Normalização

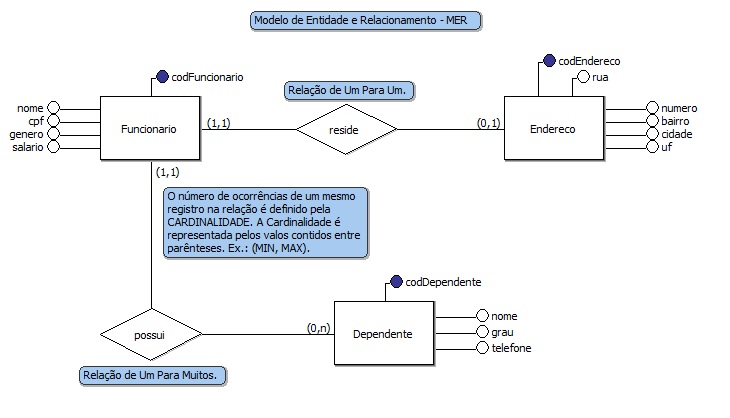
Conjunto de regras aplicado na concepção de um projeto de banco de dados que determina as melhores práticas para a construção das entidades, atributos e relacionamentos. Esse conjunto de regras é chamado de Formas Normais e seu uso evita a ocorrência de problemas de desempenho e integridade comuns a um projeto de banco de dados.

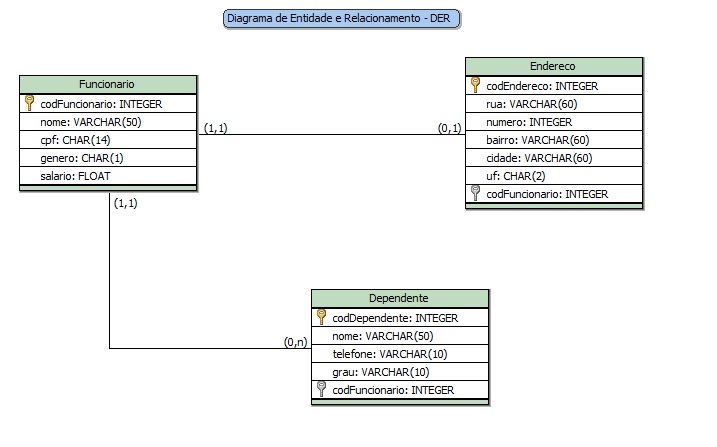
As formas normais são enumeradas entre FN1 e FN5 e são dependentes entre si, um conjunto de entidades, atributos e relaciomentos, só estão de acordo com a FN3, se antes estiver de acordo com FN2.

Relacionamento

Sempre que houver a necessidade de associar registros de uma tabela, a registros de outra tabela, será criado um relacionamento. Um relacionamento é efetivamente construído utilizando Chave Estrangeira (Foreign Key - FK).

Na construção de um relacionamento, a chave estrangeira de uma das tabelas envolvidas na relação faz referência à chave primária da outra tabela envolvida na relação.





-- Aula2 BD - 12/07/2018

/\*

Primeira Forma Normal (FN1)

Todos os campos da tabela devem ser atômicos(indivisíveis). Os dados armazenados em cada um dos

campos da tabela devem possibilitar uma busca simples.

Campo composto e multivalorado devem ser evitados.

Campo Composto - Endereco (Rio branco, 185, Centro, Rio de Janeiro, RJ)

Campo Multivalorado - Telefone (3333-6666 / 2222-7777 / 9999-3333)

-----------------------------------------------------------------------------------------------------

Segunda Forma Normal (FN2)

Todos os campos de uma tabela, que não são Chave Primária, devem ter a sua existência

diretamente, e somente, associada à Chave Primária da tabela ao qual estão contidos.

\*\*/

/\*

Estudo de caso.

Uma empresa requer armazenar os dados de seus funcionários: nome, cpf, gênero, salário e endereço.

Caso o funcionário possua dependendentes, armazenar o nome, o telefone e o grau de parentesco.

\*\*/

DROP DATABASE aula2;

CREATE DATABASE aula2;

USE aula2;

-- AUTO\_INCREMENT - Geração automática de valores sequenciais.

-- UNIQUE - Campo único, não permite duplicidade.

-- ENUM('f', 'm') - Limitar a entrada de dados em 'f' ou 'm'.

