SQL Triggers - Aula 10

Gabriel de Paula Gaspar Pinto

Exercício 1

a) Gatilhos (triggers) são um mecanismo que permite executar automaticamente um bloco de comandos SQL quando uma determinada ação ocorre sobre uma tabela, como a inserção, atualização ou exclusão de dados.

b) As instruções que podem disparar um gatilho são o INSERT, UPDATE e DELETE. Essas ações podem ser associadas a um gatilho configurado para ser executado antes (BEFORE) ou depois (AFTER) da operação ser realizada.

c) Durante a execução de um gatilho, o SGDB disponibiliza duas tabelas temporárias, o OLD e o NEW. As duas armazenam os valores das linhas afetadas. A tabela OLD representa os dados antes da modificação, enquanto a tabela NEW representa os dados após a modificação.

No caso do INSERT, apenas a tabela NEW é disponibilizada, já que a linha inserida não existia antes. Em um DELETE, somente a tabela OLD estará disponível, já que a linha será removida e não haverá novos dados. Por fim, em um UPDATE, tanto quando a tabela NEW e a tabela OLD estão disponíveis, já que uma linha não será removida ou inserida, mas somente editada, permitindo que os valores anteriores e os novos sejam comparados.

Exercício 2

tabela1

a1
1
3
1
7
1
8
4
4

tabela2

a2
1
3
1
7
1
8
4
4

tabela3

a3
2
5
6
9
10

tabela4

a4	b4
1	3
2	0
3	1
4	2
5	0
6	0
7	1
8	1
9	0
10	0

Exercício 3

```
DELIMITER $$

CREATE TRIGGER atualizar_categoria
BEFORE UPDATE ON cliente
FOR EACH ROW
BEGIN
IF NEW.total_gasto < 1000 THEN
    SET NEW.categoria = 'bronze';
ELSEIF NEW.total_gasto < 5000 THEN
    SET NEW.categoria = 'prata';
ELSE
    SET NEW.categoria = 'ouro';
END IF;
END $$

DELIMITER ;
```

Exercício 4

a)
```
DELIMITER $$

CREATE TRIGGER reduzir_quantidade_disponivel
AFTER INSERT ON emprestimo
FOR EACH ROW
BEGIN
UPDATE livro SET quantidade_disponivel = quantidade_disponivel - 1
WHERE isbn = NEW.livro_isbn;
END $$

DELIMITER ;

```

b)
```
DELIMITER $$

CREATE TRIGGER aumentar_quantidade_disponivel
AFTER UPDATE ON emprestimo
FOR EACH ROW
BEGIN
IF OLD.ativo = '1' AND NEW.ativo = '0' THEN
    UPDATE livro SET quantidade_disponivel = quantidade_disponivel + 1
    WHERE isbn = NEW.livro_isbn;
END IF;
END $$

DELIMITER ;

```

c)
```
CREATE VIEW notificacoes_disponiveis AS
SELECT
s.nome AS nome_socio,
st.numero AS telefone,
l.titulo AS livro
FROM notificacao n
JOIN socio s ON s.cpf = n.socio_cpf
JOIN socio_telefone st ON st.socio_cpf = s.cpf
JOIN livro l ON l.isbn = n.livro_isbn
WHERE l.quantidade_disponivel > 0;

```

d)
```
CREATE VIEW emprestimos_atrasados AS
SELECT s.nome AS nome_socio, l.titulo AS nome_livro, e.retirada, e.devolucao
FROM emprestimo e
JOIN socio s ON s.cpf = e.socio_cpf
JOIN livro l ON l.isbn = e.livro_isbn
WHERE e.ativo = '1' AND e.devolucao < CURDATE();

```

Exercício 5

- INSERT

```
DELIMITER $$

CREATE TRIGGER log_leilao_insert
AFTER INSERT ON leilao
FOR EACH ROW
BEGIN
INSERT INTO leilao_log (id_leilao, id_usuario, id_item, acao, hora, lance, usuario_bd)
VALUES (NEW.id_leilao, NEW.id_usuario, NEW.id_item, 'INSERT', NOW(), NEW.lance,
CURRENT_USER());
END$$

DELIMITER ;


```

- UPDATE

```
        DELIMITER $$

        CREATE TRIGGER log_leilao_update
        AFTER UPDATE ON leilao
        FOR EACH ROW
        BEGIN
        INSERT INTO leilao_log (id_leilao, id_usuario, id_item, acao, hora, lance, usuario_bd)
        VALUES (NEW.id_leilao, NEW.id_usuario, NEW.id_item, 'UPDATE', NOW(), NEW.lance,
    CURRENT_USER());
        END$$

        DELIMITER ;

```

- DELETE

```
        DELIMITER $$

        CREATE TRIGGER log_leilao_delete
        AFTER DELETE ON leilao
        FOR EACH ROW
        BEGIN
        INSERT INTO leilao_log (id_leilao, id_usuario, id_item, acao, hora, lance, usuario_bd)
        VALUES (OLD.id_leilao, OLD.id_usuario, OLD.id_item, 'DELETE', NOW(), OLD.lance,
    CURRENT_USER());
        END$$

        DELIMITER ;

```