UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO - UERJ
CIÊNCIA DA COMPUTAÇÃO
BANCO DE DADOS I

DOCUMENTAÇÃO DO DESENVOLVIMENTO DE UM PROJETO DE BANCO DE DADOS

Alline Marjorie Gueiros Camera Coelho

Rio de Janeiro, Julho de 2018

Minimundo

Super Uber é uma empresa criativa que cria experiências inovadoras. Ela trabalha atendendo às demandas de seus clientes externos, que solicitam projetos de acordo com seus próprios interesses, após reunir seus colaboradores mais capazes de desenvolver o projeto e então entregá-lo com seu alto padrão de qualidade. Além dos projetos solicitados por seus clientes, ela também desenvolve projetos internos.

A organização é dividida em vários departamentos. Departamentos são registrados por seus números de identificação únicos e nomes. Um departamento pode possuir vários funcionários trabalhando nele.

Um funcionário é cadastrado com as seguintes informações - número único de matrícula na empresa, número de registro único (cpf ou cnpj), nome, endereço residencial, e-mail, telefone, data de nascimento, salário, número de identificação único de seu cargo, número de identificação único do departamento em que está alocado e a informação que diz se ele é pessoa física ou jurídica. Um funcionário possui apenas um cargo e um departamento, e ele pode trabalhar em múltiplos projetos.

Cargos são registrados pelo número de identificação único e seu nome. Um cargo pode ser ocupado por múltiplos funcionários.

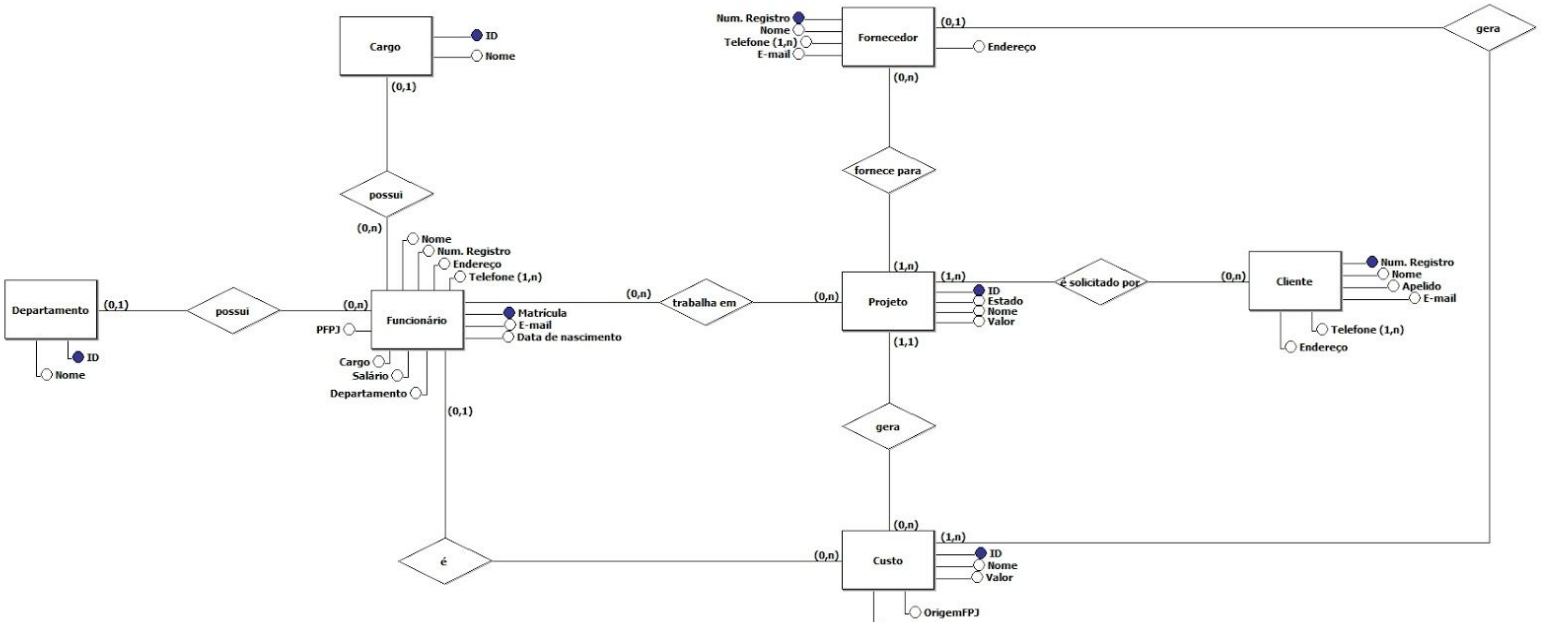
Clientes podem solicitar múltiplos projetos (mas um cliente apenas é registrado no sistema se tiver pedido pelo menos um projeto), e o mesmo projeto pode ser solicitado por mais de um cliente por razões contratuais. Clientes são cadastrados pelo número único de registro (CPF ou CNPJ), nome, apelido (dado não obrigatório), telefone, e-mail e endereço.

