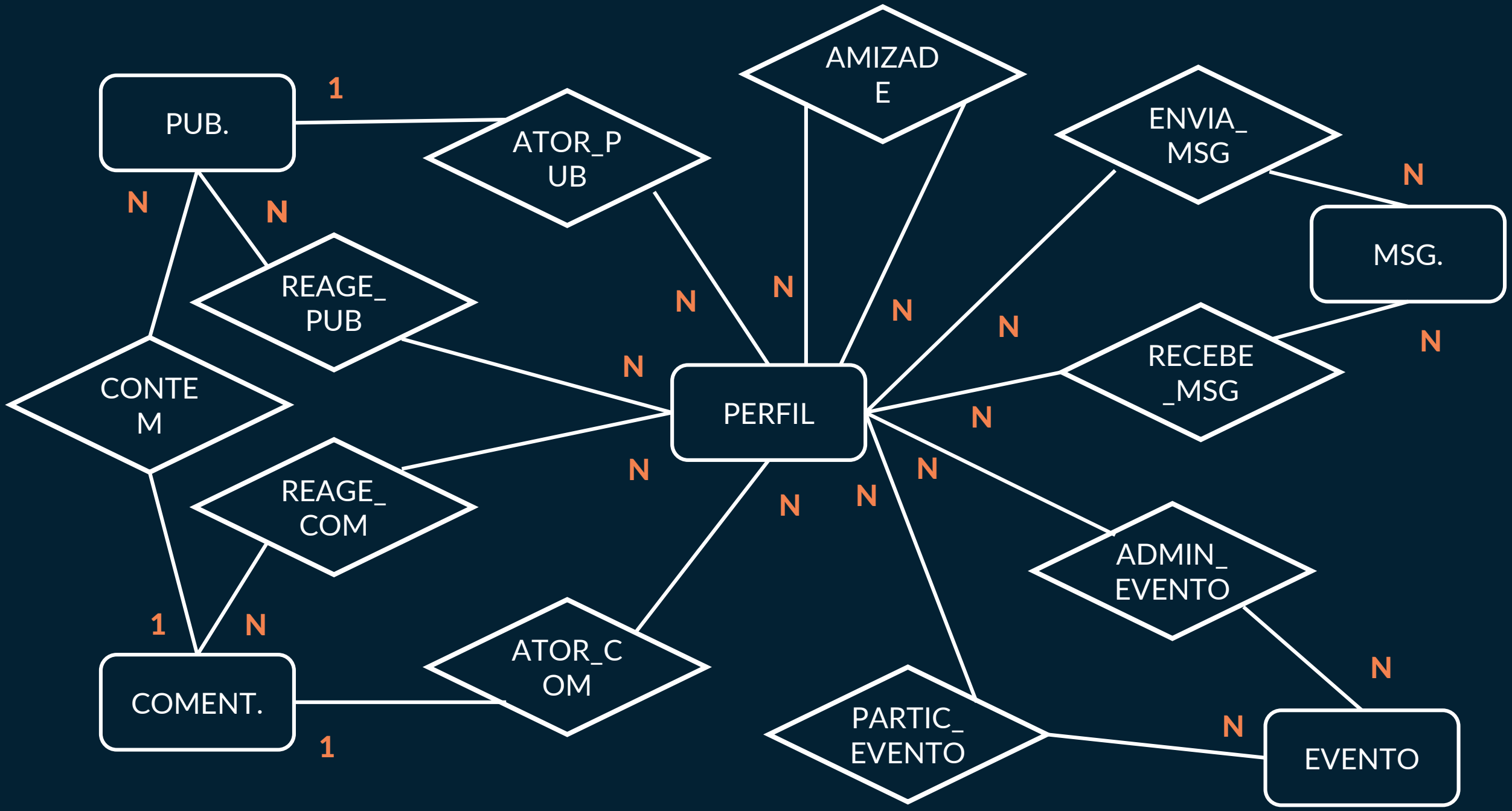
ALUNOS: ALEXANDRE E VINICIUS


PUB.
1
ATOR_PUB
AMIZADE
ENVIA_MSG
N
MSG.
N
N
N
N
N
N
N
REAGE_PUB
N
RECEBE_MSG
CONTEM
PERFIL
N
N
N
N
N
N
REAGE_COM
ADMIN_EVENTO
1
N
N
ATOR_COM
COMENT.
PARTIC_EVENTO
N
EVENTO
1

```
/* CREATE TABLES */

CREATE TABLE PERFIL(
        ID_PERFIL SERIAL PRIMARY KEY,
        NOME VARCHAR(30) NOT NULL,
        DATA_NASCIMENTO DATE NOT NULL,
        EMAIL VARCHAR(30) NOT NULL UNIQUE,
        SENHA VARCHAR(30) NOT NULL
);

CREATE TABLE PUBLICACAO(
        ID_PUBLICACAO SERIAL,
        ID_PERFIL_PUB INT NOT NULL REFERENCES PERFIL(ID_PERFIL),
        DATA_PUBLICACAO DATE NOT NULL,
        TEXTO VARCHAR(500) NOT NULL,
        PRIMARY KEY(ID_PUBLICACAO)
);

CREATE TABLE COMENTARIO(
        ID_COMENTARIO SERIAL,
        ID_PERFIL INT NOT NULL REFERENCES PERFIL(ID_PERFIL),
        ID_PUBLICACAO INT NOT NULL REFERENCES PUBLICACAO(ID_PUBLICACAO) ON DELETE CASCADE,
        COMENTARIO VARCHAR(500) NOT NULL,
        DATA_COMENTARIO DATE NOT NULL,
        PRIMARY KEY(ID_COMENTARIO)
);
```

```
CREATE TABLE AMIZADE(
        ID__A INT NOT NULL REFERENCES PERFPDELETEERFILIL(ID_PERFIL) ON CASCADE,
        ID_PERFIL_B INT NOT NULL REFERENCES PERFIL(ID_PERFIL) ON DELETE CASCADE,
        DATA_INICIO DATE NOT NULL,
        DATA_FIM DATE DEFAULT '9999-12-31',
        PRIMARY KEY(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO)
);

CREATE TABLE REAGE_PUB(
        ID_PERFIL INT NOT NULL REFERENCES PERFIL(ID_PERFIL),
        ID_PUBLICACAO INT NOT NULL REFERENCES PUBLICACAO(ID_PUBLICACAO) ON DELETE CASCADE,
        REACAO VARCHAR(15) NOT NULL,
        PRIMARY KEY(ID_PERFIL, ID_PUBLICACAO)
);

CREATE TABLE REAGE_COMENTARIO(
        ID_PERFIL INT NOT NULL REFERENCES PERFIL(ID_PERFIL),
        ID_COMENTARIO INT NOT NULL REFERENCES COMENTARIO(ID_COMENTARIO) ON DELETE CASCADE,
        REACAO VARCHAR(15) NOT NULL,
        PRIMARY KEY(ID_PERFIL, ID_COMENTARIO)
);
```

```
CREATE TABLE MENSAGEM(
        ID_MENSAGEM SERIAL,
        ID_PERFIL_ORIGEM INT NOT NULL REFERENCES PERFIL(ID_PERFIL),
        ID_PERFIL_DESTINO INT NOT NULL REFERENCES PERFIL(ID_PERFIL),
        CONTEUDO_MENSAGEM VARCHAR(500) NOT NULL,
        DATA_ENVIO_MENSAGEM DATE NOT NULL,
        PRIMARY KEY(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, ID_MENSAGEM)
);

CREATE TABLE EVENTO(
        ID_EVENTO SERIAL,
        NOME_EVENTO VARCHAR(30) NOT NULL,
        DATA_INICIO_EVENTO DATE NOT NULL,
        DATA_FIM_EVENTO DATE NOT NULL,
        DESCRICAO_EVENTO VARCHAR(500) NOT NULL,
        ENDERECO_EVENTO VARCHAR(100) NOT NULL,
        NUMERO_CONFIRMADOS INT,
        PRIMARY KEY(ID_EVENTO)
);
```

```
CREATE TABLE ADMIN_EVENTO(
        ID_PERFIL INT NOT NULL REFERENCES PERFIL(ID_PERFIL) ON DELETE CASCADE,
        ID_EVENTO INT NOT NULL REFERENCES EVENTO(ID_EVENTO) ON DELETE CASCADE,
        PRIMARY KEY(ID_PERFIL, ID_EVENTO)
);

CREATE TABLE PARTICIPA_EVENTO(
        ID_PERFIL INT NOT NULL REFERENCES PERFIL(ID_PERFIL) ON DELETE CASCADE,
        ID_EVENTO INT NOT NULL REFERENCES EVENTO(ID_EVENTO) ON DELETE CASCADE,
        STATUS_COMPARECIMENTO_EVENTO VARCHAR(10) NOT NULL,
        PRIMARY KEY(ID_PERFIL, ID_EVENTO)
);

