Universidade de São Paulo
Escola de Artes, Ciências e Humanidades
Disciplina: Laboratório de Banco de Dados
Profª Dra. Fátima Nunes.

Administração de Condomínio

Parte II

Fernando K. G. de Amorim – 10387644
João Guilherme da Costa Seike – 9784634
Lucas Pereira Castelo Branco – 10258772
Victor Gomes de O. M. Nicola – 9844881

São Paulo
2019

Nesta segunda parte do trabalho para a matéria de Laboratório de Banco de Dados, precisamos enunciar algumas regras de negócio para fazer asserções, *triggers* e visões. Esta segunda parte foi subdividida em outras quatro partes segundo o enunciado.

LETRA A

Regras de negócios para a parte (a):

1. Um membro do corpo administrativo só pode ter dois mandatos seguidos para a mesma função, após isso ele deve ficar um ano sem poder ser nomeado para a mesma função. Só pode haver uma eleição por ano e não podem haver inserções para um ano anterior ao atual.

Enunciado textual

Seguindo regras condominiais, um membro do corpo administrativo só pode ser eleito duas vezes seguidas para a mesma função, podendo se candidatar novamente para a mesma função após um ano sem ser eleito para aquela função específica. Só ocorre uma eleição de um corpo administrativo, e não obstante, devem ser impedidas entradas de eleições de anos anteriores ao ano atual vigente, as eleições só podem ser feitas a partir do ano atual. Para verificar se a eleição está de acordo com as regras, são verificadas as datas e a quantidade de vezes que aquela pessoa foi eleita para os últimos dois anos.

Solução textual em SQL padrão

```
CREATE ASSERTION eleicao_corpo_adm CHECK (
NOT EXISTS ( SELECT * FROM corpo_administrativo WHERE
( IF ( NEW.data_eleicao (YEAR) >= NOW() (YEAR) ) ) AND
( IF NOT EXISTS ( SELECT * FROM corpo_administrativo c_adm
        WHERE c_adm.data_eleicao (YEAR) =
        NEW.data_eleicao (YEAR) ) ) AND
( IF ( SELECT COUNT(*) FROM edua WHERE edua.id_sindico = NEW.id_sindico) < 2) AND
( IF ( SELECT COUNT(*) FROM edua WHERE edua.id_subsindico = NEW.id_subsindico) < 2 ) AND
( IF ( SELECT COUNT(*) FROM edua WHERE edua.id_conselheiro_1 = NEW.id_conselheiro_1) < 2 ) AND
( IF ( SELECT COUNT(*) FROM edua WHERE edua.id_conselheiro_2 = NEW.id_conselheiro_2) < 2 ) AND
( IF ( SELECT COUNT(*) FROM edua WHERE edua.id_conselheiro_3 = NEW.id_conselheiro_3) < 2 ) );
```

