Integridade e triggers no PostgreSQL

Alex Sandro
alex@ifpb.edu.br

Sumário

- *Integridade*
  - *Definição e exemplos*
  - *Tipos de integridade*
  - *Vantagens e desvantagens*
- *Triggers*
  - *Sintaxe*
  - *Considerações gerais*

Integridade

- *Integridade de Dados*
  - Manutenção e a garantia da precisão e consistência de dados durante todo o ciclo de vida da informação
    - *Correção, precisão, validade*
- *Em um SGBD, o que pode causar a perda da integridade?*
  - *Quedas durante o processamento de uma transação;*
  - *Acesso concorrente ao BD;*
  - *Distribuição dos dados em diversos computadores*

Violação de Integridade (acidental)

- ***Exemplos:***
  - *Informação incorreta;*
  - *Uma transação de venda ocorre mas o operador informa a data da transação de forma incorreta;*
  - *Um zero é esquecido ao digitar o salário de um empregado;*
  - *Um novo departamento é criado, com codDep=200, sendo inserido duas vezes na tabela;*
  - *Chave estrangeira inválida (validação na própria aplicação)*

Violação de Integridade (acidental)

- ***Mais exemplos…***
  - *Cód. de um departamento é deletado, alguns empregados recebem o novo código do departamento e um empregado ficou de for a da atualização, ficando com o extinto código*
- ***Formas de assegurar a consistência dos dados de um BD***
  - *Formalizar verificações (check);*
  - *Regras de negócio (business rules);*
- ***Solução:***
  - ***Restrições de integridade (RI)***

Integridade

- ***Como garantir a integridade do BD?***
  - ***Solução 1****: inserir as restrições no código da aplicação.*
  - ***Problemas***
    - *Sobrecarrega o programador;*
    - *Susceptível a erros;*
    - *Manutenção*
  - ***Solução 2****: inserir as restrições de integridade no próprio SGBD*
    - *O que isso traria de vantagem???*

Tipos de Integridade

- ***Integridade de domínio (de coluna)***
  - *Define o domínio de um atributo;*
  - *Deve ser verificada em inserções e atualizações. (check, null, defaults,…)*
- ***Integridade de entidade***
  - *Integridade de tabela (primary key, Unique,…)*
- ***Integridade Referencial***
  - *Restrição especificada entre duas relações para manter a consistência entre tuplas das duas relações (chave estrangeira, trigger).*

Regras de Integridade

- ***Considerações gerais***
  - *Regras de integridade são expressas em uma linguagem de alto nível;*
  - ***Objetivo***
    - *detectar a violação e tomar as ações necessárias*
- ***A partir de que ponto as regras são efetivadas?***
  - *As regras são utilizadas do estado atual do BD em diante*

Regras de integridade

- ***Exemplos***

  SE (saldo da conta for < 0)

  Operação não deve ser efetivada!

  ‘Violação da regra de integridade´

  FIM SE

- ***Vantagens***
  - *Validação é tratada no próprio SGBD*
    - *Um **check()**, por exemplo, resolveria o problema*
  - *Melhora o entendimento e manutenção da mesma*
  - *Regras podem ser atualizadas com o sistema em funcionamento*

Triggers

- ***Definição genérica***
  - *Conjunto de **instruções SQL** disparadas automaticamente quando um comando do tipo **INSERT**, **DELETE** ou **UPDATE** é executado em uma tabela.*
  - *É uma regra do tipo E-C-A (Evento-Condição-Ação)*
- ***Exemplo:***

  Ao invés de restringir contas com saldos negativos, podemos ativar uma ação que automaticamente inicia um empréstimo para uma determinada conta.
  - ***Evento? Condição? Ação?***

Triggers

- ***Nível de aplicação***
  - *São usados para reforçar **restrições de integridade** e consistência que não podem ser tratadas pelos recursos mais simples*
    - *defaults, checks, not null, etc.*
- ***Uso***
  - *Logar modificações feitas em tabelas (auditoria)*
  - *garantir críticas mais complexas*
  - *gerar o valor de uma coluna,*
  - *Manter tabelas duplicadas*
  - *Desencadear inserts em outras tabelas*

Triggers

- ***Composição de um trigger***
  - *Um **nome**: único para cada banco de dados*
  - *A **ação**: um comando INSERT, UPDATE ou DELETE*
  - *As **instruções**: Um bloco de comandos SQL*
- ***Vínculo de um trigger***
  - *Os gatilhos (triggers) são sempre vinculados a uma determinada tabela*
    - *Quando uma tabela é removida, todos os gatilhos relacionados serão excluídos automaticamente.*

Triggers

- ***Observações importantes***
  1. *Triggers não podem ser criadas para **Visões** ou **tabelas temporárias***
  2. *Triggers **não possuem parâmetros** e não podem ser explicitamente invocados.*
     - *São disparados quando os dados da **"tabela protegida"** são modificados*
  3. *Triggers são considerados como parte de uma transação*
     - *Se houver falha no seu funcionamento, os comandos serão revertidos (ROLLBACK)*

