# Unidad 2 – Vistas

## Vistas

Las vistas (incluyendo vistas actualizables) fueron introducidas en la versión 5.0 del servidor de base de datos MySQL

En este capítulo se tratan los siguientes temas:

- Creación o modificación de vistas con CREATE VIEW o ALTER VIEW
- Eliminación de vistas con DROP VIEW
- Obtención de información de definición de una vista (metadatos) con SHOW CREATE VIEW

### Sintaxis de ALTER VIEW

```
  ALTER [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
VIEW nombre_vista [(columnas)]
AS sentencia_select
  [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Esta sentencia modifica la definición de una vista existente. La sintaxis es semejante a la empleada en CREATE VIEW. Se requiere que posea los permisos CREATE VIEW y DELETE para la vista, y algún privilegio en cada columna seleccionada por la sentencia SELECT.

### Sintaxis de CREATE VIEW

```
  CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
VIEW nombre_vista [(columnas)]
AS sentencia_select
  [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Esta sentencia crea una vista nueva o reemplaza una existente si se incluye la cláusula OR REPLACE.

La sentencia_select es una sentencia SELECT que proporciona la definición de la vista. Puede estar dirigida a tablas de la base o a otras vistas.

Se requiere que posea el permiso CREATE VIEW para la vista, y algún privilegio en cada columna seleccionada por la sentencia SELECT. Para columnas incluidas en otra parte de la sentencia SELECT debe poseer el privilegio SELECT. Si está presente la cláusula OR REPLACE, también deberá tenerse el privilegio DELETE para la vista.

Toda vista pertenece a una base de datos. Por defecto, las vistas se crean en la base de datos actual. Para crear una vista en una base de datos específica, indíquela con base_de_datos.nombre_vista al momento de crearla.

```
CREATE VIEW test.v AS SELECT * FROM t;
```

Las tablas y las vistas comparten el mismo espacio de nombres en la base de datos, por eso, una base de datos no puede contener una tabla y una vista con el mismo nombre.

Al igual que las tablas, las vistas no pueden tener nombres de columnas duplicados. Por defecto, los nombres de las columnas devueltos por la sentencia SELECT se usan para las columnas de la vista .

Para dar explícitamente un nombre a las columnas de la vista utilice la cláusula columnas para indicar una lista de nombres separados con comas. La cantidad de nombres indicados en columnas debe ser igual a la cantidad de columnas devueltas por la sentencia SELECT.

Las columnas devueltas por la sentencia SELECT pueden ser simples referencias a columnas de la tabla, pero también pueden ser expresiones conteniendo funciones, constantes, operadores, etc.

Los nombres de tablas o vistas sin calificar en la sentencia SELECT se interpretan como pertenecientes a la base de datos actual. Una vista puede hacer referencia a tablas o vistas en

otras bases de datos precediendo el nombre de la tabla o vista con el nombre de la base de datos apropiada.

Las vistas pueden crearse a partir de varios tipos de sentencias SELECT. Pueden hacer referencia a tablas o a otras vistas. Pueden usar combinaciones, UNION, y subconsultas. El SELECT inclusive no necesita hacer referencia a otras tablas. En el siguiente ejemplo se define una vista que selecciona dos columnas de otra tabla, así como una expresión calculada a partir de ellas:

```
CREATE TABLE precios (costo INT, cant INT); INSERT
  INTO precios VALUES(3, 50);

CREATE VIEW vista_precios AS SELECT costo, cant, costo * cant AS value FROM precios;
  SELECT * FROM vista_precios;
```

La definición de una vista está sujeta a las siguientes limitaciones:

- La sentencia SELECT no puede contener una subconsulta en su cláusula FROM.
- La sentencia SELECT no puede hacer referencia a variables del sistema o del usuario.
- La sentencia SELECT no puede hacer referencia a parámetros de sentencia preparados.
- Dentro de una rutina almacenada, la definición no puede hacer referencia a parámetros de la rutina o a variables locales.
- Cualquier tabla o vista referenciada por la definición debe existir. Sin embargo, es posible que después de crear una vista, se elimine alguna tabla o vista a la que se hace referencia. Para comprobar la definición de una vista en busca de problemas de este tipo, utilice la sentencia CHECK

TABLE.

- La definición no puede hacer referencia a una tabla TEMPORARY, y tampoco se puede crear una vista TEMPORARY.
- Las tablas mencionadas en la definición de la vista deben existir siempre.
- No se puede asociar un disparador con una vista.

En la definición de una vista está permitido ORDER BY, pero es ignorado si se seleccionan columnas de una vista que tiene su propio ORDER BY.

Con respecto a otras opciones o cláusulas incluidas en la definición, las mismas se agregan a las opciones o cláusulas de cualquier sentencia que haga referencia a la vista creada, pero el efecto es indefinido. Por ejemplo, si la definición de una vista incluye una cláusula LIMIT, y se hace una selección desde la vista utilizando una sentencia que tiene su propia cláusula LIMIT, no está definido cuál se aplicará. El mismo principio se extiende a otras opciones como ALL, DISTINCT, o SQL_SMALL_RESULT que se ubican a continuación de la palabra reservada SELECT, y a cláusulas como INTO, FOR UPDATE, LOCK IN SHARE MODE, y PROCEDURE.

