UNIP
UNIVERSIDADE PAULISTA

UNICLUSTER

Linguagem de Programação de Banco de Dados

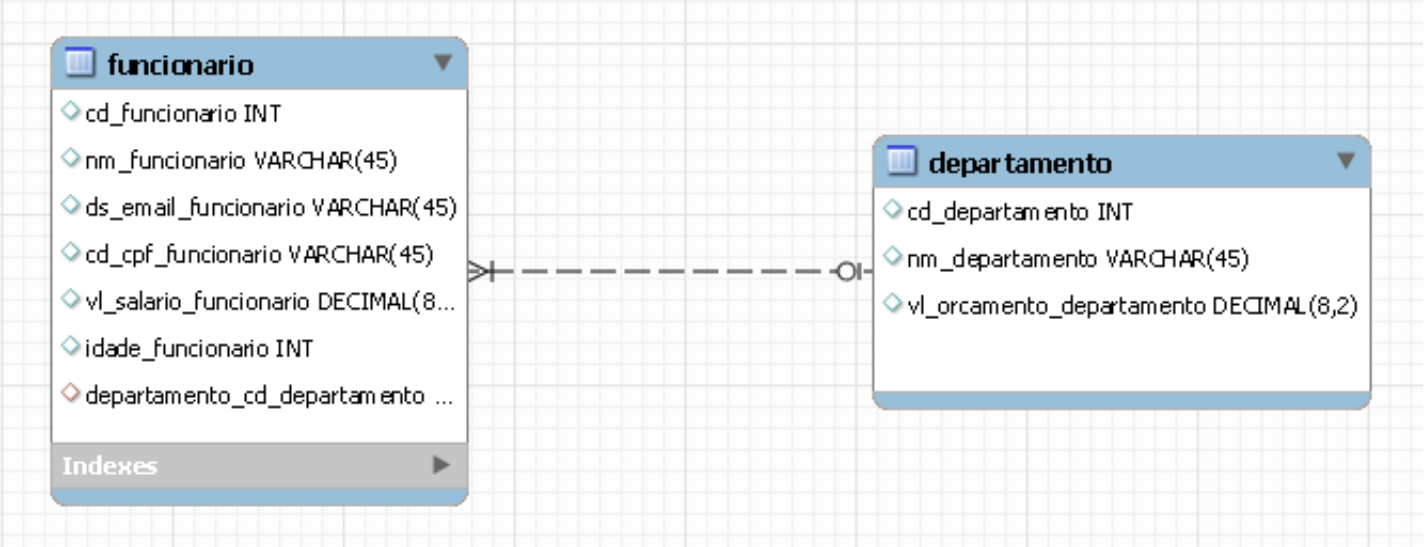
Restrições de Integridade

Carlos Arruda Baltazar

UNIP – Cidade Universitária

Exercício

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



Exercício


```
CREATE DATABASE UNIP_LPBD;

USE UNIP_LPBD;

SET SQL_SAFE_UPDATES = 0;
```

Exercício


```
CREATE TABLE departamento
(
    cd_departamento           INT,
    nm_departamento           VARCHAR(45),
    vl_orcamento_departamento DECIMAL(10,2)
);
```

Exercício


```
CREATE TABLE funcionario
(
    cd_funcionario            INT,
    nm_funcionario            VARCHAR(45),
    ds_email_funcionario      VARCHAR(45),
    cd_cpf_funcionario        VARCHAR(45),
    vl_salario_funcionario    DECIMAL(10,2),
    idade_funcionario         INT,
    cd_departamento           INT
);
```

Exercício

- Adicione restrições de chave às tabelas funcionário e departamento considerando os atributos cd_departamento e cd_funcinario como chaves primárias:

Exercício


```
ALTER TABLE departamento MODIFY cd_departamento INT NOT NULL AUTO_INCREMENT PRIMARY KEY;

ALTER TABLE funcionario MODIFY cd_funcionario INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
```

Exercício

- Após a alteração das tabelas, vamos verificar se as chaves foram criadas:

```
SHOW KEYS FROM departamento WHERE Key_name = 'PRIMARY’;

SHOW KEYS FROM funcionario WHERE Key_name = 'PRIMARY';
```

Exercício

- Vamos testar

```
SHOW KEYS FROM departamento WHERE Key_name = 'PRIMARY’;

SHOW KEYS FROM funcionario WHERE Key_name = 'PRIMARY';
```

Exercício

- Agora vamos testar as chaves:

```
INSERT INTO departamento (nm_departamento, vl_orcamento_departamento) VALUES (1,
'Desenvolvimento', 10);
```

Exercício

- Agora vamos testar as chaves:

```
INSERT INTO funcionario (cd_funcionario, nm_funcionario, ds_email_funcionario,
cd_cpf_funcionario,
vl_salario_funcionario, idade_funcionario, cd_departamento)
VALUES (1, 'Mario', 'mario.quinello@docente.unip.br', 12312312312, '20000', 44, 1);
```