/* END OF CREATE TABLES */
```

```
/* TRIGGERS E FUNCTIONS  */


-- TRIGGER 01: NÃO PERMIRTIR  QUE UMA DATA_FIM DE AMIZADE SEJA MENOR QUE UMA DATA_INICIO

CREATE OR REPLACE FUNCTION DATA_FIM_MAIOR_DATA_INICIO() RETURNS TRIGGER AS $BODY$
BEGIN
    IF(NEW.DATA_INICIO > NEW.DATA_FIM)
    THEN
        DELETE FROM AMIZADE WHERE DATA_FIM = NEW.DATA_FIM;
    END IF;
    RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDAR_DATA_AMIZADE
    AFTER INSERT OR UPDATE ON AMIZADE
    FOR EACH ROW
    EXECUTE PROCEDURE DATA_FIM_MAIOR_DATA_INICIO();
```

```
-- TRIGGER 02: VALIDAR COMENTARIO A PARTIR DE UMA AMIZADE,  O COMENTARIO SOMENTE PODERA SER FEITO
            --DURANTE UM  PERIODO DE AMIZADE

CREATE OR REPLACE FUNCTION VALIDAR_COMENTARIO_PELA_AMIZADE() RETURNS TRIGGER AS $BODY$
BEGIN
   IF NOT EXISTS(SELECT * FROM COMENTARIO
      INNER JOIN PUBLICACAO ON (NEW.ID_PUBLICACAO = PUBLICACAO.ID_PUBLICACAO)
      INNER JOIN AMIZADE ON ((NEW.ID_PERFIL = AMIZADE.ID_PERFIL_A OR NEW.ID_PERFIL = AMIZADE.ID_PERFIL_B)
AND
          (PUBLICACAO.ID_PERFIL_PUB = AMIZADE.ID_PERFIL_A OR PUBLICACAO.ID_PERFIL_PUB =
AMIZADE.ID_PERFIL_B))
      WHERE (NEW.DATA_COMENTARIO > AMIZADE.DATA_INICIO) AND (NEW.DATA_COMENTARIO <
AMIZADE.DATA_FIM))
      THEN
          DELETE FROM COMENTARIO WHERE ID_COMENTARIO = NEW.ID_COMENTARIO;
   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDA01_COMENTARIO
   AFTER INSERT OR UPDATE ON COMENTARIO
   FOR EACH ROW
   EXECUTE PROCEDURE VALIDAR_COMENTARIO_PELA_AMIZADE();
```

```
-- TRIGGER 03: VALIDAR SE A DATA DO COMENTARIO OCORREU DEPOIS DA PUBLICACAO REFERENCIADA ,
                --VERIFICANDO A DATA DESTA

CREATE OR REPLACE FUNCTION VALIDAR_COMENTARIO_PELA_PUBLICACAO() RETURNS TRIGGER AS $BODY$
BEGIN
    IF NOT EXISTS (SELECT * FROM COMENTARIO
        INNER JOIN PUBLICACAO ON (NEW.ID_PUBLICACAO = PUBLICACAO.ID_PUBLICACAO)
        WHERE NEW.DATA_COMENTARIO > PUBLICACAO.DATA_PUBLICACAO)
    THEN
        DELETE FROM COMENTARIO WHERE ID_COMENTARIO = NEW.ID_COMENTARIO;
    END IF;
    RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDA02_COMENTARIO
    AFTER INSERT OR UPDATE ON COMENTARIO
    FOR EACH ROW
    EXECUTE PROCEDURE VALIDAR_COMENTARIO_PELA_PUBLICACAO();
```

```
-- TRIGGER 04: VALIDAR A REACAO EM UMA PUBLICACAO A PARTIR DE UMA AMIZADE, VERIFICANDO APENAS SE HÁ
            --UMA AMIZADE VIGENTE

CREATE OR REPLACE FUNCTION VALIDAR_REACAO_PUB_PELA_AMIZADE() RETURNS TRIGGER AS $BODY$
BEGIN
   IF NOT EXISTS(SELECT * FROM REAGE_PUB
      INNER JOIN PUBLICACAO ON (NEW.ID_PUBLICACAO = PUBLICACAO.ID_PUBLICACAO)
      INNER JOIN AMIZADE ON ((NEW.ID_PERFIL = AMIZADE.ID_PERFIL_A OR NEW.ID_PERFIL = AMIZADE.ID_PERFIL_B)
AND
         (PUBLICACAO.ID_PERFIL_PUB = AMIZADE.ID_PERFIL_A OR PUBLICACAO.ID_PERFIL_PUB =
AMIZADE.ID_PERFIL_B))
      WHERE (CURRENT_TIMESTAMP > AMIZADE.DATA_INICIO) AND (CURRENT_TIMESTAMP < AMIZADE.DATA_FIM))
      THEN
         DELETE FROM REAGE_PUB WHERE ID_PERFIL = NEW.ID_PERFIL;
   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDA01_REACAO
   AFTER INSERT OR UPDATE ON REAGE_PUB
   FOR EACH ROW
   EXECUTE PROCEDURE VALIDAR_REACAO_PUB_PELA_AMIZADE();
```

```
