UNIP
UNIVERSIDADE PAULISTA

UNICLUSTER

Linguagem de Programação de Banco de Dados

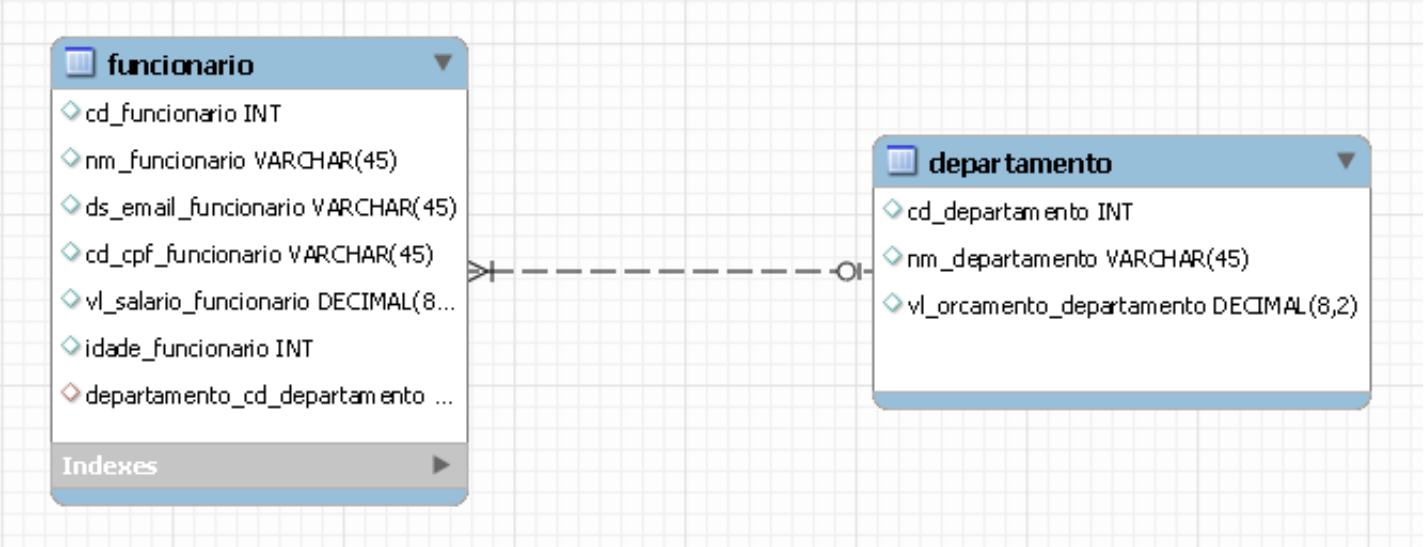
Gatilhos

Carlos & Genivaldo

UNIP – Cidade Universitária

Exercício

- Para o laboratório vamos implementar o seguinte banco de dados:



Exercício


```
CREATE TABLE departamento
(
        cd_departamento INT NOT NULL AUTO_INCREMENT,
        nm_departamento VARCHAR(45) NOT NULL,
        vl_orcamento DECIMAL(8,2),    PRIMARY KEY(cd_departamento)
);
```

Exercício


```
CREATE TABLE funcionario(
        cd_funcionario INT NOT NULL AUTO_INCREMENT,
        nm_funcionario VARCHAR(45) NOT NULL,
        ds_email_funcionario VARCHAR(45),
        cd_cpf_funcionario VARCHAR(45) UNIQUE NOT NULL,
        vl_salario_funcionario DECIMAL(8,2) NOT NULL,
        idade_funcionario INT,
        departamento_cd_departamento INT,
        PRIMARY KEY (cd_funcionario),
        FOREIGN KEY (departamento_cd_departamento) REFERENCES departamento (cd_departamento)
);
```

Exercício


```
CREATE TABLE auditoria
(
        cd_auditoria INT NOT NULL AUTO_INCREMENT,
        dt_auditoria DATE,
        usuario VARCHAR(40) NOT NULL,
        mensagem VARCHAR(100) NULL,
        PRIMARY KEY (cd_auditoria)
);
```