Exercício

- Outra restrição que pode ser exercitada neste modelo é a do Vazio ou Null; veja que nenhum dos campos em ambas as tabelas (com exceção das PK) atribui a característica de Not Null, logo, nenhuma dessas colunas é obrigatória.
- Visando uma melhor consistência destas informações, vamos modificar as tabelas seguindo as regras abaixo:
  - “O nome do funcionário precisa ser obrigatoriamente preenchido”
  - “O nome do departamento precisa ser obrigatoriamente preenchido”

Exercício


```
ALTER TABLE funcionario MODIFY nm_funcionario VARCHAR(45) NOT NULL;

ALTER TABLE departamento MODIFY nm_departamento VARCHAR(45) NOT NULL;
```

Exercício


```
INSERT INTO funcionario (cd_funcionario, nm_funcionario, ds_email_funcionario,
cd_cpf_funcionario, vl_salario_funcionario,
idade_funcionario, cd_departamento) VALUES (2, NULL, 'teste@docente.unip.br',
12312312312, '20000', 44, 1);
```

```
INSERT INTO departamento (cd_departamento, nm_departamento, vl_orcamento_departamento)
VALUES (2, NULL, 10);
```

Neste

Exercício

- Neste mesmo contexto, podemos verificar a **restrição de integridade baseada no domínio**; veja o modelo inicialmente implementado para a tabela funcionario; o campo cd_departamento foi definido como um inteiro:

Exercício


```
INSERT INTO funcionario (cd_funcionario, nm_funcionario, ds_email_funcionario,
cd_cpf_funcionario, vl_salario_funcionario,
idade_funcionario, cd_departamento) VALUES (3, 'Jose Valente', 'valente@docente.unip.br',
12312312312, '20000', 44, 'A');
```

Exercício


```
SELECT * FROM funcionario;

DELETE FROM funcionario WHERE cd_funcionario = 3;
```

Exercício

- Finalmente vamos estabelecer a Integridade Referencial entre as tabelas do nosso modelo, recapitulando: “A Integridade Referencial visa garantir que os valores dos atributos que são chave estrangeira (Foreign Key) em uma relação R1 possuem valores correspondentes nas chaves primárias (Primary Key) na tabela referenciada R2.”

- Com base nesta afirmação, vamos atribuir um relacionamento entre as tabelas funcionario e departamento:

Exercício

```
ALTER TABLE funcionario ADD CONSTRAINT fk_funcionario_departamento FOREIGN KEY
(cd_departamento)
REFERENCES departamento (cd_departamento);
```

Exercício

- Com o relacionamento estabelecimento, vamos validar a restrição inserindo **um funcionário que tenha a sua respectiva chave estrangeira válida**:

```
INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario,
vl_salario_funcionario, cd_departamento)
VALUES ('Mario', 'mario.quinello@docente.uunip.br', '12312312312', '20000', '1');
```

Exercício

- Agora repita o teste, porém, com um valor na FK que não esteja presente na tabela referenciada (departamento); o SGBD não deve permitir que este registro salvo na base de dados:

```
INSERT INTO funcionario (nm_funcionario, ds_email_funcionario, cd_cpf_funcionario,
vl_salario_funcionario, cd_departamento)
VALUES ('Mario', 'mario.quinello@docente.uunip.br', '123123123123', '20000', '10');
```

Exercício

- Por fim, vamos criar uma restrição baseada na regra de negócio utilizando a cláusula CHECK: “O valor do orçamento para cada departamento não pode ser igual a zero.” Aplicando a alteração na tabela departamento:

```
ALTER TABLE departamento ADD CONSTRAINT Valor_ORC CHECK (vl_orcamento_departamento > 0);
```

Exercício

- Agora valide se a restrição foi corretamente implementada efetuando a inserção abaixo:

```
INSERT INTO departamento (cd_departamento, nm_departamento, vl_orcamento_departamento)
VALUES (2, 'Testes', 0);
```

Exercício

- Um atributo com uma restrição de integridade de unicidade obedece o conceito de chave Candidata. Como sabemos, este tipo de campo permite qualificar o registro e garantir a sua unicidade similar a uma chave primária, mesmo que não utilizado com esta função.

Exercício

INSERT INTO departamento (cd_departamento, nm_departamento, vl_orcamento_departamento) VALUES (1, 'Desenvolvimento', 100000);

Exercício

- Veja que o campo cd_cpf_funcionario tem, por natureza da informação, a garantia de ser único; desta forma, podemos alterar sua configuração e implementar a característica UNIQUE e torná-lo uma chave Candidata:

```
ALTER TABLE funcionario ADD CONSTRAINT UNIQUE_CPF_funcionario UNIQUE (cd_cpf_funcionario);
```

Exercício

- Para testar esta restrição, faça duas inserções na tabela funcionários mantendo o CPF; perceba que o incremento da PK ficou a cargo do SGBD como definido no modelo inicial acima:

```
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);
```

OBRIGADO