-- trigger para controlar que solo los docentes se puedan adscribir a una unidad academica.

delimiter $$

create trigger docente\_unidad\_academica\_tgr

before insert

on unidad\_usuario for each row

begin

declare rol\_usuario varchar(50);

select rol into rol\_usuario

from Usuario

where idUsuario = new.Usuario\_idUsuario;

if rol\_usuario <> "Docente" then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El usuario asignado a la unidad academica debe tener el rol de Docente";

end if;

end $$

delimiter ;

-- trigger para controlar que solo los secretarios academicos puedan gestionar una facultad.

delimiter $$

create trigger secretario\_facultad\_tgr

before insert

on Facultad\_usuario for each row

begin

declare rol\_usuario varchar(50);

select rol into rol\_usuario

from Usuario

where idUsuario = new.Usuario\_idUsuario;

if rol\_usuario <> "Secretario Academico" then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El usuario asignado a la facultad debe tener el rol de Secretario Academico";

end if;

end $$

delimiter ;

-- trigger para controlar que solo los estudiantes puedan pertenezcan a un programa academico.

delimiter $$

create trigger estudiante\_programaAcademico\_tgr

before insert

on programa\_usuario for each row

begin

declare rol\_usuario varchar(50);

select rol into rol\_usuario

from Usuario

where idUsuario = new.Usuario\_idUsuario;

if rol\_usuario <> "Estudiante" then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El usuario asignado al programa academico debe tener el rol de estudiante";

end if;

end $$

delimiter ;

-- trigger para controlar que solo los estudiantes y docentes sean responsables de un evento.

delimiter $$

create trigger usuario\_organizador\_tgr

before insert

on usuario\_organizador for each row

begin

declare rol\_usuario varchar(50);

select rol into rol\_usuario

from Usuario

where idUsuario = new.Usuario\_idUsuario;

if rol\_usuario <> "Estudiante" and rol\_usuario <> "Docente" then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El usuario asignado a la organizacion del evento debe ser un docente o estudiante.";

end if;

end $$

delimiter ;

-- trigger para controlar que solo los secretarios academicos puedan realizar la evaluacion de un evento.

delimiter $$

create trigger EvaluacionEvento\_tgr

before insert

on EvaluacionEvento for each row

begin

declare rol\_usuario varchar(50);

select rol into rol\_usuario

from Usuario

where idUsuario = new.Usuario\_idUsuario;

if rol\_usuario <> "secretario academico" then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El usuario asignado a la evaluacion del evento debe ser un secretario academico.";

end if;

end $$

delimiter ;

-- trigger para que cuando se evalue un evento cambie el estado del mismo

delimiter $$

create trigger cambiar\_estado\_evento\_tgr

after insert

on evaluacionevento for each row

begin

update evento

set estadoevento = new.estado

where idevento = new.evento\_idevento;

end $$

delimiter ;

-- trigger para asegurarse de que al registrar un evento su estado sea en revision

delimiter $$

create trigger estadoevento\_revision\_por\_defecto\_tgr

before insert on evento for each row

begin

if new.estadoevento <> "Revisión" then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El estado por defecto de un evento al añadirlo debe ser (revision)";

end if;

end $$

delimiter ;

-- verificar que la hora de finalizacion sea mayor a la hora de inicio

delimiter $$

create trigger verificar\_horafin\_evento\_tgr

before insert on evento for each row

begin

if new.horafinevento <= new.horainicioevento then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "La hora de fin del evento debe ser posterior a la hora de inicio";

end if;

end $$

delimiter ;

-- trigger para verificar que no se asigne 2 veces un usuario organizador al mismo evento

delimiter $$

create trigger evitar\_usuarioorganizador\_duplicado\_tgr

before insert on usuario\_organizador for each row

begin

if exists (

select 1 from usuario\_organizador where

usuario\_idusuario = new.Usuario\_idUsuario and

Evento\_idEvento = new.Evento\_idEvento) then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El usuario ya fue asignado como organizador de este evento.";

end if;

end $$

delimiter ;

-- trigger para verificar que no se asigne 2 veces un docente a la misma unidad

delimiter $$

create trigger evitar\_docente\_duplicado\_tgr

before insert on unidad\_usuario for each row

begin

if exists (

select 1 from unidad\_usuario where

usuario\_idusuario = new.Usuario\_idUsuario and

UnidadAcademica\_idUnidadAcademica = new.UnidadAcademica\_idUnidadAcademica) then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El docente ya fue asignado a esta unidad.";

end if;

end $$

delimiter ;

-- trigger para verificar que no se asigne 2 veces un estudiante al mismo programa

delimiter $$

create trigger evitar\_estudiante\_duplicado\_tgr

before insert on programa\_usuario for each row

begin

if exists (

select 1 from programa\_usuario where

usuario\_idusuario = new.Usuario\_idUsuario and

programaacademico\_idprogramaacademico = new.programaacademico\_idprogramaacademico) then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El estudiante ya fue asignado a este programa.";

end if;

end $$

delimiter ;

-- trigger para verificar que no se asigne 2 veces un secretario academico a la misma facultad

delimiter $$

create trigger evitar\_secretarioacademico\_duplicado\_tgr

before insert on facultad\_usuario for each row

begin

if exists (

select 1 from facultad\_usuario where

usuario\_idusuario = new.Usuario\_idUsuario and

Facultad\_idFacultad = new.Facultad\_idFacultad) then

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "El secretario academico ya fue asignado a esta facultad.";

end if;

end $$

delimiter ;

-- trigger para revisar que la fecha del evento sea futura o igual a la actual

delimiter $$

create trigger revisar\_evento\_fecha\_tgr

before insert on Evento for each row

begin

if new.fechaEvento < CURDATE() then

signal sqlstate "45000"

set message\_text = "La fecha del evento debe ser mayor o igual a la fecha actual.";

end if;

end$$

delimiter;

-- trigger para verificar que una organizacion externa no este repetida en un evento

delimiter $$

create trigger organizacion\_externa\_repetida\_evento\_tgr

before insert on Participa for each row

begin

if exists(

select 1 from participa

where OrganizaciónExterna\_idOrganizacionExterna = NEW.OrganizaciónExterna\_idOrganizacionExterna

and Evento\_idEvento = NEW.Evento\_idEvento) then

signal sqlstate "45000"

set message\_text = "La organizacion externa ya esta registrada en este evento.";

end if;

end$$

DELIMITER ;

-- trigger para que un eveneto en revision o rechazado no pueda tener acta

DELIMITER $$

create trigger validar\_acta\_evaluacion\_trg

before insert on EvaluacionEvento

for each row

begin

if new.evaluacion\_actaurl is not null then

if new.estado <> "aprobado" then

signal sqlstate "45000"

set message\_text = 'No se puede asignar acta a un evento que está en revisión o es rechazado.';

end if;

end if;

end$$

delimiter ;

-- trigger que al actualizar la evaluacion de un evento no pueda tener acta si su estado es revision o rechazado

delimiter $$

CREATE TRIGGER validar\_acta\_actualizacion\_evaluacion\_trg

before update on EvaluacionEvento

for each row

begin

if new.evaluacion\_actaurl is not null then

if new.estado <> "aprobado" then

signal sqlstate "45000"

set message\_text = 'No se puede asignar acta a un evento que está en revisión o es rechazado.';

end if;

end if;

end$$

DELIMITER ;