BASES DE DATOS

FES Aragón

ICO

MTI. Omar Mendoza González

Vistas

- Una vista de base de datos es un resultado de una consulta SQL de cero, una o varias tablas.

- Las vistas tienen la misma estructura que una tabla: filas y columnas. La única diferencia es que sólo se almacena de ellas la definición, no los datos.

Vistas

- 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.
- 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.

Limitaciones de las Vistas

- 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.

Limitaciones de las Vistas

- 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.

Limitaciones de las Vistas

- Las tablas mencionadas en la definición de la vista deben existir siempre.
- No se puede asociar un disparador con una vista.
- Las columnas de la vista deben ser referencias a columnas simples y no columnas derivadas

Sintaxis Vistas

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE |
    TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Sintaxis Vistas

ALTER [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}] VIEW *nombre_vista* [(*columnas*)]

AS *sentencia_select*

DROP VIEW [IF EXISTS] *nombre_vista* [, *nombre_vista*] ...

SHOW CREATE VIEW *nombre_vista*

Ejemplo de Vista

```
CREATE OR REPLACE  ALGORITHM=UNDEFINED
DEFINER=root@localhost
SQL SECURITY DEFINER
VIEW alumno_datos AS
SELECT clave_alu, CONCAT_WS(' ',
    ap_paterno,ap_materno, nombre) as nombrealu,
IF(sexo='M','MASCULINO',IF(sexo='F','FEMENINO','ND
    ')) as sexo, curp
FROM alumnos
```

Ejemplo de Vista

```
CREATE VIEW alumno_curso AS
SELECT a.clave_alu AS idalu, CONCAT_WS( '
   ', ap_paterno, ap_materno, a.nombre ) AS
   nombrealu, abreviatura AS curso
   FROM alumnos a
   LEFT JOIN alumno_salon b ON ( a.clave_alu
   = b.clave_alu )
   LEFT JOIN cursos c ON ( b.id_curso =
   c.id_curso )
```

Vistas

- CREATE VIEW or ALTER VIEW
- DROP VIEW
- SHOW CREATE VIEW

Usar tablas temporales

- Cuando estamos trabajando con tablas muy grandes, suele suceder que ocasionalmente necesitemos ejecutar algunas consultas sobre un pequeño subconjunto de una gran cantidad de datos.

- Crear una tabla temporal es tan sencillo como agregar la palabra TEMPORARY a una sentencia típica CREATE TABLE

- Una tabla temporal existe mientras dure la conexión a MySQL. Cuando se interrumpe la conexión MySQL remueve automáticamente la tabla y libera el espacio que ésta usaba.

Usar tablas temporales

```
CREATE TEMPORARY TABLE tabla_temp
(
   campo1 tipoDato,
   campo2 tipoDeDato,
   ...
) TYPE = HEAP;
```

- MySQL también permite especificar que una tabla temporal sea creada en memoria si dicha tabla se declara del tipo HEAP