PMR 3304 - Sistemas de Informação

Laboratório Aula 03

Profs. Marcos S. G. Tsuzuki e José Reinaldo Silva

PMR - EPUSP

9 de setembro de 2019

Desenvolvendo Entidade-Relação

Vamos inicialmente criar um novo ambiente. Para que cada aluno tenha a sua versão, mesmo em outras turmas, o diretório deverá conter o seu número USP, não coloque os símbolos < e >, apenas o número USP direto após loja.

```
1 rails new loja
```

Em seguida, avance para o diretório recém criado.

```
1 cd loja<NUSP>
```

Vamos criar a tabela **Produto**

```
1 rails generate scaffold Produto descricao:string unidade:string valunit:
    decimal
```

Vamos criar a tabela **Cliente**

```
1 rails generate scaffold Cliente nome:string endereco:string cidade:
    string CEP:string UF:string CNPJ:string IE:string
```

Vamos criar a tabela **Vendedor**

```
1 rails generate scaffold Vendedor nome:string salario:decimal
    faixacomissao:string
```

Vamos criar a tabela **Pedido**

```
1 rails generate scaffold Pedido numpedido:integer prazoentrega:integer
    vendedor:references cliente:references
```

Vamos criar a tabela **Item**

```
1 rails generate scaffold Item quantidade:integer produto:references
    pedido:belongs_to
```

Foram criados os seguintes arquivos de migration

```ruby
class CreateProdutos < ActiveRecord::Migration[5.2]
  def change
    create_table :produtos do |t|
      t.string :descricao
      t.string :unidade
      t.decimal :valunit

      t.timestamps
    end
  end
end
```

```ruby
class CreateClientes < ActiveRecord::Migration[5.2]
  def change
    create_table :clientes do |t|
      t.string :nome
      t.string :endereco
      t.string :cidade
      t.string :CEP
      t.string :UF
      t.string :CNPJ
      t.string :IE

      t.timestamps
    end
  end
end
```

```ruby
class CreateVendedors < ActiveRecord::Migration[5.2]
  def change
    create_table :vendedors do |t|
      t.string :nome
      t.decimal :salario
      t.string :faixacomissao

      t.timestamps
    end
  end
end
```

```ruby
class CreatePedidos < ActiveRecord::Migration[5.2]
  def change
    create_table :pedidos do |t|
      t.integer :numpedido
      t.integer :prazoentrega
      t.references :vendedor, foreign_key: true
      t.references :cliente, foreign_key: true
```

```
8
9         t.timestamps
10      end
11    end
12  end
```

```
1  class CreateItems < ActiveRecord::Migration[5.2]
2    def change
3      create_table :items do |t|
4        t.integer :quantidade
5        t.references :produto, foreign_key: true
6        t.belongs_to :pedido, foreign_key: true
7
8        t.timestamps
9      end
10   end
11 end
```