Si se crea una vista y luego se modifica el entorno de proceso de la consulta a través de la modificación de variables del sistema, puede afectar los resultados devueltos por la vista:

La cláusula opcional ALGORITHM es una extensión de MySQL al SQL estándar. ALGORITHM puede tomar tres valores: MERGE, TEMPTABLE, o UNDEFINED. El algoritmo por defecto es UNDEFINED si no se encuentra presente la cláusula ALGORITHM. El algoritmo afecta la manera en que MySQL procesa la vista.

- Para MERGE, el texto de una sentencia que haga referencia a la vista y

la definición de la vista son mezclados de forma que parte de la definición de la vista reemplaza las partes correspondientes de la consulta.

- Para TEMPTABLE, los resultados devueltos por la vista son colocados en una tabla temporal, la cual es luego utilizada para ejecutar la sentencia.
- Para UNDEFINED, MySQL determina el algoritmo que utilizará. En ese caso se prefiere MERGE por sobre TEMPTABLE si es posible, ya que MERGE por lo general es más eficiente y porque la vista no puede ser actualizable si se emplea una tabla temporal.

Una razón para elegir explícitamente TEMPTABLE es que los bloqueos en tablas subyacentes pueden ser liberados después que la tabla temporal fue creada, y antes de que sea usada para terminar el procesamiento de la sentencia. Esto podría resultar en una liberación del bloqueo más rápida que en el algoritmo MERGE, de modo que otros clientes que utilicen la vista no estarán bloqueados mucho tiempo.

El algoritmo de una vista puede ser UNDEFINED en tres situaciones:

- No se encuentra presente una cláusula ALGORITHMen la sentencia CREATE VIEW.
- La sentencia CREATE VIEW tiene explícitamente una cláusula ALGORITHM = UNDEFINED.
- Se especificó ALGORITHM = MERGE para una vista que solamente puede ser procesada usando una tabla temporal. En este caso, MySQL emite una advertencia y establece el algoritmo en UNDEFINED.

Como se dijo anteriormente, MERGE provoca que las partes correspondientes de la definición de la vista se combinen dentro de la sentencia que hace referencia a la vista. El siguiente ejemplo muestra brevemente cómo funciona el algoritmo MERGE. El ejemplo asume que hay una vista v_merge con esta definición:

```
CREATE ALGORITHM = MERGE VIEW v_merge (vc1, vc2) AS SELECT
  c1, c2 FROM t WHERE c3 > 100;
```

Ejemplo 1: Suponiendo que se utilice esta sentencia:

```
SELECT * FROM v_merge;
```

MySQL la gestiona del siguiente modo:

- v_merge se convierte en t
- * se convierte en vc1, vc2, que corresponden a c1, c2

- Se agrega la cláusula WHERE de

la vista La sentencia ejecutada resulta

ser:

```
SELECT c1, c2 FROM t WHERE c3 > 100;
```

Ejemplo 2: Suponiendo que se utilice esta sentencia:

```
SELECT * FROM v_merge WHERE vc1 < 100;
```

Esta sentencia se gestiona en forma similar a la anterior, a excepción de que vc1 < 100 se convierte en c1 < 100 y la cláusula WHERE de la vista se agrega a la cláusula WHERE de la sentencia empleando un conector AND (y se agregan paréntesis para asegurarse que las partes de la cláusula se ejecutarán en el orden de precedencia correcto). La sentencia ejecutada resulta ser:

```
SELECT c1, c2 FROM t WHERE (c3 > 100) AND (c1 < 100);
```

Necesariamente, la sentencia a ejecutar tiene una cláusula WHERE con esta forma:

```
WHERE (WHERE de la sentencia) AND (WHERE de la vista)
```

El algoritmo MERGE necesita una relación uno-a-uno entre los registros de la vista y los registros de la tabla subyacente. Si esta relación no se sostiene, debe emplear una tabla temporal en su lugar. No se tendrá una relación uno-a-uno si la vista contiene cualquiera de estos elementos:

- Funciones agregadas (SUM(), MIN(), MAX(), COUNT(), etcétera)
- DISTINCT
- GROUP BY
- HAVING
- UNION o UNION ALL

- Hace referencia solamente a valores literales (en tal caso, no hay una tabla subyacente)

Algunas vistas son actualizables. Esto significa que se las puede emplear en sentencias como UPDATE, DELETE, o INSERT para actualizar el contenido de la tabla subyacente. Para que una vista sea actualizable, debe haber una relación uno-a-uno entre los registros de la vista y los registros de la tabla subyacente. Hay otros elementos que impiden que una vista sea actualizable. Más específicamente, una vista no será actualizable si contiene:

