RESOLUCIÓN PRACTICA SQL v 2016

Práctica Nº 7: Vistas

```
1)
CREATE VIEW `vw_instructores` AS
select  concat(i.nombre,' ',i.apellido) AS 'Nombre y Apellido', i.tel, i.email
from instructores i;

select * from vw_instructores;

2)

CREATE VIEW listado_cursos
AS

SELECT c.`nom_plan` 'Nombre del Plan', p.`desc_plan` 'Descripcion del Plan',
       c.`nro_curso` 'Nro. del Curso', c.`fecha_ini` 'Fecha Inicio',
       c.`fecha_fin` 'Fecha Fin', c.`salon`, c.`cupo`,
       count(*) 'Cantidad Alumnos'
FROM   `cursos` c inner join
       `plan_capacitacion` p on c.`nom_plan`=p.`nom_plan` inner join
       `inscripciones` i on c.`nom_plan`=i.`nom_plan`
                         and c.`nro_curso`=i.`nro_curso`
WHERE  year(c.fecha_ini)=year(CURRENT_DATE)
group by c.`nom_plan`, p.`desc_plan`, c.`nro_curso`, c.`fecha_ini`,
         c.`fecha_fin`, c.`salon`, c.`cupo`;


select * from listado_cursos;

3)
CREATE VIEW planesactuales

AS SELECT cur.nom_plan, desc_plan, cur.nro_curso, fecha_ini, fecha_fin, salon, cupo, count(
dni ) cantidad
FROM cursos cur
INNER JOIN plan_capacitacion plan ON cur.nom_plan = plan.nom_plan
INNER JOIN inscripciones insc ON cur.nro_curso = insc.nro_curso
WHERE fecha_ini >= "2009-01-01"
GROUP BY cur.nom_plan, desc_plan, cur.nro_curso, fecha_ini, fecha_fin, salon, cupo ;

CREATE TEMPORARY TABLE costoactplan

SELECT nom_plan, max( fecha_desde_plan ) max
FROM valores_plan
GROUP BY 1 ;

SELECT plan.nom_plan, desc_plan, nro_curso, fecha_ini, fecha_fin, salon, cupo, cantidad,
valor_plan
FROM planesactuales plan
INNER JOIN  costoactplan valores ON plan.nom_plan = valores.nom_plan
and plan.max = valores.fecha_desde_plan


4)

CREATE VIEW alumplan AS SELECT a.dni, concat( nombre, " ", apellido ) Nomyape, e.nom_plan,
e.nro_curso, avg( nota ) prom
```

```
FROM alumnos a
INNER JOIN inscripciones i ON a.dni = i.dni
INNER JOIN evaluaciones e ON e.dni = i.dni
AND e.nom_plan = i.nom_plan
AND e.nro_curso = i.nro_curso
GROUP BY 1 , 2, 3, 4



CREATE TEMPORARY TABLE impagos SELECT dni, count( * ) cantidad
FROM cuotas
WHERE fecha_pago IS NULL
GROUP BY 1 ;



SELECT Nomyape, nom_plan, nro_curso, prom, cantidad
FROM alumplan a
LEFT JOIN impagos i ON a.dni = i.dni
```

Práctica Nº 8: INSERT VALUES, UPDATE, DELETE y TRUNCATE

1)

```
start transaction;

INSERT INTO instructores
VALUES
( "44-44444444-4", "Daniel", "Tapia", "444-444444", "dotapia@gmail.com", "Ayacucho 4444",
NULL );
COMMIT;
```

2)

```
start transaction;

INSERT INTO plan_capacitacion
VALUES
( "Administrador de BD", "Instalación y configuración MySQL. Lenguaje SQL. Usuarios y
permisos", 300, "presencial");


INSERT INTO `plan_temas`
VALUES
( "Administrador de BD", "Instalación MySQL", "Distintas configuraciones de instalación"),
( "Administrador de BD", "Configuración DBMS", "Variables de entorno, su uso y
configuración"),
( "Administrador de BD", "Lenguaje SQL", " DML, DDL y TCL"),
( "Administrador de BD", "Usuarios y Permisos", "Permisos de usuarios y DCL");

INSERT INTO examenes
VALUES
( "Administrador de BD", 1),
( "Administrador de BD", 2),
( "Administrador de BD", 3),
("Administrador de BD", 4);
```