Triggers

- ***Observações importantes (cont.)***

4. *No caso de uma **falha detectada** dentro do trigger, deve-se lançar uma exceção ou instrução **Rollback**.*
5. *Um trigger pode executar **comandos** contidos em seu corpo ou acionar **stored's procedure** e outros **triggers** indiretamente*
6. ***Momento de um trigger***
   - *Indica-se na sua criação o momento de disparo (**AFTER ou BEFORE**) do bloco de comandos que será executado automaticamente ao invocar uma **modificação**.*

Triggers

- ***Trigger x Performance***
  - *Afeta a performance*
  - *Triggers aninhados podem consumir uma boa quantidade de recursos do sistema*
  - *Triggers são reativos.*
    - *Constraints não fazem modificações*

Triggers no PostgreSQL

- ***Introdução***
  - *Uma **função** especial definida pelo usuário, invocada automaticamente sempre que um evento INSERT, UPDATE, DELETE ou **TRUNCATE** acontece em uma tabela*
- ***Níveis de Trigger***
  - **`row`***: se uma instrução afeta 10 linhas, o trigger será invocada 10 vezes*
  - **`instrução`***: se uma instrução afeta 10 linhas, o trigger será invocado uma única vez*

Triggers no PostgreSQL

- ***Criação de um Trigger***
  - *PostgreSQL requer a definição de uma **função** que será responsável pela ação do trigger*
    - *O padrão SQL permite que você defina os comandos SQL diretamente no trigger.*
  - ***Requisitos**:*
    - *Criar a função de trigger usando* `CREATE FUNCTION`
    - *Ligar a função de trigger a uma tabela usando* `CREATE TRIGGER`
  - *Considerações*
    - *A função de trigger é similar a uma função comum, exceto que não recebe argumentos e seu tipo de retorno é TRIGGER*

Triggers no PostgreSQL

- ***Sintaxe: Trigger Function (tf)***

```
CREATE FUNCTION <nome_função>
RETURNS TRIGGER
AS $$
   -- corpo da função
   RETURN NEW; -- Ou RETURN OLD
$$;
```

  - *A **tf** recebe dados inerentes ao seu ambiente de chamada por meio de uma estrutura especial denominada **TriggerData**, as quais contém um conjunto de variáveis locais (OLD, NEW,…)*
  - *Uma vez definida a **tf**, esta pode ser ligada a uma ação específica na tabela.*

Triggers no PostgreSQL

- ***Variáveis diponíveis dentro da função***
  - **`NEW`***: tipo RECORD, variável que mantém a nova linha da tabela para operações INSERT/UPDATE em gatilhos de nível de linha*
    - *Assume NULL para trigger em nível de instrução ou operações **DELETE***
  - **`OLD`***: tipo RECORD, variável que mantém a linha antiga da tabela para operações DELETE/UPDATE em gatilhos de nível de linha*
    - *Assume NULL para trigger em nível de instrução ou operações **INSERT***

Triggers no PostgreSQL

- ***Variáveis diponíveis dentro da função***
  - *Quando um comando **INSERT** é executado, o novo registro é copiado para a variável **NEW**. A variável **OLD** assume **NULL***
  - *Quando um comando **UPDATE** é executado, o **registro original** (antigo) fica acessível através da variável **OLD** e o **registro modificado** (atual) pode ser acessado através da variável **NEW***
  - *Quando o comando **DELETE** é executado, o registro excluído é armazenado na variável **OLD**. A variável **NEW** assume **NULL***

Triggers no PostgreSQL

- ***Variáveis diponíveis dentro da função***
  - **`TG_OP`**: *tipo TEXT, sinaliza o tipo de evento ocorrido na tabela que disparou o trigger*
  - **`TG_LEVEL`**: *tipo TEXT, assume os valores ROW ou STATEMENT dependendo da definição do trigger.*
  - **`TG_WHEN`**: *tipo TEXT, assume os valores BEFORE ou AFTER de acordo com a definição do trigger.*

Triggers no PostgreSQL

- ***Exemplo: Trigger de Auditoria***
  - *Criando as tabelas*

```
CREATE TABLE empregado(
  nome varchar(45) NOT NULL,
  salario numeric(6,2)
);
CREATE TABLE audiEmpregado(
  operacao char(1) NOT NULL,
  momento timestamp NOT NULL,
  idUsuario text NOT NULL,
  nomeEmp varchar(45) NOT NULL,
  salario numeric(6,2)
);
```

