Instituto Federal

Campus Goiânia

Bacharelado em Sistemas de Informação

Banco de Dados II

Prof. Dory Gonzaga Rodrigues

MySQL®

ORACLE®

PostgreSQL
the world's most advanced open source database

Agenda

- Objetos Avançados

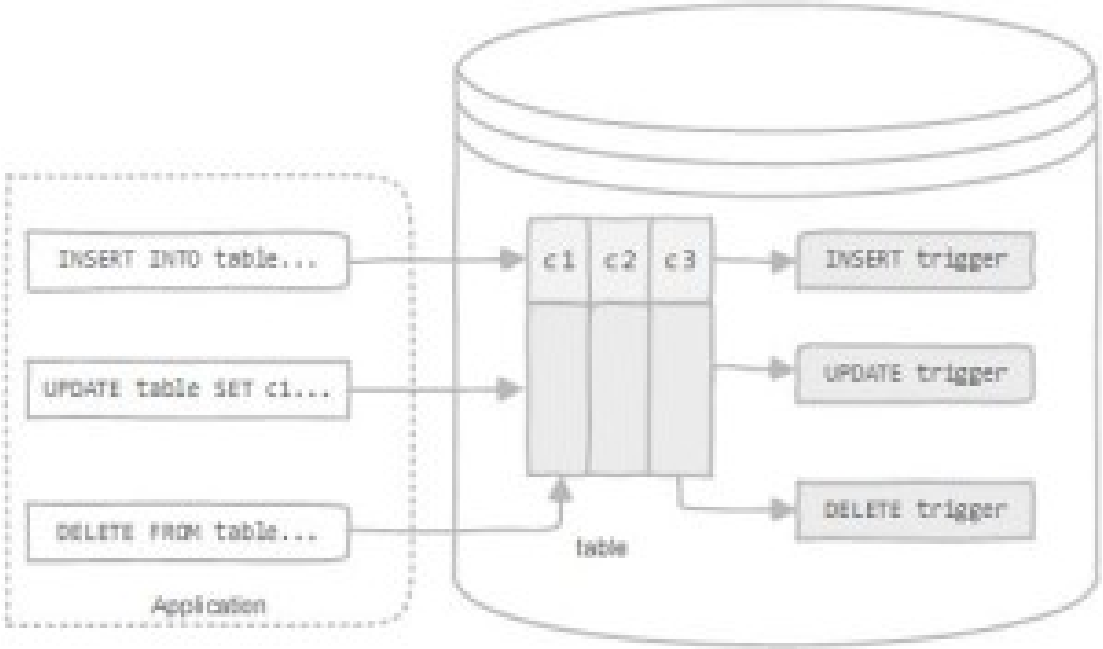
  - TRIGGERS

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Uma TRIGGER no PostgreSQL é uma função invocada/chamada automaticamente sempre que um evento (Insert, Update, ou Delete) ocorre.



OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGER x FUNCTION/PROCEDURE

A principal diferença entre uma TRIGGER e uma FUNCTION/PROCEDURE é que um gatilho aguarda a ocorrência de determinado evento para ser executado automaticamente, já a função ou procedimento só é executado quando invocado explicitamente.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGER

Podemos dizer então que uma TRIGGER é

- Um tipo específico de função ou procedimento armazenado.

- Um conjunto de instruções SQL armazenadas no catálogo de banco de dados.

- Executado ou acionado sempre que um evento associado com uma tabela ocorre, por exemplo, inserção, atualização ou exclusão.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Vantagens do uso de TRIGGERS

- Proporcionam uma forma alternativa de verificar a integridade dos dados.

- Pode pegar erros de lógica de negócios na camada de banco de dados.

- Fornece uma forma alternativa de executar tarefas agendadas. Neste caso, você não tem que esperar para executar tarefas agendadas porque os gatilhos são invocados automaticamente antes ou após a alteração de dados realizada na tabela.