Projetos são registrados pelo número de identificação único, nome, estado atual do projeto (se ele apenas foi Criado, se está Em andamento, se foi Concluído ou Cancelado) e valor oferecido por seu(s) cliente(s) (dado não obrigatório). Eles podem gerar vários custos e demandas de vários fornecedores. Podem ter vários funcionários trabalhando no mesmo projeto. Projetos também podem ser solicitados internamente, sem relação com clientes.

Custos são registrados pelo número de identificação único, nome (a natureza do custo), valor, origemPJ e origemF - custos podem ser originado de um funcionário PJ ou fornecedor, então há dois dados não obrigatórios referentes à origem do custo, já que ele não pode ser gerado por um funcionário PJ ou um fornecedor ao mesmo tempo. Cada custo está necessariamente associado à apenas um projeto e um funcionário PJ/fornecedor.

Fornecedores são registrados pelo nome, número de registro único (CPF ou CNPJ), telefone, endereço e e-mail. Um fornecedor deve fornecer seu trabalho para um ou mais projetos.

Modelagem conceitual ER



https://image.ibb.co/kV1kro/MER.jpg

Entidades

Departamento

- **ID**: Chave primária. Sua restrição de domínio é aceitar apenas números inteiros.
- **Nome**: Nome do departamento. Texto até 30 caracteres.

Cargo

- **ID**: Chave primária. Número de identificação único de cada cargo. Inteiro.
- **Nome**: Nome do cargo. Texto até 30 caracteres.

Funcionário

- **Matrícula**: Chave primária. Número de identificação único de cada funcionário. Inteiro.
- **Número de registro**: Número único de registro do funcionário - CPF ou CNPJ. Texto até 14 caracteres.
- **Nome**: Nome do funcionário. Texto até 80 caracteres.
- **Endereço**: Endereço do funcionário. Texto até 200 caracteres.
- **Telefone**: Número de telefone do funcionário. Texto até 13 caracteres.
    - não é seguro armazenar números de telefone como inteiros, pois além deles potencialmente resultarem em inteiros grandes, estes também podem incluir caracteres alfabéticos. 13 caracteres são suficientes para armazenar telefones nacionais.
- **E-mail**: Endereço de e-mail do funcionário. Texto até 80 caracteres.
- **Data de nascimento**: Data de nascimento do funcionário. Admite valores do tipo Data.
- **Salário**: Valor de salário do funcionário. Admite valores do tipo *Double*.
- **PFPJ**: Informa se o funcionário é Pessoa Física ou Pessoa Jurídica. Admite valores do tipo *TINYINT*, que funciona como o tipo *Boolean*.
- **Departamento**: Chave estrangeira. Referencia a chave Departamento(ID).
- **Cargo**: Chave estrangeira. Referencia a chave Cargo(ID).

Projeto

- **ID**: Chave primária. Número de identificação único de cada projeto. Inteiro.

- **Nome**: Nome do projeto. Texto até 30 caracteres.
- **Valor**: Valor do projeto. Admite valores do tipo *Double*. Dado não obrigatório.
- **Estado**: Estado do projeto. Texto até 12 caracteres.

Fornecedor

- **Número de registro**: Chave primária. Número de CPF ou CNPJ do fornecedor. Texto até 14 caracteres.
- **Nome**: Nome do fornecedor. Texto até 50 caracteres.
- **Telefone**: Número de telefone do fornecedor. Texto até 13 caracteres.
- **E-mail**: Endereço de e-mail do fornecedor. Texto até 80 caracteres.
- **Endereço**: Endereço do fornecedor. Texto até 200 caracteres.