Triggers no PostgreSQL

- ***Exemplo: Função de Trigger de Auditoria***

```
CREATE OR REPLACE FUNCTION auditEmp()
RETURNS TRIGGER AS $$
BEGIN
  IF (TG_OP='DELETE') THEN
     INSERT INTO audiEmpregado
     SELECT 'D',now(),user, OLD.nome, OLD.salario;
     RETURN OLD;
  ELSIF (TG_OP='INSERT') THEN
     INSERT INTO audiEmpregado
     SELECT 'I',now(),user, NEW.nome, NEW.salario;
     RETURN NEW;
  ELSIF (TG_OP='UPDATE') THEN
     INSERT INTO audiEmpregado
     SELECT 'U',now(),user, NEW.nome, NEW.salario;
     RETURN NEW;
  END IF;
END; $$
LANGUAGE plpgsql;
```

Triggers no PostgreSQL

- ***Sintaxe***
  - *Trigger*

```
CREATE TRIGGER <nome_trigger>
{BEFORE|AFTER}
{evento [OR ...]}
ON <nome_tabela>
[FOR [EACH] {ROW|STATEMENT}]
EXECUTE PROCEDURE <trigger_function(args)>;
```

  - ***Momento:*** BEFORE or AFTER (event)
  - ***Evento****:* INSERT, UPDATE, DELETE ou TRUNCATE
  - **FOR EACH ROW***:* nível de linha
  - **FOR EACH STATEMENT***:* nível de instrução

Triggers no PostgreSQL

- **Considerações**
  - *Os **momentos de disparo** são definidos para atuarem quando o **respectivo comando de modificação** for executado*
    - *Um trigger de **INSERT** não vai ser disparado quando a tabela sofre um **UPDATE***
  - *Pode haver problema na remoção/atualização em cascata*
    - *A não ser que a tabela tenha uma **Foreign Key** habilitada com **cascade delete**, ou **update delete**, e o SGBD permita. (Exemplo: SQL Server permite cascata apenas em 1 tabela)*

Trigger para múltipla ação

- ***Triggers de Múltipla Ação***
  - *Um gatilho pode ser criado em uma tabela para múltiplas ações nessa tabela (Atuação em eventos de INSERT, UPDATE, DELETE ou TRUNCATE)*
    - *Determine os eventos no commando CREATE TRIGGER, na seção correspondente.*
    - *Se um Trigger tem a ação DELETE, por exemplo, e não há uma instrução que alcance RETURN OLD, a **remoção não será efetivada**.*
    - *Independentemente se o momento é BEFORE ou AFTER, os triggers data NEW e OLD estarão disponíveis**

Triggers no PostgreSQL

- ***Exemplo: Trigger de Auditoria***

```
CREATE TRIGGER tgAuditEmpregado
AFTER INSERT OR UPDATE OR DELETE ON empregado
FOR EACH ROW
EXECUTE PROCEDURE auditEmp();
```

```
INSERT INTO empregado VALUES ('Alex',3000.00);
INSERT INTO empregado VALUES ('Luiz',3250.00);
INSERT INTO empregado VALUES ('Damires',2800.00);
UPDATE empregado SET salario=4000.00
WHERE nome='Alex';
DELETE FROM empregado WHERE nome='Damires';
```

Data Output | Explain | Messages | Notifications

	operacao character (1)	momento timestamp without time zone	idusuario text	nomeemp character varying (45)	salario numeric (6,2)
1	I	2019-03-22 11:24:32.366519	postgres	Alex	3000.00
2	I	2019-03-22 11:24:32.366519	postgres	Luiz	3250.00
3	U	2019-03-22 11:25:55.581855	postgres	Alex	4000.00
4	I	2019-03-22 11:24:32.366519	postgres	Damires	2800.00
5	D	2019-03-22 11:29:08.677961	postgres	Damires	2800.00

Triggers no PostgreSQL

- ***Recursividade***
  - *Triggers disparados em uma tabela, podem disparar triggers em outra tabela*
    - *Todos os triggers são considerados como uma única transação*
    - *No SQLServer, por exemplo, pode haver até 16 níveis de recursividade*
- ***Comandos não aceitáveis em um trigger***
  - *Todos os CREATE'S (database, table, view, procedure, index, etc.)*
  - *Todos os comandos DROP*
  - *Alter Table e Alter Database*

Triggers no PostgreSQL

- ***Alterando o nome de um trigger***

```
ALTER TRIGGER <nome_trigger> ON <tabela>
RENAME TO <novo_nome>;
```

- ***Desabilitando a atuação de um trigger***

```
ALTER TABLE <tabela>
DISABLE TRIGGER <nome_trigger> | ALL;
```

  - *O ALL desabilita todos os triggers relacionados à tabela*

- ***Removendo um trigger***

```
DROP TRIGGER <nome_trigger>
ON <tabela>;
```

Vamos aos Exercícios de Fixação!

Lista08 e Lista09: Triggers.pdf

Referências Bibliográficas

- *Introduction to PostgreSQL Trigger.* http://www.postgresqltutorial.com/introduction-postgresql-trigger/
- *Creating the first trigger in PostgreSQL.* http://www.postgresqltutorial.com/creating-first-trigger-postgresql/
- *Trigger Procedures. (PostgreSQL Documentation)* https://www.postgresql.org/docs/9.2/plpgsql-trigger.html