

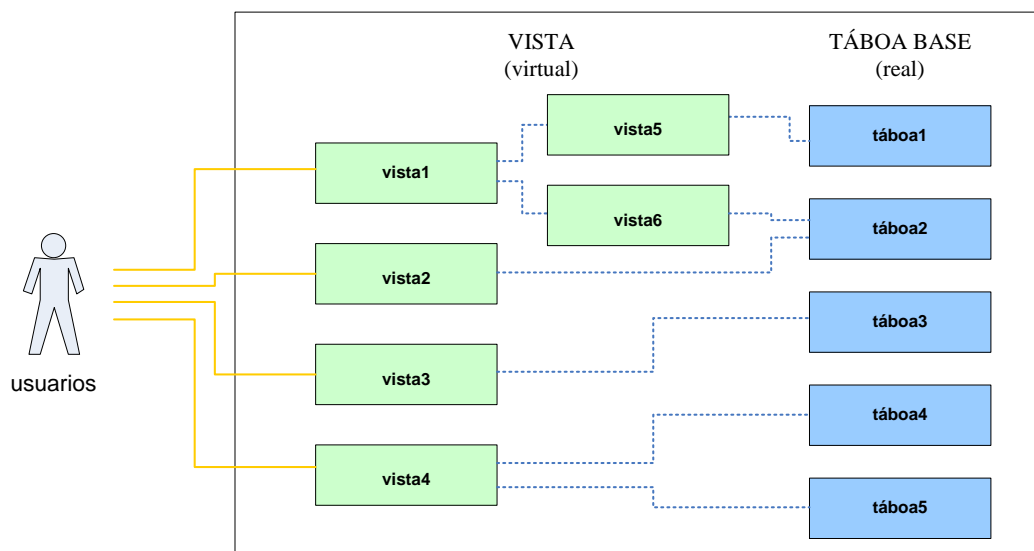
1. Vistas

1.1. Las vistas

Una vista es una tabla virtual que no tiene existencia física como una tabla base, pero que los usuarios ven como una tabla más. Distintos usuarios pueden tener una percepción distinta del contenido de una misma base de datos en función de las vistas que utilizan para acceder a los datos.

Las vistas tienen un nombre asociado y la misma estructura que una tabla, compuesta de filas y columnas. La diferencia fundamental entre las vistas y las tablas base es que de las vistas sólo se almacena la definición de su estructura, y los datos sólo se almacenan en las tablas base.

Las vistas son como un filtro entre los usuarios y los datos almacenados en las tablas base. Este filtro está definido como una selección de datos mediante una sentencia `SELECT` sobre una o más tablas, o sobre otras vistas.



Los permisos que tenga el usuario limitan las operaciones que puede hacer en las bases de datos. Si un usuario puede consultar datos (`SELECT`), y realizar el resto de operaciones de manipulación de datos (`INSERT`, `UPDATE`, `DELETE`), también tendrá permiso para hacer las mismas operaciones utilizando vistas. Para crear o modificar vistas, necesita tener permisos especiales, tal y como se verá más adelante.

La mayoría de los SGBDR permiten la creación y manejo de vistas.

