Banco de Dados II

Trigger

PROF. DR. THIAGO ELIAS

Trigger

- Triggers são procedimentos armazenados que são acionados por algum evento e em determinado momento. Estes eventos podem ser inserções (INSERT), atualizações (UPDATE) e exclusões (DELETE) (ou TRUNCATE), e os momentos, no caso de tabelas, podem ser dois: antes da execução do evento (BEFORE) ou depois (AFTER).
- No caso de visões, há a opção de INSTEAD OF

 Um diferencial das triggers deste banco de dados para outros é que no PostgreSQL as triggers são sempre associadas a funções de triggers e, nos demais, criamos o corpo da trigger na própria declaração desta.

- O PostgreSQL possui dois tipos de triggers:
 - o triggers-por-linha: é disparada uma vez para cada registro afetado pela instrução que disparou a trigger.
 - o triggers-por-instrução: é disparada somente uma vez quando a instrução é executada.
- A utilização delas pode ser definida de acordo com a quantidade de vezes que a trigger deverá ser executada.
- Por exemplo, se uma instrução UPDATE for executada, e esta afetar seis linhas, temos que a trigger de nível de linha será executada seis vezes, enquanto que a trigger a nível de instrução será chamada apenas uma vez por instrução SQL.

 O gatilho fica associado à tabela especificada e executa a função especificada nome_da_função quando determinados eventos ocorrerem.

Sintaxe:

CREATE TRIGGER nome { BEFORE | AFTER } { evento [OR ...] }
ON tabela [FOR [EACH] { ROW | STATEMENT }]
EXECUTE PROCEDURE nome_da_função (argumentos)

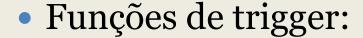
 O gatilho pode ser especificado para disparar antes de tentar realizar a operação na linha (antes das restrições serem verificadas e o comando INSERT, UPDATE ou DELETE ser tentado), ou após a operação estar completa (após as restrições serem verificadas e o INSERT, UPDATE ou DELETE ter completado).

Funções de trigger:

 São funções que não recebem nenhum parâmetro e retornam o tipo trigger.

- Obs1: As funções de gatilho chamadas por gatilhos-porinstrução devem sempre retornar NULL.
- Obs2: As funções de gatilho chamadas por gatilhos-por-linha podem retornar uma linha da tabela (exemplo: new). Devemos retornar uma linha para indicar ao PostgreSQL que ele deve continuar realizando a operação.

- O PostgreSQL disponibiliza duas variáveis importantes para serem usadas em conjunto com as triggers-por-linha: NEW e OLD.
 - A variável NEW, no caso do INSERT, armazena o registro que está sendo inserido. No caso do UPDATE, armazena a nova versão do registro depois da atualização.
 - A variável OLD, no caso do DELETE, armazena o registro que está sendo excluído. No caso do UPDATE, armazena a antiga versão do registro depois da atualização.



CREATE FUNCTION empregados_gatilho() RETURNS trigger AS \$empregados_gatilho\$
BEGIN

-- Verificar se foi fornecido o nome e o salário do empregado IF NEW.nome IS NULL THEN RAISE EXCEPTION 'O nome do empregado não pode ser nulo'; END IF;

RETURN NEW; END; \$empregados_gatilho\$ LANGUAGE plpgsql;

Trigger INSTEAD OF em Visões de BD

- Considere a tabela funcionário com os atributos código, nome e e-mail.
- Também considere a visão VISAO_FUNCIONARIO que mostra toda a tabela funcionário.
- No exemplo abaixo, sempre que for realizado um INSERT na VISAO_FUNCIONARIO, este será substituído por um INSERT na tebela FUNCIONARIO

```
CREATE FUNCTION FUNCAO_1()
RETURNS trigger AS $$
BEGIN
IF (TG_OP = 'INSERT') THEN
INSERT INTO funcionario VALUES (NEW.codigo_func, NEW.nome, NEW.email);
RETURN NEW;
END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

CREATE TRIGGER disparador
INSTEAD OF INSERT ON VISAO_FUNCIONARIO
FOR EACH ROW
EXECUTE PROCEDURE FUNCAO_1();

- Podemos ter mais de uma trigger associada ao mesmo evento e momento.
- Neste caso a ordem de execução das triggers é definida pela ordem alfabética de seus nomes.

Triggers Recursivas:

- Se uma função de trigger executar comandos SQL, estes comandos podem disparar triggers novamente. Isto é conhecido como cascatear triggers.
- É possível que o cascateamento cause chamadas recursivas da mesma trigger. Por exemplo, um trigger para INSERT pode executar um comando que insere uma linha adicional na mesma tabela, fazendo com que o trigger para INSERT seja disparado novamente.
- É responsabilidade do programador evitar recursões infinitas nestes casos.

Alterando nome do Trigger:

ALTER TRIGGER nome ON tabela RENAME TO novo_nome

• Excluindo um Trigger:

DROP TRIGGER nome ON tabela [CASCADE | RESTRICT]

Habilitando ou desabilitando Triggers:

ALTER TABLE nome_tabela
DISABLE TRIGGER nome_trigger

ALTER TABLE nome_tabela
DISABLE TRIGGER ALL

 Para habilitar uma trigger basta substituir o parâmetro DISABLE por ENABLE

• Exercício 1 de 4:

Crie uma tabela aluno com as colunas matrícula e nome.
 Depois crie um trigger que não permita o cadastro de alunos cujo nome começa com a letra "a".

• Exercício 2 de 4:

- Primeiro crie uma tabela chamada Funcionário com os seguintes campos: código (int), nome (varchar(30)), salário (int), data_última_atualização (timestamp), usuário_que_atualizou (varchar(30)). Na inserção desta tabela, você deve informar apenas o código, nome e salário do funcionário. Agora crie um Trigger que não permita o nome nulo, a salário nulo e nem negativo. Faça testes que comprovem o funcionamento do Trigger.
 - ▼ Obs: Raise Exception, 'now' e current_user

• Exercício 3 de 4:

- Agora crie uma tabela chamada Empregado com os atributos nome e salário. Crie também outra tabela chamada Empregado_auditoria com os atributos: operação (char(1)), usuário (varchar), data (timestamp), nome (varchar), salário (integer). Agora crie um trigger que registre na tabela Empregado_auditoria a modificação que foi feita na tabela empregado (E,A,I), quem fez a modificação, a data da modificação, o nome do empregado que foi alterado e o salário atual dele.
 - ▼ Obs: variável especial TG_OP

• Exercício 4 de 4:

O Crie a tabela Empregado2 com os atributos código (serial e chave primária), nome (varchar) e salário (integer). Crie também a tabela Empregado2_audit com os seguintes atributos: usuário (varchar), data (timestamp), id (integer), coluna (text), valor_antigo (text), valor_novo(text). Agora crie um trigger que não permita a alteração da chave primária e insira registros na tabela Empregado2_audit para refletir as alterações realizadas na tabela Empregado2.