Custo

- **ID**: Chave primária. Número de identificação único de cada custo. Inteiro.
- **Nome**: Breve descrição do tipo de custo. Texto até 30 caracteres.
- **Valor**: Valor do custo. Admite valores do tipo *Double*.
- **ProjetoID**: Chave estrangeira. Referencia a chave Projeto(ID).
- **OrigemFPJ**: Chave estrangeira. Referencia a chave Funcionario(Matrícula).
- **OrigemForn**: Chave estrangeira. Referencia a chave Fornecedor(ID).

Cliente

- **Número de registro**: Chave primária. Número de CPF ou CNPJ do cliente. Texto até 14 caracteres.
- **Nome**: Nome do cliente. Texto até 80 caracteres.
- **Apelido**: Apelido do cliente. Dado não obrigatório. Texto até 20 caracteres.
- **Telefone**: Número de telefone do cliente. Texto até 13 caracteres.
- **E-mail**: Endereço de e-mail do cliente. Texto até 80 caracteres.
- **Endereço**: Endereço do cliente. Texto até 200 caracteres.

Relacionamentos

Funcionário x Departamento

Um funcionário trabalha em apenas um departamento.
Um departamento pode ter vários funcionários. É possível criar um departamento e não associá-lo à ninguém até ele ser ocupado por funcionários.

Funcionário x Cargo

Um funcionário possui apenas um cargo.
Um cargo pode ser associado à vários funcionários. É possível criar um cargo e não associá-lo à ninguém até a vaga ser ocupada.

Funcionário x Projeto

Um funcionário pode trabalhar em um ou mais projetos. Há funcionários que não trabalham em projetos.
Um ou mais funcionários podem trabalhar no mesmo projeto. Um projeto pode ser criado antes da alocação de funcionários à ele ser feita.

Funcionário x Custo

Um funcionário PJ pode gerar um ou mais custos. Um funcionário PF não gera custos.
Um custo pode ou não ser gerado por um funcionário PJ.

Projeto x Fornecedor

Um projeto pode receber um ou mais fornecedores. Podem existir projetos que são desenvolvidos sem fornecedores.
Um fornecedor deve trabalhar para um ou mais projetos.

Projeto x Custo

Um projeto pode gerar um ou mais custos. Podem existir projetos que não geram custos.
Um custo deve estar associado apenas ao projeto que o gerou.