-- TRIGGER 05: VALIDAR SE A DATA DA REACAO , INDEPENDENTE SE ESTA SE TRATANDO DE UM INSERT OU UPDATE,
            --OCORREU DEPOIS DA PUBLICACAO REFERENCIADA

CREATE OR REPLACE FUNCTION VALIDAR_REACAO_PELA_PUBLICACAO() RETURNS TRIGGER AS $BODY$
BEGIN
   IF NOT EXISTS (SELECT * FROM REAGE_PUB
      INNER JOIN PUBLICACAO ON (NEW.ID_PUBLICACAO = PUBLICACAO.ID_PUBLICACAO)
      WHERE CURRENT_TIMESTAMP > PUBLICACAO.DATA_PUBLICACAO)
   THEN
      DELETE FROM REAGE_PUB WHERE ID_PUBLICACAO = NEW.ID_PUBLICACAO;
   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDA02_REACAO
   AFTER INSERT OR UPDATE ON REAGE_PUB
   FOR EACH ROW
   EXECUTE PROCEDURE VALIDAR_REACAO_PELA_PUBLICACAO();
```

```
-- TRIGGER 06: VALIDAR REACAO A UM COMENTARIO A PARTIR DE UMA AMIZADE, VERIFICANDO APENAS SE HÁ
            --AMIZADE VIGENTE

CREATE OR REPLACE FUNCTION VALIDAR_REACAO_COMENTARIO_PELA_AMIZADE() RETURNS TRIGGER AS $BODY$
BEGIN
   IF NOT EXISTS(SELECT * FROM REAGE_COMENTARIO
      INNER JOIN COMENTARIO ON (NEW.ID_COMENTARIO = COMENTARIO.ID_COMENTARIO)
      INNER JOIN AMIZADE ON ((NEW.ID_PERFIL = AMIZADE.ID_PERFIL_A OR NEW.ID_PERFIL = AMIZADE.ID_PERFIL_B)
AND
          (COMENTARIO.ID_PERFIL = AMIZADE.ID_PERFIL_A OR COMENTARIO.ID_PERFIL = AMIZADE.ID_PERFIL_B))
      WHERE (CURRENT_TIMESTAMP > AMIZADE.DATA_INICIO) AND (CURRENT_TIMESTAMP < AMIZADE.DATA_FIM))
      THEN
          DELETE FROM REAGE_COMENTARIO WHERE ID_PERFIL = NEW.ID_PERFIL;
   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDA01_REACAO_COMENTARIO
   AFTER INSERT OR UPDATE ON REAGE_COMENTARIO
   FOR EACH ROW
   EXECUTE PROCEDURE VALIDAR_REACAO_COMENTARIO_PELA_AMIZADE();
```

```
-- TRIGGER 07: VALIDAR SE A DATA DA REACAO, INDEPENDENTE SE ESTA SE TRATANDO DE UM INSERT OU
            --UPDATE, OCORREU DEPOIS DA POSTAGEM DO COMENTARIO REFERENCIADO

CREATE OR REPLACE FUNCTION VALIDAR_REACAO_PELO_COMENTARIO() RETURNS TRIGGER AS $BODY$
BEGIN
   IF NOT EXISTS (SELECT * FROM REAGE_COMENTARIO
      INNER JOIN COMENTARIO ON (NEW.ID_COMENTARIO = COMENTARIO.ID_COMENTARIO)
      WHERE CURRENT_TIMESTAMP > COMENTARIO.DATA_COMENTARIO)
   THEN
      DELETE FROM REAGE_COMENTARIO WHERE ID_COMENTARIO = NEW.ID_COMENTARIO;
   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDA02_REACAO_COMENTARIO
   AFTER INSERT OR UPDATE ON REAGE_COMENTARIO
   FOR EACH ROW
   EXECUTE PROCEDURE VALIDAR_REACAO_PELO_COMENTARIO();
```

```
-- TRIGGER 08: VALIDAR O ENVIO DE UMA MENSAGEM, ELA SÓ PODE SER ENVIADA PARA UM AMIGO. SERÁ
            --VERIFICADO O HISTORICO DE AMIZADES, COMPARANDO COM A DATA DE ENVIO DA MENSAGEM

CREATE OR REPLACE FUNCTION VALIDAR_MENSAGEM_PELA_AMIZADE() RETURNS TRIGGER AS $BODY$
BEGIN
   IF NOT EXISTS(SELECT * FROM MENSAGEM
      INNER JOIN AMIZADE ON ((NEW.ID_PERFIL_ORIGEM = AMIZADE.ID_PERFIL_A OR NEW.ID_PERFIL_ORIGEM =
AMIZADE.ID_PERFIL_B) AND
          (NEW.ID_PERFIL_DESTINO = AMIZADE.ID_PERFIL_A OR NEW.ID_PERFIL_DESTINO = AMIZADE.ID_PERFIL_B))
      WHERE (NEW.DATA_ENVIO_MENSAGEM > AMIZADE.DATA_INICIO)
            AND (NEW.DATA_ENVIO_MENSAGEM < AMIZADE.DATA_FIM)
            AND NEW.ID_PERFIL_DESTINO <> NEW.ID_PERFIL_ORIGEM)
      THEN
          DELETE FROM MENSAGEM WHERE ID_PERFIL_ORIGEM = NEW.ID_PERFIL_ORIGEM;