1.1.1. Sentencia `CREATE VIEW` y utilización de vistas

La sentencia `CREATE VIEW` permite crear nuevas vistas, o sustituir vistas que ya existen. Sintaxis:

```
CREATE [OR REPLACE]
[DEFINER = usuario]
[SQL SECURITY { DEFINER | INVOKER }]
VIEW nombre_vista [(lista_columnas)]
AS sentencia_select
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

- La cláusula opcional OR REPLACE permite borrar una vista que exista con el mismo nombre y crear la nueva vista. Si no se utiliza esta cláusula y ya existe una vista con el nombre de la que se va a crear, se produce una condición de error y no se crea la nueva vista.
- Las cláusulas DEFINER y SQL SECURITY especifican el contexto de seguridad en el momento de la ejecución de la vista. Con DEFINER se puede indicar el nombre del usuario que va a ser considerado como el creador de la vista. Si no se especifica nada, se toma CURRENT_USER que hace referencia al usuario actual que está creando la vista.
- La cláusula SQL_SECURITY permite indicar si en la ejecución de la vista se utilizan privilegios del creador de la vista (DEFINER - valor por defecto) o del usuario que la llama (INVOKER).
- El nombre de la vista, *nombre_vista*, tiene que cumplir las mismas condiciones que cualquier nombre de tabla, en cuanto a caracteres y longitud permitidos.
- *sentencia_select* representa una sentencia SELECT, que selecciona los datos de definición de la vista. Los datos pueden ser seleccionados de tablas base o de otras vistas. La sentencia SELECT utilizada para crear una vista tiene una serie de restricciones:
 - No puede contener subconsultas en la cláusula FROM.
 - No puede contener referencias a variables del sistema o de usuario.
 - No puede hacer referencia a tablas de tipo temporal, y no se pueden crear vistas temporales.
- La cláusula WITH CHECK OPTION se utiliza para crear vistas actualizables, que pueden ser utilizadas para insertar o modificar datos de las tablas. Cuando se utiliza la cláusula, sólo permitirá insertar o modificar filas que cumplan las condiciones contempladas en la cláusula WHERE de la *sentencia_select*.
- Las opciones LOCAL y CASCADED se pueden utilizar cuando en la definición de la vista se utilicen otras vistas. Cuando se utiliza la palabra LOCAL, la cláusula CHECK OPTION sólo comprueba las condiciones de la *sentencia_select* de la vista que se está creando. Cuando se utiliza la palabra CASCADED, se comprueban las condiciones de todas las vistas que intervienen en la definición. Si no se especifica nada se toma, por defecto, CASCADED.

El usuario que cree las vistas debe tener el privilegio CREATE VIEW y los privilegios adecuados sobre las columnas a las que se hace referencia en el SELECT. En caso de utilizar la opción OR REPLACE, también es necesario que tenga el privilegio DROP VIEW para borrar la vista.

Las vistas se utilizan como si fueran tablas base junto con SELECT, INSERT, DELETE o UPDATE.

Aquí podemos consultar la documentación sobre esta sentencia:

<https://dev.mysql.com/doc/refman/8.0/en/create-view.html>

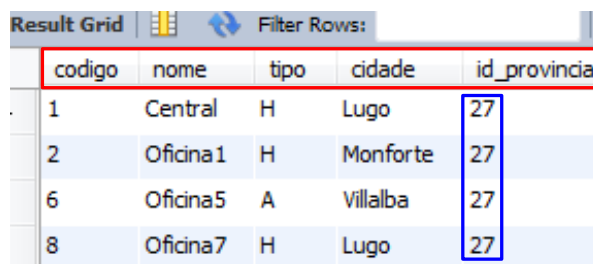
Ejemplos

- Ejemplo de creación de una vista para manejar la información de los departamentos que pertenecen a la provincia de Lugo (*id_provincia* = 27).

```
create view departamento_lugo
as
select codigo, nome, tipo, cidade, id_provincia
from departamento
where id_provincia = 27;
```

- Ejemplo de utilización de la vista anterior para que los usuarios sólo puedan consultar los departamentos de su provincia, y no las de otras provincias.

```
select * from departamento_lugo;
```



codigo	nome	tipo	cidade	id_provincia
1	Central	H	Lugo	27
2	Oficina1	H	Monforte	27
6	Oficina5	A	Villalba	27
8	Oficina7	H	Lugo	27

Cuando se hace la consulta anterior con la vista de ejemplo como filtro, sólo se muestran los departamentos que tengan en *id_provincia* el valor 27, y las columnas se ven con los nombres que tienen en la tabla *departamento*.

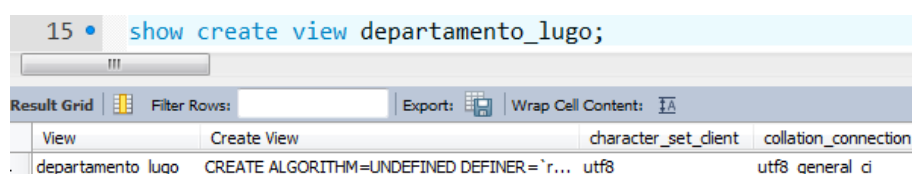
1.1.2. Sentencia SHOW CREATE VIEW

La información sobre las vistas creadas se guarda en el diccionario de datos, al igual que el resto de objetos de las bases de datos. En MySQL, la información sobre las vistas se puede consultar en *information_schema.views* mediante una sentencia SELECT y dispone, además, de la sentencia SHOW CREATE VIEW para consultar información sobre las vistas. Sintaxis:

```
SHOW CREATE VIEW nombre_vista
```


Ejemplo: Mostrar información sobre la vista *departamento_lugo*.

