

Ciência da Computação

UNIVALI / CTTMAR

Bancos de Dados II

6º Período

Aula 03

Professor: Marcelo Magnani (marcelom@univali.br)

Período: 2015-2 C/H: 72

Cronograma

- *Triggers*

Triggers

- Uma *trigger* (gatilho) é um procedimento armazenado que é executado quando um dado evento ocorre (o gatilho é disparado).



Triggers

- Uma função de gatilho pode ser criada para executar antes (**BEFORE**) ou após (**AFTER**) as consultas **INSERT, UPDATE OU DELETE**, uma vez para cada registro (linha) modificado ou por instrução SQL.
- Logo que ocorre um desses eventos do gatilho, a função é disparada automaticamente para tratar o evento.

Triggers

- A função de gatilho deve ser declarada como uma função que **não recebe argumentos** e que **retorna o tipo TRIGGER**.
- Após criar a função de gatilho, estabelecemos o gatilho pelo comando CREATE TRIGGER.
- Uma função de gatilho pode ser utilizada por vários gatilhos.

Triggers

- Quando uma função que retorna o tipo trigger é executada, o PostgreSQL cria algumas variáveis especiais na memória. Essas variáveis podem ser usadas no corpo das funções.

Triggers


- **NEW:** Variável (tipo RECORD) que armazena os valores novos da tupla. Disponível para operações de INSERT/UPDATE e “row-level triggers”. Possui valor NULL para operação DELETE ou “statement-level triggers”.
- **OLD:** Variável (tipo RECORD) que armazena os valores antigos da tupla. Disponível para operações de UPDATE/DELETE e “row-level triggers”. Possui valor NULL para operação INSERT ou “statement-level triggers”.
- **TG_OP:** Variável (tipo TEXTO) que armazena a string da operação (INSERT, UPDATE, DELETE ou TRUNCATE).
- **TG_TABLE_NAME:** Variável (tipo TEXTO) que armazena a string da tabela que disparou a trigger.

```

CREATE OR REPLACE FUNCTION atualiza_situacao_aluno() RETURNS trigger AS $$
DECLARE
    final float := 0;
    cont int := 0;
BEGIN
    IF NEW.m1 is not null THEN
        final := final + NEW.m1;
        cont := cont+1;
    END IF;
    IF NEW.m2 is not null THEN
        final := final + NEW.m2;
        cont := cont+1;
    END IF;
    IF NEW.m3 is not null THEN
        final := final + NEW.m3;
        cont := cont+1;
    END IF;
    NEW.mf = final/cont;
    IF NEW.mf >= 5.75 THEN
        NEW.situacao := 'Aprovado';
    ELSE
        NEW.situacao := 'Reprovado';
    END IF;
    IF cont < 3 THEN
        NEW.situacao := NEW.situacao || ' Aguardando...';
    END IF;
    RETURN NEW;

END;
$$ LANGUAGE plpgsql;

```

| aluno  |
|------------------------------------------------------------------------------------------------------------------------------------------|
| «column» *PK id: integer m1: decimal(10,2) m2: decimal(10,2) m3: decimal(10,2) mf: decimal(10,2) situacao: varchar(50) |
| «PK» + PK_aluno(integer) |

```

CREATE TRIGGER situacao_aluno BEFORE INSERT OR UPDATE ON aluno
FOR EACH ROW EXECUTE PROCEDURE atualiza_situacao_aluno();

```



```


CREATE OR REPLACE FUNCTION log_evento() RETURNS trigger AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO aluno_log (id_aluno, acao, data, dados_novos)
            VALUES (NEW.id, TG_OP, now(), NEW);
        RETURN NEW;
    ELSEIF TG_OP = 'DELETE' THEN
        INSERT INTO aluno_log (id_aluno, acao, data, dados_antigos)
            VALUES (OLD.id, TG_OP, now(), OLD);
        RETURN NULL;
    ELSEIF TG_OP = 'UPDATE' THEN
        INSERT INTO aluno_log (id_aluno, acao, data, dados_novos, dados_antigos)
            VALUES (NEW.id, TG_OP, now(), NEW, OLD);
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;

```

```

CREATE TRIGGER log_aluno AFTER INSERT OR UPDATE OR DELETE ON aluno
FOR EACH ROW EXECUTE PROCEDURE log_evento();

```

| aluno_log  | |
|-----------------------------------------------------------------------------------------------|-----------------------|
| «column» | |
| *PK id: integer | |
| id_aluno: integer | |
| acao: varchar(255) | |
| data: timestamp | |
| «PK» | |
| + | PK_aluno_log(integer) |

