CREATE OR REPLACE FUNCTION ver\_comentario() RETURNS TRIGGER AS $$

DECLARE

pub record;

BEGIN

select \* into pub

from publicacao

where publicacao.id = new.publicacao\_id;

if (exists (select 1

from bloqueia

where bloqueia.bloqueado\_id = new.comentou

and bloqueia.bloqueou\_id = pub.userid

and bloqueia.data\_inicio <= pub.data

and (bloqueia.data\_fim is null or bloqueia.data\_fim >= pub.data)) ) then

RAISE EXCEPTION 'Usuário % não pode comentar, pois estava bloqueado na data do comentário', NEW.COMENTOU;

END IF;

if (not (select perfil\_aberto from perfil where username = pub.userid)

and not exists (select 1 from seguidor where e\_seguido = pub.userid and seguidor = new.comentou and status like 'aprovado')) then

raise exception 'Usuário % não pode comentar, pois não é amigo confirmado do usuário % que tem perfil bloqueado', new.comentou, pub.userid;

end if;

return new;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION ver\_like() RETURNS TRIGGER AS $$

DECLARE

pub record;

BEGIN

select \* into pub

from publicacao

where publicacao.id = new.publicacao\_id;

if (exists (select 1

from bloqueia

where bloqueia.bloqueado\_id = new.curtiu

and bloqueia.bloqueou\_id = pub.userid

and bloqueia.data\_inicio <= pub.data

and (bloqueia.data\_fim is null or bloqueia.data\_fim >= pub.data)) ) then

RAISE EXCEPTION 'Usuário % não pode curtir, pois estava bloqueado na data da curtida', NEW.CURTIU;

END IF;

if ((select perfil\_aberto from perfil where username = pub.userid) = false and not exists (select 1 from seguidor where e\_seguido = pub.userid and seguidor = new.curtiu and status like 'aprovado'))then

RAISE EXCEPTION 'Usuário % não pode curtir, pois não é amigo confirmado do usuário % que tem perfil bloqueado', NEW.CURTIU, pub.userid;

end if;

return new;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION ver\_tag() RETURNS TRIGGER AS $$

DECLARE

pub record;

BEGIN

select \* into pub

from publicacao

where publicacao.id = new.publicacao\_id;

if (exists (select 1

from bloqueia

where bloqueia.bloqueado\_id = new.marcado

and bloqueia.bloqueou\_id = pub.userid

and bloqueia.data\_inicio <= pub.data

and (bloqueia.data\_fim is null or bloqueia.data\_fim >= pub.data)) ) then

RAISE EXCEPTION 'Usuário % não pode ser tagged, pois estava bloqueado na data da tag', NEW.MARCADO;

END IF;

if ((select perfil\_aberto from perfil where username = pub.userid) = false and not exists (select 1 from seguidor where e\_seguido = pub.userid and seguidor = new.marcado and status like 'aprovado'))then

RAISE EXCEPTION 'Usuário % não pode ser tagged, , pois não é amigo confirmado do usuário % que tem perfil bloqueado', NEW.MARCADO, pub.userid;

end if;

return new;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION ver\_status() RETURNS TRIGGER AS $$

BEGIN

if (new.status <> 'aprovado' and new.status <> 'pendente' and new.status <> 'recusado') then

raise exception 'status inexistente!';

elsif (TG\_OP = 'UPDATE' and ((old.status = 'aprovado' and new.status <> 'aprovado' ) or

(old.status = 'pendente' and (new.status <> 'aprovado' or new.status <> 'recusado')) or

(old.status = 'recusado' and new.status <> 'pendente')) ) then

raise exception 'Não é possível essa mudança de status!';

else return new;

end if;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION ver\_bloq() RETURNS TRIGGER AS $$

DECLARE

BEGIN

IF NEW.DATA\_INICIO >= NEW.DATA\_FIM or new.data\_fim > now() THEN

RAISE EXCEPTION 'Datas inválidas!';

END IF;

if (TG\_OP = 'UPDATE') then

if (old.bloqueado\_id <> new.bloqueado\_id or old.bloqueado\_id <> new.bloqueado\_id or old.data\_inicio <> new.data\_inicio) then

raise exception 'Não é possível mudar atributos a não ser data final';

elsif (old.data\_fim is not null) then

raise exception 'Não é possível mudar uma data que não é nula';

end if;

return new;

end if;