```
show create view departamento_lugo
```



View	Create View	character_set_client	collation_connection
departamento_lugo	CREATE ALGORITHM=UNDEFINED DEFINER='r...' utf8	utf8	utf8_general_ci

El resultado de la sentencia anterior se muestra en la zona *Result Grid* de Workbench. Se muestran cuatro columnas con información sobre el nombre de la vista, la sentencia de creación, el juego de caracteres asociado y el sistema de colación. La información sobre la sentencia de creación no se puede ver porque es muy larga y no entra en la pantalla. Para poder verla en un formato legible se pueden seguir estos pasos:

- Seleccionar el contenido de la columna *Create View* donde está la sentencia de creación.
- Hacer clic con el botón derecho del ratón.
- Eliger la opción *Copy Field (unquoted)*.
- Abrir una nueva pestaña de consulta SQL y pegar en ella el contenido de la columna.
- Pinchar en el icono  *Beautify/Reformat SQL script* y aparece el código en la versión "bonita" de Workbench.

Otro ejemplo de creación de vistas:

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = 'root'@'localhost'
  SQL SECURITY DEFINER
VIEW 'departamento_lugo' AS
  select
'departamento'. 'codigo' AS 'dl_codigo',
'departamento'. 'nome' AS 'dl_nome',
'departamento'. 'tipo' AS 'dl_tipo',
'departamento'. 'ciudad' Las 'dl_cidade',
'departamento'. 'id_provincia' AS 'dl_provincia'
  from
    'departamento'
  where
('departamento'. 'id_provincia' = 27) WITH CASCADED CHECK OPTION
```

1.1.3. Sentencia ALTER VIEW

Permite cambiar la definición de una vista que existe. La sintaxis es similar a la de la sentencia CREATE VIEW, y es lo mismo que utilizar la sentencia CREATE OR REPLACE VIEW. Sintaxis:

Podemos consultar su estructura en:

<https://dev.mysql.com/doc/refman/8.0/en/alter-view.html>

El usuario que modifique vistas tiene que tener el privilegio CREATE VIEW, DROP VIEW y los privilegios adecuados sobre las columnas a las que se hace referencia en el SELECT.

1.1.4. Sentencia DROP VIEW

Para borrar vistas se utiliza la sentencia DROP VIEW. Los usuarios que la utilicen tienen que tener el privilegio DROP VIEW.

Podemos consultar su estructura en:

<https://dev.mysql.com/doc/refman/8.0/en/drop-view.html>

La opción IF EXISTS permite evitar que se produzca un error cuando la vista que se intenta borrar no existe.

Se pueden borrar varias vistas con la misma sentencia poniendo los nombres separados por comas.

1.2. Manipulación de datos a través de las vistas

Las operaciones de manipulación de datos autorizadas sobre las vistas son las mismas que sobre las tablas (consultas, inserción, modificación y borrado). Como se ha comentado anteriormente, todas las vistas pueden ser utilizadas para consultar datos, pero no todas las vistas pueden ser utilizadas para actualizar datos sobre las tablas.

Se llama vista 'actualizable' (*updatable*) a aquella que puede ser utilizada para manipular los datos de las tablas con las que está relacionada mediante INSERT, UPDATE o DELETE.

El manual de referencia de MySQL expone detalladamente las condiciones que tiene que cumplir una vista para que sea 'actualizable'. Un resumen de esas condiciones:

- Debe haber una relación 'uno a uno' entre las filas de la vista y las filas de la tabla con la que está relacionada.
- No contiene:
 - Funciones de agrupación (AVG(), SUM(), COUNT(), ...).
 - Cláusulas: DISTINCT, GROUP BY, HAVING, UNION o UNION ALL.
 - Subconsultas en la lista de selección.
 - Ciertas combinaciones con JOIN.
 - Referencias a vistas no 'actualizables' en la cláusula FROM.
- Para insertar filas utilizando una vista, esta debe incorporar en la lista de selección todas las columnas que no admiten valores nulos (propiedad NOT NULL), y no puede contener expresiones; sólo nombres de columnas.