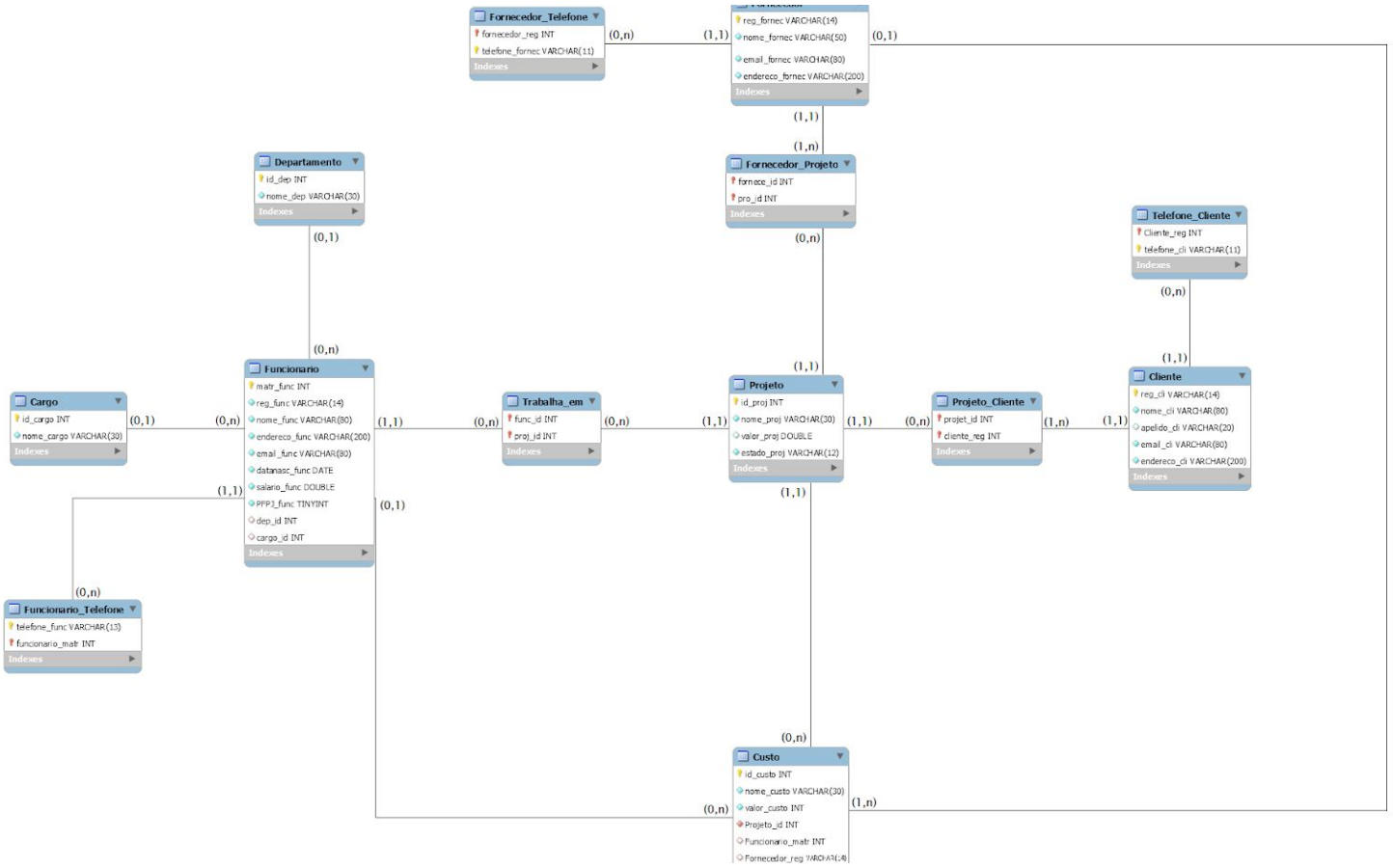
Projeto x Cliente

Um projeto pode ser solicitado por um ou mais clientes. Há projetos internos que não são solicitados por clientes.
Um cliente deve solicitar um ou mais projetos.

Custo x Fornecedor

Um fornecedor deve gerar um ou mais custos.
Um custo pode ou não ser gerado por um fornecedor.

Modelagem lógica relacional



https://image.ibb.co/hqKiBo/DER.png

De acordo com os tipos de relacionamento presentes, foram feitas adições de tabelas e atribuições de chaves estrangeiras.

Relacionamentos n-n (eg. Projeto x Funcionário) geram uma nova tabela com as chaves estrangeiras de entidade envolvida.

Relacionamentos 1-n (eg. Funcionário x Custo) não geram uma nova tabela, mas a entidade que pode aparecer múltiplas vezes adquire a chave estrangeira da entidade que só pode aparecer uma vez.

Com atributos multivalorados (eg. Telefone), foi necessária a criação de uma tabela para guardar os números de telefone e a chave da entidade relacionada, que formam uma chave primária composta.

Restrições de integridade estruturais

O conjunto de valores permitidos em cada coluna é bem definido no momento de criação de cada uma, respeitando a **integridade de domínio**.
Um campo pode receber o valor NULL, desde que ele não faça parte de uma coluna obrigatória, respeitando a **integridade de vazio**.
Todas as chaves primárias tem campos obrigatórios (não admitem valores nulos), respeitando a **integridade de chave**.
Todas as chaves estrangeiras possuem chaves primárias correspondentes em outras tabelas, respeitando a **integridade referencial**.

Logo, as restrições de integridade estruturais são obedecidas no banco.

Restrições semânticas

- Um funcionário estagiário não pode ter salário superior à R$2000;
- Um funcionário com contrato PJ não pode ser estagiário.

Consultas em Álgebra Relacional

```
σ matr_func=42 (Funcionario)
```

Retorna o funcionário com id 42.

```
π nome_func (σ dep_id=1 (Funcionario))
```

Retorna a lista de funcionários que trabalham no departamento de id 1.

```
π nome_func (Funcionario) ∪ π nome_cli (Cliente)
```

Retorna a lista de funcionários e clientes da empresa.

```
funcionarios_dep1 = π nome_func (σ dep_id=1 (Funcionario))
π nome_func (Funcionario) - funcionarios_dep1
```

Retorna a lista de funcionários da empresa que não trabalham no departamento de id 1.

```
a0 = (Trabalha_em) ⋈ Trabalha_em.proj_id = Projeto_Cliente.proj_id (Projeto_Cliente)
ProjIds = π Trabalha_em.proj_id (a0)
FuncIds = π Trabalha_em.func_id (a0)
CliIds = π Projeto_Cliente.cli_id (a0)

ProjNomesIds = π nome_proj, proj_id (ProjIds ⋈ (Trabalha_em.proj_id = Projeto.id_proj) Projeto)
FuncNomes = π nome_func (FuncIds ⋈ (Trabalha_em.func_id = Funcionario.matr_func) Funcionario)
CliNomes = π nome_cli (CliIds ⋈ (Projeto_Cliente.cli_id = Cliente.reg_cli) Cliente)

ProjFuncs = ProjNomesIds x FuncNomes
ProjFuncs x CliNomes
```

