

MySQL

09_ Vistas



SQL_ Vistas

Contenido

1. Vistas	4
2. Vistas de vistas	10
3. Vistas actualizables	13

SQL_ Vistas

Introducción

En MySQL, una vista es una tabla virtual que se define mediante una consulta SQL. Es decir, una vista es una representación lógica de los datos en una o varias tablas, que se puede utilizar como si fuera una tabla física.

Las vistas en MySQL se utilizan para simplificar las consultas SQL y ocultar la complejidad de las consultas subyacentes. Por ejemplo, se puede crear una vista que contenga los datos de varias tablas y utilizar esta vista en lugar de escribir una consulta SQL compleja que una las tablas.

Las vistas se pueden utilizar para restringir el acceso a ciertos datos. Por ejemplo, se puede crear una vista que muestre sólo los datos relevantes para un determinado usuario y concederle permisos sólo para acceder a esa vista en lugar de a la tabla subyacente.

Las vistas también pueden ser útiles para mejorar el rendimiento de las consultas. Por ejemplo, se puede crear una vista que contenga los resultados de una consulta compleja y utilizar esta vista en lugar de ejecutar la consulta cada vez que se necesiten los resultados.

Las vistas en MySQL se crean utilizando la sentencia `CREATE VIEW`, que permite definir la consulta que define la vista. Una vez creada la vista, se puede utilizar en consultas SQL de la misma manera que se utiliza una tabla física.

En resumen, una vista en MySQL es una tabla virtual que se define mediante una consulta SQL. Las vistas se utilizan para simplificar las consultas SQL, ocultar la complejidad de las consultas subyacentes, restringir el acceso a ciertos datos y mejorar el rendimiento de las consultas. Las vistas se crean utilizando la sentencia `CREATE VIEW` y se utilizan en consultas SQL de la misma manera que se utiliza una tabla física.



SQL_ Vistas

1. Vistas

Una vista es una alternativa para mostrar datos de una o varias tablas. Una vista es como una tabla virtual que almacena una consulta.

Entonces, una vista almacena una consulta como un objeto para utilizarse posteriormente. Las tablas consultadas en una vista se llaman tablas base. En general, se puede dar un nombre a cualquier consulta y almacenarla como una vista.

Una vista suele llamarse también tabla virtual porque los resultados que retorna y la manera de referenciarlas es la misma que para una tabla.

Las vistas permiten:

- ocultar información: permitiendo el acceso a algunos datos y manteniendo oculto el resto de la información que no se incluye en la vista. El usuario opera con los datos de una vista como si se tratara de una tabla, pudiendo modificar en algunos casos tales datos.
- simplificar la administración de los permisos de usuario: se pueden dar al usuario permisos para que solamente pueda acceder a los datos a través de vistas, en lugar de concederle permisos para acceder a ciertos campos, así se protegen las tablas base de cambios en su estructura.
- mejorar el rendimiento: se puede evitar tipear instrucciones repetidamente almacenando en una vista el resultado de una consulta compleja que incluya información de varias tablas.

Podemos crear vistas con: un subconjunto de registros y campos de una tabla; una unión de varias tablas; una combinación de varias tablas; un resumen estadístico de una tabla; un subconjunto de otra vista, combinación de vistas y tablas.

Una vista se define usando un "select".

La sintaxis básica parcial para crear una vista es la siguiente:

```
create view NOMBREVISTA as SENTENCIAS
SELECT from TABLA;
```

El contenido de una vista se muestra con un "select":

```
select * from NOMBREVISTA;
```

En el siguiente ejemplo creamos la vista "vista_empleados", que es resultado de una combinación en la cual se muestran 4 campos:

```
create view vista_empleados as
select concat(apellido,' ',e.nombre) as nombre,
       sexo,
       s.nombre as seccion,
       cantidadhijos
from empleados as e
join secciones as s on codigo=seccion;
```

SQL_ Vistas

Para ver la información contenida en la vista creada anteriormente tipeamos:

```
select nombre, seccion, cantidadhijos from vista_empleados;
```

Podemos realizar consultas a una vista como si se tratara de una tabla:

```
select seccion,count(*) as cantidad  
from vista_empleados;
```

Los nombres para vistas deben seguir las mismas reglas que cualquier identificador. Para distinguir una tabla de una vista podemos fijar una convención para darle nombres, por ejemplo, colocar el sufijo "vista" y luego el nombre de las tablas consultadas en ellas.

