.

A continuación, en la cláusula FROM indicamos, únicamente, la tabla, digámoslo así, principal. Cuando se hace una consulta de tipo INNER JOIN entre dos tablas, las consideramos, esquemáticamente, como situadas una a cada lado de la unión. En la cláusula FROM indicamos sólo la tabla que consideramos "a la izquierda" de dicha unión.

Después, añadimos la cláusula INNER JOIN, seguida por el nombre de la tabla que, en nuestro esquema, está "a la derecha" de la unión.

El siguiente paso es especificar la cláusula ON, con la condición que queremos usar para relacionar ambas tablas. En este ejemplo, la condición de la unión es que el campo id de la tabla provincias coincida con el campo id_provincia de la tabla usuarios. Fíjate el que campo id de la tabla provincias sí hemos tenido que desambigüarlo, ya que ambas tablas tienen un campo con el nombre id. Por esta razón, lo hemos llamado provincias.id. El campo id_provincia, en cambio, no necesita desambigüación, ya que sólo una de las tablas (usuarios) tiene un campo con ese nombre. El hecho de que la cláusula ON esté tabulada es, únicamente, a efectos de que, visualmente, la consulta resulte más legible. Programáticamente es irrelevante.

Por último, tenemos la cláusula WHERE en la que, al igual que en el caso de la consulta del apartado anterior, especificamos una condición para la consulta. Por lógica, la condición se refiere a la tabla "izquierda" (la mencionada en la cláusula FROM). No obstante, si la tabla derecha tuviese algún campo con el mismo nombre, sería imprescindible efectuar una desambigüación.

Como buena práctica, para cuando se incluyen campos de dos o más tablas en una consulta SQL, es adecuado desambigüar los campos siempre, incluso aunque, en este momento no parezca necesario. Esto se debe a que, a lo largo de la vida de una aplicación, las estructuras de las tablas pueden variar y, donde hoy no hay conflictos con los nombres de los campos, mañana pueden surgir, obligándonos a modificar el código. La consulta que hemos visto en este apartado debería, por lo tanto, quedar así:

```
SELECT usuarios.login, provincias.provincia
FROM usuarios
INNER JOIN provincias
```

```
ON provincias.id = usuarios.id_provincia
WHERE usuarios.premium = 'S';";
```

EN CONCLUSIÓN

En este artículo hemos sentado las bases de las uniones (JOIN), mostrando un ejemplo que ilustra al objetivo y la forma de uso de las más simples: las INNER JOIN o uniones internas. La base de datos y un script que ejecuta estas consultas pueden obtenerse en este enlace. En el próximo artículo iremos un paso más allá, con las uniones externas.

ATENCIÓN. En el script que ilustra estos artículos accedemos a la base de datos por PDO. Si no estás familiarizado con este modo de trabajo, te recomiendo leer, antes de seguir adelante, este artículo y, este otro.