Retorna a lista de nomes de funcionários e de nomes de clientes que estão envolvidos nos mesmos projetos, que também estão listados por seus nomes e ids.

```
func1 = π proj_id (σ func_id=1 (Trabalha_em))
func2 = π proj_id (σ func_id=2 (Trabalha_em))
func1 ∩ func2
```

Retorna a lista de projetos em comum que os funcionários de matrículas 1 e 2 estão trabalhando em.

```
Projetos = π proj_id (Trabalha_em)
Funcs = π func_id (Trabalha_em)
ProjFuncs = Projetos x Funcs
ProjFuncs ÷ Funcs
```

Retorna a lista de projetos em que todos os funcionários ativos em projetos estão trabalhando em.

```
π nome_func, nome_dep, id_dep (Funcionario ⋈ (Funcionario.dep_id = 2) Departamento)
```

Retorna a lista de nomes de funcionários do departamento de id 2, o nome deste departamento e sua id

```
π nome_func, nome_dep, id_dep (Funcionario ⋈ Funcionario.dep_id = Departamento.id_dep Departamento)
```

Retorna uma lista de todos os nomes de funcionários e seus nomes de departamentos e ids. Se um funcionário não estiver associado à algum departamento, será impresso *'null'* no lugar de seu nome e id.

```
π nome_func, nome_dep, id_dep (Funcionario ⋈ Funcionario.dep_id = Departamento.id_dep Departamento)
```

Retorna uma lista de todos os nomes de departamentos e suas ids, com os nomes dos funcionários que trabalham neles. Se um departamento não tiver funcionários trabalhando nele, será impresso *'null'* no lugar do nome do funcionário.

```
π nome_func, nome_dep, id_dep (Funcionario ⋈ Funcionario.dep_id = Departamento.id_dep Departamento)
```

Retorna uma lista de todos os nomes de funcionários, nomes de departamentos e ids de departamentos independentemente dos funcionários estarem associados à departamentos ou de cada departamento ter funcionários trabalhando nele. Em qualquer um desses casos, *'null'* é impresso no lugar da informação que não existe.

Consultas em SQL

SELECT * FROM funcionario WHERE matr_func=1;
Retorna o funcionário de matrícula 1.

SELECT * FROM projeto WHERE id_proj<>1;
Retorna a lista de projetos que não tem ID 1.

SELECT * FROM funcionario WHERE matr_func NOT IN(1,2,3);
Retorna a lista de funcionários que não tem matrícula 1, 2 ou 3.

SELECT * FROM funcionario WHERE nome_func LIKE 'J%';
Retorna a lista de funcionários que tem nomes começando com ‘J’.

SELECT * FROM funcionario ORDER BY nome_func asc;
Retorna a lista de funcionários em ordem alfabética.

SELECT funcionario.matr_func as 'funcionario', proj_id as 'projeto' from trabalha_em, funcionario where funcionario.matr_func = trabalha_em.func_matr;

SELECT count(*) FROM funcionario;
Retorna o número de funcionários registrados.

CREATE DATABASE database CHARACTER SET utf8 COLLATE utf8_unicode_ci;
Cria um banco de dados com charset UTF8 e collate utf8-unicode-ci.

DROP DATABASE database;
Destrói um banco de dados.

(Exemplos de criações de tabelas e inserções de dados nelas no Script DDL)

DROP TABLE Departamento;
Destrói a tabela Departamento.

DESCRIBE Departamento;
Descreve o conteúdo da tabela Departamento.

ALTER TABLE funcionario
ADD GENERO TINYINT NOT NULL;
Adiciona um novo campo na tabela Funcionário.

```
INSERT INTO Custo (id_custo, nome_custo, valor_custo, projeto_id,
funcionario_matr) VALUES (666, 'Serviço de dev', 3000,
        (SELECT id_proj FROM Projeto WHERE nome_proj='Ragnarok'),
        (SELECT matr_func FROM Funcionario WHERE nome_func='Lisa Lisa')
);
```

Adiciona um novo custo buscando o projeto e funcionário associados por seus nomes (‘Ragnarok’ e ‘Lisa Lisa’).

```
TRUNCATE table Cliente;
```

Apaga o registro de todos os clientes.