INSERT INTO examenes_temas

```
VALUES
( "Administrador de BD", "Instalación MySQL", 1),
( "Administrador de BD", "Configuración DBMS", 2),
( "Administrador de BD", "Lenguaje SQL", 3),
( "Administrador de BD", "Usuarios y Permisos", 4);

--Agrego los materiales nuevos:

INSERT INTO materiales (cod_material, desc_material, url_descarga, autores, tamanio,
fecha_creacion, cant_disponible, punto_pedido, cantidad_a_pedir )
VALUES
( "AP-010", "DBA en MySQL", "www.afatse.com.ar/apuntes?AP=010", "José Román", 2,
"2009/03/01", 0, 0, 0),
("AP-011", "SQL en MySQL", " www.afatse.com.ar/apuntes?AP=011", "Juan López", 3,
"2009/04/01", 0, 0, 0) ;

INSERT INTO materiales_plan
VALUES ( "Administrador de BD", "UT-001", 0),
("Administrador de BD", "UT-002", 0),
("Administrador de BD", "UT-003", 0),
("Administrador de BD", "UT-004", 0),
("Administrador de BD", "AP-010", 0) ,
("Administrador de BD", "AP-011", 0) ;

INSERT INTO valores_plan
VALUES ( "Administrador de BD", "2009/02/01", 150)

COMMIT;
```

UPDATE

```
3)
start transaction;

UPDATE cursos SET cupo = cupo * 1.25 WHERE cupo >=20
UPDATE cursos SET cupo = cupo * 1.50 WHERE cupo <20

COMMIT;

4)
start transaction;

UPDATE instructores SET cuil_supervisor = "44-44444444-4"
WHERE cuil IN("55-55555555-5","66-66666666-6")

COMMIT;

5)

SELECT cuil INTO @supervisor
FROM instructores
WHERE cuil = "44-44444444-4";
start transaction;


SELECT cuil INTO @intructor1
FROM instructores
WHERE cuil = "55-55555555-5";
```

```
SELECT cuil INTO @intructor2
FROM instructores
WHERE cuil = "66-66666666-6";


start transaction;

UPDATE instructores SET cuil_supervisor = @supervisor
WHERE cuil IN (@instructor1,@instructor2)


COMMIT;
```

6)

```
Select dni into @dni_v
WHERE nombre = "Victor" AND apellido = "Hugo";



 start transaction;


UPDATE alumnos SET direccion = "Italia 2323", tel = “23232323”
WHERE dni =@dni_v
COMMIT;
```

7)

```
start transaction;


DELETE FROM valores_plan WHERE nom_plan = "Administrador de BD" ;
DELETE FROM materiales_plan WHERE nom_plan = "Administrador de BD"
DELETE FROM examenes_temas WHERE nom_plan = "Administrador de BD"
DELETE FROM examenes WHERE nom_plan = "Administrador de BD"
DELETE FROM plan_temas WHERE nom_plan = "Administrador de BD"
DELETE FROM plan_capacitacion WHERE nom_plan = "Administrador de BD"

COMMIT;
```

8) Verifico que el cod_material a eliminar no esté utilizado en materiales plan, si existe, primero eliminar el cod_material de materiales_plan

```
SELECT *
FROM `materiales_plan`
WHERE cod_material IN ( "AP-008", "AP-009")
```

No existe ninguna fila, entonces elimino el material

```
start transaction;

DELETE FROM materiales
WHERE cod_material IN ("AP-008","AP-009")

COMMIT;
```

9)