Los campos y expresiones de la consulta que define una vista DEBEN tener un nombre. Se debe colocar nombre de campo cuando es un campo calculado o si hay 2 campos con el mismo nombre. Note que en el ejemplo, al concatenar los campos "apellido" y "nombre" colocamos un alias; si no lo hubiésemos hecho aparecería un mensaje de error porque dicha expresión DEBE tener un encabezado, MySQL no lo coloca por defecto.

Los nombres de los campos y expresiones de la consulta que define una vista DEBEN ser únicos (no puede haber dos campos o encabezados con igual nombre). Note que en la vista definida en el ejemplo, al campo "s.nombre" le colocamos un alias porque ya había un encabezado (el alias de la concatenación) llamado "nombre" y no pueden repetirse, si sucediera, aparecería un mensaje de error.

Al crear una vista, MySQL verifica que existan las tablas a las que se hacen referencia en ella.

Se aconseja probar la sentencia "select" con la cual definiremos la vista antes de crearla para asegurarnos que el resultado que retorna es el imaginado.

Otra sintaxis para definir una vista es la siguiente:

```
create view NOMBREVISTA (NOMBRESDEENCABEZADOS)  
as  
SENTENCIASSELECT  
from TABLA;
```

Creamos otra vista de "empleados" denominada "vista_empleados_ingreso" que almacena la cantidad de empleados por año:

```
create view vista_empleados_ingreso(fecingreso,cantidad) as  
select extract(year from fechaingreso) as fecingreso,  
       count(*) as cantidad  
from empleados  
group by fecingreso;
```

La diferencia es que se colocan entre paréntesis los encabezados de las columnas que aparecerán en la vista. Nos facilita la lectura de una vista.

SQL_ Vistas

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para probar las vistas:

```
drop table if exists empleados;
drop table if exists secciones;

create table secciones(
  codigo int auto_increment primary key,
  nombre varchar(30),
  sueldo decimal(5,2)
);

create table empleados(
  legajo int primary key auto_increment,
  documento char(8),
  sexo char(1),
  apellido varchar(40),
  nombre varchar(30),
  domicilio varchar(30),
  seccion int not null,
  cantidadhijos int,
  estadocivil char(10),
  fechaingreso date
);

insert into secciones(nombre,sueldo) values('Administracion', 300);
insert into secciones(nombre,sueldo) values('Contaduría', 400);
insert into secciones(nombre,sueldo) values('Sistemas', 500);

insert
into
empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values ('22222222','f','Lopez','Ana','Colon 123',1,2,'casado','1990-10-10');
insert
into
empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('23333333','m','Lopez','Luis','Sucre 235',1,0,'soltero','1990-02-10');
insert
into
empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('24444444','m','Garcia','Marcos','Sarmiento 1234',2,3,'divorciado','1998-07-12');
insert
into
empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('25555555','m','Gomez','Pablo','Bulnes 321',3,2,'casado','1998-10-09');
insert
into
empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('26666666','f','Perez','Laura','Peru 1254',3,3,'casado','2000-05-09');

drop view if exists vista_empleados;
```