- Funciones agregadas (SUM(), MIN(), MAX(), COUNT(), etcétera)
- DISTINCT
- GROUP BY
- HAVING
- UNION o UNION ALL
- Una subconsulta en la lista de columnas del SELECT
- Join
- Una vista no actualizable en la cláusula FROM
- Una subconsulta en la cláusula WHERE que hace referencia a una tabla en la cláusula FROM
- Hace referencia solamente a valores literales (en tal caso no hay una) tabla subyacenta para actualizar.
- ALGORITHM = TEMPTABLE (utilizar una tabla temporal siempre resulta en una vista no actualizable)

Con respecto a la posibilidad de agregar registros mediante sentencias INSERT, es necesario que las columnas de la vista actualizable también cumplan los siguientes requisitos adicionales:

- No debe haber nombres duplicados entre las columnas de la vista.
- La vista debe contemplar todas las columnas de la tabla en la base de datos que no tengan indicado un valor por defecto.
- Las columnas de la vista deben ser referencias a columnas simples y no columnas derivadas. Una columna derivada es una que deriva de una expresión. Estos son algunos ejemplos de columnas derivadas:

```
3.14159
col1 + 3 UPPER(col2)
col3 / col4
  (subquery)
```

No puede insertar registros en una vista conteniendo una combinación de columnas simples y derivadas, pero puede actualizarla si actualiza únicamente las columnas no derivadas. Considere esta vista:

```
CREATE VIEW v AS SELECT col1, 1 AS col2 FROM t;
```

En esta vista no pueden agregarse registros porque col2 es derivada de una expresión. Pero será actualizable si no intenta actualizar col2. Esta actualización es posible:

```

```

Esta actualización no es posible porque se intenta realizar sobre una columna derivada:

```
UPDATE v SET col2 = 0;
```

A veces, es posible que una vista compuesta por múltiples tablas sea actualizable, asumiendo que es procesada con el algoritmo MERGE. Para que esto funcione, la vista debe usar inner join (no outer join o UNION). Además, solamente puede actualizarse una tabla de la definición de la vista, de forma que la cláusula SET debe contener columnas de sólo una tabla de la vista.

Las vistas que utilizan UNION ALL no se pueden actualizar aunque teóricamente fuese posible hacerlo, debido a que en la implementación se emplean tablas temporales para procesarlas.

En vistas compuestas por múltiples tablas, INSERT funcionará si se aplica sobre una única tabla. DELETE no está soportado.

La cláusula WITH CHECK OPTION puede utilizarse en una vista actualizable para evitar inserciones o actualizaciones excepto en los registros en que la cláusula WHERE de la sentencia_select se evalúe como true.

En la cláusula WITH CHECK OPTION de una vista actualizable, las palabras reservadas LOCAL y CASCADED determinan el alcance de la verificación cuando la vista está definida en términos de otras vistas. LOCAL restringe el CHECK OPTION sólo a la vista que está siendo definida. CASCADED provoca que las vistas subyacentes también sean verificadas. Si no se indica, el valor por defecto es CASCADED. Considere las siguientes definiciones de tabla y vistas:

```
CREATE TABLE t1 (a INT);

CREATE VIEW v1 AS SELECT * FROM t1 WHERE a < 2 WITH
  CHECK OPTION;

CREATE VIEW v2 AS SELECT * FROM v1 WHERE a > 0 WITH
  LOCAL CHECK OPTION;

CREATE VIEW v3 AS SELECT * FROM v1 WHERE a > 0 WITH
  CASCADED CHECK OPTION;
```

Las vistas v2 y v3 están definidas en términos de otra vista, v1.

v2 emplea check option LOCAL, por lo que las inserciones sólo atraviesan la verificación de v2.

v3 emplea check option CASCADED de modo que las inserciones no solamente atraviesan su propia verificación sino tambien las de las vistas subyacentes. Las siguientes sentencias demuestran las diferencias:

La posibilidad de actualización de las vistas puede verse afectada por el valor de la variable del sistema updatable_views_with_limit.

La sentencia CREATE VIEW fue introducida en MySQL 5.0.1. La cláusula WITH CHECK OPTION fue implementada en MySQL 5.0.2.

INFORMATION_SCHEMA contiene una tabla VIEWS de la cual puede obtenerse información

sobre los objetos de las vistas.

**Sintaxis de DROP VIEW**

```
  DROP VIEW [IF EXISTS]
nombre_vista [, nombre_vista] ... [RESTRICT
  | CASCADE]
```

DROP VIEW elimina una o más vistas de la base de datos. Se debe poseer el privilegio DROP en cada vista a eliminar.

La cláusula IF EXISTS se emplea para evitar que ocurra un error por intentar eliminar una vista inexistente. Cuando se utiliza esta cláusula, se genera una NOTE por cada vista inexistente.

**Sintaxis de SHOW CREATE VIEW**

```
SHOW CREATE VIEW nombre_vista
```

Muestra la sentencia CREATE VIEW que se utilizó para crear la vista.