Cuando se crea una vista, el servidor le añade un indicador que toma el valor verdadero (*true*) cuando la vista es 'actualizable' y el valor falso (*false*) si no lo es. Esta información se puede consultar en la columna IS_UPDATABLE de la tabla VIEW de la base de datos INFORMATION_SCHEMA, aunque en algunos casos puede tener el valor verdadero y no puede ser utilizada para cualquier operación de actualización de datos. Esto significa que cuando se ejecuta una sentencia INSERT, UPDATE o DELETE sobre una vista, el servidor sabe si están permitidas esas operaciones sobre la vista, y en caso de que no estén muestra el mensaje de error correspondiente.

Ejemplos de algunas sentencias SELECT que no se podrían utilizar para crear una vista 'actualizable':

```
select dni,apellidos,nome,salario_bruto *20/100
from empleado;
```

Porque contiene una expresión y no puede ser utilizada para insertar filas, aunque podría ser utilizada para hacer modificaciones y borrado de filas

```
select id_provincia, ciudad , count(*)
from departamento
group by id_provincia;
```

Porque utiliza la cláusula GROUP BY, y además la función de agrupamiento COUNT(*)

```
select dni,apellidos, nombre, salario_bruto,
(select avg() from empleado) as media_salario
from empleado
where salario_bruto >= 50000;
```

Porque contiene una subconsulta en la lista de selección.

```
select apellidos, nome, salario_bruto
from empleado
where salario_bruto between 50000 and 70000;
```

No puede ser utilizada para insertar filas, porque no incluye la columna dni que no admite valores null.

1.3. Ventajas del uso de vistas

De forma resumida, los beneficios más importantes que aportan las vistas son los siguientes:

- Se almacenan en el servidor, por lo que el consumo de recursos y eficacia siempre serán óptimos. Cuando se crea una vista se hace una compilación de la sentencia *SELECT* que contiene, y se guarda asociada al nombre de la vista. De esta manera el tiempo de ejecución de una vista es menor que si se ejecuta la sentencia *SELECT* directamente, ya que ésta tiene que pasar por el proceso de compilación.
- Una vista es un camino fácil para guardar consultas complejas en la propia base de datos. Ocurre con frecuencia que los usuarios solicitan información que requiere realizar consultas complejas que pueden llevar mucho tiempo diseñar y probar, y que pueden ser utilizadas en otro momento. Si estas consultas se hacen con vistas, quedarán almacenadas en el servidor y estarán accesibles cuando se necesiten.
- Proporcionan flexibilidad en la formulación de consultas muy complejas. Una vista puede ser creada con la idea de facilitar la formulación de sentencias complejas que no son soportadas en una única sentencia *SELECT*.
- Son una herramienta importante para mantener la confidencialidad de los datos. Pueden ser utilizadas como filtro entre los usuarios y las tablas base, de manera que los usuarios no acceden directamente a las tablas y no tienen que ver todos los datos de las tablas si no que solo ven los datos que fueron seleccionados en la consulta con la que se creó la vista.
Se puede limitar el acceso a los usuarios a determinadas filas o columnas de una o más tablas, sin tener que conceder privilegios a los usuarios sobre las tablas, siendo suficiente conceder privilegios sobre las vistas.
A los desarrolladores de software se les facilita el acceso a las vistas con los datos que necesitan, pero no se les da acceso a las tablas.
- Ayudan a mantener la integridad de dominio. Una vista puede restringir el acceso a las filas en función del contenido de una o más columnas cuando incluyen condiciones en

la cláusula `WHERE` en la consulta utilizada en su creación. También, pueden utilizarse para comprobar si los valores guardados en esas columnas cumplen las condiciones establecidas, cuando se insertan o modifican filas. Por ejemplo, podemos crear una vista asociada a la tabla `clientes` que compruebe que los datos almacenados en la columna `localidad` pertenezcan al dominio {'Lugo','Sarria','Ourense','Ferrol','Vigo'}.