   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER VALIDA01_MENSAGEM
   AFTER INSERT OR UPDATE ON MENSAGEM
   FOR EACH ROW
   EXECUTE PROCEDURE VALIDAR_MENSAGEM_PELA_AMIZADE();
```

```
-- TRIGGER 09:  ADMINS E PARTICIPANTES DE UM EVENTO QUE JA ACABOU, PRECISAM SER DELETADOS

CREATE OR REPLACE FUNCTION VERIFICA_EVENTO_ATUALIZADO() RETURNS trigger AS $BODY$
BEGIN
   IF((SELECT NEW.DATA_FIM_EVENTO FROM EVENTO WHERE ID_EVENTO = NEW.ID_EVENTO) < current_timestamp)
THEN
      DELETE FROM ADMIN_EVENTO WHERE ADMIN_EVENTO.ID_EVENTO = NEW.ID_EVENTO;
      DELETE FROM PARTICIPA_EVENTO WHERE PARTICIPA_EVENTO.ID_EVENTO = NEW.ID_EVENTO;
   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER PERIODO_CERTO_EVENTO
   AFTER INSERT OR UPDATE ON EVENTO
   FOR EACH ROW
   EXECUTE PROCEDURE VERIFICA_EVENTO_ATUALIZADO();
```

```
-- TRIGGER 10: NÃO SE PODE SER ADMIN DE UM EVENTO QUE JA ACABOU, SERA VERIFICADO SE DATA_FIM DE UM
                -- EVENTO JA PASSOU

CREATE OR REPLACE FUNCTION VERIFICA_DATA_ADMIN() RETURNS trigger AS $BODY$
BEGIN
   IF((SELECT DATA_FIM_EVENTO FROM EVENTO WHERE EVENTO.ID_EVENTO = NEW.ID_EVENTO) < current_timestamp)
THEN
      DELETE FROM ADMIN_EVENTO WHERE ID_EVENTO = NEW.ID_EVENTO;

   END IF;
   RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER PERIODO_CERTO_ADMIN
   AFTER INSERT OR UPDATE ON ADMIN_EVENTO
   FOR EACH ROW
   EXECUTE PROCEDURE VERIFICA_DATA_ADMIN();
```

```
-- TRIGGER 11: NÃO PODE PARTICIPAR DE UM EVENTO QUE JA ACABOU, SERA VERIFICADO SE DATA_FIM DE UM
                -- EVENTO JA PASSOU

CREATE OR REPLACE FUNCTION VERIFICA_DATA_PARTICIPA() RETURNS trigger AS $BODY$
BEGIN
    IF((SELECT DATA_FIM_EVENTO FROM EVENTO WHERE EVENTO.ID_EVENTO = NEW.ID_EVENTO) < current_timestamp)
THEN
        DELETE FROM PARTICIPA_EVENTO WHERE ID_EVENTO = NEW.ID_EVENTO;
    END IF;
    RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER PERIODO_CERTO_PARTICIPA
    AFTER INSERT ON PARTICIPA_EVENTO
    FOR EACH ROW
    EXECUTE PROCEDURE VERIFICA_DATA_PARTICIPA();
```

```
 -- FUNCAO 01: VERIFICAR A POPULARIDADE DE UM DADO PERFIL, ANALISANDO AS REACOES ÀS SUAS PUBLICACOES E COMENTARIOS

CREATE OR REPLACE FUNCTION POPULARIDADE(PESSOA INT) RETURNS varchar(15) AS $BODY$
DECLARE
    coments REAGE_COMENTARIO%rowtype;
    pubs REAGE_PUB%rowtype;
    pontos int;
    classificacao varchar(15);
BEGIN
    pontos:= 0;
    FOR coments IN SELECT * FROM REAGE_COMENTARIO INNER JOIN COMENTARIO ON (REAGE_COMENTARIO.ID_COMENTARIO = COMENTARIO.ID_COMENTARIO) WHERE (COMENTARIO.ID_PERFIL = PESSOA) LOOP
        IF (coments.REACAO = 'LIKE') THEN
            pontos := pontos + 10;
        END IF;
        IF (coments.REACAO = 'LOVE') THEN
            pontos := pontos + 25;
        END IF;
        IF (coments.REACAO = 'ANGER') THEN
            pontos := pontos - 10;
        END IF;
    END LOOP;
```

```
FOR pubs IN SELECT * FROM REAGE_PUB INNER JOIN PUBLICACAO ON (REAGE_PUB.ID_PUBLICACAO =
PUBLICACAO.ID_PUBLICACAO) WHERE (PUBLICACAO.ID_PERFIL_PUB = PESSOA) LOOP
        IF (pubs.REACAO = 'LIKE') THEN
            pontos := pontos + 10;
        END IF;
        IF (pubs.REACAO = 'LOVE') THEN
            pontos := pontos + 25;

        END IF;