Solução em código implementada

Esta é uma imagem da solução em código implementada dentro do SGBD PostgreSQL:

```sql
CREATE OR REPLACE FUNCTION fc_deleta_apartamentos()
RETURNS trigger AS $tr_dapartamentos$
BEGIN

    CREATE TABLE table_holder AS (SELECT fk_id_moradia FROM adm_condominio.Moradia_Edificio WHERE fk_id_edificio = old.id_edificio);


    DELETE FROM adm_condominio.Moradia_Pessoa WHERE fk_id_moradia IN
        (SELECT id_moradia from adm_condominio.Moradia WHERE id_moradia IN (SELECT * FROM table_holder));

    DELETE FROM adm_condominio.Condominio_Moradia WHERE  fk_id_moradia IN (SELECT * FROM table_holder);
    DELETE FROM adm_condominio.Apartamento WHERE  fk_id_moradia IN (SELECT * FROM table_holder);
    DELETE FROM adm_condominio.Moradia_Edificio WHERE fk_id_edificio = old.id_edificio;
    DELETE FROM adm_condominio.Moradia WHERE  id_moradia IN (SELECT * FROM table_holder);
    DROP TABLE table_holder;
RETURN NEW;
END;
$tr_dapartamentos$ LANGUAGE plpgsql;

CREATE TRIGGER tr_deleta_apartamentos
AFTER DELETE ON adm_condominio.Edificio
FOR EACH ROW
EXECUTE PROCEDURE fc_deleta_apartamentos();
```

```sql
CREATE OR REPLACE FUNCTION eleicao_corpo_adm() RETURNS TRIGGER AS $$
BEGIN
    DROP TABLE IF EXISTS elec_dois_ult_anos;
    CREATE TEMP TABLE elec_dois_ult_anos
    (
        id_corpo INT,
        id_sindico INT,
        id_subsindico INT,
        id_conselheiro_1 INT,
        id_conselheiro_2 INT,
        id_conselheiro_3 INT
    );

    INSERT INTO elec_dois_ult_anos
    SELECT id_corpo, id_sindico, id_subsindico, id_conselheiro_1, id_conselheiro_2, id_conselheiro_3
    FROM adm_condominio.corpo_administrativo adm_ca
    WHERE DATE_PART('year', adm_ca.data_eleicao) = DATE_PART('year', (NEW.data_eleicao - interval '1 year'))
    UNION
    SELECT id_corpo, id_sindico, id_subsindico, id_conselheiro_1, id_conselheiro_2, id_conselheiro_3
    FROM adm_condominio.corpo_administrativo adm_ca
    WHERE DATE_PART('year', adm_ca.data_eleicao) = DATE_PART('year', (NEW.data_eleicao - interval '2 years'));


    -- verificar ano de eleicao
    IF (DATE_PART('year', NEW.data_eleicao) >= DATE_PART('year',NOW())) THEN

    -- verifica se ja ocorreu uma eleicao naquele ano
    IF NOT EXISTS
    (
        SELECT *
        FROM adm_condominio.corpo_administrativo adm_ca
        WHERE DATE_PART('year', adm_ca.data_eleicao) = DATE_PART('year', NEW.data_eleicao)
    )
    THEN
```

```
-- verifica síndico ja foi eleito 2 vezes
IF
(
    SELECT COUNT(*) FROM elec_dois_ult_anos edua
    WHERE edua.id_sindico = NEW.id_sindico
) < 2 THEN

-- verifica sub-síndico ja foi eleito 2 vezes
IF
(
    SELECT COUNT(*) FROM elec_dois_ult_anos edua
    WHERE edua.id_subsindico = NEW.id_subsindico
) < 2 THEN

-- verifica conselheiro 1 ja foi eleito 2 vezes
IF
(
    SELECT COUNT(*) FROM elec_dois_ult_anos edua
    WHERE edua.id_conselheiro_1 = NEW.id_conselheiro_1
) < 2 THEN


-- verifica conselheiro 2 ja foi eleito 2 vezes
IF
(
    SELECT COUNT(*) FROM elec_dois_ult_anos edua
    WHERE edua.id_conselheiro_2 = NEW.id_conselheiro_2
) < 2 THEN


-- verifica conselheiro 3 ja foi eleito 2 vezes
IF
(
    SELECT COUNT(*) FROM elec_dois_ult_anos edua
    WHERE edua.id_conselheiro_3 = NEW.id_conselheiro_3
) < 2 THEN

    RETURN NEW;
```

```
    -- else conselheiro 3 ja eleito
    ELSE RAISE EXCEPTION 'O conselheiro 3 já foi eleito duas vezes seguidas nos últimos anos!';
    END IF;
    -- else conselheiro 2 ja eleito
    ELSE RAISE EXCEPTION 'O conselheiro 2 já foi eleito duas vezes seguidas nos últimos anos!';
    END IF;
    -- else conselheiro 1 ja eleito
    ELSE RAISE EXCEPTION 'O conselheiro 1 já foi eleito duas vezes seguidas nos últimos anos!';
    END IF;
    -- else subsindico ja eleito
    ELSE RAISE EXCEPTION 'O sub-síndico já foi eleito duas vezes seguidas nos últimos anos!';
    END IF;
    -- else sindico ja eleito
    ELSE RAISE EXCEPTION 'O síndico já foi eleito duas vezes seguidas nos últimos anos!';
    END IF;
    -- else ja ocorreu eleicao
    ELSE RAISE EXCEPTION 'Um corpo já foi eleito esse ano!';
    END IF;
    -- else ano anterior ao atual
    ELSE RAISE EXCEPTION 'Você não pode eleger um corpo para um ano anterior ao atual!';
    END IF;

    DROP TABLE IF EXISTS elec_dois_ult_anos;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER verifica_eleicao BEFORE INSERT ON adm_condominio.corpo_administrativo
    FOR EACH ROW EXECUTE PROCEDURE eleicao_corpo_adm();
```