SQL_ Vistas

```
create view vista_empleados as
select concat(apellido, ' ', e.nombre) as nombre,
       sexo,
       s.nombre as seccion,
       cantidadhijos
from empleados as e
join secciones as s on codigo=seccion;

select nombre, seccion, cantidadhijos from vista_empleados;

select seccion, count(*) as cantidad
from vista_empleados;

select seccion, count(*) as cantidad
from vista_empleados
group by seccion;

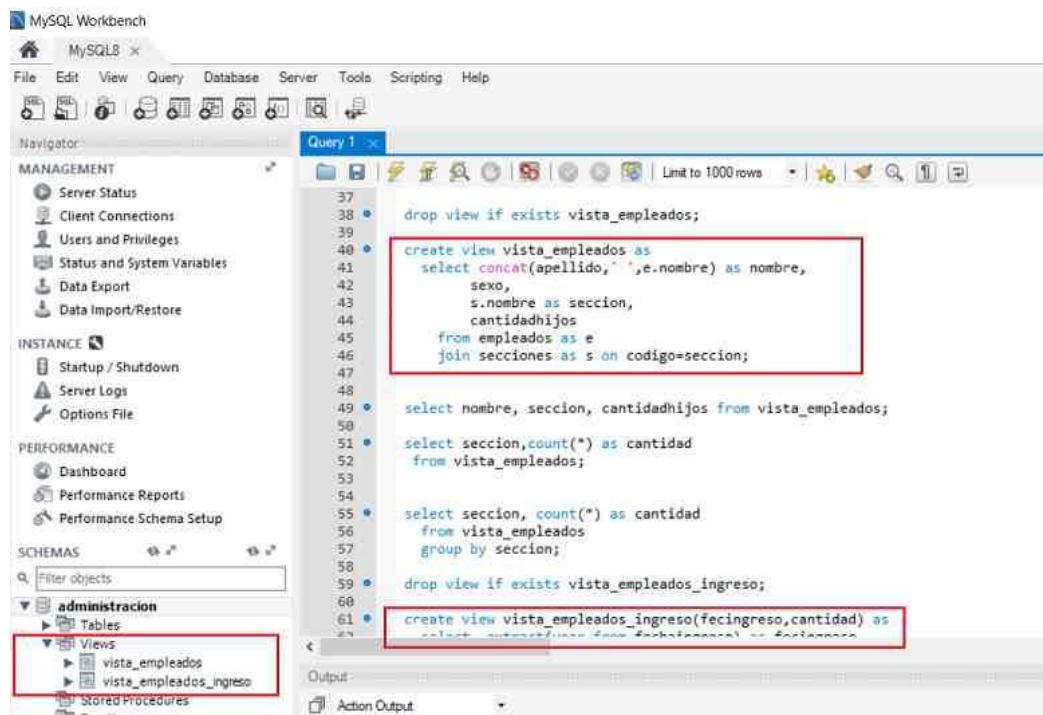
drop view if exists vista_empleados_ingreso;

create view vista_empleados_ingreso(fecingreso, cantidad) as
select extract(year from fechaingreso) as fecingreso,
       count(*) as cantidad
from empleados
group by fecingreso;

select fecingreso, cantidad from vista_empleados_ingreso;
```

SQL_ Vistas

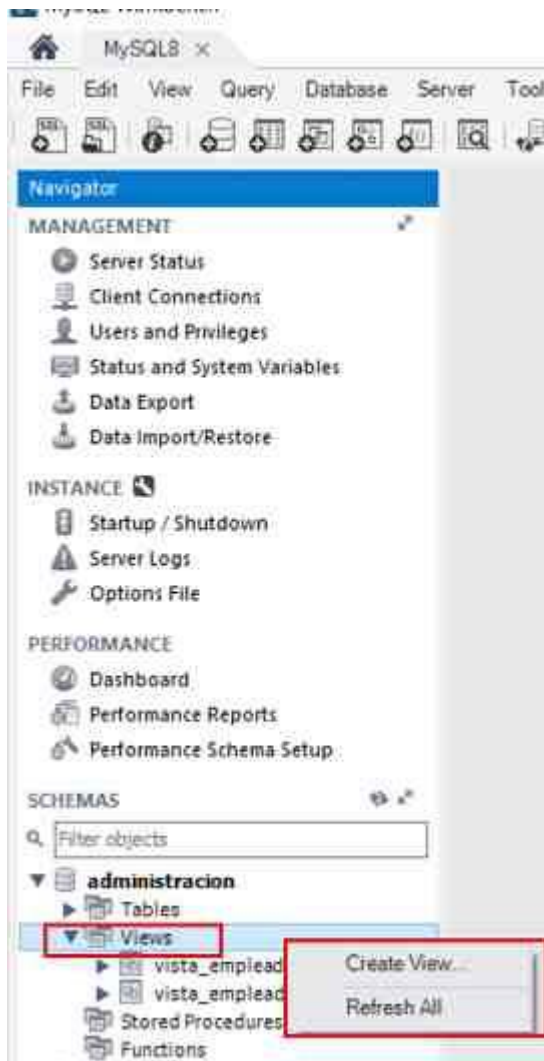
Cuando se crean vistas podemos consultarlas desde el "Workbench":



"Workbench" suministra un editor especializado para codificar una vista:

Si presionamos con el botón derecha del mouse sobre "view" podemos seleccionar "Create view" y proceder a codificar en forma individual una vista (de forma similar podemos crear una tabla):

SQL_ Vistas



SQL_ Vistas

2. Vistas basadas en otras vistas

En MySQL podemos crear una vista y dentro del comando 'select' que definimos cuando creamos la vista podemos hacer referencia a otra vista ya existente.