        IF (pubs.REACAO = 'ANGER') THEN
            pontos := pontos - 10;
        END IF;
    END LOOP;
```

```
IF (pontos >= 100) THEN
        classificacao := 'super popular';
    END IF;
    IF (pontos >= 50 AND pontos < 100) THEN
        classificacao := 'popular';
    END IF;
    IF (pontos >= 25 AND pontos < 50) THEN
        classificacao := 'neutra';
    END IF;
    IF (pontos >= 0 AND pontos < 25) THEN
        classificacao := 'comum';
    END IF;
    IF (pontos < 0) THEN
        classificacao := 'estranha';
    END IF;

    RETURN classificacao AS RESULT;
END;
$BODY$
LANGUAGE plpgsql;
```

```
-- FUNCAO 02: LOG, RETORNARÁ TODAS AS RELACOES DESEMPENHADAS PELOS PERFIS

CREATE OR REPLACE FUNCTION LOG() RETURNS TABLE(NOME VARCHAR(15), ID_PERFIL INT , AMIGOS INT,
PUBLICACOES INT,
                                    COMENTARIOS INT, MENSAGENS_ENVIADAS INT, MENSAGENS_RECEBIDAS INT,
                                    PARTICIPANDO_EVENTOS INT, ADMIN_EVENTOS INT, REACOES_PUBLICACOES INT,
REACOES_COMENTARIOS INT) AS $$
DECLARE
    qtd_a int;
    qtd_p int;
    qtd_c int;
    qtd_m int;
    qtd_mr int;
    qtd_pe int;
    qtd_ae int;
    qtd_rp int;
    qtd_rc int;
    resp PERFIL%rowtype;
    aux PERFIL%rowtype;
BEGIN
```

```
FOR aux IN SELECT * FROM PERFIL LOOP
        qtd_p := 0;
        qtd_a := 0;
        qtd_c := 0;
        qtd_m := 0;
        qtd_mr := 0;
        qtd_pe := 0;
        qtd_ae := 0;
        qtd_rp := 0;
        qtd_rc := 0;

        FOR resp IN SELECT * FROM PUBLICACAO where aux.ID_PERFIL = PUBLICACAO.ID_PERFIL_PUB LOOP
            qtd_p := qtd_p + 1;
        END LOOP;

        FOR resp IN SELECT ID_COMENTARIO FROM COMENTARIO where aux.ID_PERFIL = COMENTARIO.ID_PERFIL LOOP
           qtd_c := qtd_c + 1;
        END LOOP;

        FOR resp IN SELECT * FROM AMIZADE where (aux.ID_PERFIL = AMIZADE.ID_PERFIL_A) OR (aux.ID_PERFIL =
AMIZADE.ID_PERFIL_B) LOOP
            qtd_a := qtd_a + 1;
        END LOOP;
```

```
        FOR resp IN SELECT ID_MENSAGEM FROM MENSAGEM where aux.ID_PERFIL = MENSAGEM.ID_PERFIL_ORIGEM
LOOP
            qtd_m := qtd_m + 1;
        END LOOP;

        FOR resp IN SELECT ID_MENSAGEM FROM MENSAGEM where aux.ID_PERFIL = MENSAGEM.ID_PERFIL_DESTINO
LOOP
            qtd_mr := qtd_mr + 1;
        END LOOP;

        FOR resp IN SELECT PARTICIPA_EVENTO.ID_PERFIL FROM PARTICIPA_EVENTO where aux.ID_PERFIL =
PARTICIPA_EVENTO.ID_PERFIL LOOP
            qtd_pe := qtd_pe + 1;
        END LOOP;

        FOR resp IN SELECT ADMIN_EVENTO.ID_PERFIL FROM ADMIN_EVENTO where aux.ID_PERFIL =
ADMIN_EVENTO.ID_PERFIL LOOP
            qtd_ae := qtd_ae + 1;
        END LOOP;
```

```
        FOR resp IN SELECT REAGE_PUB.ID_PERFIL FROM REAGE_PUB where aux.ID_PERFIL = REAGE_PUB.ID_PERFIL LOOP
            qtd_rp := qtd_rp + 1;
        END LOOP;

        FOR resp IN SELECT REAGE_COMENTARIO.ID_PERFIL FROM REAGE_COMENTARIO where aux.ID_PERFIL =
REAGE_COMENTARIO.ID_PERFIL LOOP
            qtd_rc := qtd_rc + 1;
        END LOOP;

        RETURN QUERY SELECT aux.NOME, aux.ID_PERFIL, qtd_a, qtd_p, qtd_c, qtd_m, qtd_mr, qtd_pe, qtd_ae, qtd_rp, qtd_rc;
    END LOOP;



END;
$$
LANGUAGE plpgsql;

/* END OF TRIGGERS E FUNCTIONS  */
```