CREATE TABLE funcionario (

codFuncionario INTEGER PRIMARY KEY AUTO\_INCREMENT,

nome VARCHAR(50) NOT NULL,

cpf CHAR(14) NOT NULL UNIQUE,

genero ENUM('f', 'm') NOT NULL,

salario FLOAT NOT NULL);

/\*

Declaração de Chave Estrangeira

FOREIGN KEY(cod\_func) REFERENCES funcionario (codFuncionario)

O campo 'cod\_func' da tabela é uma chave estrangeira e faz referência ao campo

'codFuncionario' da tabela 'funcionario'.

\*\*/

CREATE TABLE endereco (

codEndereco INTEGER PRIMARY KEY AUTO\_INCREMENT,

rua VARCHAR(60) NOT NULL,

numero INTEGER NOT NULL,

bairro VARCHAR(60) NOT NULL,

cidade VARCHAR(60) NOT NULL,

uf CHAR(2) NOT NULL,

cod\_func INTEGER NOT NULL UNIQUE,

FOREIGN KEY(cod\_func) REFERENCES funcionario (codFuncionario));

CREATE TABLE dependente (

codDependente INTEGER PRIMARY KEY AUTO\_INCREMENT,

nome VARCHAR(50) NOT NULL,

telefone VARCHAR(10) NOT NULL,

grau VARCHAR(10) NOT NULL,

cod\_func INTEGER NOT NULL,

FOREIGN KEY(cod\_func) REFERENCES funcionario (codFuncionario));

SHOW TABLES;

DESC funcionario;

DESC endereco;

DESC dependente;

-- Inserir dados nas tabelas.

INSERT INTO funcionario VALUES(NULL, 'Bia', '123.432.432-21', 'f', 4000);

INSERT INTO funcionario VALUES(NULL, 'Edu', '222.432.432-21', 'm', 3500);

INSERT INTO funcionario VALUES(NULL, 'Ana', '333.432.432-21', 'f', 5100);

INSERT INTO funcionario VALUES(NULL, 'Nat', '444.432.432-21', 'f', 6200);

INSERT INTO funcionario VALUES(NULL, 'Rui', '555.432.432-21', 'm', 5300);

SELECT \* FROM funcionario;

INSERT INTO endereco VALUES(NULL, 'Rua a', 523, 'Bairro q', 'Cidade z', 'RJ', 3);

INSERT INTO endereco VALUES(NULL, 'Rua s', 315, 'Bairro w', 'Cidade x', 'RJ', 1);

INSERT INTO endereco VALUES(NULL, 'Rua d', 1358, 'Bairro e', 'Cidade c', 'SP', 2);

INSERT INTO endereco VALUES(NULL, 'Rua f', 20, 'Bairro r', 'Cidade v', 'SP', 5);

INSERT INTO endereco VALUES(NULL, 'Rua g', 117, 'Bairro t', 'Cidade b', 'RJ', 4);

SELECT \* FROM endereco;

INSERT INTO dependente VALUES(NULL, 'Beto', 'Filho', '4444-2222', 3);

INSERT INTO dependente VALUES(NULL, 'Rita', 'Filha', '5555-2222', 3);

INSERT INTO dependente VALUES(NULL, 'Mila', 'Filha', '6644-2222', 1);

INSERT INTO dependente VALUES(NULL, 'Cadu', 'Filho', '3344-2222', 5);

INSERT INTO dependente VALUES(NULL, 'Duda', 'Filha', '7744-2222', 5);

INSERT INTO dependente VALUES(NULL, 'Hugo', 'Conjuge', '2244-2222', 4);

INSERT INTO dependente VALUES(NULL, 'Rafa', 'Conjuge', '9999-2222', 5);

SELECT \* FROM dependente;

ALTER TABLE dependente CHANGE COLUMN telefone grauParent VARCHAR(10) NOT NULL;

ALTER TABLE dependente CHANGE COLUMN grau telefone VARCHAR(10) NOT NULL;

-- Pesquisas.

-- Exibir os endereços do Rio de Janeiro.

SELECT \* FROM endereco WHERE uf = 'RJ';

-- Exibir os dependentes com grau de parentesco filho e filha, em uma ordem que os filhos

-- sejam exibidos primeiro.

SELECT \* FROM dependente

WHERE grauParent = 'Filho' OR grauParent = 'Filha'

ORDER BY grauParent DESC, nome;