**Trilha Banco de Dados**

Curso: **ADS – Análise e Desenvolvimento de Sistemas**

Ano atualização: **2024/2**

Nome da Trilha: **OT06 – TRIGGERS**

**1. O que são TRIGGERS?**

*Triggers* (ou gatilhos) são mecanismos utilizados em sistemas de gerenciamento de banco de dados (SGBDs) para executar automaticamente uma ação em resposta a um evento específico que ocorre em uma tabela ou visão. Esses eventos podem ser, por exemplo, uma inserção, atualização ou exclusão de dados. As *triggers* são frequentemente usadas para manter a integridade dos dados, garantir a consistência de registros, realizar auditorias ou executar cálculos automáticos quando uma determinada condição é atendida.

Existem dois principais tipos de *triggers*:

*Before Trigger:* Executa a ação antes que o evento (inserção, atualização ou exclusão) seja realizado na tabela.

*After Trigger*: Executa a ação depois que o evento for completado com sucesso.

**2. Explique quando usar e quando se deve aplicar as *TRIGGERS***

As *triggers* devem ser aplicadas quando há a necessidade de garantir que certas ações sejam executadas automaticamente em resposta a operações no banco de dados. Exemplos de casos em que triggers são úteis incluem:

* Manutenção de integridade referencial: Por exemplo, garantir que uma exclusão em uma tabela principal automaticamente remova ou altere registros em tabelas relacionadas (cascade delete).
* Auditoria de alterações: Quando for necessário rastrear modificações feitas em dados, como a criação de um log de alterações.
* Validação de dados: Certificar-se de que valores inseridos ou atualizados atendem a regras de negócios específicas antes de serem gravados na tabela.
* Cálculos automáticos: Automatizar cálculos ou atualizações de valores em campos derivados, como atualização de totais em uma tabela de pedidos.

Porém, deve-se evitar o uso excessivo de *triggers*, pois elas podem impactar negativamente o desempenho do banco de dados, especialmente em sistemas com grandes volumes de dados ou operações frequentes. Além disso, o uso de *triggers* complexas pode tornar a manutenção e depuração do banco de dados mais difíceis.

**3. Explique sobre a instrução *ROLLBACK TRANSACTION***

A instrução *ROLLBACK TRANSACTION* é utilizada para reverter todas as modificações feitas em uma transação de banco de dados. Quando uma transação é iniciada (geralmente com o comando *BEGIN TRANSACTION*), todas as operações realizadas, como inserções, atualizações e exclusões, são temporárias até que uma instrução de *commit* (*COMMIT*) seja executada para tornar essas alterações permanentes. Caso ocorra algum erro ou condição que requeira reverter as alterações, a instrução *ROLLBACK* é usada para desfazer as modificações feitas na transação e restaurar o banco de dados ao estado anterior.

A sintaxe básica de ROLLBACK é:

ROLLBACK TRANSACTION;

**4. Como criar uma *Trigger*, estrutura:**

A criação de uma *trigger* envolve especificar o evento que irá acionar a execução da *trigger*, o tipo de *trigger* (antes ou depois do evento), a tabela envolvida e a ação a ser realizada. A estrutura básica de uma *trigger* é a seguinte:

CREATE TRIGGER nome\_da\_trigger

{BEFORE | AFTER} {INSERT | UPDATE | DELETE}

ON nome\_da\_tabela

FOR EACH ROW

BEGIN

-- Ações a serem executadas

END;

Abaixo está uma legenda das instruções utilizadas no exemplo acima:

**nome\_da\_trigger:** o nome da *trigger*.

**{BEFORE | AFTER}:** define se a *trigger* será executada antes ou depois do evento.

**{INSERT | UPDATE | DELETE}:** especifica qual tipo de operação irá acionar a *trigger*.

**nome\_da\_tabela:** a tabela onde o evento ocorre.

**FOR EACH ROW:** define que a trigger será executada para cada linha afetada pelo evento (em caso de INSERT, UPDATE ou DELETE em várias linhas).