Vamos executar todas as migrations: **rails db:migrate**. Temos um arquivo contendo a criação do banco de dados **db/seeds.rb**. Este arquivo deverá atualizado e executado com o comando: **rails db:seed**.

```
1  # This file should contain all the record creation needed to seed the
       database with its default values.
2  # The data can then be loaded with the rails db:seed command (or created
       alongside the database with db:setup).
3  #
4  # Examples:
5  #
6  #   movies = Movie.create([{ name: 'Star Wars' }, { name: 'Lord of the
       Rings' }])
7  #   Character.create(name: 'Luke', movie: movies.first)
8
9  P01 = Produto.create(descricao: 'PARAFUSO' , unidade: 'KG' , valunit:
       1.25)
10 P02 = Produto.create(descricao: 'QUEIJO'   , unidade: 'KG' , valunit:
       0.97)
11 P03 = Produto.create(descricao: 'CHOCOLATE', unidade: 'BAR', valunit:
       0.87)
12 P04 = Produto.create(descricao: 'VINHO'    , unidade: 'L'  , valunit:
       2.00)
13 P05 = Produto.create(descricao: 'ACUCAR'   , unidade: 'SAC', valunit:
       0.30)
14 P06 = Produto.create(descricao: 'LINHA'    , unidade: 'M'  , valunit:
       1.80)
15 P07 = Produto.create(descricao: 'OURO'     , unidade: 'G'  , valunit:
       6.18)
```

```
16 P08 = Produto.create(descricao: 'MADEIRA'  , unidade: 'M'  , valunit:
      0.25)
17 P09 = Produto.create(descricao: 'CANO'     , unidade: 'M'  , valunit:
      1.97)
18 P10 = Produto.create(descricao: 'PAPEL'    , unidade: 'M'  , valunit:
      1.05)
19 P11 = Produto.create(descricao: 'LINHO'    , unidade: 'M'  , valunit:
      0.11)
20
21
22 C01 = Cliente.create(nome: 'ANA'      , endereco: 'RUA 17 N.19'        ,
       cidade: 'NITEROI'       , CEP: '24358310', UF: 'RJ', CNPJ: '
      121132131/0001-34', IE: '2134')
23 C02 = Cliente.create(nome: 'FLAVIO'   , endereco: 'AV PRES VARGAS 10'  ,
       cidade: 'SAOPAULO'      , CEP: '22763931', UF: 'SP', CNPJ: '
      225341261/0001-09', IE: '6374')
24 C03 = Cliente.create(nome: 'JORGE'    , endereco: 'RUA CAIAPO 13'      ,
       cidade: 'CURITIBA'      , CEP: '30078500', UF: 'PR', CNPJ: '
      142563748/0001-09', IE: '7532')
25 C04 = Cliente.create(nome: 'LUCIA'    , endereco: 'RUA ITABIRA 123'    ,
       cidade: 'BELO HORIZONTE', CEP: '22124391', UF: 'MG', CNPJ: '
      193847562/0001-08', IE: '8847')
26 C05 = Cliente.create(nome: 'MAURICIO' , endereco: 'AV PAULISTA 1236'   ,
       cidade: 'SAO PAULO'     , CEP: '30126830', UF: 'SP', CNPJ: '
      123546923/0001-06', IE: '9584')
27 C06 = Cliente.create(nome: 'EDMAR'    , endereco: 'RUA DA PRAIA SN'    ,
       cidade: 'SALVADOR'      , CEP: '30079300', UF: 'BA', CNPJ: '
      839265472/0001-09', IE: '2345')
28 C07 = Cliente.create(nome: 'RODOLFO'  , endereco: 'LARGO DA LAPA 27'   ,
       cidade: 'RIO DE JANEIRO', CEP: '30078300', UF: 'RJ', CNPJ: '
      192746453/0001-09', IE: '4563')
29 C08 = Cliente.create(nome: 'BETH'     , endereco: 'AV CLIMERIO 45'     ,
       cidade: 'SAO PAULO'     , CEP: '25679300', UF: 'SP', CNPJ: '
      494874635/0001-09', IE: '1523')
30 C09 = Cliente.create(nome: 'PAULO'    , endereco: 'TV MORAES C/3'      ,
       cidade: 'LONDRINA'      , CEP: '30077500', UF: 'PR', CNPJ: '
      857920345/0001-08', IE: '1234')
31 C10 = Cliente.create(nome: 'LIVIO'    , endereco: 'AV BEIRA MAR 1256'  ,
       cidade: 'FLORIANOPOLIS' , CEP: '30046500', UF: 'SC', CNPJ: '
      134583049/0001-05', IE: '5874')
32 C11 = Cliente.create(nome: 'SUSANA'   , endereco: 'RUA LOPES MENDES 12',
       cidade: 'NITEROI'       , CEP: '30225900', UF: 'RJ', CNPJ: '
      374856745/0001-03', IE: '4837')
33 C12 = Cliente.create(nome: 'RENATO'   , endereco: 'RUA MEIRELES 123'   ,
       cidade: 'SAO PAULO'     , CEP: '30438700', UF: 'SP', CNPJ: '
      123748393/0001-04', IE: '1123')
34 C13 = Cliente.create(nome: 'SEBASTIAO', endereco: 'RUA DA IGREJA 10'   ,
       cidade: 'UBERABA'       , CEP: '22841650', UF: 'MG', CNPJ: '
      284756393/0001-05', IE: '3321')
```

```
35 C14 = Cliente.create(nome: 'JOSE'     , endereco: 'QUADRA 3 BL 3'      ,
       cidade: 'BRASILIA'      , CEP: '34875630', UF: 'DF', CNPJ: '
      847365351/0001-01', IE: '4532')
36
37 V01 = Vendedor.create(nome: 'JOSE'    , salario: 1800, faixacomissao: 'C
      ')
38 V02 = Vendedor.create(nome: 'CARLOS'  , salario: 2490, faixacomissao: 'A