	nome	id_perfil	amigos	publicacoes	comentarios	mensagens_enviadas	mensagens_recebidas	participando_eventos	admin_eventos	reacoes_publicacoes	reacoes_comentarios
1	DANTE	1	2	2	1	3	1	2	3	1	1
2	NERO	2	3	2	2	3	4	3	1	0	1
3	VIRGIL	3	3	1	0	1	1	1	2	2	0
4	EVA	4	2	0	2	1	2	1	0	1	0

	nome	id_perfil	amigos	publicacoes	comentarios	mensagens_enviadas	mensagens_recebidas	participando_eventos	admin_eventos	reacoes_publicacoes	reacoes_comentarios
1	DANTE	1	2	2	1	3	1	1	1	0	0
2	NERO	2	3	2	1	1	2	1	0	0	0
3	VIRGIL	3	2	1	0	1	1	0	1	2	0
4	EVA	4	1	0	1	0	0	0	0	0	0

-- TRIGGER 01: NÃO PERMIRTIR QUE UMA DATA_FIM DE AMIZADE SEJA MENOR QUE UMA DATA_INICIO

```
INSERT INTO PERFIL(NOME, DATA_NASCIMENTO, EMAIL, SENHA) VALUES ('DANTE', '1970-07-15', 'dante@gmail.com', '12345678');
INSERT INTO PERFIL(NOME, DATA_NASCIMENTO, EMAIL, SENHA) VALUES ('NERO', '1990-02-22', 'nero@gmail.com', '12345678');
INSERT INTO PERFIL(NOME, DATA_NASCIMENTO, EMAIL, SENHA) VALUES ('VIRGIL', '1970-07-15', 'virgil@gmail.com', '12345678');
INSERT INTO PERFIL(NOME, DATA_NASCIMENTO, EMAIL, SENHA) VALUES ('EVA', '1000-11-06', 'eva@gmail.com', '12345678');
```

```
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO) VALUES (1, 2, '2003-03-03');
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO, DATA_FIM) VALUES (1, 3, '2002-03-03', '2028-01-01');
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO) VALUES (2, 4, '2010-08-08');
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO) VALUES (3, 2, '2001-11-12');
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO, DATA_FIM) VALUES (4, 3, '2012-03-03', '2010-01-01');
```

	nome	id_perfil	amigos
1	DANTE	1	2
2	NERO	2	3
3	VIRGIL	3	3
4	EVA	4	2

	nome	id_perfil	amigos
1	DANTE	1	2
2	NERO	2	3
3	VIRGIL	3	2
4	EVA	4	1

```
-- TRIGGER 02: VALIDAR COMENTARIO A PARTIR DE UMA AMIZADE,  O COMENTARIO SOMENTE PODERA SER FEITO
            --DURANTE UM  PERIODO DE AMIZADE
-- TRIGGER 03: VALIDAR SE A DATA DO COMENTARIO OCORREU DEPOIS DA PUBLICACAO REFERENCIADA ,
            --VERIFICANDO A DATA DESTA

INSERT INTO PUBLICACAO(ID_PERFIL_PUB, DATA_PUBLICACAO, TEXTO) VALUES (1, '2010-01-13', 'XXXXXXXXXXXXX');
INSERT INTO PUBLICACAO(ID_PERFIL_PUB, DATA_PUBLICACAO, TEXTO) VALUES (1, '2018-03-25', 'XXXXXXXXXXXXX');
INSERT INTO PUBLICACAO(ID_PERFIL_PUB, DATA_PUBLICACAO, TEXTO) VALUES (2, '2017-08-25', 'XXXXXXXXXXXXX');
INSERT INTO PUBLICACAO(ID_PERFIL_PUB, DATA_PUBLICACAO, TEXTO) VALUES (2, '2019-01-06', 'XXXXXXXXXXXXX');
INSERT INTO PUBLICACAO(ID_PERFIL_PUB, DATA_PUBLICACAO, TEXTO) VALUES (3, '2017-08-25', 'XXXXXXXXXXXXX');

INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
            VALUES (4, 3, 'COMENT1', '2020-01-06');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
            VALUES (2, 1, 'COMENT2', '2018-09-25');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
            VALUES (4, 1, 'COMENT3', '2019-08-25');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
            VALUES (1, 3, 'COMENT4', '2019-01-13');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
            VALUES (2, 5, 'COMENT5', '2016-08-25');
```

	nome	id_perfil	amigos	publicacoes	comentarios
1	DANTE	1	2	2	1
2	NERO	2	3	2	2
3	VIRGIL	3	3	1	0
4	EVA	4	2	0	2

	nome	id_perfil	amigos	publicacoes	comentarios
1	DANTE	1	2	2	1
2	NERO	2	3	2	1
3	VIRGIL	3	2	1	0
4	EVA	4	1	0	1