Verificar si el instructor "44-44444444-4", tiene registros en cursos_instructores, evaluaciones y si existe como cuil_supervisor. En los dos primeros casos no se debería

eliminar, en el tercero debemos actualizar el cuil del supervisor con otro supervisor si está asignado o con valor nulo:

```
start transaction;

UPDATE instructores SET cuil_supervisor = NULL
WHERE cuil_supervisor = "44-44444444-4";
DELETE FROM instructores WHERE cuil = "44-44444444-4";

COMMIT;
```

10)

```
start transaction;

DELETE FROM inscripciones WHERE nom_plan = "Marketing 3" AND nro_curso =1

COMMIT;
```

11)
Verfico cuales son los instructores que tienen de supervisor a Elías

```
SELECT *
FROM `instructores`
WHERE cuil_supervisor = "99-99999999-9";


start transaction;


DELETE FROM INSTRUCTORES
WHERE cuil_supervisor = "99-99999999-9";

ROLLBACK;    --para poder resolver el ejercicio 12
```

12)

```
SELECT cuil
INTO @vcuil
FROM instructores
WHERE cuil = "99-99999999-9";

SELECT *
FROM `instructores`
WHERE cuil_supervisor = @vcuil ;


start transaction;

DELETE
FROM `instructores`
WHERE cuil_supervisor = @vcuil ;

COMMIT;
```

13) Verifico los apuntes de Erica de Forifregoro:

```
SELECT *
FROM materiales
WHERE autores LIKE "%Erica de Forifregoro%"
```

Verifico en qué planes se usa:

```
SELECT *
FROM `materiales_plan`
WHERE cod_material = "AP-006"
```

Elimino primero el cod_material en materiales_plan y luego en materiales

```
start transaction;

DELETE FROM `materiales_plan` WHERE cod_material = "AP-006";

DELETE FROM materiales WHERE autores LIKE "%Erica de Forifregoro%

COMMIT;
```

Práctica Nº 9: INSERT SELECT, UPDATE y DELETE con JOINS

Práctica en clase: 1 a 7

Práctica recomendada:

1)

```
start transaction;


insert into `valores_plan`( `nom_plan`, `fecha_desde_plan`, `valor_plan`)

select val.`nom_plan`,'20090601', val.`valor_plan`*1.2

from valores_plan val

inner join

(

 select vp.`nom_plan`, max(vp.`fecha_desde_plan`) ult_fecha

 from valores_plan vp

 group by vp.`nom_plan`

) fechas

        on val.`nom_plan`=fechas.nom_plan

        and val.`fecha_desde_plan`=fechas.ult_fecha;


commit;
```

2)

Solución A.

```
start transaction;


insert into `valores_plan`( `nom_plan`, `fecha_desde_plan`, `valor_plan`)
select val.`nom_plan`,'20090801',
                                case
                                    when val.`valor_plan`<90 then val.`valor_plan`*1.2
                                    else val.`valor_plan`*1.12
                                end
from valores_plan val
inner join
(
 select vp.`nom_plan`, max(vp.`fecha_desde_plan`) ult_fecha
 from valores_plan vp
 group by vp.`nom_plan`
) fechas
        on val.`nom_plan`=fechas.nom_plan
        and val.`fecha_desde_plan`=fechas.ult_fecha;


commit;
```