Script DDL

```
CREATE TABLE IF NOT EXISTS Cargo (
	id_cargo int not null primary key,
	nome_cargo varchar(30) not null
);

CREATE TABLE IF NOT EXISTS Departamento (
	id_dep int not null primary key,
	nome_dep varchar(30) not null
);

CREATE TABLE IF NOT EXISTS Funcionario (
	matr_func int not null primary key,
	reg_func varchar(14) not null,
	nome_func varchar(80) not null,
	endereco_func varchar(200) not null,
	email_func varchar(80) not null,
	datanasc_func date not null,
	salario_func double not null,
	pfpj_func tinyint(1) not null,
	dep_id int,
	cargo_id int,

	CONSTRAINT dep_id_fk
		FOREIGN KEY (dep_id) REFERENCES Departamento (id_dep),
	CONSTRAINT cargo_id_fk
		FOREIGN KEY (cargo_id) REFERENCES Cargo (id_cargo),
	CONSTRAINT reg_func_uniq
		UNIQUE (reg_func)
);

CREATE TABLE IF NOT EXISTS Projeto (
	id_proj int not null primary key,
	nome_proj varchar(30) not null,
	valor_proj double,
	estado_proj varchar(12) not null
);

CREATE TABLE IF NOT EXISTS Cliente (
	reg_cli varchar(14) not null primary key,
```

```
	nome_cli varchar(80) not null,
	email_cli varchar(80) not null,
	apelido_cli varchar(20),
	endereco_cli varchar(200) not null
);

CREATE TABLE IF NOT EXISTS Fornecedor (
	reg_fornec int not null primary key,
	nome_fornec varchar(50) not null,
	email_fornec varchar(80) not null,
	endereco_fornec varchar(200) not null
);

CREATE TABLE IF NOT EXISTS Custo (
	id_custo int not null primary key,
	nome_custo varchar(30) not null,
	valor_custo double not null,
	projeto_id int,
	funcionario_matr int,
	fornecedor_reg varchar(14),

	CONSTRAINT projeto_id_fk
		FOREIGN KEY (projeto_id) REFERENCES Projeto(id_proj),
	CONSTRAINT funcionario_matr_fk
		FOREIGN KEY (funcionario_matr) REFERENCES
Funcionario(matr_func),
	CONSTRAINT fornecedor_reg_fk
		FOREIGN KEY (fornecedor_reg) REFERENCES
Fornecedor(reg_fornec)
);

CREATE TABLE IF NOT EXISTS Funcionario_Telefone (
	telefone_func varchar(13) not null primary key,
	func_matr int,

	CONSTRAINT func_matr
		FOREIGN KEY (func_matr) REFERENCES Funcionario(matr_func)
);

CREATE TABLE IF NOT EXISTS Trabalha_em (
	func_matr int,
	proj_id int,
```

```
        CONSTRAINT func_matr_fk
                FOREIGN KEY (func_matr) REFERENCES Funcionario(matr_func),
        CONSTRAINT proj_id_fk
                FOREIGN KEY (proj_id) REFERENCES Projeto(id_proj)
);

CREATE TABLE IF NOT EXISTS Projeto_Cliente (
        projet_id int,
        cliente_reg varchar(14),

        CONSTRAINT projet_id_fk
                FOREIGN KEY (projet_id) REFERENCES Projeto(id_proj),
        CONSTRAINT cliente_id_fk
                FOREIGN KEY (cliente_reg) REFERENCES Cliente(reg_cli)
);

CREATE TABLE IF NOT EXISTS Cliente_Telefone (
        telefone_cli varchar(13) not null primary key,
        cli_reg varchar(14),

        CONSTRAINT cli_reg_fk
                FOREIGN KEY (cli_reg) REFERENCES Cliente(reg_cli)
);

CREATE TABLE IF NOT EXISTS Funcionario_Custo (
        funcionario_matr int,
        custo_id int,

        CONSTRAINT funcionario_matr_fk
                FOREIGN KEY (funcionario_matr) REFERENCES
Funcionario(matr_func),
        CONSTRAINT custo_id_fk
                FOREIGN KEY (custo_id) REFERENCES Custo(id_custo)
);

CREATE TABLE IF NOT EXISTS Custo_Fornecedor (
        cust_id int,
        forn_reg varchar(14),

        CONSTRAINT cust_id_fk
                FOREIGN KEY (cust_id) REFERENCES Custo(id_custo),
```

```
	CONSTRAINT forn_reg_fk
		FOREIGN KEY (forn_reg) REFERENCES Fornecedor(reg_fornec)
);

CREATE TABLE IF NOT EXISTS Fornecedor_Telefone (
	telefone_fornec varchar(13) not null primary key,
	fornecedor_reg varchar(14),

	CONSTRAINT fornecedor_reg_fk
		FOREIGN KEY (fornecedor_reg) REFERENCES
Fornecedor(reg_fornec)
);

CREATE TABLE IF NOT EXISTS Fornecedor_Projeto (
	fornece_reg varchar(14),
	pro_id int,

	CONSTRAINT fornece_reg_fk
		FOREIGN KEY (fornece_reg) REFERENCES Fornecedor(reg_fornec),
	CONSTRAINT pro_id_fk
		FOREIGN KEY (pro_id) REFERENCES Projeto(id_proj)
);

INSERT INTO Cargo (id_cargo, nome_cargo) VALUES (1, 'Designer');
INSERT INTO Cargo (id_cargo, nome_cargo) VALUES (2, 'Desenvolvedor');
INSERT INTO Cargo (id_cargo, nome_cargo) VALUES (3, 'Contador');
INSERT INTO Cargo (id_cargo, nome_cargo) VALUES (4, 'Gerente de TI');
INSERT INTO Cargo (id_cargo, nome_cargo) VALUES (5, 'Faxineiro');
INSERT INTO Cargo (id_cargo, nome_cargo) VALUES (6, 'Recepcionista');

INSERT INTO Departamento (id_dep, nome_dep) VALUES (1, 'Financeiro');
INSERT INTO Departamento (id_dep, nome_dep) VALUES (2, 'UX');