-- TRIGGER 04: VALIDAR A REACAO EM UMA PUBLICACAO A PARTIR DE UMA AMIZADE, VERIFICANDO APENAS SE HÁ
--UMA AMIZADE VIGENTE
-- TRIGGER 05: VALIDAR SE A DATA DA REACAO , INDEPENDENTE SE ESTA SE TRATANDO DE UM INSERT OU UPDATE,
--OCORREU DEPOIS DA PUBLICACAO REFERENCIADA

```
INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (1, 4, 'LIKE');
INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (4, 2, 'ANGER');
INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (3, 1, 'LOVE');
INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (3, 2, 'LOVE');
```

	nome	id_perfil	amigos	publicacoes	reacoes_publicacoes
1	DANTE	1	2	2	1
2	NERO	2	3	2	0
3	VIRGIL	3	3	1	2
4	EVA	4	2	0	1

	nome	id_perfil	amigos	publicacoes	reacoes_publicacoes
1	DANTE	1	2	2	0
2	NERO	2	3	2	0
3	VIRGIL	3	2	1	2
4	EVA	4	1	0	0

```
-- TRIGGER 06: VALIDAR REACAO A UM COMENTARIO A PARTIR DE UMA AMIZADE, VERIFICANDO APENAS SE HÁ
		--AMIZADE VIGENTE
-- TRIGGER 07: VALIDAR SE A DATA DA REACAO, INDEPENDENTE SE ESTA SE TRATANDO DE UM INSERT OU
		--UPDATE, OCORREU DEPOIS DA POSTAGEM DO COMENTARIO REFERENCIADO
/*
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
		VALUES (4, 3, 'COMENT1', '2020-01-06');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
		VALUES (2, 1, 'COMENT2', '2018-09-25');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
		VALUES (4, 1, 'COMENT3', '2019-08-25');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
		VALUES (1, 3, 'COMENT4', '2019-01-13');
INSERT INTO COMENTARIO(ID_PERFIL, ID_PUBLICACAO, COMENTARIO, DATA_COMENTARIO)
		VALUES (2, 5, 'COMENT5', '2016-08-25’);
*/
--INSERT INTO REAGE_COMENTARIO(ID_PERFIL, ID_COMENTARIO, REACAO) VALUES (1, 3, 'ANGER');
INSERT INTO REAGE_COMENTARIO(ID_PERFIL, ID_COMENTARIO, REACAO) VALUES (2, 4, 'LOVE');
INSERT INTO REAGE_COMENTARIO(ID_PERFIL, ID_COMENTARIO, REACAO) VALUES (3, 1 'LOVE');
--INSERT INTO REAGE_COMENTARIO(ID_PERFIL, ID_COMENTARIO, REACAO) VALUES (1, 5, 'LIKE');
```

	nome	id_perfil	amigos	comentarios	reacoes_comentarios
1	DANTE	1	2	1	1
2	NERO	2	3	2	1
3	VIRGIL	3	3	0	0
4	EVA	4	2	2	0

	nome	id_perfil	amigos	comentarios	reacoes_comentarios
1	DANTE	1	2	1	0
2	NERO	2	3	1	0
3	VIRGIL	3	2	0	0
4	EVA	4	1	1	0

```
-- TRIGGER 08: VALIDAR O ENVIO DE UMA MENSAGEM, ELA SÓ PODE SER ENVIADA PARA UM AMIGO. SERÁ
            --VERIFICADO O HISTORICO DE AMIZADES, COMPARANDO COM A DATA DE ENVIO DA MENSAGEM
/*
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO) VALUES (1, 2, '2003-03-03');
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO, DATA_FIM) VALUES (1, 3, '2002-03-03', '2028-01-01');
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO) VALUES (2, 4, '2010-08-08');
INSERT INTO AMIZADE(ID_PERFIL_A, ID_PERFIL_B, DATA_INICIO) VALUES (3, 2, '2001-11-12');
*/
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (1, 2, MSG1', '2010-05-15');
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (1, 3, MSG2', '2030-12-02');
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (2, 4, MSG3', '2010-05-09');
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (2, 2, MSG4', '2010-05-22');
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (3, 1, MSG5', '2010-05-21');
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (1, 3, MSG6', '2010-05-01');
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (2, 4, MSG7', '2012-08-12');
INSERT INTO MENSAGEM(ID_PERFIL_ORIGEM, ID_PERFIL_DESTINO, CONTEUDO_MENSAGEM,
            DATA_ENVIO_MENSAGEM) VALUES (4, 1, MSG8', '2010-05-30');
```

	nome	id_perfil	amigos	mensagens_enviadas	mensagens_recebidas
1	DANTE	1	2	3	1
2	NERO	2	3	3	4
3	VIRGIL	3	3	1	1
4	EVA	4	2	1	2

	nome	id_perfil	amigos	mensagens_enviadas	mensagens_recebidas
1	DANTE	1	2	3	1
2	NERO	2	3	1	1
3	VIRGIL	3	2	1	1
4	EVA	4	1	0	0

```
-- TRIGGER 09:  ADMINS E PARTICIPANTES DE UM EVENTO QUE JA ACABOU, PRECISAM SER DELETADOS
-- TRIGGER 10: NÃO SE PODE SER ADMIN DE UM EVENTO QUE JA ACABOU, SERA VERIFICADO SE DATA_FIM DE UM
        -- EVENTO JA PASSOU


INSERT INTO EVENTO(NOME_EVENTO, DATA_INICIO_EVENTO, DATA_FIM_EVENTO, DESCRICAO_EVENTO,
        ENDERECO_EVENTO, NUMERO_CONFIRMADOS)
                VALUES ('EVENTO B', '2013-02-28', '2019-04-04', DESCR1', 'RUA XXXX', 34);
INSERT INTO EVENTO(NOME_EVENTO, DATA_INICIO_EVENTO, DATA_FIM_EVENTO, DESCRICAO_EVENTO,
        ENDERECO_EVENTO, NUMERO_CONFIRMADOS)
                VALUES ('FESTA X', '2014-11-21', '2014-12-30', DESCR2', 'RUA YYYY', 35);
INSERT INTO EVENTO(NOME_EVENTO, DATA_INICIO_EVENTO, DATA_FIM_EVENTO, DESCRICAO_EVENTO,
        ENDERECO_EVENTO, NUMERO_CONFIRMADOS)
                VALUES ('FESTA Y', '2017-05-08', '2017-07-26', DESCR3', 'RUA ZZZZ', 36);

INSERT INTO ADMIN_EVENTO(ID_PERFIL, ID_EVENTO) VALUES (1, 1);
INSERT INTO ADMIN_EVENTO(ID_PERFIL, ID_EVENTO) VALUES (1, 3);
INSERT INTO ADMIN_EVENTO(ID_PERFIL, ID_EVENTO) VALUES (1, 2);
INSERT INTO ADMIN_EVENTO(ID_PERFIL, ID_EVENTO) VALUES (3, 1);
INSERT INTO ADMIN_EVENTO(ID_PERFIL, ID_EVENTO) VALUES (3, 2);
INSERT INTO ADMIN_EVENTO(ID_PERFIL, ID_EVENTO) VALUES (2, 2);
```