Solución B.

```
-- Atencion: El orden es importante
start transaction;


insert into `valores_plan`( `nom_plan`, `fecha_desde_plan`, `valor_plan`)
select val.`nom_plan`,'20090801', val.`valor_plan`*1.12
from valores_plan val
inner join
(
 select vp.`nom_plan`, max(vp.`fecha_desde_plan`) ult_fecha
 from valores_plan vp
```

```
 group by vp.`nom_plan`
) fechas
        on val.`nom_plan`=fechas.nom_plan
        and val.`fecha_desde_plan`=fechas.ult_fecha
where val.`valor_plan`>=90;

insert into `valores_plan`( `nom_plan`, `fecha_desde_plan`, `valor_plan`)
select val.`nom_plan`,'20090801', val.`valor_plan`*1.2
from valores_plan val
inner join
(
 select vp.`nom_plan`, max(vp.`fecha_desde_plan`) ult_fecha
 from valores_plan vp
 group by vp.`nom_plan`
) fechas
        on val.`nom_plan`=fechas.nom_plan
        and val.`fecha_desde_plan`=fechas.ult_fecha
where val.`valor_plan`<90;

commit;

3)
start transaction;
insert into plan_capacitacion
select 'Marketing 1 Presen', desc_plan,hs,'presencial'
from `plan_capacitacion`
where nom_plan= 'Marketing 1';

insert into plan_temas
select 'Marketing 1 Presen', titulo,detalle
from plan_temas
where nom_plan= 'Marketing 1';
```

```
insert into `examenes`
select 'Marketing 1 Presen', nro_examen
from `examenes`
where nom_plan= 'Marketing 1';

insert into `examenes_temas`
select 'Marketing 1 Presen',titulo,nro_examen
from `examenes_temas`
where nom_plan= 'Marketing 1';

insert into `valores_plan`(`nom_plan`, `fecha_desde_plan`, `valor_plan`)
select 'Marketing 1 Presen',fecha_desde_plan,valor_plan*1.5
from `valores_plan`
where nom_plan= 'Marketing 1' and year(fecha_desde_plan)=2015;

/*
para realizar esta parte del ejercicio hay que haber realizado los 2 anteriores
*/
commit;
4)
start transaction;
update
`instructores` i   inner join
`cursos_instructores` ci
                      on ci.`cuil`=i.`cuil`
set `cuil_supervisor` ='66-66666666-6'
where ci.`nom_plan`='Reparac PC Avanzada';
commit;
5)
start transaction;
update
```

```
 `cursos_horarios` ch inner join

     `cursos_instructores` ci on ch.`nom_plan`=ci.`nom_plan`

                               and ch.`nro_curso`=ci.`nro_curso` inner join

     `cursos` c on ci.`nom_plan`=c.`nom_plan`

                and ci.`nro_curso`=c.`nro_curso`



set ch.`hora_inicio`=ADDTIME(ch.`hora_inicio`,-010000)

,ch.`hora_fin`=ADDTIME(ch.`hora_fin`,-010000)

where ci.`cuil`='66-66666666-6' and ch.`hora_inicio`='160000'

      and year(c.`fecha_ini`)=2009;

commit;

6)
start transaction;

drop temporary table if exists exa_elim;
create temporary table exa_elim
(
select ev.`nom_plan`,ev.`nro_examen`, AVG(ev.nota) promedio
from `evaluaciones` ev
group by ev.`nom_plan`,ev.`nro_examen`
having promedio<5.5
);

delete ev,et
from evaluaciones ev
inner join exa_elim
on ev.`nom_plan`=exa_elim.nom_plan and ev.`nro_examen`=exa_elim.nro_examen
inner join `examenes_temas` et on et.`nom_plan`=exa_elim.nom_plan
                               and et.`nro_examen`=exa_elim.nro_examen;

delete ex
from exa_elim
inner join `examenes` ex on ex.`nom_plan`=exa_elim.`nom_plan`
                         and ex.`nro_examen`=exa_elim.`nro_examen`;

commit;

7)
start transaction;

delete insc
from inscripciones insc inner join
(
select distinct dni
from cuotas
where fecha_pago is null
and anio=year(CURRENT_DATE)-1
) deudores
on insc.`dni`=deudores.dni
where year(insc.`fecha_inscripcion`)=year(CURRENT_DATE);
```

```
commit;
```