En el concepto anterior creamos la vista 'vista_empleados' con la siguiente sintaxis:

```
create view vista_empleados as
select concat(apellido,' ',e.nombre) as nombre,
       sexo,
       s.nombre as seccion,
       cantidadhijos
from empleados as e
join secciones as s on codigo=seccion;
```

Podemos crear una nueva vista basada en la vista 'vista_empleados' que nos retorne todos los empleados que tienen hijos:

```
create view vista_empleados_con_hijos as
select nombre,
       sexo,
       seccion,
       cantidadhijos
from vista_empleados
where cantidadhijos>00;
```

Como vemos en la cláusula from hacemos referencia a la vista ya existente llamada 'vista_empleados'.

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists empleados;
drop table if exists secciones;

create table secciones(
codigo int auto_increment primary key,
nombre varchar(30),
sueldo decimal(5,2)
);

create table empleados(
legajo int primary key auto_increment,
documento char(8),
sexo char(1),
apellido varchar(40),
nombre varchar(30),
domicilio varchar(30),
seccion int not null,
```

SQL_ Vistas

```
cantidadhijos int,
estadocivil char(10),
fechaingreso date
);

insert into secciones(nombre,sueldo) values('Administracion', 300);
insert into secciones(nombre,sueldo) values('Contaduría', 400);
insert into secciones(nombre,sueldo) values('Sistemas', 500);

insert                                into                                empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values ('22222222','f','Lopez','Ana','Colon 123',1,2,'casado','1990-10-10');
insert                                into                                empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('23333333','m','Lopez','Luis','Sucre 235',1,0,'soltero','1990-02-10');
insert                                into                                empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('24444444','m','Garcia','Marcos','Sarmiento 1234',2,3,'divorciado','1998-07-12');
insert                                into                                empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('25555555','m','Gomez','Pablo','Bulnes 321',3,2,'casado','1998-10-09');
insert                                into                                empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechaingreso)
values('26666666','f','Perez','Laura','Peru 1254',3,3,'casado','2000-05-09');

drop view if exists vista_empleados;

create view vista_empleados as
select concat(apellido,' ',e.nombre) as nombre,
       sexo,
       s.nombre as seccion,
       cantidadhijos
from empleados as e
join secciones as s on codigo=seccion;

select nombre, seccion, cantidadhijos from vista_empleados;

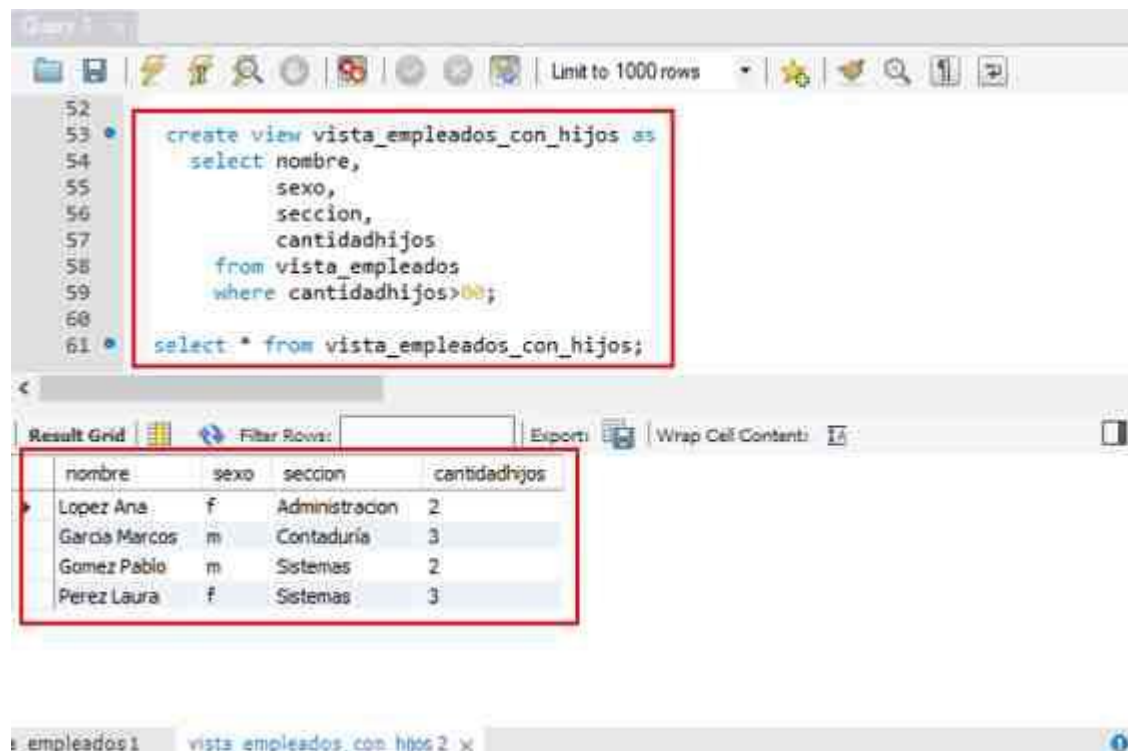
drop view if exists vista_empleados_con_hijos;