	nome	id_perfil	admin_eventos
1	DANTE	1	3
2	NERO	2	1
3	VIRGIL	3	2
4	EVA	4	0

	nome	id_perfil	admin_eventos
1	DANTE	1	1
2	NERO	2	0
3	VIRGIL	3	1
4	EVA	4	0

```
-- TRIGGER 09:  ADMINS E PARTICIPANTES DE UM EVENTO QUE JA ACABOU, PRECISAM SER DELETADOS
-- TRIGGER 11: NÃO PODE PARTICIPAR DE UM EVENTO QUE JA ACABOU, SERA VERIFICADO SE DATA_FIM DE UM
        -- EVENTO JA PASSOU


INSERT INTO EVENTO(NOME_EVENTO, DATA_INICIO_EVENTO, DATA_FIM_EVENTO, DESCRICAO_EVENTO,
        ENDERECO_EVENTO, NUMERO_CONFIRMADOS)
                VALUES ('EVENTO B', '2013-02-28', '2019-04-04', DESCR1', 'RUA XXXX', 34);
INSERT INTO EVENTO(NOME_EVENTO, DATA_INICIO_EVENTO, DATA_FIM_EVENTO, DESCRICAO_EVENTO,
        ENDERECO_EVENTO, NUMERO_CONFIRMADOS)
                VALUES ('FESTA X', '2014-11-21', '2014-12-30', DESCR2', 'RUA YYYY', 35);
INSERT INTO EVENTO(NOME_EVENTO, DATA_INICIO_EVENTO, DATA_FIM_EVENTO, DESCRICAO_EVENTO,
        ENDERECO_EVENTO, NUMERO_CONFIRMADOS)
                VALUES ('FESTA Y', '2017-05-08', '2017-07-26', DESCR3', 'RUA ZZZZ', 36);

INSERT INTO PARTICIPA_EVENTO(ID_PERFIL, ID_EVENTO, STATUS_COMPARECIMENTO_EVENTO) VALUES (1, 1, 'SIM');
INSERT INTO PARTICIPA_EVENTO(ID_PERFIL, ID_EVENTO, STATUS_COMPARECIMENTO_EVENTO) VALUES (1, 2, 'NAO');
INSERT INTO PARTICIPA_EVENTO(ID_PERFIL, ID_EVENTO, STATUS_COMPARECIMENTO_EVENTO) VALUES (2, 1, 'NAO');
INSERT INTO PARTICIPA_EVENTO(ID_PERFIL, ID_EVENTO, STATUS_COMPARECIMENTO_EVENTO) VALUES (2, 3, 'SIM');
INSERT INTO PARTICIPA_EVENTO(ID_PERFIL, ID_EVENTO, STATUS_COMPARECIMENTO_EVENTO) VALUES (2, 2, 'SIM');
INSERT INTO PARTICIPA_EVENTO(ID_PERFIL, ID_EVENTO, STATUS_COMPARECIMENTO_EVENTO) VALUES (3, 2, 'NAO');
INSERT INTO PARTICIPA_EVENTO(ID_PERFIL, ID_EVENTO, STATUS_COMPARECIMENTO_EVENTO) VALUES (4, 3, 'SIM');
```

	nome	id_perfil	participando_eventos
1	DANTE	1	2
2	NERO	2	3
3	VIRGIL	3	1
4	EVA	4	1

	nome	id_perfil	participando_eventos
1	DANTE	1	1
2	NERO	2	1
3	VIRGIL	3	0
4	EVA	4	0

```
-- TESTANDO A FUNCAO 01: POPULARIDADE (LIKE  +10, LOVE + 25,  ANGER -10)
-- CLASSIFICACAO: SUPER POPULAR (PONTOS>=100), POPULAR (50<=PONTOS < 100), NEUTRO(25<=PONTOS<50),
--COMUM(0<=PONTOS<25) E ESTRANHO(PONTOS<0).


INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (1, 4, 'LIKE');
INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (4, 4, 'ANGER');
INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (3, 1, 'LOVE');
INSERT INTO REAGE_PUB(ID_PERFIL, ID_PUBLICACAO, REACAO) VALUES (3, 2, 'LOVE');


INSERT INTO REAGE_COMENTARIO(ID_PERFIL, ID_COMENTARIO, REACAO) VALUES (2, 4, 'LOVE'); -- 1
INSERT INTO REAGE_COMENTARIO(ID_PERFIL, ID_COMENTARIO, REACAO) VALUES (3, 1 'LOVE');  -- 4

SELECT POPULARIDADE(3);
SELECT POPULARIDADE(1);
```

	popularidade
1	comum

	popularidade
1	popular

OBRIGADO