Práctica Nº 10: DCL

Para corroborar los cambios realizados en los permisos consultar con sentencias SQL la base de datos information_schema las tables: table_privileges y user_privileges

```
1)
CREATE USER usuario@localhost identified by 'entre';

2)
SET PASSWORD FOR 'usuario'@'localhost' = PASSWORD('entrar');

3)
GRANT SELECT ON agencia_personal.* to usuario@localhost

4)
GRANT UPDATE ON agencia_personal.personas to usuario@localhost;
GRANT INSERT ON agencia_personal.personas to usuario@localhost;
GRANT DELETE ON agencia_personal.personas to usuario@localhost;

O en un solo commando otorgo todos los permisos (SELECT, INSERT, UPDATE, DELETE)

GRANT all PRIVILEGES ON agencia_personal.personas to usuario@localhost;

5)
REVOKE SELECT ON agencia_personal.* FROM usuario@localhost;
REVOKE INSERT ON agencia_personal.* FROM usuario@localhost;
REVOKE DELETE ON agencia_personal.* FROM usuario@localhost;
REVOKE UPDATE ON agencia_personal.* FROM usuario@localhost;

O le quito todos los permisos de un solo commando

REVOKE all PRIVILEGES ON agencia_personal.* FROM usuario@localhost;

6)
GRANT UPDATE ON agencia_personal.vw_contratos to usuario@localhost;
GRANT INSERT ON agencia_personal.vw_contratos to usuario@localhost;
GRANT DELETE ON agencia_personal.vw_contratos to usuario@localhost;
```

Práctica Nº 12: STORE PROCEDURES y FUNCTIONS

Practica en clase: Ej.: 1, 2, 3, 6, 7, 8, 11 y 12.

Practica Sugerida: Ej.: 4, 5, 9 y 10.

```
1)

CREATE PROCEDURE `plan_lista_precios_actual`()

BEGIN

drop temporary table if exists valor_actual;
```

```
create temporary table valor_actual
(
 select vp.`nom_plan`, max(vp.`fecha_desde_plan`) ult_fecha
 from `valores_plan` vp
 group by vp.`nom_plan`
);

select pc.`nom_plan`, pc.`modalidad`, vp.`valor_plan` valor_actual
from `plan_capacitacion` pc
inner join valor_actual va
                         on  pc.`nom_plan`=va.nom_plan
inner join `valores_plan` vp
                           on  va.`nom_plan`=vp.`nom_plan`
                           and va.ult_fecha=vp.`fecha_desde_plan`;


drop temporary table if exists valor_actual;

END;

2)
CREATE PROCEDURE `plan_lista_precios_a_fecha`(IN fecha_hasta DATE)
BEGIN
drop temporary table if exists valor_actual;

create temporary table valor_actual
(
 select vp.`nom_plan`, max(vp.`fecha_desde_plan`) ult_fecha
 from `valores_plan` vp
 where vp.`fecha_desde_plan`<=fecha_hasta
 group by vp.`nom_plan`
);
```

```
select pc.`nom_plan`, pc.`modalidad`, vp.`valor_plan` valor_a_fecha
from `plan_capacitacion` pc
inner join valor_actual va
                          on  pc.`nom_plan`=va.nom_plan
inner join `valores_plan` vp
                            on  va.`nom_plan`=vp.`nom_plan`
                            and va.ult_fecha=vp.`fecha_desde_plan`;



drop temporary table if exists valor_actual;
END;
```

3)

```
DROP PROCEDURE `plan_lista_precios_actual`;

CREATE PROCEDURE `plan_lista_precios_actual`()
BEGIN

call plan_lista_precios_a_fecha(CURRENT_DATE);

END;
```