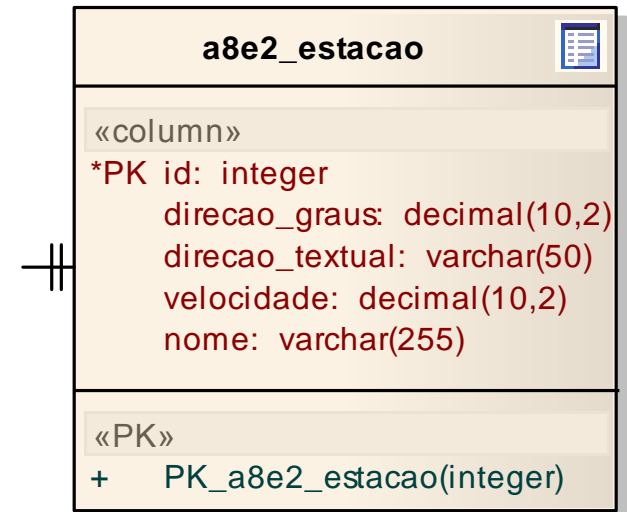
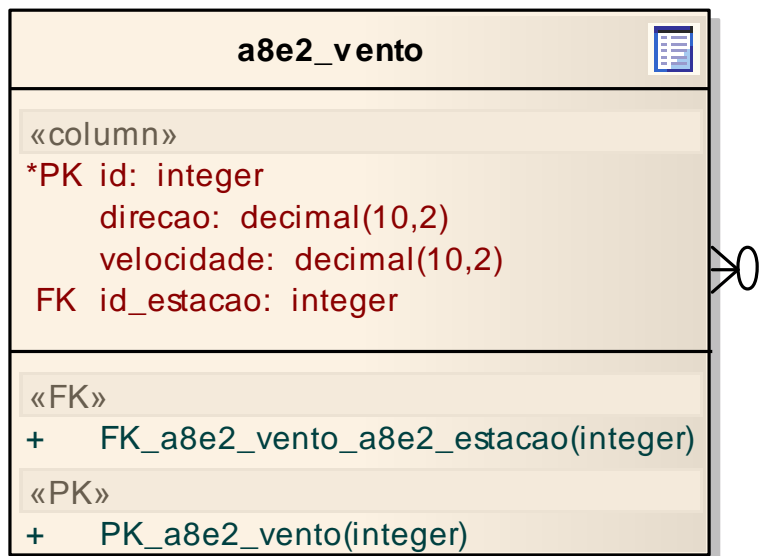
Exercício 01

- Crie uma função de gatilho que, antes de inserir ou atualizar a tabela a7e4_calculadoralog, preencha a coluna “resultado” com o valor da operação especificada.
 - INSERT INTO a7e4_calculadoralog (valor1, valor2, operacao) VALUES (5,6,'+');
 - UPDATE a7e4_calculadoralog SET valor1 = 10 WHERE id = 2;

| a7e4_calculadoralog | |
|---------------------|---------------------------------|
| «column» | |
| *PK | id: integer |
| * | valor1: decimal(10,2) |
| * | operacao: char(10) |
| * | valor2: decimal(10,2) |
| * | resultado: decimal(10,2) |
| «PK» | |
| + | PK_a7e4_calculadoralog(integer) |

Exercício 02

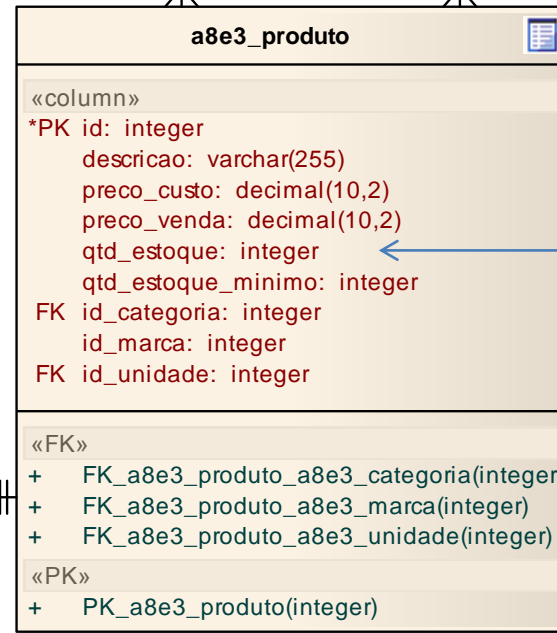
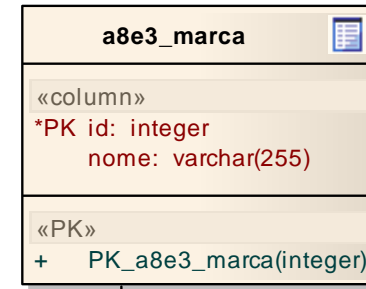
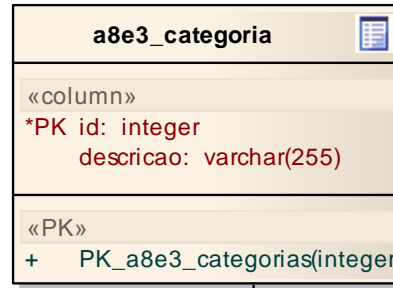
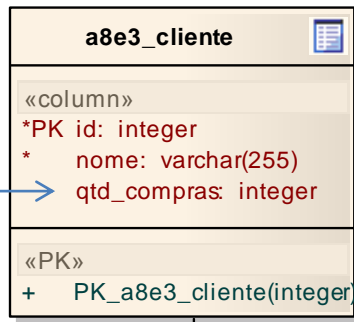
- Crie uma função de gatilho que ao inserir ou atualizar um registro na tabela vento, preencha as informações de condição atual na tabela estacao.
 - INSERT INTO vento (direcao, velocidade,id_estacao) VALUES (50, 10, 1);--direção textual na estação: Oeste
 - INSERT INTO vento (direcao, velocidade,id_estacao) VALUES (270, 10, 1); --direção textual na estação: Leste



Exercício 03

- Crie uma trigger que ao inserir ou excluir um registro na tabela venda, atualize a coluna correspondente a quantidade de compras do cliente. 1
- Crie uma trigger que ao inserir, atualizar ou excluir um registro na tabela venda_produto:
 - Calcule e preencha a coluna do preço total do item 2
 - Atualize a quantidade do estoque do produto; 3
 - Atualize a coluna preço total na tabela “venda”; 4

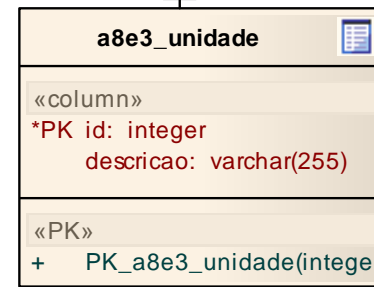
1



3



2



4