**BEGIN...END:** define o bloco de código a ser executado quando a *trigger* for acionada.

Exemplo de *trigger* para auditar alterações em uma tabela de funcionários:

CREATE TRIGGER auditoria\_alteracao\_funcionarios

AFTER UPDATE ON funcionarios

FOR EACH ROW

BEGIN

INSERT INTO log\_alteracoes (funcionario\_id, data\_alteracao, campo\_alterado)

VALUES (OLD.id, NOW(), 'Salário');

END;

Neste exemplo, sempre que um salário for atualizado na tabela funcionarios, a *trigger* irá adicionar um registro na tabela log\_alteracoes com a ID do funcionário, a data da alteração e o campo alterado.

**5. Quais as principais instruções para usar as *Triggers*?**

As principais instruções e funções utilizadas dentro de uma trigger são:

* **INSERT:** Para inserir dados em uma tabela. Usada dentro de uma trigger para registrar informações em uma tabela de log ou realizar outras inserções baseadas nas alterações.

Exemplo:

INSERT INTO log\_de\_auditoria (usuario, operacao, data)

VALUES ('admin', 'UPDATE', NOW());

* **UPDATE:** Para modificar dados em uma tabela. Pode ser usada em triggers AFTER para realizar ajustes ou cálculos adicionais quando um dado for alterado.

Exemplo:

UPDATE produtos SET estoque = estoque - 1 WHERE id = NEW.id;

* **DELETE**: Para excluir dados de uma tabela. Pode ser usada para remover registros relacionados ou registrar a exclusão em uma tabela de log.

Exemplo:

DELETE FROM log\_de\_auditoria WHERE data < NOW() - INTERVAL 1 YEAR;

* **NEW e OLD**: Referências especiais usadas para acessar os valores das colunas antes e depois de uma modificação em uma trigger de tipo UPDATE:

NEW: Refere-se aos valores após a atualização ou inserção.

OLD: Refere-se aos valores antes da atualização ou exclusão.

Exemplo de *trigger* de atualização:

CREATE TRIGGER verifica\_salario

BEFORE UPDATE ON funcionarios

FOR EACH ROW

BEGIN

IF NEW.salario < OLD.salario THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'O salário não pode ser reduzido';

END IF;

END;

* **SIGNAL:** Usada para gerar uma exceção, interrompendo a execução da transação quando uma condição especificada é atendida.

Essas instruções são fundamentais para a manipulação de dados dentro de triggers e para a implementação de lógica personalizada no banco de dados.

ATIVIDADES

1. **Crie uma trigger para que quando o salário de um vendedor for alterado, ele não possa ser menor do que o já cadastrado no sistema.**

CREATE TRIGGER salario\_mudanca\_vendedor

ON vendedor

FOR EACH ROW

BEGIN

UPDATE vendedor SET salario = NEW.salario WHERE NEW.salario>OLD.salario;

END

1. **Crie uma trigger para que quando seja inserido um novo vendedor no sistema, ele não possa ser inserido com salário inicial menor que R$ 1200,00.**

DELIMITER $$

CREATE TRIGGER vendedor\_insert\_trigger

BEFORE INSERT ON vendedor

FOR EACH ROW

BEGIN

IF

NEW.salario<1200.00 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'O salário inicial não pode ser menor que R$ 1.200,00';

END IF;

END$$

DELIMITER ;

1. **Crie uma trigger para que quando um vendedor for deletado do quadro de vendedores ativos, esses dados sejam inseridos em uma nova tabela no banco chamada Vendedores\_desligados.**

DELIMITER $$

CREATE TRIGGER vendedores\_after\_delete

AFTER DELETE ON vendedor

FOR EACH ROW

BEGIN

INSERT INTO vendedores\_desligados (idvendedor,nome,salario,data\_desligamento) VALUES (OLD.idvendedor,OLD.nome,OLD.salario,NOW());

END$$

DELIMITER ;