6)

```
CREATE PROCEDURE `alumnos_pagos_deudas_a_fecha`(IN fecha_limite DATE, IN dni_alumno
INTEGER(11), OUT pagado FLOAT(9,3), OUT cant_adeudado INTEGER(11))
BEGIN

select @pagado:=sum(cuo.`importe_pagado`)
from cuotas cuo
where cuo.dni=dni_alumno and cuo.`fecha_pago` is not null
      and cuo.`fecha_emision`<=fecha_limite;
```

```
select @cant_adeudado:=count(*)

from cuotas cuo

where cuo.dni=dni_alumno and cuo.`fecha_pago` is null

      and cuo.`fecha_emision`<=fecha_limite;


set pagado:=@pagado;

set cant_adeudado:=@cant_adeudado;


END;


7)

CREATE FUNCTION `alumnos_deudas_a_fecha`(dni_alumno INTEGER(11), fecha_limite DATE)

    RETURNS float(9,3)

BEGIN

declare cant_adeudado integer(11);


select count(*) into cant_adeudado

from cuotas cuo

where cuo.dni=dni_alumno and cuo.`fecha_pago` is null

      and cuo.`fecha_emision`<=fecha_limite;




return cant_adeudado;

END;


8)

CREATE PROCEDURE `alumno_inscripcion`(IN dni_alumno INTEGER(11), IN plan CHAR(20), IN curso
INTEGER(11))

BEGIN

start transaction;
```

```
     insert into inscripciones

     values (plan, curso,dni_alumno,CURRENT_DATE);



     insert into cuotas

     values (plan,curso,dni_alumno, year(adddate(CURRENT_DATE,interval 1 month)),

             month(adddate(CURRENT_DATE,interval 1 month)),CURRENT_DATE, null,null);

commit;



END;



11)
CREATE PROCEDURE `stock_movimiento`(IN cod_mat CHAR(6), IN cant_movida INTEGER(11), OUT
stock INTEGER(11))
BEGIN

     declare url varchar(50);

     start transaction;
     /*
       Consulto por la url para no saber si es un apunte o un útil
     */
     select url_descarga into url
     from materiales
     where cod_material=cod_mat;


     if url is null then
        update materiales set cant_disponible=cant_disponible+cant_movida
        where cod_material=cod_mat;
     end if;

     select cant_disponible into stock
     from materiales
     where cod_material=cod_mat;

     if stock>=0 then
        commit;

       else
         rollback;

         select cant_disponible into stock
         from materiales
         where cod_material=cod_mat;
     end if;

END;
```

```
CREATE PROCEDURE `stock_ingreso`(IN cod_mat CHAR(6), IN cant_movida INTEGER(11), OUT stock
INTEGER(11))

BEGIN
     call stock_movimiento(cod_mat,cant_movida,stock);
END;

CREATE PROCEDURE `stock_egreso`(IN cod_mat CHAR(6), IN cant_movida INTEGER(11), OUT stock
INTEGER(11))
    NOT DETERMINISTIC
    SQL SECURITY DEFINER
    COMMENT ''
BEGIN
     call stock_movimiento(cod_mat,(-1)*cant_movida,stock);
END;

12)
CREATE PROCEDURE `alumno_anula_inscripcion`(IN plan CHAR(20), IN curso INTEGER(11), IN
alumno INTEGER(11))
BEGIN
     declare cuotas_pagas integer(11);

     select count(*) into cuotas_pagas
     from cuotas
     where nom_plan=plan and nro_curso=curso and dni=alumno
           and fecha_pago is not null;

     if cuotas_pagas<=0 then
        start transaction;

        delete from cuotas where nom_plan=plan and nro_curso=curso
                                 and dni=alumno and fecha_pago is null;

        delete from inscripciones where nom_plan=plan and nro_curso=curso and dni=alumno;

        commit;

     end if;

END;
```

Práctica Nº 13: TRIGGERS

Practica en clase: Ej.: 1, 2, 3 y 4.

