Vistas en MySQL

En este artículo vamos a conocer qué son las vistas en MySQL y como pueden ayudarnos a hacer que nuestra aplicación web con base de datos funcione más rápido, mejorando la experiencia del usuario.
Por decirlo brevemente, las vistas constituyen una herramienta muy apta para acelerar las consultas de selección (SELECT), es decir, aquellas que recuperan datos de una o más tablas. Están disponibles, con ligeras variaciones de sintaxis, en todas las implementaciones de SQL, aunque nosotros, por razones obvias, vamos a analizar su funcionamiento en MySQL.

QUÉ SON LAS VISTAS
Una vista es una tabla virtual que se genera a partir de una consulta de selección. Dicho de otro modo. Escribimos una consulta de selección (sobre una o más tablas) para leer los datos, y almacenamos el resultado en una vista. Si usas PHPMyAdmin, la vista te aparecerá como un elemento más de la base de datos en curso. Al pulsar sobre ella, verás el resultado de la consulta de selección, como si fuera una tabla más, aunque con ciertas limitaciones (no puedes editar o añadir registros, por ejemplo) porque lo que estás viendo no es una tabla, sino el resultado de una consulta.
La ventaja que tienes es que si actualizas las tablas originales (en el proceso en curso, o desde cualquier otro punto de la aplicación), la vista queda, automáticamente, actualizada. Por lo tanto, ya no tienes que repetir la consulta de selección para recuperar los nuevos datos de la tabla. Sólo tienes que mirar en la vista, que tien todos los datos de la consulta original actualizados en tiempo real.
EL ESCENARIO
Como siempre, lo veremos todo más claro con un ejemplo. Pensemos en una base de datos con dos tablas: una de usuarios, y otra de provincias. En ambas tablas tenemos un campo id autoincrementable. En la tabla de usuarios hay un campo que le relaciona con la provincia de la tabla de provincias donde reside el usuario. Este campo, que hemos llamado id_provincia tiene el valor 0 en aquellos usuarios de los que aún no tenemos su provincia de residencia.
Si hacemos una lectura de los usuarios con sus provincias de residencia nos saldrá una lista de aquellos usuarios que tienen una provincia asignada. La consulta podría ser algo así:

```
SELECT
usuarios.id,
usuarios.login,
usuarios.premium,
provincias.provincia
FROM usuarios
INNER JOIN provincias
ON provincias.id = usuarios.id_provincia
ORDER BY usuarios.id;
```

Sí ahora alguien actualiza la tabla, (por ejemplo asignando provincias a los usuarios que no la tienen) y queremos los datos actualizados, deberemos repetir la consulta. En cambio, si la consulta la hemos almacenado en una vista, ya sólo tendremos que leer dicha vista, que se mantiene actualizada.
CREAR Y USAR LAS VISTAS
La forma de crear una vista sobre una consulta es, genéricamente, la siguiente:

```
CREATE VIEW IF NOT EXISTS nombre_de_la_vista AS consulta_de_seleccion;
```

En el caso de la consulta anterior, esto podría quedar así:

```
CREATE VIEW IF NOT EXISTS usuarios_con_provincia AS
SELECT
usuarios.id,
```

usuarios.login,
usuarios.premium,
provincias.provincia
FROM usuarios
INNER JOIN provincias
ON provincias.id = usuarios.id_provincia
ORDER BY usuarios.id;
Ahora, para recuperar los resultados actualizados en tiempo real sólo tenemos que recuperar la vista, así:

```
SELECT * FROM nombre_de_la_vista;
```

Esto presenta tres grandes ventajas:

Esta última consulta se ejecuta más rápido.
Cómo hemos dicho, está actualizada si se han modificado las tablas originales.
Es mucho más simple y corta de escribir que la consulta original. Por lo tanto, si tenemos que hacer la consulta de lectura en varios puntos de nuestra aplicación, estamos obteniendo un código más limpio y optimizado.

UN EJEMPLO

A continuación reproducimos un script que hace uso de lo que hemos visto en este artículo, mostrándonos cómo funciona. Se llama vistas.php:

```
<?php
 echo "<pre>";

 $conexion = new PDO('mysql:host=localhost;dbname=pruebas_de_vistas;charset=UTF8', 'root', '');
 $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

 /* Creamos una vista a partir de una consulta que nos recupera los usuarios que tienen provincia
 asignada, con el nombre de dicha provincia. */
 $consulta = "CREATE VIEW IF NOT EXISTS usuarios_con_provincia AS ";
 $consulta .= "SELECT ";
 $consulta .= "usuarios.id, ";
 $consulta .= "usuarios.login, ";
 $consulta .= "usuarios.premium, ";
 $consulta .= "provincias.provincia ";
 $consulta .= "FROM ";
 $consulta .= "usuarios ";
 $consulta .= "INNER JOIN provincias ";
 $consulta .= "ON provincias.id = usuarios.id_provincia ";
 $consulta .= "ORDER BY usuarios.id;";
 $conexion->query($consulta);

 /* Recuperamos el contenido de la vista recién creada. */
 $consulta = "SELECT * FROM ";
 $consulta .= "usuarios_con_provincia;";
 $hacerConsulta = $conexion->query($consulta);
 $resultadosDeLaVista = $hacerConsulta->fetchAll(PDO::FETCH_ASSOC);
 $hacerConsulta->closeCursor();

 /* Mostramos la vista */
 echo "LA VISTA ORIGINAL:<br>";
 echo "==================<br>";
```

```
var_dump($resultadosDeLaVista);

/* Ahora las tablas originales sufren modificaciones en la aplicación. */
$consulta = "INSERT INTO provincias (provincia) VALUES ('TOLEDO');";
$conexion->query($consulta);
$id_ultima_provincia = $conexion->lastInsertId();
$consulta = "UPDATE usuarios SET id_provincia = '".$id_ultima_provincia."' WHERE id_provincia = '0';";
$conexion->query($consulta);

/* Releemos la vista SIN RECREARLA */
$consulta = "SELECT * FROM ";
$consulta .= "usuarios_con_provincia;";
$hacerConsulta = $conexion->query($consulta);
$resultadosDeLaVista = $hacerConsulta->fetchAll(PDO::FETCH_ASSOC);
$hacerConsulta->closeCursor();

/* Mostramos la vista */
echo "LA VISTA TRAS MODIFICAR LAS TABLAS:<br>";
echo "==================================<br>";
var_dump($resultadosDeLaVista);
?>
```

Instala la base de datos, y carga este script en tu navegador, para que veas como funciona y puedas experimentar, que es la mejor manera de familiarizarse con algo. Todo el material que necesitas está en este enlace.