INSERT INTO Departamento (id_dep, nome_dep) VALUES (3, 'TI');
INSERT INTO Departamento (id_dep, nome_dep) VALUES (4, 'Assistência');
INSERT INTO Departamento (id_dep, nome_dep) VALUES (5, 'Limpeza');

INSERT INTO Funcionario (matr_func, reg_func, nome_func, endereco_func,
email_func, datanasc_func, salario_func, pfpj_func)
	VALUES (1, '12345678912', 'Joseph Joestar', 'Aquele lugar',
'joseph@joestar.com', 1960-09-27, 9999, 1);

INSERT INTO Funcionario (matr_func, reg_func, nome_func, endereco_func,
```

```
email_func, datanasc_func, salario_func, pfpj_func)
	VALUES (2, '12345678921', 'Jonathan Joestar', 'Aquele lugar ali',
'jonathan@joestar.com', 1968-04-04, 30000, 2);

INSERT INTO Funcionario (matr_func, reg_func, nome_func, endereco_func,
email_func, datanasc_func, salario_func, pfpj_func)
	VALUES (3, '12345678900', 'Lisa Lisa', 'Aquele lugar ala',
'lisalisa@joestar.com', 2000-01-01, 32000, 2);

INSERT INTO Funcionario (matr_func, reg_func, nome_func, endereco_func,
email_func, datanasc_func, salario_func, pfpj_func)
	VALUES (4, '12345678911', 'Jotaro Kujo', 'Aquele lugar niponico',
'jotaro@joestar.com', 1990-01-01, 2000, 1);

INSERT INTO Funcionario (matr_func, reg_func, nome_func, endereco_func,
email_func, datanasc_func, salario_func, pfpj_func)
	VALUES (5, '12345678922', 'Josuke Higashikata', 'Aquele niponico',
'jojo@joestar.com', 1999-01-01, 200, 1);

INSERT INTO Projeto (id_proj, nome_proj, valor_proj, estado_proj) VALUES (1,
'Ragnarok', 9999, 'Em andamento');
INSERT INTO Projeto (id_proj, nome_proj, valor_proj, estado_proj) VALUES (2,
'MTV', 999, 'Cancelado');
INSERT INTO Projeto (id_proj, nome_proj, valor_proj, estado_proj) VALUES (3,
'Aquele projeto', 8888, 'Criado');
INSERT INTO Projeto (id_proj, nome_proj, valor_proj, estado_proj) VALUES (4,
'FIRJAN', 20000, 'Concluído');
INSERT INTO Projeto (id_proj, nome_proj, valor_proj, estado_proj) VALUES (5,
'Snake Eater', 999999, 'Concluído');

INSERT INTO Cliente (reg_cli, nome_cli, email_cli, endereco_cli) VALUES
('09296295000160', 'Azul', 'azul@azul.com', 'Ali po');
INSERT INTO Cliente (reg_cli, nome_cli, email_cli, endereco_cli) VALUES
('12343243234336', 'SENAC', 'senac@senac.com', 'Ali aaaaa');
INSERT INTO Cliente (reg_cli, nome_cli, email_cli, apelido_cli, endereco_cli)
VALUES ('12333333311', 'Danilo', 'danilo@gol.com', 'Dani', 'Ali a');

INSERT INTO Fornecedor (reg_fornec, nome_fornec, email_fornec,
endereco_fornec) VALUES ('61190658000106', 'Dio', 'dio@dio.com', 'aliiiiii');

INSERT INTO Custo (id_custo, nome_custo, valor_custo, projeto_id,
funcionario_matr) VALUES (1, 'Serviço de UX', 3000, 1, 1);
```

```
INSERT INTO Custo (id_custo, nome_custo, valor_custo, projeto_id,
fornecedor_reg) VALUES (2, 'Monitor novo Apple', 4000, 1, '61190658000106');

INSERT INTO Funcionario_Telefone (telefone_func, func_matr) VALUES
('21999999999', 1);
INSERT INTO Funcionario_Telefone (telefone_func, func_matr) VALUES
('21999999991', 1);
INSERT INTO Funcionario_Telefone (telefone_func, func_matr) VALUES
('21999999992', 1);
INSERT INTO Funcionario_Telefone (telefone_func, func_matr) VALUES
('21999999993', 2);
INSERT INTO Funcionario_Telefone (telefone_func, func_matr) VALUES
('21999999994', 4);
INSERT INTO Funcionario_Telefone (telefone_func, func_matr) VALUES
('21999999995', 6);

INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (1, 1);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (2, 1);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (4, 1);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (1, 2);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (2, 2);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (4, 2);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (1, 3);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (2, 3);
INSERT INTO Trabalha_em (func_matr, proj_id) VALUES (1, 4);

INSERT INTO Projeto_Cliente (projet_id, cliente_reg) VALUES (1,
'09296295000160');
INSERT INTO Projeto_Cliente (projet_id, cliente_reg) VALUES (1,
'12343243234336');
INSERT INTO Projeto_Cliente (projet_id, cliente_reg) VALUES (2,
'86786739434343');
INSERT INTO Projeto_Cliente (projet_id, cliente_reg) VALUES (3,
'09296295000160');

INSERT INTO Cliente_Telefone (telefone_cli, cli_reg) VALUES ('21992299221',
'09296295000160');
INSERT INTO Cliente_Telefone (telefone_cli, cli_reg) VALUES ('21993399330',
'12343243234336');
INSERT INTO Cliente_Telefone (telefone_cli, cli_reg) VALUES ('21903309329',
'12343243234336');
INSERT INTO Cliente_Telefone (telefone_cli, cli_reg) VALUES ('21937367643',
```