      ')
39 V03 = Vendedor.create(nome: 'JOAO'    , salario: 2780, faixacomissao: 'C
      ')
40 V04 = Vendedor.create(nome: 'ANTONIO' , salario: 9500, faixacomissao: 'C
      ')
41 V05 = Vendedor.create(nome: 'FELIPE'  , salario: 4600, faixacomissao: 'A
      ')
42 V06 = Vendedor.create(nome: 'JONAS'   , salario: 2300, faixacomissao: 'A
      ')
43 V07 = Vendedor.create(nome: 'JOAO'    , salario: 2650, faixacomissao: 'C
      ')
44 V08 = Vendedor.create(nome: 'JOSIAS'  , salario:  870, faixacomissao: 'B
      ')
45 V09 = Vendedor.create(nome: 'MAURICIO', salario: 2930, faixacomissao: 'B
      ')
46
47 D01 = Pedido.create(numpedido: 121, prazoentrega: 20, cliente: C07,
      vendedor: V01)
48 D02 = Pedido.create(numpedido:  97, prazoentrega: 20, cliente: C01,
      vendedor: V07)
49 D03 = Pedido.create(numpedido: 101, prazoentrega: 15, cliente: C01,
      vendedor: V07)
50 D04 = Pedido.create(numpedido: 137, prazoentrega: 20, cliente: C01,
      vendedor: V05)
51 D05 = Pedido.create(numpedido: 148, prazoentrega: 20, cliente: C01,
      vendedor: V07)
52 D06 = Pedido.create(numpedido: 189, prazoentrega: 15, cliente: C02,
      vendedor: V06)
53 D07 = Pedido.create(numpedido: 104, prazoentrega: 30, cliente: C03,
      vendedor: V07)
54 D08 = Pedido.create(numpedido: 203, prazoentrega: 30, cliente: C05,
      vendedor: V09)
55 D09 = Pedido.create(numpedido:  98, prazoentrega: 20, cliente: C07,
      vendedor: V01)
56 D10 = Pedido.create(numpedido: 143, prazoentrega: 30, cliente: C08,
      vendedor: V02)
57 D11 = Pedido.create(numpedido: 105, prazoentrega: 15, cliente: C10,
      vendedor: V04)
58 D12 = Pedido.create(numpedido: 111, prazoentrega: 20, cliente: C11,
      vendedor: V04)
59 D13 = Pedido.create(numpedido: 103, prazoentrega: 20, cliente: C11,
      vendedor: V03)
```

```
60 D14 = Pedido.create(numpedido:  91, prazoentrega: 20, cliente: C11,
      vendedor: V03)
61 D15 = Pedido.create(numpedido: 138, prazoentrega: 20, cliente: C11,
      vendedor: V03)
62 D16 = Pedido.create(numpedido: 108, prazoentrega: 15, cliente: C12,
      vendedor: V08)
63 D17 = Pedido.create(numpedido: 119, prazoentrega: 30, cliente: C13,
      vendedor: V09)
64 D18 = Pedido.create(numpedido: 127, prazoentrega: 10, cliente: C07,
      vendedor: V03)
65
66 I01 = Item.create(pedido: D01, produto: P02, quantidade: 10)
67 I02 = Item.create(pedido: D01, produto: P03, quantidade: 35)
68 I03 = Item.create(pedido: D02, produto: P11, quantidade: 20)
69 I04 = Item.create(pedido: D03, produto: P03, quantidade:  9)
70 I05 = Item.create(pedido: D03, produto: P04, quantidade: 18)
71 I06 = Item.create(pedido: D03, produto: P08, quantidade:  5)
72 I07 = Item.create(pedido: D09, produto: P11, quantidade:  5)
73 I08 = Item.create(pedido: D05, produto: P09, quantidade:  8)
74 I09 = Item.create(pedido: D05, produto: P03, quantidade:  7)
75 I10 = Item.create(pedido: D05, produto: P11, quantidade:  3)
76 I11 = Item.create(pedido: D05, produto: P02, quantidade: 10)
77 I12 = Item.create(pedido: D05, produto: P04, quantidade: 30)
78 I13 = Item.create(pedido: D07, produto: P07, quantidade: 32)
79 I14 = Item.create(pedido: D08, produto: P03, quantidade:  6)
80 I15 = Item.create(pedido: D06, produto: P04, quantidade: 45)
81 I16 = Item.create(pedido: D10, produto: P03, quantidade: 20)
82 I17 = Item.create(pedido: D10, produto: P04, quantidade: 10)
83 I18 = Item.create(pedido: D11, produto: P04, quantidade: 10)
84 I19 = Item.create(pedido: D12, produto: P02, quantidade: 10)
85 I20 = Item.create(pedido: D12, produto: P04, quantidade: 70)
86 I21 = Item.create(pedido: D13, produto: P07, quantidade: 37)
87 I22 = Item.create(pedido: D14, produto: P11, quantidade: 40)
88 I23 = Item.create(pedido: D15, produto: P05, quantidade: 10)
89 I24 = Item.create(pedido: D15, produto: P11, quantidade: 35)
90 I25 = Item.create(pedido: D15, produto: P07, quantidade: 18)
91 I26 = Item.create(pedido: D16, produto: P08, quantidade: 17)
92 I27 = Item.create(pedido: D17, produto: P11, quantidade: 40)
93 I28 = Item.create(pedido: D17, produto: P08, quantidade:  6)
94 I29 = Item.create(pedido: D17, produto: P05, quantidade: 10)
95 I30 = Item.create(pedido: D17, produto: P07, quantidade: 43)
96 I31 = Item.create(pedido: D04, produto: P08, quantidade:  8)
```