create view vista_empleados_con_hijos as
select nombre,
       sexo,
       seccion,
       cantidadhijos
from vista_empleados
where cantidadhijos>00;
```

SQL_ Vistas

```
select * from vista_empleados_con_hijos;
```

Genera una salida similar a esta:



The screenshot shows a SQL IDE interface. The top pane displays SQL code for creating and querying a view. The bottom pane shows the result grid of the query.

```
52  
53 • create view vista_empleados_con_hijos as  
54     select nombre,  
55            sexo,  
56            seccion,  
57            cantidadhijos  
58     from vista_empleados  
59     where cantidadhijos > 0;  
60  
61 • select * from vista_empleados_con_hijos;
```

The result grid shows the following data:

nombre	sexo	seccion	cantidadhijos
Lopez Ana	f	Administracion	2
Garcia Marcos	m	Contaduria	3
Gomez Pablo	m	Sistemas	2
Perez Laura	f	Sistemas	3

SQL_ Vistas

3. Vistas actualizables: insert, update y delete

En MySQL se puede utilizar la vista para efectuar a través de ésta inserciones, modificaciones y borrados de la tabla base de la vista.

Es importante tener en cuenta que si se modifican los datos de una vista, se modifica la tabla base.

Se puede insertar, actualizar o eliminar datos de una tabla a través de una vista, teniendo en cuenta lo siguiente, las modificaciones que se realizan a las vistas:

- No pueden afectar a más de una tabla consultada.
- No se pueden cambiar los campos resultado de un cálculo.
- La vista no puede tener funciones de agrupamiento (count - max - min - sum - avg)
- No puede tener la cláusula distinct, left join, outer join, union, group by, having.
- No puede tener subconsultas en la cláusula select.

La vista debe ser bastante sencilla para luego sea actualizable.

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL donde analizamos una vista actualizable:

```
drop table if exists alumnos;
drop table if exists profesores;

create table alumnos(
  documento char(8),
  nombre varchar(30),
  nota decimal(4,2),
  codigoprofesor int,
  primary key(documento)
);

create table profesores (
  codigo int auto_increment,
  nombre varchar(30),
  primary key(codigo)
);

insert into alumnos values('30111111','Ana Algarbe', 5.1, 1);
insert into alumnos values('30222222','Bernardo Bustamante', 3.2, 1);
insert into alumnos values('30333333','Carolina Conte',4.5, 1);
insert into alumnos values('30444444','Diana Dominguez',9.7, 1);
insert into alumnos values('30555555','Fabian Fuentes',8.5, 2);
insert into alumnos values('30666666','Gaston Gonzalez',9.70, 2);
```

SQL_ Vistas

```
insert into profesores(nombre) values ('Maria Luque');
insert into profesores(nombre) values ('Jorje Dante');

drop view if exists vista_nota_alumnos_aprobados;

-- Creamos una vista con los datos de todos los alumnos que tienen
-- una nota mayor o igual a 7, junto con el nombre del profesor que
-- lo calificó
create view vista_nota_alumnos_aprobados as
select documento,
       a.nombre as nombrealumno,
       p.nombre as nombreprofesor,
       nota,
       codigoprofesor
from alumnos as a
join profesores as p on a.codigoprofesor=p.codigo
where nota>=7;

select * from vista_nota_alumnos_aprobados;

-- Mediante la vista insertamos un nuevo alumno calificado por el profesor
-- con código 1
insert into vista_nota_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor)
values('99999999','Rodriguez Pablo', 10, 1);

select * from vista_nota_alumnos_aprobados;

-- si consultamos la tabla base: alumnos tenemos una nueva fila con el alumno
-- insertado
select * from alumnos;

-- modificamos la nota de un alumno aprobado mediante la vista
update vista_nota_alumnos_aprobados set nota=10
where documento='30444444';

select * from alumnos;
```

SQL_ Vistas

Acotaciones

Si efectuamos un insert mediante la vista creada e insertamos un alumno con una nota inferior a 7 luego dicha fila se inserta en la tabla base pero no se visualiza en la vista:

```
insert vista_nota_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor)
values('88888888','Laura Robles', 3, 1);

-- se cargó en la tabla 'alumnos'
select * from alumnos;

-- no se visualiza en la vista
select * from vista_nota_alumnos_aprobados;
```

Para evitar este tipo de inconsistencias se ha creado la cláusula 'with check option'. Si agregamos ésta cláusula cuando creamos la vista luego no se harán inserciones, borrados o actualizaciones cuando los cambios no se visualizan en la vista.

Probemos de crear ahora la vista con la cláusula 'with check option' y tratemos de insertar una fila que no se visualiza en la vista:

```
drop table if exists alumnos;
drop table if exists profesores;

create table alumnos(
  documento char(8),
  nombre varchar(30),
  nota decimal(4,2),
  codigoprofesor int,
  primary key(documento)
);

create table profesores (
  codigo int auto_increment,
  nombre varchar(30),
  primary key(codigo)
);

insert into alumnos values('30111111','Ana Algarbe', 5.1, 1);
insert into alumnos values('30222222','Bernardo Bustamante', 3.2, 1);
insert into alumnos values('30333333','Carolina Conte',4.5, 1);
insert into alumnos values('30444444','Diana Dominguez',9.7, 1);
insert into alumnos values('30555555','Fabian Fuentes',8.5, 2);
insert into alumnos values('30666666','Gaston Gonzalez',9.70, 2);

insert into profesores(nombre) values ('Maria Luque');
insert into profesores(nombre) values ('Jorje Dante');
```

SQL_ Vistas

```
drop view if exists vista_nota_alumnos_aprobados;

create view vista_nota_alumnos_aprobados as
select documento,
       a.nombre as nombrealumno,
       p.nombre as nombreprofesor,
       nota,
       codigoprofesor
from alumnos as a
join profesores as p on a.codigoprofesor=p.codigo
where nota>=7
with check option;

-- Se genera error ya que luego este alumno no aparecerá en la vista
update vista_nota_alumnos_aprobados set nota=1
where documento='30444444';

-- Se genera un error
insert vista_nota_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor)
values('73777777','Raquel Montes', 3, 1);

-- Se efectúa la inserción en forma correcta
insert vista_nota_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor)
values('73777777','Raquel Montes', 7, 1);

select * from vista_nota_alumnos_aprobados;
```


SQL_ Vistas

Se produce un error al tratar de insertar un alumno con nota menor a 7:

The screenshot shows a SQL IDE interface. The top pane displays a query with the following SQL code:

```
41 with check option;
42
43 -- Se genera error ya que luego este alumno no aparecerá en la vista
44 update vista_notas_alumnos_aprobados set nota=1
45 where documento='30444444';
46
47 -- Se genera un error
48 insert vista_notas_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor)
49 values('73777777', 'Raquel Montes', 3, 1);
50
51 -- Se efectúa la inserción en forma correcta
52 insert vista_notas_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor)
53 values('73777777', 'Raquel Montes', 7, 1);
54
55 select * from vista_notas_alumnos_aprobados;
```

The bottom pane shows the 'Result Grid' with the following data:

documento	nombrealumno	nombreprofesor	nota	codigoprofesor
30444444	Diana Dominguez	Maria Luque	9.70	1
30555555	Fabian Fuentes	Jorge Dante	8.50	2
30666666	Gaston Gonzalez	Jorge Dante	9.70	2
73777777	Raquel Montes	Maria Luque	7.00	1

The bottom pane also shows the 'Action Output' with the following log:

#	Time	Action	Message
13	11:07:34	drop view if exists vista_notas_alumnos_aprobados	0 row(s) affected
14	11:07:34	create view vista_notas_alumnos_aprobados as select documento, nombrealumno, nota, codigoprofesor from alumnos_aprobados	0 row(s) affected
15	11:07:34	update vista_notas_alumnos_aprobados set nota=1 where documento='30444444'	Error Code: 1369. CHECK OPTION violated
16	11:07:34	insert vista_notas_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor) values('73777777', 'Raquel Montes', 3, 1);	Error Code: 1369. CHECK OPTION violated
17	11:07:34	insert vista_notas_alumnos_aprobados(documento, nombrealumno, nota, codigoprofesor) values('73777777', 'Raquel Montes', 7, 1);	1 row(s) affected
18	11:07:34	select * from vista_notas_alumnos_aprobados LIMIT 0, 1000	4 row(s) returned