Banco de Dados

AULA 12 – TRIGGERS

Introdução

- Lógica de processamento procedural, armazenada no SGBD e disparada automaticamente pelo servidor sob condições específicas.
- Aplicações ou usuários não ativam as triggers. Elas são disparadas por atividades realizadas sobre o SGBD, sob forma de um gatilho.

Introdução

- Com a utilização de triggers é possível definir regras mais complexas do que instruções do tipo CHECK.
- Triggers representam regras do mundo (negócio) que definem a integridade ou consistência do Banco de Dados (BD).
- O processo está associado com operações de INSERT, UPDATE ou DELETE.
- São disparadas após o comando ter sido completado, mas fazem parte da transação corrente.

Vantagens de uso

- Sempre é disparada quando o evento ocorre, evitando esquecimentos ou falta de conhecimento sobre o banco de dados.
- É administrada de forma centralizada. O DBA define suas restrições (situação ou condição), eventos (momento específico da situação) e ações (procedimentos que serão executados).
- Ativação central, a execução da trigger é realizada no servidor, independente do aplicativo executado pelo cliente.

Aplicabilidade

- Regras de negócio: Garantia de integridade e consistência.
- Aplicação lógica: Outro procedimento que executa instruções de consulta, verificação, etc.
- Segurança: Verificação e teste de regras de segurança mais complexas.
- Replicação de dados: Dados em tabelas que são compartilhadas podem ser gerados automaticamente.

Implementação

```
CREATE TRIGGER name
{ BEFORE | AFTER } { event [ OR ... ] } ON
table [ FOR [ EACH ] { ROW | STATEMENT } ]
EXECUTE PROCEDURE funcname ( arguments );
```

Implementação

- **`funcname:`** é o nome de uma função desenvolvida em uma linguagem suportada.
- **`event:`** é a operação que pode ser INSERT, UPDATE ou DELETE.
- **`table:`** diz respeito à tabela a qual a trigger está associada.
- Uma função será executada sempre que ocorrer um determinado evento, em uma determinada tabela, de acordo com a definição.

Exemplo 1

Ao ser alterado ou excluído um filme, será excluído o estúdio ao qual o filme está associado.
A trigger é executada enquanto a transação ainda não foi confirmada.

```
CREATE TRIGGER estudio_sem_ref
AFTER DELETE OR UPDATE ON filmes
FOR EACH ROW
EXECUTE PROCEDURE apaga_estudio_sem_ref ();
```

Transação

- É uma unidade de programa que executa uma ou mais atualizações no banco de dados.
- Diversas propriedades devem ser respeitadas para que a transação não viole a integridade dos dados armazenados.
- Cada instrução SQL é considerada uma transação.
- Se a instrução modificar diversos registros, todos serão modificados ou nenhum.
- Instruções BEGIN, COMMIT, ROLLBACK permitem definir início de transação, confirmação ou retorno a uma situação anterior.

Propriedades (ACID)

- Atomicidade: Todas as atualizações feitas por uma transação são efetivadas no BD ou nenhuma delas.
- Consistência: Uma transação normal deve preservar a consistência do banco de dados.
- Isolamento: Eventos dentro de uma transação devem ser transparentes para outras transações executando concorrentemente (sincronização de transações).
- Durabilidade: sempre que uma transação é executada com sucesso, o SGBD deve garantir que o seu resultado sobreviva a qualquer falha subsequente.

Exemplo 2

Atualização em contas correntes.
Em alguns casos é possível usar pontos de salvamento e retornar até o ponto em que a transação havia funcionado.

```
BEGIN TRANSACTION;

UPDATE contas
SET saldo = saldo - 100.00
WHERE codigo = 5;

SAVEPOINT meu_ponto_de_salvamento;

UPDATE contas
SET saldo = saldo + 100.00
WHERE codigo = 5;

ROLLBACK TO SAVEPOINT meu_ponto_de_salvamento;

UPDATE contas
SET saldo = saldo + 100.00
WHERE conta = 6;

COMMIT TRANSACTION;
```

Exemplo 3

Verificar se está sendo realizada uma atualização na chave primária da cidade e não permite alteração caso isso ocorrer.

```
CREATE OR REPLACE FUNCTION fatuchavecid ()
     RETURNS TRIGGER
     AS $$
     BEGIN
          IF (NEW.CodCid <> OLD.Codcid) THEN
               RAISE NOTICE 'Não é permitido atualizar o
                             campo CodCid';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     END;
     $$ LANGUAGE plpgsql;

CREATE TRIGGER tatuchavecid
BEFORE UPDATE ON Cidade
FOR EACH ROW
EXECUTE PROCEDURE fatuchavecid ();

UPDATE Cidade SET CodCid=115 WHERE CodCid=15;
```

Exemplo 4

Verificar se uma participação está sendo incluída para uma pessoa com a função coordenador e impede que a pessoa seja alguém vinculado a um cliente.

```
CREATE OR REPLACE FUNCTION fvalidapartic ()
    RETURNS TRIGGER
    AS $$
    DECLARE
        vcodclipess INTEGER;
    BEGIN
        SELECT codcli INTO vcodclipess
        FROM pessoa
        WHERE codpess=NEW.codpess;
        IF ((NEW.codfunc=4) AND (vcodclipess IS NOT NULL)) THEN
            RAISE NOTICE 'Pessoas de clientes não podem ser
                                    coordenadores';
            RETURN NULL;
        ELSE
            RETURN NEW;
        END IF;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER Tvalidapartic
BEFORE INSERT ON participanteativ
FOR EACH ROW
EXECUTE PROCEDURE fvalidapartic ();
```

Exemplo 5

Criar uma função que será chamada na trigger para controlar o custo estimado de um projeto, baseado no valor das atividades.

```
CREATE OR REPLACE FUNCTION fAtuCustoEstProj ()
    RETURNS TRIGGER
    AS $$
    BEGIN
        IF TG_OP='DELETE' OR TG_OP='UPDATE' THEN
            UPDATE projeto
            SET custoest=custoest-OLD.custoest
            WHERE id=OLD.idproj;
        END IF;
        IF TG_OP='INSERT' OR TG_OP='UPDATE' THEN
            UPDATE projeto
            SET custoest=custoest+NEW.custoest
            WHERE id=NEW.idproj;
        END IF;
        RETURN NULL;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER tAtuCustoEstProj
AFTER DELETE OR UPDATE OR INSERT ON atividade
FOR EACH ROW
EXECUTE PROCEDURE fAtuCustoEstProj ();
```

Leitura recomendada

- SILBERSCHATZ, Abraham. Sistemas de bancos de dados. 3. ed. São Paulo: Makron Books, 1999. Páginas 109 a 113.

Exercícios

Ver Lista 11