**CREATE VIEW MySQL: Crear vistas en una base de datos**

Vamos a repasar la sentencia **CREATE VIEW MySQL**, esta tiene como finalidad crear tablas virtuales a partir de **consultas SELECT** a otras tablas.

Las vistas de una [base de datos MySQL](https://www.anerbarrena.com/programacion/php/mysql/) tienen la misma estructura de filas y columnas de una tabla, **no puede haber una vista con el mismo nombre** de una tabla.

**Sintaxis de CREATE VIEW MySQL**

|  |  |
| --- | --- |
| 1  2 | CREATE [OR REPLACE] VIEW nombre\_vista [column\_list]  AS consulta\_SELECT |

**crear vistas en MySQL**:

* **OR REPLACE**: Reemplaza una vista existente en caso de coincidir en nombre.
* **nombre\_vista**: Nombre de la vista a crear.
* **column\_list**: Listado de columnas a crear.
* **consulta\_SELECT**: Consulta SELECT que queremos realizar para obtener la información que contendrá la vista.

**Ventajas de usar vistas en MySQL**

* **Privacidad de la información**: Mostramos a los usuarios con acceso a la vista únicamente la información que creamos conveniente. De esta manera no se tiene acceso a la tabla original con todas sus filas y columnas.
* **Optimización del rendimiento de la base de datos**: Podemos crear de querys sobre vistas complejas, es decir, vistas cuya información ha sido extraída y creada a través de unas SELECT complejas. De esta manera nos ahorramos estar ejecutando queys pesadas y atacamos directamente al resultado de dichas querys.
* **Tablas de prueba**: Para los desarrolladores que no tengan entornos de preproducción es muy útil usar las vistas para no tener miedo a perder información.

**Ejemplo para crear vistas con CREATE VIEW MySQL**

**ejemplos para crear vistas en MySQL**:

|  |  |
| --- | --- |
| 1  2  3 | CREATE VIEW  v\_futbolistas AS  SELECT \* FROM futbolistas; |

En el código superior estaríamos creando una vista con todas las filas y columnas de la tabla futbolistas.

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE VIEW  v\_futbolistas AS  SELECT futbolistas.id, nombre, apellidos FROM futbolistas  INNER JOIN tarjetas\_amarillas  ON futbolistas.id = tarjetas\_amarillas.id\_futbolista; |

En este otro código estaríamos obteniendo una vista con los datos de los futbolistas que contienen tarjetas amarillas usando la sentencia **INNER JOIN**.

|  |  |
| --- | --- |
| 1  2  3  4 | CREATE VIEW  v\_futbolistas AS  SELECT futbolistas.id, nombre, apellidos FROM futbolistas  WHERE edad < 30; |

En este otro código estaríamos obteniendo una vista con los datos de los futbolistas que tienen menos de 30 años sentencia **WHERE**.

**DROP VIEW MySQL: Eliminar una vista creada**

**Sintaxis de DROP VIEW MySQL**

|  |  |
| --- | --- |
| 1 | DROP VIEW [IF EXISTS] nombre\_vista1, nombre\_vista2 ...; |

sintaxis:

* Es recomendable usar la **claúsula IF EXISTS**, de esta manera no se mostrará error en caso de que la vista a eliminar no exista.
* Se pueden borrar tantas vistas como deseemos, sus nombres deberán ir separadas de una coma.

Hay que recordar que para poder **eliminar una vista** hay que tener los privilegios necesarios para realizar dicha acción.

**Ejemplo de DROP VIEW MySQL**

|  |  |
| --- | --- |
| 1 | DROP VIEW [IF EXISTS] vista\_usuarios; |

**ALTER VIEW MySQL: Modificar las vistas de una base de datos**

Modificar las vistas existentes en una base de datos.

**Sintaxis de ALTER VIEW para modificar vistas en**

**MySQL**

|  |  |
| --- | --- |
| 1  2 | ALTER VIEW nombre\_vista [column\_list]  AS consulta\_SELECT |

Tal y como podéis ver en la sintaxis solo hay que poner el **nombre de la vista y columnas** que queramos modificar seguidas de la **consulta select**.

**Ejemplo de ALTER VIEW MySQL**

Veamos un sencillo **ejemplo para modificar una vista** de una base de datos:

|  |  |
| --- | --- |
| 1  2  3 | ALTER VIEW OR REPLACE  v\_usuarios AS  SELECT \* FROM usuarios WHERE edad < 30; |

Ventajas:

* Facilita el manejo de grandes volúmenes de información, haciéndolos más fáciles y rápidos de manipular.
* Brinda mayor seguridad a la información.
* Mejora la metodología de trabajo, se hace mas organizada.
* Facilita la realización de consultas en la BD por lo que se facilita la toma de decisiones.

Desventajas:

* Rendimiento ya que las vistas crean una apariencia de tabla por lo que el SMBD debe traducir las consultas definidas en una vista.
* restricciones de actualización, cuando se solicita la actualización de una fila de la vista se debe traducir en una petición de actualización a la tabla de origen de la vista, esto en vistas muy grandes puede causar problemas y evitar que las actualizaciones se realicen.

**Ventajas de utilizar view’s:**

* Una vista permite simplificar las sentencias complejas de consulta. Ejemplo: cuando no piden un reporte y tenemos que consultar varias tablas y luego crear los filtros, solo tendríamos que crear un view con la consulta hacia las diferentes tablas y luego llamamos el view desde una sentencia SQL y con la cláusula WHERE filtrariamos los resultados.
* Además podemos utilizar las vista para proporcionar una capa de seguridad adicional a nuestra base de datos.

**Desventajas de utilizar view’s:**

* Al que tener cuidado al utilizar las vista porque consume mucho procesador, porque no se almacena la data, sino la sentencia SQL y se ejecuta cada vez que llamamos el VIEW.
* Las vista dependen de la estructura de la tabla así que si las estructura cambia, el VIEW debe modificarse también.