Vamos agora executar questionamentos ao banco de dados. Para este fim, vamos entrar em modo console executando **rails c**. Uma das operações mais comuns, realizadas sobre um banco de dados é a de examinar (selecionar) as informações armazenadas. Neste item iremos mostrar várias situações de utilização do comando SELECT.

A. Listar todos os produtos com todos os campo.

```
1 Produto.all
```

B. Listar todos os clientes com todos os campo.

```
1 Cliente.all
```

C. Listar todo o conteúdo de vendedor

```
1 Vendedor.all
```

D. Listar os itens do pedido com a quantidade igual a 35.

```
1 Item.where(quantidade: 35)
```

E. Listar os clientes que moram em NITEROI

```
1 Cliente.where(cidade: "NITEROI")
```

F. Listar os clientes e seus respectivos endereços, que morem em São Paulo ou estejam na faixa de CEP entre 30077-000 e 30079-000.

```
1 Cliente.where(CEP: 30077-000 .. 30079-000)
```

G. Listar todos os pedidos que não tenham prazo de entrega igual a 15 dias.

```
1 Pedido.where.not(prazoentrega: 15)
```

H. Listar os produtos que tenham o valor unitário na faixa de R$ $0,32$ até R$ $2,00$.

```
1 Produto.where(valunit: 0.02 .. 2.00)
```

I. Listar todos os produtos que tenham a sua unidade começando por 'K'

```
1  Produto.where("unidade like ?", "k%")
```

J. Listar os vendedores que não começam por 'JO'

```
1 Vendedor.where.not("nome like ?", "jo%")
```

K. Listar os vendedores que são da faixa de comissão A e B

```
1 Vendedor.where(faixacomissao: "A" .. "B")
```

L. Mostrar os clientes que não tenham inscrição estadual

```
Cliente.where("IE is null")
```

M. Mostrar em ordem alfabética a lista de vendedores

```
Vendedor.order("nome desc")
```

N. Listar os clientes, ordenados por estado e cidade de forma descendente

```
-----------------------------------------------------------------------
```

O. Listar a descrição e o valor unitário de todos os produtos que tenham a unidade 'KG', em ordem crescente de valor unitário

```
-----------------------------------------------------------------------
```

Realizando cálculos com informação selecionada, veremos casos utilizando funções sobre conjuntos.