Practica Sugerida:

1)

```
/*

Este TRIGGER se tiene que crear después de la inserción o da error de clave foránea.

Tienen que usar los valores de la fila a ingresar, o sea, de new. Porque como es un INSERT y
entonces no existen los valores viejos de la fila.

*/

CREATE TRIGGER `alumnos_before_ins_tr` AFTER INSERT ON `alumnos`
```

```
  FOR EACH ROW
BEGIN
     insert into alumnos_historico
     values (new.dni,CURRENT_TIMESTAMP,new.nombre,new.apellido,
             new.tel,new.email,new.direccion,CURRENT_USER);
END;

/*
Este TRIGGER puede crearse antes o después es a gusto de C/U.
Tienen que usar los valores nuevos de la fila, o sea, de new.
*/
CREATE TRIGGER `alumnos_before_upd_tr` AFTER UPDATE ON `alumnos`
  FOR EACH ROW
BEGIN
     insert into alumnos_historico
     values (new.dni,CURRENT_TIMESTAMP,new.nombre,new.apellido,
             new.tel,new.email,new.direccion,CURRENT_USER);
END;

2)
/*
Este TRIGGER se tiene que crear después de la inserción o da error de clave foránea
Se tiene que preguntar si la cant_disponible nueva es NULL o no porque si es NULL es un
apunte y si es un apunte no tiene cantidad por lo que no hace falta modificar la cantidad.
Además la cantidad  en la tabla stock_movimientos no puede ser NULL.
Tienen que usar los valores de la fila a ingresar, o sea, de new. Porque como es un INSERT y
entonces no existen los valores viejos de la fila
*/
CREATE TRIGGER `materiales_after_ins_tr` AFTER INSERT ON `materiales`
  FOR EACH ROW
BEGIN
     if new.cant_disponible is not null then
     insert into stock_movimientos( cod_material, cantidad_movida,
                                    cantidad_restante, usuario_movimiento)
```

```
     values (new.cod_material,new.cant_disponible,new.cant_disponible,CURRENT_USER);

     end if;

END;



/*

Se tiene que preguntar si la cant_disponible nueva es NULL o no porque si es NULL es un
apunte y si es un apunte no tiene cantidad por lo que no hace falta modificar la cantidad.
Además la cantidad  en la tabla stock_movimientos no puede ser NULL.

Acá hay que calcular la diferencia entre el valor anterior y el nuevo valor para poder
calcular la cantidad_movida.

*/

CREATE TRIGGER `materiales_before_upd_tr` BEFORE UPDATE ON `materiales`

  FOR EACH ROW

BEGIN

   if new.cant_disponible is not null then

     set @cant_movida=new.cant_disponible-old.cant_disponible;

     if @cant_movida!=0 then

          insert into stock_movimientos( cod_material, cantidad_movida,

                                         cantidad_restante, usuario_movimiento)

          values (new.cod_material,@cant_movida,

                 new.cant_disponible,CURRENT_USER);

     end if;

   end if;

END;



3)

/*

Acá el antes o el después no tiene importancia.

Tienen usar los valores de la CP del curso del los nuevos valores porque como es un INSERT
no existen los viejos

*/

CREATE TRIGGER `inscripciones_after_ins_tr` AFTER INSERT ON `inscripciones`

  FOR EACH ROW

BEGIN
```

```
     update cursos
     set cant_inscriptos=cant_inscriptos+1
     where nom_plan=new.nom_plan and nro_curso=new.nro_curso;
END;

/*
Acá es muy importante que saquen el dato de la CP del curso de los valores anteriores de la fila (old.) porque luego de borrar no existen los valores (new.) y daría error si usan el new.
*/
CREATE TRIGGER `inscripciones_after_del_tr` AFTER DELETE ON `inscripciones`
  FOR EACH ROW
BEGIN
     update cursos
     set cant_inscriptos=cant_inscriptos-1
     where nom_plan=old.nom_plan and nro_curso=old.nro_curso;
END;

4)
/*
Este TRIGGER tiene que ser ANTES de insertar porque no se puede modificar el valor de la nueva fila (new) si ya se insertó.
ATENCIÓN con el SET. Es como si fuese una variable.
*/
CREATE TRIGGER `valores_plan_before_ins_tr` BEFORE INSERT ON `valores_plan`
  FOR EACH ROW
BEGIN
    set new.usuario_alta=CURRENT_USER;
END;
```