Transcrição

A asserção precisa primeiro ter a declaração de uma tabela temporária com os dados do SELECT apresentado, que insere as tuplas dos dois últimos anos relacionados às eleições.
As verificações começam da seguinte forma:

1. É criada a tabela temporária e os dados são inseridas na mesma através de INSERT INTO com SELECT e um UNION, o só é possível encadear tuplas de anos diferentes com o UNION para este caso.
2. É verificado se o ano da eleição é igual ou superior ao ano atual.
3. É verificado com base na data da inserção a ser feita, se já foi eleito um corpo administrativo no ano inserido.
4. Começam então uma série de verificações para cada um dos cargos do corpo a ser inserido, caso um dos cargos tenha uma pessoa que esteja sendo eleita, e já possui dois mandatos seguidos dos dois últimos anos, uma exceção é gerada, aquela pessoa não pode ser elegida por cerca de um ano.
5. As exceções se aplicam a cada um dos cargos e caso as datas não passem nas verificações também.
6. Por fim a tabela temporária é excluída.

Está incluso também a criação do *trigger* para a asserção, o *trigger* é executado para cada tupla e antes de ser inserido.

Regras de negócios para a parte (a):

2. Um condômino só pode reservar até 3 espaços do condomínio no máximo por mês do ano. O condômino não pode reservar mais de um espaço por dia. O condomínio deve garantir que só deve haver uma reserva de um dado espaço no dia, que deve se encerrar até as 22h e deve durar até no máximo 6h de duração.

Enunciado textual

Para que um condomínio tenha um controle das reservas dos espaços, são necessários também que algumas regras sejam seguidas como em qualquer outro local da sociedade. Respeitando a lei do silêncio, as reservas dos espaços devem se encerrar até as 22h do mesmo dia da reserva e cada reserva deve possuir duração máxima de 6 horas para não atrapalhar outros moradores. Claro que para isso, é necessário que não ocorram conflitos de reservas e datas. Uma pessoa que já alugou um espaço no dia, não pode alugar outro para dar chance aos outros moradores de poderem alugar. O espaço só pode ser reservado uma vez por dia, ou seja, caso o mesmo tenha sido alugado, não poderá ser alugado novamente no dia. E por fim, o condômino só pode alugar um máximo de 3 vezes por mês em seu condomínio, garantindo que não fique alugando espaços todos os dias.

Solução textual em SQL padrão

```
CREATE ASSERTION reserva_espaco_unico CHECK (
        NOT EXISTS ( SELECT * FROM reserva WHERE
        ( IF ( SELECT NEW.hora_final <= '22:00:00')) AND
        ( IF ( SELECT NEW.hora_final - NEW.hora_inicial <= '06:00:00' )) AND
        ( IF NOT EXISTS ( SELECT * FROM reserva WHERE fk_id_pessoa = NEW. fk_id_pessoa AND
                        data = NEW.data )) AND
        ( IF NOT EXISTS ( SELECT * FROM reserva WHERE fk_id_espaco = NEW.fk_id_espaco AND
                        data = NEW.data )) AND
        ( IF ( SELECT COUNT(*) FROM reserva
                WHERE fk_id_pessoa = NEW.fk_id_pessoa AND
                data (MONTH) = NEW.data (MONTH) AND
                data (YEAR) = NEW.data (YEAR)) < 3 ));
```

