Banco de Dados

anthonyferreiralamarca@gmail.com

Assertion

- Especifica tipos adicionais de restrições que estão fora do escopo das restrições embutidas no modelo relacional
  - Integridade de entidade
  - Chave única
  - Integridade referencial

- O que é uma restrição mesmo?

Assertion

- Comando DDL
  - Create Assertion
- Recebe um nome de restrição
- Especificada por uma condição semelhante à cláusula WHERE de uma consulta SQL
- Exemplo
  - Criar uma restrição de que o salário de um funcionário não pode ser maior que o salário do gerente do departamento para o qual o funcionário trabalha

Assertion

- CREATE ASSERTION RESTRICAO_SALARIAL

  CHECK (NOT EXISTS (SELECT *

  FROM FUNCIONARIO F, FUNCIONARIO G, DEPARTAMENTO D

  WHERE F.SALARIO >G.SALARIO AND F.DNR = D.DNUMERO AND D.CPF_GERENTE = G.CPF));

Assertion

- O SGBD é responsável por garantir que a nova restrição definida não seja violada
- A clausula WHERE pode ser utilizada
- Mas geralmente utiliza-se EXISTS e NOT EXISTS para especificá-la
- Se alguma tupla no BD fizer a condição de um comando ASSERTION seja avaliada como FALSE
- A restrição é violada
- A restrição só será satisfeita se nenhuma combinação de tuplas do BD violar a restrição

Assertion

- Para cria-las, basta especificar uma consulta que seleciona quaisquer tuplas que violam a condição desejada
- Ao usar NOT EXISTS, o resultado da consulta deve ser vazio para satisfazer a condição
- Como o exemplo anterior

Assertion

- Deve-se utilizar CREATE ASSERTION, em casos em que as restrições de domínio não será suficiente para especificar a condição da restrição
- Pois as restrições de domínio são implementadas de modo mais eficiente pelo SGBD
- Devido ao alto custo de testar e mante-las, desenvolvedores omitem suporte a elas

Trigger

- Especifica ações automáticas que o SBD realizará quando certos eventos e condições ocorrem
- É um procedimento que está associado a uma tabela
- Exemplo
  - Um gerente quer ser informado quando as despesas de viagem de um funcionário excederem certo limite, recebendo uma mensagem sempre que isto ocorrer

Trigger

- Eventos
  - INSERT
  - UPDATE
  - DELETE
- Pode-se definir vários TRIGGERS em uma base de dados
  - Para cada evento, pode-se definir apenas um TRIGGERS

Trigger

- Sintaxe:

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    TRIGGER trigger_name trigger_time trigger_event
    ON tbl_name FOR EACH ROW trigger_stmt
```

- DEFINER: Cláusula opcional para definir o usuário para criar o gatilho;
- trigger_name: define o nome do gatilho;
- trigger_time: define se o TRIGGER será ativado antes (BEFORE) ou depois (AFTER) do comando que o disparou;
- trigger_event: aqui se define qual será o evento, INSERT, REPLACE, DELETE ou UPDATE;
- tbl_name: nome da tabela cujos eventos podem disparar o gatilho;
- trigger_stmt: as definições do que o TRIGGER deverá fazer quando for disparado.

Trigger

- Operadores OLD e NEW
- Refere-se as tuplas responsáveis pelos eventos que dispararam o gatilho
- Através desses operadores é possível acessar os valores das tuplas envolvidas
- O comportamento de OLD e NEW depende do evento disparado

Trigger

- Operadores OLD e NEW
- INSERT
  - O operador NEW.nome_coluna permite verificar o valor enviado a ser inserido em uma coluna de uma tabela. OLD.nome_coluna não está disponível
- DELETE
  - O operador OLD.nome_coluna permite verificar o valor excluído ou a ser excluído. NEW.nome_coluna não está disponível
- UPDATE
  - Tanto o OLD.nome_coluna quanto o NEW.nome_coluna estão disponíveis

Trigger - Restrições

- Não pode-se chamar diretamente uma TRIGGER
- Não é permitido iniciar e finalizar transações em meio à TRIGGERS
- Não se pode criar um TRIGGER para uma tabela temporária ou para uma visão

Trigger

- CREATE TABLE `produtos` (
  `COD_PRODUTO` int(3) NOT NULL,
  `DESCRICAO` varchar(100) DEFAULT NULL,
  `ESTOQUE` int(5) DEFAULT '0',
  PRIMARY KEY (`COD_PRODUTO`));

Trigger

- CREATE TABLE `pedido` (
  `COD_VENDA` int(3) PRIMARY KEY,
  `DATA_VENDA` date DEFAULT NULL,
  `QUANTIDADE` int(3) DEFAULT NULL,
  `PRODUTO` int(3) DEFAULT NULL,
  CONSTRAINT `PRODUTO_FK` FOREIGN KEY (`PRODUTO`) REFERENCES `produtos` (`COD_PRODUTO`));

Trigger

- Sempre que um cliente compre um produto, o estoque deste determinado produto deve ser atualizado
- Sempre que um cliente desiste da compra feita, devolver a quantidade daquele produto ao estoque
- Caso o cliente altere o pedido, requerendo mais ou menos do produto, atualize o estoque

Trigger

- DROP TRIGGER IF EXISTS ATUALIZA_ESTOQUE_ VENDA_AI;
- CREATE TRIGGER ATUALIZA_ESTOQUE_VENDA_AI

```
AFTER INSERT ON PEDIDO FOR EACH ROW
BEGIN
        UPDATE PRODUTOS P
        SET ESTOQUE = ESTOQUE - NEW.QUANTIDADE
        WHERE P.COD_PRODUTO = NEW.PRODUTO;
END
```

Trigger

- DROP TRIGGER IF EXISTS ATUALIZA_ESTOQUE_DEV_AD;
- CREATE TRIGGER ATUALIZA_ESTOQUE_DEV_AD

```
AFTER DELETE ON PEDIDO FOR EACH ROW
BEGIN
        UPDATE PRODUTOS P
        SET ESTOQUE = ESTOQUE + OLD.QUANTIDADE
        WHERE P.COD_PRODUTO = OLD.PRODUTO;
END
```

Trigger

- DROP TRIGGER IF EXISTS ATUALIZA_ESTOQUE_MOD_AU;
- CREATE TRIGGER ATUALIZA_ESTOQUE_MOD_AU

```
AFTER UPDATE ON PEDIDO FOR EACH ROW
BEGIN
        UPDATE PRODUTOS P
        SET ESTOQUE = ESTOQUE + OLD.QUANTIDADE
                - NEW.QUANTIDADE
         WHERE P.COD_PRODUTO = OLD.PRODUTO;
END
```

Exercício

- CLIENTES (cod_cli, nome, endereço, telefone)
- PRODUTOS (cod_prod, nome, preço)
- VENDAS (cod_venda, data, valor_total, cliente)
- ITENS_VENDA (cod_item, produto, venda, quantidade)
- FIDELIDADE (num_cartão, cliente, pontos)

Exercício

- Criar gatilhos para
- Quando um cliente for inserido, criar automaticamente o seu cartão de fidelidade
- Ao inserir uma venda, creditar um ponto para cada real do valor total da venda, no cartão de fidelidade do cliente
- Idem na exclusão
- Atualizar o valor total da venda de um produto, assim que sua quantidade for definida em ITENS_VENDAS
- Altere os pontos de fidelidade do cliente com o novo VALOR_TOTAL