P. Mostrar o novo salário fixo dos vendedores, de faixa de comissão "C", calculando com base no reajuste de 75% acrescido de R$ 120,00 de bonificação. Ordenar pelo nome do vendedor.

```
-----------------------------------------------------------------------
```

Q. Mostrar o menor e o maior salário de vendedor.

```
Vendedor.maximum(:salario)
Vendedor.minimum(:salario)
```

R. Mostrar a quantidade total pedida para o produto "VINHO". Utilize o código abaixo como início, e descubra qual o código para o "VINHO".

```
Item.group(:produto_id).sum(:quantidade)
```

S. Qual a média dos salários fixos dos vendedores?

```
Vendedor.average(:salario)
```

T. Quantos vendedores ganham acima de R$ 2.500,00 de salário fixo?

```
Vendedor.where("salario > 2500").count(:all)
```

Utilizando a cláusula DISTINCT: Normalmente, vários registros dentro de uma tabela podem conter os mesmos valores, com exceção da chave primária. Com isso, muitas consultas podem trazer informações erradas. A cláusula DISTINCT, aplicada em uma consulta, foi criada para não permitir que certas redundâncias, obviamente necessárias, causem problemas.

U. Quais são as unidades de produtos, diferentes, na tabela produto?

```
1 Produto.distinct.select(:unidade)
```

Agrupando informações selecionadas (GROUP BY): Utilizando a cláusula GROUP BY, é possível organizar a seleção de dados em grupos determinados.

V. listar o número de produtos que cada pedido contém.

```
1 Item.group(:pedido_id).count()
```

Inicialmente, os registros são ordenados de forma ascendente por número do pedido. Num segundo passo, é aplicada a operação COUNT(*) para cada grupo de registros que tenha o mesmo número de pedido. Após a operação de contagem de cada grupo, o resultado da consulta utilizando a cláusula GROUP BY é apresentado. Geralmente, a cláusula GROUP BY é utilizada em conjunto com as operações COUNT e AVG.

X. Listar os pedidos que têm mais do que 3 produtos

```
1 Item.group(:pedido_id).having("count() > 3").select(:pedido_id)
```

Utilizando consultas encadeadas (Subqueries): O que é uma subquery? Em linhas gerais, é quando o resultado de uma consulta é utilizado por outra consulta, de forma encadeada e contido no mesmo comando SQL.

Y. Que produtos participam em qualquer pedido cuja quantidade seja 10?

```
1 Produto.where(id:  Item.where('Quantidade = ?', 10))
```

Z. Quais os vendedores que ganham um salário fixo abaixo da média?

```
1 ----------------------------------------------------------------------
```

RECUPERANDO DADOS DE VÁRIAS TABELAS (JOINS). Até agora viemos trabalhando com a recuperação de dados sobre uma única tabela, mas o conceito de banco de dados reúne, evidentemente, várias tabelas diferentes. Para que possamos recuperar informações de um banco de dados temos, muitas vezes, a necessidade de acessar simultaneamente várias tabelas. Algumas dessas consultas necessitam realizar uma junção (JOIN) entre tabelas, para desta poder extrair as informações necessárias para a consulta formulada.

AA. Quais os clientes que têm prazo de entrega superior a 15 dias e que pertencem aos estados de São Paulo (SP) ou Rio de Janeiro (RJ)?

```
1 Pedido.joins(:cliente).where('prazoentrega > 15 and (UF = "RJ" or UF = "
    SP")')
```

AB. Mostrar os clientes e seus respectivos prazos de entrega, ordenados do maior para o menor.

```
1 ----------------------------------------------------------------------
```

AC. Apresentar os vendedores (ordenados) que emitiram pedidos com prazos de entrega superiores a 15 dias e que tenham salários fixos igual ou superiores a R$ 1.000, 00.

```
1 ----------------------------------------------------------------------
```

AD. Mostre os clientes (ordenados) que têm prazo de entrega maior que 15 dias para o produto "QUEIJO" e que sejam do Rio de Janeiro.

```
1 ----------------------------------------------------------------------
```

AE. Mostre todos os vendedores que venderam chocolate em quantidade superior a 10 Kg

```
1 ----------------------------------------------------------------------
```

AF. Quantos clientes fizeram pedido com o vendedor João?

```
1 ----------------------------------------------------------------------
```

AG. Quantos clientes da cidade do Rio de Janeiro, e Niterói tiveram os seus pedidos tirados com o vendedor João?

```
1 ----------------------------------------------------------------------
```

AH. Quais os produtos que não estão presentes em nenhum pedido?

```
1 ----------------------------------------------------------------------
```