Trabalho 4 banco de dados

Felippe 2020202291

A)  
create table TB\_Empregado (

RG\_Empregado varchar(8) not null constraint PK\_Empregado primary key,

Nome\_Empregado varchar(40),

CPF\_Empregado varchar(11),

RG\_Supervisor varchar(8) constraint Fk\_RGSupervisor Foreign key (RG\_Supervisor)

references TB\_Empregado(RG\_Empregado),

Salario\_Empregado money,

Depto\_Empregado int)  
  
B)   
CREATE TRIGGER trg\_CheckIdadeDependente

ON TB\_Dependente

INSTEAD OF INSERT

AS

BEGIN

IF EXISTS (

SELECT \*

FROM inserted

WHERE

(DATEDIFF(YEAR, Data\_Nascimento, GETDATE()) > 18 AND Sexo = 'M') OR

(DATEDIFF(YEAR, Data\_Nascimento, GETDATE()) > 21 AND Sexo = 'F')

)

BEGIN

RAISERROR('Inserção não permitida para dependentes com a idade e sexo especificados.', 16, 1);

END

ELSE

BEGIN

INSERT INTO TB\_Dependente (RG\_Responsavel, Nome\_dependente, Data\_Nascimento, Relacao, Sexo)

SELECT RG\_Responsavel, Nome\_dependente, Data\_Nascimento, Relacao, Sexo

FROM inserted;

END

END;

C)   
CREATE TRIGGER trg\_CheckDependente

ON TB\_Dependente

INSTEAD OF INSERT

AS

BEGIN

DECLARE @ano\_atual INT;

SET @ano\_atual = YEAR(GETDATE()); -- Obtém o ano atual

-- Verifica as condições para impedir a inserção

IF EXISTS (

SELECT 1

FROM inserted

WHERE

-- Calcula a idade usando apenas o ano de nascimento

(CASE

WHEN Sexo = 'M' AND @ano\_atual - YEAR(Data\_Nascimento) > 18 THEN 1

WHEN Sexo = 'F' AND @ano\_atual - YEAR(Data\_Nascimento) > 21 THEN 1

ELSE 0

END) = 1

)

BEGIN

-- Gera um erro se a condição for atendida

RAISERROR('Não é permitido inserir dependentes com mais de 18 anos do sexo masculino ou mais de 21 anos do sexo feminino.', 16, 1);

ROLLBACK TRANSACTION;

END

ELSE

BEGIN

-- Se as condições não forem atendidas, realiza a inserção normalmente

INSERT INTO TB\_Dependente (RG\_Responsavel, Nome\_dependente, Data\_Nascimento, Relacao, Sexo)

SELECT RG\_Responsavel, Nome\_dependente, Data\_Nascimento, Relacao, Sexo

FROM inserted;

END

END;

-- TESTANDO A TRIGGER

-- rejeitada, Idade > 18

INSERT INTO TB\_Dependente (RG\_Responsavel, Nome\_dependente, Data\_Nascimento, Relacao, Sexo) VALUES ('34567891', 'João Silva', '2001-01-01', 'Filho', 'M');

-- rejeitada, Idade > 21

INSERT INTO TB\_Dependente (RG\_Responsavel, Nome\_dependente, Data\_Nascimento, Relacao, Sexo) VALUES ('34567891', 'Maria Silva', '1998-01-01', 'Filha', 'F');

-- aceita, Idade <= 18

INSERT INTO TB\_Dependente (RG\_Responsavel, Nome\_dependente, Data\_Nascimento, Relacao, Sexo) VALUES ('34567891', 'Lucas Silva', '2012-01-01', 'Filho', 'M');

D)  
CREATE TRIGGER trg\_ValidarSupervisor

ON TB\_Empregado

FOR INSERT, UPDATE

AS

BEGIN

-- Verifica se o RG do Supervisor é igual ao RG do Empregado

IF EXISTS (

SELECT 1

FROM inserted i

WHERE i.RG\_Empregado = i.RG\_Supervisor

)

BEGIN

RAISERROR('Um empregado não pode ser supervisor de si mesmo.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

-- Verifica se o RG do Supervisor é válido (existe na tabela TB\_Empregado)

IF EXISTS (

SELECT i.RG\_Supervisor

FROM inserted i

LEFT JOIN TB\_Empregado e ON i.RG\_Supervisor = e.RG\_Empregado

WHERE i.RG\_Supervisor IS NOT NULL AND e.RG\_Empregado IS NULL

)

BEGIN

RAISERROR('O RG do Supervisor não é válido. O supervisor deve ser um empregado cadastrado.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

END;

--1º Teste

--Por causa da Trigger, não deve existir um caso em que o empregado seja o seu próprio Supervisor