- São muito úteis para auditar/verificar as alterações de dados em tabelas.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Desvantagens

Só pode fornecer uma validação estendida e eles não podem substituir todas as validações. Algumas validação necessariamente devem ser feitas na camada de aplicação. Por exemplo:

Validar entradas do usuário nos formulários deve ser feita no lado do cliente usando JavaScript ou no lado do servidor usando linguagens de script, tais como JSP, PHP, ASP.NET, Perl , etc.

Como os TRIGGERS são chamados e executados de forma invisível das aplicações cliente, fica difícil descobrir o que acontece na camada de banco de dados. Além da possibilidade de sobrecarga do servidor de banco de dados.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Definindo Gatilhos

- Um gatilho é uma função especial definida pelo usuário que se liga a uma tabela.

- Para criar um novo gatilho, você deve definir uma função de gatilho primeiro e, em seguida, vincular essa função de gatilho a uma tabela.

- No PostgreSQL temos dois tipos principais de Gatilhos:

1) Nível de linha
2) Declaração

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Definindo Gatilhos

- No PostgreSQL temos dois tipos principais de Gatilhos:

As diferenças entre os dois são quantas vezes o disparador é invocado e em que momento.

1) Nível de linha: o gatilho será acionado a cada linha afetada por um evento (comando SQL esperado);

2) Declaração/Instrução: o gatilho será invocado apenas uma única vez quando o evento (comando SQL esperado) ocorrer, independente de quantas linhas forem afetadas.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Definindo Gatilhos

- O PostgreSQL implementa o padrão SQL, porém com alguns recursos específicos:

- Gatilhos para o evento TRUNCATE.
- Gatilhos de Declaração/Instrução em VIEWs.
- Requer uma função como a ação do Gatilho, enquanto o padrão SQL permite que você use qualquer comandos SQL.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Criando um Gatilho

- Para criar um gatilho no PostgreSQL devemos:

- Crie uma função de gatilho usando a instrução CREATE FUNCTION.
- Vincule esta função a uma tabela usando a instrução CREATE TRIGGER.
- Uma função de gatilho é semelhante a uma função comum, exceto que não tem argumentos e tem o tipo de retorno de TRIGGER como segue:

```
CREATE FUNCTION nome_function() RETURN trigger AS
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Criando um Gatilho

- Para criar um gatilho no PostgreSQL devemos:

- Em seguida, use a instrução **CREATE TRIGGER** para vincular o evento à função a ser executada.

- Sintaxe:

```
CREATE TRIGGER trigger_name {BEFORE | AFTER | INSTEAD OF} {event [OR ...]}
ON table_name
[FOR EACH {ROW | STATEMENT}]
EXECUTE PROCEDURE nome_function
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Definindo Gatilhos

- Um gatilho pode ser definido para ser chamado antes ou depois da alteração dos dados em uma tabela pela comandos de: **INSERT**, **UPDATE** ou **DELETE**.

BEFORE	**INSERT**	- ativado antes dos dados serem inseridos na tabela.
AFTER	**INSERT**	- ativado após a inserção dos dados na tabela.
BEFORE	**UPDATE**	- ativado antes da atualização dos dados na tabela.
AFTER	**UPDATE**	- ativado após a atualização dos dados na tabela.
BEFORE	**DELETE**	- ativado antes da exclusão dos dados da tabela.
AFTER	**DELETE**	- ativado após a exclusão dos dados da tabela.

- O **INSTEAD OF** é usado somente para **INSERT**, **UPDATE** ou **DELETE** nas visualizações.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Exemplo de um Gatilho

1) Vamos criar uma tabela Autor_Auditoria como segue:

```
CREATE TABLE autor_auditoria (
        idautor SERIAL,
        autor_id INTEGER NOT NULL,
        nome_autor VARCHAR(60) NOT NULL,
        data_exclusao TIME NOT NULL
);
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Exemplo de um Gatilho

2) Vamos criar uma função que copie os dados do autor para a tabela de auditoria quando ocorrer a exclusão:

```
CREATE OR REPLACE FUNCTION auditoria_exclusao_autor()
RETURNS trigger AS $$
BEGIN
    INSERT INTO autor_auditoria(autor_id, nome_autor,data_exclusao)
    VALUES( OLD.idautor, OLD.nome_autor,now() );
    RETURN OLD;
END;$$
LANGUAGE plpgsql;
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Exemplo de um Gatilho

3) Vamos criar o gatilho vinculado a finção auditoria_exclusao_autor():

```
CREATE TRIGGER exclusao_autor
BEFORE DELETE
ON autor
FOR EACH ROW
EXECUTE PROCEDURE auditoria_exclusao_autor();
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

Alterando um Gatilho

```
ALTER TRIGGER exclusao_autor
ON autor
RENAME TO delete_autor
```

Desabilitando um Gatilho

```
ALTER TABLE autor DISABLE TRIGGER delete_autor

ALTER TABLE autor DISABLE TRIGGER ALL
```

Removendo um Gatilho

```
DROP TRIGGER [IF EXISTS] delete_autor ON autor;
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

As palavras reservadas **OLD** e **NEW** são muito importantes e bastante utilizadas dentro de uma TRIGGER.

OLD - refere-se ao registro existente antes de alterar os dados

NEW - refere-se aos dados do novo registro, ou seja, após a alteração dos dados

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

EXERCÍCIO: Construir uma tabela de Auditoria que irá conter todas as ações de **UPDATE** que ocorrerem na tabela **MUSICA**.

1 PASSO – Criar a tabela de auditoria

```
CREATE TABLE musica_auditoria (
    id              SERIAL,
    idMusica        INTEGER NOT NULL,
    oldNomeMusica   VARCHAR(60) DEFAULT NULL,
    newNomeMusica   VARCHAR(60) DEFAULT NULL,
    oldDuracao      TIME  DEFAULT NULL,
    newDuracao      TIME  DEFAULT NULL,
    dataevento      TIMESTAMP DEFAULT NULL,
    acao            VARCHAR(50) DEFAULT NULL,
    PRIMARY KEY (id)  );
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

2 PASSO – Construir a função que copie os dados originais e novos da música para a tabela de auditoria assim que ocorrer a alteração destes dados:

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

2 PASSO – Construir a função que copie os dados originais e novos da música para a tabela de auditoria assim que ocorrer a alteração destes dados:

```
CREATE OR REPLACE FUNCTION auditoria_musica_update()
RETURNS trigger AS $$
BEGIN
    INSERT INTO musica_auditoria(   idMusica, oldNomeMusica, newNomeMusica,
                                    oldduracao, newduracao, dataevento, acao)
    VALUES ( OLD . IdMusica, OLD.NomeMusica, NEW.NomeMusica, OLD.duracao,
                NEW.duracao, NOW(), 'update');
    RETURN NEW;
END;$$
LANGUAGE plpgsql;
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

3 PASSO – Construir o Gatilho que seja acionado antes do comando UPDATE ser executado na tabela música.

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: TRIGGERS

3 PASSO – Construir o Gatilho que seja acionado antes do comando UPDATE ser executado na tabela música.

```
CREATE TRIGGER musica_update
BEFORE UPDATE
ON musica
FOR EACH ROW
EXECUTE PROCEDURE auditoria_musica_update();
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Objetos Avançados: EXECUTANDO UMA TRIGGERS

Diferente de um procedimento, onde deve ocorrer a chamada explícita da função, uma trigger é acionada automaticamente quando ocorrer a ação esperada na tabela base.

EXEMPLO

```
SELECT * FROM musica where idmusica = 1;
SELECT * FROM musica_auditoria;

UPDATE musica
SET NomeMusica = 'Vai dar tudo certo !'
WHERE idMusica = 1;

SELECT * FROM musica_auditoria;
```

OBJETOS AVANÇADOS

SQL – AVANÇADA

- Atividade

1) Construir a função que copie os dados originais da música para a tabela de auditoria assim que ocorrer uma exclusão destes dados;

2) Construir a função que copie os novos dados da música para a tabela de auditoria assim que ocorrer a inclusão destes dados.