Exercícioo


```
INSERT INTO departamento (nm_departamento, vl_orcamento) VALUES ('Qualidade', 200000);

INSERT INTO departamento (nm_departamento, vl_orcamento) VALUES ('Processos', 300000);

INSERT INTO departamento (nm_departamento, vl_orcamento) VALUES ('Produto', 400000);

INSERT INTO departamento (nm_departamento, vl_orcamento) VALUES ('Diretoria', 500000);
```

Exercícioo

INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario, vl_salario_funcionario, idade_funcionario, cd_departamento) VALUES ('Mario', 'mario.quinello@docente.unip.br', 11111111111, '20000', 44, 1),

INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario, vl_salario_funcionario,idade_funcionario, departamento_cd_departamento) VALUES ('Mario', 'mario.quinello@docente.unip.br', 22222222222, '20000', 44,1);

INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario, vl_salario_funcionario,idade_funcionario, departamento_cd_departamento) VALUES ('Mario', 'mario.quinello@docente.unip.br', 33333333333, '20000', 44,Null);

INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario, vl_salario_funcionario,idade_funcionario, departamento_cd_departamento) VALUES ('Mario', 'mario.quinello@docente.unip.br', 44444444444, '20000', 44,2);

Exercício

- Nosso primeiro gatilho será utilizado para monitorar a inserção na tabela funcionário; utilize o script abaixo para criar esta trigger:

```
-- Auditando registros na tabela funcionario - INSERT
DELIMITER //
CREATE TRIGGER auditoria_func_ins AFTER INSERT ON funcionario
        FOR EACH ROW
        BEGIN
                INSERT INTO auditoria (dt_auditoria, usuario, mensagem)
                VALUES (CURRENT_TIMESTAMP, CURRENT_USER(), 'Novo registro inserido na tabela funcionario!' );
        END
```

Exercício

- Para testar a implementação, insira um novo registro na tabela funcionário:

```
-- Testando a trigger auditoria_func_ins
INSERT INTO funcionário (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario, vl_salario_funcionario,
idade_funcionario, cd_departamento) VALUES ('Humberto', 'humberto@humberto.unip.br', ‘55555555555’, 20000, 42,
1);
```

Exercício

- Verifique as tabelas funcionário e auditoria; veja que o registro 1 da tabela auditoria foi disparado pela nossa trigger que armazenou um registro (1) auditando o evento:

```
SELECT * FROM funcionario;

SELECT * FROM auditoria;
```

Exercício

- Agora vamos criar uma trigger para monitorar os registros removidos da tabela funcionários; neste caso vamos utilizar o conceito de OLD para armazenar a informação sobre qual registro foi removido:

```
-- Auditando registros na tabela funcionario - DELETE
DELIMITER //
CREATE TRIGGER auditoria_func_del AFTER DELETE ON funcionario
        FOR EACH ROW
        BEGIN
                INSERT INTO auditoria (dt_auditoria, usuario, mensagem)
                VALUES (CURRENT_TIMESTAMP, CURRENT_USER(), concat('O registro ', OLD.cd_funcionario, ' foi
removido!' ));
        END;
```

Exercício

- Para testar esta trigger, remova o registro 6 da tabela funcionários:

```
DELETE FROM funcionario WHERE cd_funcionario = 6;
```

Exercício

- Verifique as tabelas funcionário e auditoria:

```
SELECT * FROM funcionario;

SELECT * FROM auditoria;
```

Exercício

- Agora vamos impedir que uma determinada informação seja alterada baseado em uma regra de negócio implementada via gatilho; veja que neste caso a verificação ocorre antes que o UPDATE seja efetivado (BEFORE):

“O valor do salário do funcionário não pode ser maior que 5% do orçamento do setor onde atua.”