'86786739434343');
INSERT INTO Cliente_Telefone (telefone_cli, cli_reg) VALUES ('21973673639', '09296295000160');
INSERT INTO Cliente_Telefone (telefone_cli, cli_reg) VALUES ('21936553875', '09296295000160');

INSERT INTO Funcionario_Custo (funcionario_matr, custo_id) VALUES (1, 1);

INSERT INTO Custo_Fornecedor (cust_id, forn_reg) VALUES (2, '61190658000106');

INSERT INTO Fornecedor_Telefone (telefone_fornec, fornecedor_reg) VALUES ('21999995555', '61190658000106');
INSERT INTO Fornecedor_Telefone (telefone_fornec, fornecedor_reg) VALUES ('21999996666', '61190658000106');

INSERT INTO Fornecedor_Projeto (fornece_reg, pro_id) VALUES ('61190658000106', 1);

Avaliação de qualidade

Projeto

- **Dependências funcionais**: todos os campos são funcionalmente dependentes da chave primária;
- **Formas normais**: segue as três formas normais.

Cliente

- **Dependências funcionais**: todos os campos são funcionalmente dependentes da chave primária;
- **Formas normais**: não segue a 3FN, pois o campo ‘Apelido’ depende do também do campo ‘Nome’ para ser determinado.

Funcionário

- **Dependências funcionais**: todos os campos são funcionalmente dependentes da chave primária;
- **Formas normais**: segue as três formas normais.