INSERT INTO TB\_Empregado (RG\_Empregado, Nome\_Empregado, CPF\_Empregado, RG\_Supervisor, Salario\_Empregado, Depto\_Empregado)

VALUES ('99999999', 'Alonso Prado', '88888888888', '99999999', 5000.00, 1);

--2º Teste

-- Este comando deve falhar porque o RG\_Supervisor '00000000' não existe nos inserts da tabela TB\_Empregado

INSERT INTO TB\_Empregado (RG\_Empregado, Nome\_Empregado, CPF\_Empregado, RG\_Supervisor, Salario\_Empregado, Depto\_Empregado)

VALUES ('88888888', 'Luiz Pereira', '77777777777', '00000000', 4500.00, 2);

--3º Teste

-- Este comando deve ser bem-sucedido, assumindo que o RG\_Supervisor '34567891' existe na tabela

INSERT INTO TB\_Empregado (RG\_Empregado, Nome\_Empregado, CPF\_Empregado, RG\_Supervisor, Salario\_Empregado, Depto\_Empregado)

VALUES ('77777777', 'Pedro Gedro', '66666666666', '34567891', 5500.00, 3);

SELECT \* FROM TB\_Empregado;

--4º Teste

-- Este comando deve falhar por causa da trigger

UPDATE TB\_Empregado

SET RG\_Supervisor = '77777777'

WHERE RG\_Empregado = '77777777';

SELECT \* FROM TB\_Empregado;  
  
E)  
ALTER TABLE TB\_Projeto ADD hora\_total int;

CREATE TRIGGER TR\_LimiteHorasProjeto

ON TB\_Projeto

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @CidadeProjeto varchar(40);

DECLARE @TotalHoras int;

SELECT @CidadeProjeto = Cidade\_Projeto

FROM inserted;

SELECT @TotalHoras = SUM(hora\_total)

FROM TB\_Projeto

WHERE Cidade\_Projeto = @CidadeProjeto;

IF @TotalHoras > 2000

BEGIN

ROLLBACK TRANSACTION;

RAISERROR('O total de horas para projetos na cidade %s excede 2.000 horas.', 16, 1, @CidadeProjeto);

END

END;

INSERT INTO TB\_Projeto (Numero\_Projeto, Nome\_Projeto, Cidade\_Projeto, hora\_total)

VALUES (1706, 'Novo Projeto Excedente', 'Curitiba', 1000);

INSERT INTO TB\_Projeto (Numero\_Projeto, Nome\_Projeto, Cidade\_Projeto, hora\_total)

VALUES (1707, 'Projeto Teste Excedente', 'Curitiba', 1500);

SELECT \* FROM TB\_Projeto WHERE Numero\_Projeto = 1707;  
  
F)  
CREATE PROCEDURE sp\_ContarProjetosEmpregado

AS

BEGIN

SELECT

e.RG\_Empregado,

e.Nome\_Empregado,

COUNT(ep.Numero\_Projeto) AS Quantidade\_Projetos

FROM

TB\_Empregado e

LEFT JOIN

TB\_Empregado\_Projeto ep ON e.RG\_Empregado = ep.RG\_Empregado

GROUP BY

e.RG\_Empregado, e.Nome\_Empregado

ORDER BY

Quantidade\_Projetos DESC;

END;

--Execução

EXEC sp\_ContarProjetosEmpregado;  
  
G)   
CREATE PROCEDURE sp\_IncrementarSalarioPorProjeto

AS

BEGIN

WITH ProjetosPorEmpregado AS (

SELECT

ep.RG\_Empregado,

COUNT(ep.Numero\_Projeto) AS Quantidade\_Projetos

FROM

TB\_Empregado\_Projeto ep

GROUP BY

ep.RG\_Empregado

)

UPDATE e

SET e.Salario\_Empregado = e.Salario\_Empregado + (p.Quantidade\_Projetos \* 50)

FROM

TB\_Empregado e

INNER JOIN

ProjetosPorEmpregado p ON e.RG\_Empregado = p.RG\_Empregado;

END;

--Execução

EXEC sp\_IncrementarSalarioPorProjeto;

--Verificação

SELECT RG\_Empregado, Nome\_Empregado, Salario\_Empregado

1. FROM TB\_Empregado;  
     
   H)  
   CREATE TRIGGER trg\_AtualizarSalarioPorProjeto

ON TB\_Empregado\_Projeto

AFTER INSERT

AS

BEGIN

-- Atualiza o salário de cada empregado com R$50 para cada novo projeto

UPDATE e

SET e.Salario\_Empregado = e.Salario\_Empregado + 50

FROM TB\_Empregado e

JOIN inserted i ON e.RG\_Empregado = i.RG\_Empregado;

END;