tisBDdocs

Z01 Adicional: vistas y tablas temporales

En esta sesión vamos a mostrar dos tipos de objetos de uso habitual en cualquier SGBD: las vistas y las tablas temporales. Mientras que las primeras son objetos persistentes que nos permiten resumir consultas complejas y de uso frecuente, las segundas, como su nombre indica, son objetos de corta vida, durante una sesión o una simple consulta.

CONTENIDOS

Vistas

Para esta sección es necesario conectarse al servidor con un usuario con permisos para crear vistas.

Una vista es un objeto que se define a partir de una consulta y que se comporta como una tabla si bien, dependiendo de la consulta en la que se basa, se pueden hacer más o menos cosas: consultar una vista siempre será posible pero insertar o borrar filas en una vista o modificar un valor ya depende de cómo sea esa definición.

Para crear una vista disponemos de la orden:

```
CREATE VIEW nombrevista AS consulta
```

Una vista es un objeto persistente, por lo tanto, para eliminarla del catálogo, hay que ejecutar:

```
DROP VIEW nombrevista
```

También se puede consultar la definición de una vista ya definida (aunque esto ya es particular de MySQL, en otros gestores tienen sus propios métodos):

```
SHOW CREATE VIEW nombrevista
```

Toda la información sobre vistas en MySQL se puede consultar en http://dev.mysql.com/doc/refman/5.0/en/create-view.html.

Esta sesión se ha planteado en formato demostración por lo que se recomienda ejecutar una a una las siguientes sentencias y buscar los motivos por los que unas sentencias se ejecutan sin problemas y otras no.

Tras conectarse al servidor hay que elegir una base de datos en la que se nos permita crear vistas:

use mibd;

Vamos a copiar los contenidos de algunas tablas de TiendaOnLine a tablas locales que sean de nuestra propiedad

```
drop table if exists mitv;
drop table if exists miarticulo;

create table miarticulo (
cod varchar(7),
nombre varchar(45),
pvp decimal(7,2),
marca varchar(15),
imagen blob,
urlimagen varchar(100),
especificaciones text,
primary key (cod)) engine=innodb;

insert into miarticulo select * from tiendaonline.articulo;

create table mitv (
cod varchar(7),
panel varchar(45),
pantalla smallint(6),
resolucion varchar(15),
hdreadyfullhd varchar(6),
tdt tinyint(1),
primary key (cod),
foreign key (cod) references miarticulo (cod)) engine=innodb;

insert into mitv select * from tiendaonline.tv;
```

Creamos la primera vista:

```
create view vma as select cod,nombre,pvp from miarticulo;
```

Una vista se comporta como una tabla y puede consultarse.

```
select * from vma where pvp between 500 and 700;
```

Una vista se puede generar a partir de cualquier consulta, y tiene la característica añadida de poder restringir el acceso a solo un subconjunto de las filas posibles.

```
create view vmb as select cod,nombre,pvp from miarticulo
where pvp between 500 and 700;

select * from vmb;

(1) select cod,pvp from vmb where cod in (select cod from
mitv) order by pvp;
```

Insertando

Una vista, bajo ciertas condiciones, permite insertar nuevos datos.

```
(2) insert into vma (cod,nombre,pvp) values
('B001','MiArtículo',499);

select * from miarticulo where cod='B001';
select * from vma where cod='B001';
select * from vmb where cod='B001';


(3) insert into vmb (cod,nombre,pvp) values
('B002','MiOtroArtículo',701);

select * from vmb where cod='B002';
select * from miarticulo where cod='B002';
select * from vma where cod='B002';
```

Borrando

Creamos otra vista para facilitar la comprobación de las acciones que solicitamos. En este caso vamos a intentar eliminar filas.

```
create view bart as select * from miarticulo where cod
like 'B%';

select * from bart;

(4) delete from vma where cod='B001';
```

```
(5) delete from vmb where cod='B002';

select * from bart;
```

Modificando

Ahora comprobaremos la orden `update`. Esta orden tiene las mismas restricciones que `delete`.

```
select * from bart;

(6) update vma set pvp = 800 where cod='B002';

select * from bart;

(7) update vmb set pvp = 600 where cod='B002';

select * from bart;
```