Solução em código implementada

Esta é uma imagem da solução em código implementada dentro do SGBD PostgreSQL:

```sql
CREATE OR REPLACE FUNCTION reserva_espaco_unico() RETURNS TRIGGER AS $$
BEGIN
    -- verifica se o horario de encerramento da reserva eh superior a 22h
    IF
    (SELECT NEW.hora_final) <= time '22:00:00' THEN

    -- verifica se a duracao da reserva eh maior que 6h
    IF
    (SELECT NEW.hora_final - NEW.hora_inicial) <= interval '06:00:00' THEN

    -- verifica se ele ja alugou um espaco no dia
    IF NOT EXISTS
    (
        SELECT * FROM adm_condominio.reserva adm_r
        WHERE adm_r.fk_id_pessoa = NEW.fk_id_pessoa AND
        adm_r.data = NEW.data
    ) THEN

    -- verifica se o espaco ja foi alugado naquele dia
    IF NOT EXISTS
    (
        SELECT * FROM adm_condominio.reserva adm_r
        WHERE adm_r.fk_id_espaco = NEW.fk_id_espaco AND
        adm_r.data = NEW.data
    ) THEN
```

```sql
    -- verifica se ele ja alugou 3 espaços no mes do mesmo ano
    IF
    (
        SELECT COUNT(*) FROM adm_condominio.reserva adm_r
        WHERE adm_r.fk_id_pessoa = NEW.fk_id_pessoa AND
        DATE_PART('month', adm_r.data) = DATE_PART('month', NEW.data) AND
        DATE_PART('year', adm_r.data) = DATE_PART('year', NEW.data)
    ) < 3 THEN
            RETURN NEW;

    -- else pessoa ja tem mais que 3 reservas
    ELSE RAISE EXCEPTION 'O condômino já tem mais que 3 reservas!';
    END IF;
    -- else espaco reservado no dia
    ELSE RAISE EXCEPTION 'Esse espaço já foi reservado neste dia!';
    END IF;
    -- else condomino ja reservou um espaco no dia
    ELSE RAISE EXCEPTION 'Este condômino já reservou um espaço hoje!';
    END IF;
    -- else duracao 6h
    ELSE RAISE EXCEPTION 'A duração da reserva ultrapassa 6 horas!';
    END IF;
    -- else hora final 22h
    ELSE RAISE EXCEPTION 'A hora final da reserva deve ser até 22h!';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

```sql
CREATE TRIGGER reserva_espacos BEFORE INSERT ON adm_condominio.reserva
    FOR EACH ROW EXECUTE PROCEDURE reserva_espaco_unico();
```

Transcrição

O código implementado funciona da seguinte maneira:

1. É feita uma verificação para se o horário final da reserva é menor ou igual a 22h, que é o horário do silêncio.
2. A segunda verificação feita é em relação ao horário de duração da reserva, o limite de duração para uma reserva é de 6h.
3. Como terceira verificação, é executado uma operação de SELECT para verificar se não existe nenhuma reserva daquele condômino na mesma data, caso exista, será impedido de fazê-la.
4. Na quarta verificação, um outro SELECT é executado para verificar se o espaço já foi alugado naquele dia por outra pessoa.
5. E por último, contabiliza quantas vezes o condômino fez solicitações de reserva, e caso passem de 3, ele não permite que ele faça outra reserva, sendo necessário esperar o próximo mês do ano.

Após isso, é criado o *trigger* com a asserção descrita na função acima, executando o *trigger* sempre antes da inserção e para cada tupla.