--inserindo bloqueio com data nula se ja existe outro com data nula ou com data inicio interceptando outras datas

if exists (

select 1

from bloqueia

where bloqueou\_id = new.bloqueou\_id

and bloqueado\_id = new.bloqueado\_id

and (

(data\_fim is null and new.data\_fim is null)

or

(new.data\_fim is null and data\_fim >= new.data\_inicio)

)

) then

RAISE EXCEPTION 'USUÁRIO % JÁ ESTÁ BLOQUEADO ATUALMENTE', new.BLOQUEADO\_ID;

-- a data final ou a inicial da nova tupla esta no meio de uma data de bloqueio ja existente

elseif exists(

select 1

from bloqueia

where bloqueou\_id = new.bloqueou\_id

and bloqueado\_id = new.bloqueado\_id

and (

(data\_inicio <= new.data\_inicio and data\_fim >= new.data\_inicio)

or

(data\_inicio <= new.data\_fim and data\_fim >= new.data\_fim)

or

(new.data\_inicio <= data\_inicio and new.data\_fim >= data\_inicio)

or

(new.data\_inicio <= data\_fim and new.data\_fim >= data\_fim)

)

) then

RAISE EXCEPTION 'Data de bloqueio inválida! Usuário % já estava bloqueado em alguma data dentro desse período.', NEW.BLOQUEADO\_ID;

end if;

return new;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION atualiza\_comentarios() RETURNS TRIGGER AS $$

DECLARE

valor\_antigo int;

valor\_novo int;

BEGIN

IF (TG\_OP = 'INSERT') THEN

select num\_comentarios into valor\_antigo from publicacao where id = new.publicacao\_id;

valor\_novo = valor\_antigo + 1;

UPDATE PUBLICACAO SET num\_comentarios = valor\_novo where id = new.publicacao\_id;

RETURN NEW;

ELSIF (TG\_OP = 'DELETE') THEN

select num\_comentarios into valor\_antigo from publicacao where id = old.publicacao\_id;

valor\_novo = valor\_antigo - 1;

UPDATE PUBLICACAO SET num\_likes = valor\_novo where id = old.publicacao\_id;

RETURN old;

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION atualiza\_likes() RETURNS TRIGGER AS $$

DECLARE

valor\_antigo int;

valor\_novo int;

BEGIN

IF (TG\_OP = 'INSERT') THEN

select num\_likes into valor\_antigo from publicacao where id = new.publicacao\_id;

valor\_novo = valor\_antigo + 1;

UPDATE PUBLICACAO SET num\_likes = valor\_novo where id = new.publicacao\_id;

RETURN NEW;

ELSIF (TG\_OP = 'DELETE') THEN

select num\_likes into valor\_antigo from publicacao where id = old.publicacao\_id;

valor\_novo = valor\_antigo - 1;

UPDATE PUBLICACAO SET num\_likes = valor\_novo where id = old.publicacao\_id;

RETURN old;

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_likes

AFTER INSERT OR DELETE ON "like"

FOR EACH ROW EXECUTE PROCEDURE atualiza\_likes();

CREATE TRIGGER update\_comentarios

AFTER INSERT OR DELETE ON comentario

FOR EACH ROW EXECUTE PROCEDURE atualiza\_comentarios();

CREATE TRIGGER verifica\_comentario

BEFORE INSERT OR UPDATE ON COMENTARIO

FOR EACH ROW EXECUTE PROCEDURE ver\_comentario();

CREATE TRIGGER verifica\_like

BEFORE INSERT OR UPDATE ON "like"

FOR EACH ROW EXECUTE PROCEDURE ver\_like();

CREATE TRIGGER verifica\_tag

BEFORE INSERT OR UPDATE ON TAG

FOR EACH ROW EXECUTE PROCEDURE ver\_tag();

CREATE TRIGGER verifica\_bloqueio

BEFORE INSERT OR UPDATE ON BLOQUEIA

FOR EACH ROW EXECUTE PROCEDURE ver\_bloq();

CREATE TRIGGER verifica\_status

BEFORE INSERT OR UPDATE ON SEGUIDOR

FOR EACH ROW EXECUTE PROCEDURE ver\_status();