Con más de una tabla

La vista se puede definir sobre varias tablas, pero eso afecta a las órdenes que se pueden ejecutar y en qué condiciones.

```
create view vat as
  select a.cod,nombre,pvp,resolucion,tdt
  from miarticulo a, mitv t
  where a.cod=t.cod
    and pvp between 800 and 1200;


select * from vat order by pvp desc;


(8) insert into vat values ('B003','OtroMás',1100,null,null);
(9) insert into vat (cod,nombre,pvp,resolucion,tdt) values
('B003','OtroMás',1100,null,null);
(10) insert into vat (cod,nombre,pvp) values
('B003','OtroMás',1100);

select * from vat;
select * from bart;


(11) insert into vat (resolucion,tdt) values ('800x600',1);
(12) insert into vat (cod,resolucion,tdt) values
('B004','800x600',1);
```

```
(13) delete from vat where cod='A0694';

select * from vat;
select * from bart;

(14) update vat set pvp = 999 where cod='A0694';
(15) update vat set resolucion = '800x600',tdt=1 where
cod='A0694';
(16) update vat set pvp = 850, resolucion = 'ninguna',tdt=1
where cod='A0694';


select * from vat;


(17) update vat set cod = 'B004' where cod='A0694';

select * from vat;
select * from bart;
select * from mitv where cod='A0694';
```

With check option

Vamos redefinir `bart` y crear otra vista `bart2`.

```
drop view if exists bart2;
drop view if exists bart;


create view bart as select * from miarticulo where cod
like 'B%';
(18) create view bart2 as select * from miarticulo where
cod like 'B%' with check option;


select * from bart;
select * from bart2;


insert into bart2 (cod,nombre,pvp) values
('B010','Artículo B10',1999);
insert into bart2 (cod,nombre,pvp) values
('C010','Artículo C10',1999);


select * from bart;
select * from bart2;
select * from miarticulo where pvp=1999;
```

```
insert into bart (cod,nombre,pvp) values
('C010','Artículo C10',1999);
select * from bart;
select * from bart2;
select * from miarticulo where pvp=1999;
```

Finalmente, eliminamos todas las vistas y tablas creadas.

```
drop view bart;
drop view bart2;
drop view vat;
drop view vmb;
drop view vma;
drop table mitv;
drop table miarticulo;
```

El modificador WITH CHECK OPTION obliga a cualquier operación que se haga sobre la vista a cumplir las condiciones del *where*: si "bart2" no tuviera esa opción, podríamos insertar en la vista cualquier artículo que, finalmente, se almacenaría en "miarticulo", el artículo existiría en mi base de datos pero no podría verlo por la definición de "bart2"; con WITH CHECK OPTION, el sistema no me deja insertar más que artículos cuyo "cod" empiece por 'B'.

Tablas temporales

Nos hemos tomado la libertad de considerar tablas temporales tanto las tablas temporary en MySQL, como las subconsultas utilizadas como columna en la select o como tabla en el from.

Se entiende por tabla temporal aquella que se crea y utiliza en un contexto limitado, bien sea una orden concreta de SQL (el caso genérico de las subconsultas) o una sesión o conexión. Las tablas `TEMPORARY`, por ejemplo, son objetos que desaparecen automáticamente cuando se cierra la sesión de usuario. Una subconsulta es accesible solo mientras se ejecuta una determinada orden. En esta lección solo vamos a tratar los casos de subconsultas, las tablas `TEMPORARY` no necesitan más explicación.

Subconsultas como columnas calculadas

Una consulta puede utilizarse como una columna más de cualquier consulta, solo hay que tener la precaución de darle nombre. Aunque MySQL no lo necesita realmente, parece conveniente utilizar la palabra reservada AS para mejorar la comprensión de cada parte de la consulta. Por ejemplo, la siguiente orden genera una tabla con 2 columnas a partir de subconsultas completas:

```
use tiendaonline;
select
```

```
       (select max(pvp) from articulo) as mart,
       (select max(precio) from linped) as mlinped;
```

En estos casos se utilizan las subconsultas para realizar cálculos sobre ellas y dentro de una consulta sobre un cierto conjunto de filas:

```
select cod,pvp,
       pvp/(select max(pvp) from articulo)*100 as
tpcpvp,
       pvp/(select max(precio) from linped)*100 as
tpcprecio
from articulo
where pvp between 1000 and 1200;
```