Exercício


```
-- Auditando registros na tabela funcionario - UPDATE
DELIMITER //
CREATE TRIGGER auditoria_sal_func BEFORE UPDATE ON funcionario
        FOR EACH ROW
        BEGIN
                IF NEW.vl_salario_funcionario > (0.05 * (SELECT vl_orcamento_departamento FROM departamento WHERE cd_departamento =
OLD.cd_departamento)) THEN
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'O registro não foi gravado pois ultrapassa o orçamento previsto!';
                END IF;
        END;
```

Exercício

- Para validar esse gatilho, efetue a atualização abaixo:

```
UPDATE funcionario SET vl_salario_funcionario = 25000 WHERE cd_funcionario = 1;
```

- Veja que não houve alteração para o funcionário com código 1:

```
SELECT * FROM funcionario;
```

Exercício

1. “Verificar no ato de cadastro do funcionário se o mesmo tem idade entre 16 e 70 anos; caso esteja fora dos limites, apontar na tabela auditoria com o respectivo código e a informação se a idade é SUPERIOR ou INFERIOR.”
2. “Verificar no ato de cadastro do departamento se o mesmo tem orçamento previsto; caso não tenha, impedir o INSERT e informar o usuário (SIGNAL SQLState).”
3. “Verificar no ato de cadastro do funcionário se o e-mail do mesmo foi informado; caso não, gravar registro na tabela auditoria informando o código do funcionário que não possui e-mail.”

Exercício

1. “Verificar no ato de cadastro do funcionário se o mesmo tem idade entre 16 e 70 anos; caso esteja fora dos limites, apontar na tabela auditoria com o respectivo código e a informação se a idade é SUPERIOR ou INFERIOR.”

```
DELIMITER //CREATE TRIGGER auditoria_idade_func AFTER INSERT ON funcionarioFOR EACH ROW
        BEGIN
                IF NEW.idade_funcionario < 16 THEN
                        INSERT INTO auditoria (dt_auditoria, usuario, mensagem)
                        VALUES (CURRENT_TIMESTAMP, CURRENT_USER(),
                        concat('O funcionario ', NEW.cd_funcionario, ' tem idade INFERIOR ao esperado!' ));
                ELSEIF NEW.idade_funcionario > 70 THEN
                        INSERT INTO auditoria (dt_auditoria, usuario, mensagem)
                        VALUES (CURRENT_TIMESTAMP, CURRENT_USER(),
                        concat('O funcionario ', NEW.cd_funcionario, ' tem idade SUPERIOR ao esperado!' ));
                END IF;
        END
 //DELIMITER ;
```

Exercício


```
INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario, vl_salario_funcionario, idade_funcionario, cd_departamento)
VALUES ('Roberto', 'Roberto@Roberto.unip.br', 77777777777, '20000', 80, 1);
```

Exercício

2. “Verificar no ato de cadastro do departamento se o mesmo tem orçamento previsto; caso não tenha, impedir o INSERT e informar o usuário (SIGNAL SQLState).”

```
DELIMITER //
CREATE TRIGGER auditoria_orc_dep AFTER INSERT ON departamentoFOR EACH ROW
	BEGIN
		IF NEW.vl_orcamento_departamento <= 0 THEN
			SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'O registro não foi gravado pois não possui valor de orçamento previsto!’;
		END IF;
	END;
```

Exercício


```
INSERT INTO departamento (nm_departamento, vl_orcamento_departamento)
VALUES ('Logistica', 0);
```

Exercício

3. “Verificar no ato de cadastro do funcionário se o e-mail do mesmo foi informado; caso não, gravar registro na tabela auditoria informando o código do funcionário que não possui e-mail.”

```
DELIMITER //CREATE TRIGGER auditoria_email_func AFTER INSERT ON funcionarioFOR EACH ROW
        BEGIN
                IF NEW.ds_email_funcionario is NULL THEN
                        INSERT INTO auditoria (dt_auditoria, usuario, mensagem)
                        VALUES (CURRENT_TIMESTAMP, CURRENT_USER(),
                        concat('O funcionario ', NEW.cd_funcionario, ' não tem e-mail cadastrado!' ));
                END IF;
        END;
```

Exercício


```
INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario, vl_salario_funcionario,
idade_funcionario, cd_departamento)
VALUES ('Roberto', NULL, 77777777777, '20000', 80, 1);
```

OBRIGADO