Subconsultas en el `from`

A veces es necesario incluir una subconsulta como una tabla más a enlazar en otra consulta de orden superior. Nuevamente, hay que darle nombre a esa tabla temporal para poder hacer referencia a ella. En este caso, aprovechamos la consulta anterior para utilizar en otra más compleja:

```
select tv.cod,resolucion,tdt,tpcpvp
from tv,
  (select cod,pvp,
       pvp/(select max(pvp) from articulo)*100 as
tpcpvp,
       pvp/(select max(precio) from linped)*100 as
tpcprecio
   from articulo
   where pvp between 1000 and 1200) as calc
where tv.cod=calc.cod;
```

Invertir una tabla

Lo que en otros sistemas se conoce como columnas PIVOT (en SQL Server, por ejemplo) consiste en construir columnas a partir de la información contenida en las filas. Aquí vamos a calcular cuántos pedidos se han realizado en los meses de octubre, noviembre y diciembre de cualquier año. Antes, vamos a comprobar los datos de los que disponemos:

```
select month(fecha) mes, count(*)
from pedido
where month(fecha) in (10,11,12)
```

```
group by month(fecha)
order by month(fecha);
```

Lo que queremos conseguir es que los meses se conviertan en columnas. Para ello:

```
select
    (select count(*)
     from pedido
     where month(fecha) =10) as octubre,
    (select count(*)
     from pedido
     where month(fecha) =11) as noviembre,
    (select count(*)
     from pedido
     where month(fecha) =12) as diciembre;
```

Complicándolo un poco, veamos la cantidad de veces que se solicita cada artículo en cada uno de esos mismos meses:

```
select articulo, month(fecha) mes, count(*) veces
from pedido p join linped l on (p.numpedido=l.numpedido)
where month(fecha) in (10,11,12)
group by articulo,month(fecha)
order by articulo,month(fecha);
```

Con la misma intención que antes, convertiremos los meses en columnas haciendo uso de subconsultas:

```
select distinct articulo,
    (select count(*)
     from pedido p join linped l on
(p.numpedido=l.numpedido)
     where month(fecha) =10 and l.articulo=ll.articulo)
octubre,
    (select count(*)
     from pedido p join linped l on
(p.numpedido=l.numpedido)
     where month(fecha) =11 and l.articulo=ll.articulo)
noviembre,
    (select count(*)
     from pedido p join linped l on
```

```
(p.numpedido=l.numpedido)
     where month(fecha) =12 and l.articulo=ll.articulo)
diciembre
from linped ll;
```

La misma consulta pero variando ligeramente los orígenes de los datos, lo que evita el uso del `distinct`. Nótese que ahora las subconsultas hacen referencia a la tabla del `select` externo:

```
select cod as articulo,
  (select count(*)
   from pedido p join linped l on
(p.numpedido=l.numpedido)
   where month(fecha) =10 and l.articulo=cod) octubre,
  (select count(*)
   from pedido p join linped l on
(p.numpedido=l.numpedido)
   where month(fecha) =11 and l.articulo=cod) noviembre,
  (select count(*)
   from pedido p join linped l on
(p.numpedido=l.numpedido)
   where month(fecha) =12 and l.articulo=cod) diciembre
from articulo
where cod in (select articulo
              from pedido p join linped l on
(p.numpedido=l.numpedido)
              where month(fecha) in (10,11,12));
```

Tablas TEMPORARY

Aprovechamos la consulta anterior para crear una tabla temporal. Recuérdese que esta tabla desaparecerá del sistema en cuanto cerremos la conexión al servidor. Aquí se define a partir de una consulta pero estas tablas se pueden crear como cualquier otra con la orden `CREATE TABLE`.

```
create temporary table mitabla
    select cod articulo,
      (select count(*)
       from pedido p join linped l on
(p.numpedido=l.numpedido)
       where month(fecha) =10 and l.articulo=cod) octubre,
      (select count(*)
       from pedido p join linped l on
(p.numpedido=l.numpedido)
```

```
        where month(fecha) =11 and l.articulo=cod) noviembre,
      (select count(*)
       from pedido p join linped l on
(p.numpedido=l.numpedido)
       where month(fecha) =12 and l.articulo=cod) diciembre
    from articulo
    where cod in (select articulo
                  from pedido p join linped l on
(p.numpedido=l.numpedido)
                  where month(fecha) in (10,11,12));
```

Consultemos la tabla recién creada para comprobar su contenido

```
select * from mitabla where octubre>0 and noviembre>0;
```

No nos tenemos que preocupar de eliminarla puesto que ya lo hará el propio sistema.