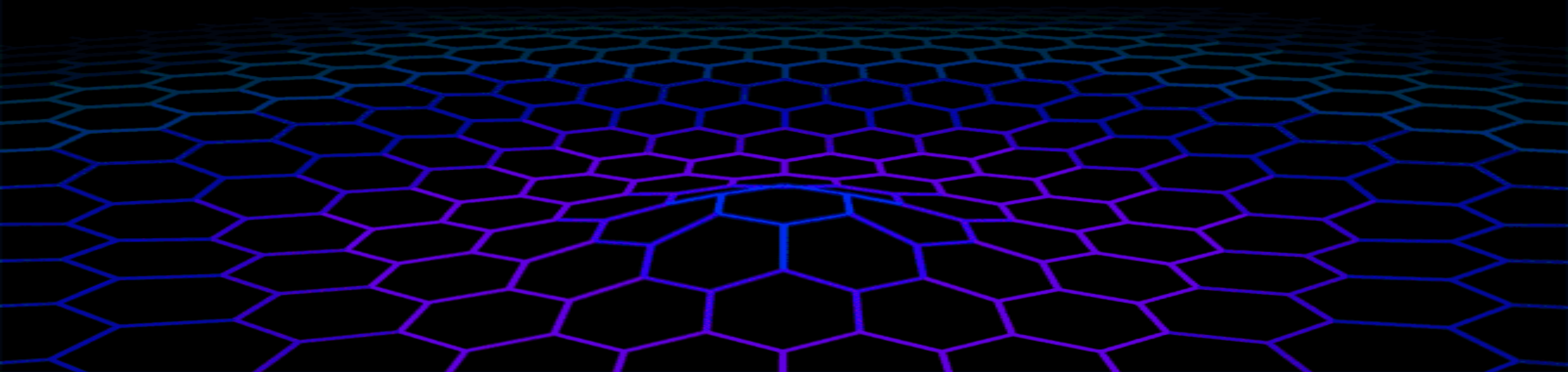
Faculdades senac

Banco de Dados II



Banco de Dados 2

REVISÃO (a saga continua)

SQL


SQL


PostgreSQL

SQL


Microsoft
SQL Server


ORACLE
PL/SQL


MySQL

Install

Linux – (no Terminal)

```
sudo su (super usuário)

apt update (iniciar a atualização de pacotes)

apt install mysql-server (instalar o MySQL)

mysql --version (verificar se o MySQL foi instalado)

service mysql start (iniciar o serviço MySQL)

mysql -u root -p (acessar o MySQL)
```

Windows – (no PowerShell)

```
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor
3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://community.c
hocolatey.org/install.ps1’))(instalar o Chocolatey)

choco install mysql (instalar o MySQL)

mysql --version (verificar se o MySQL foi instalado)

mysql –u root –p (acessar o MySQL)
```

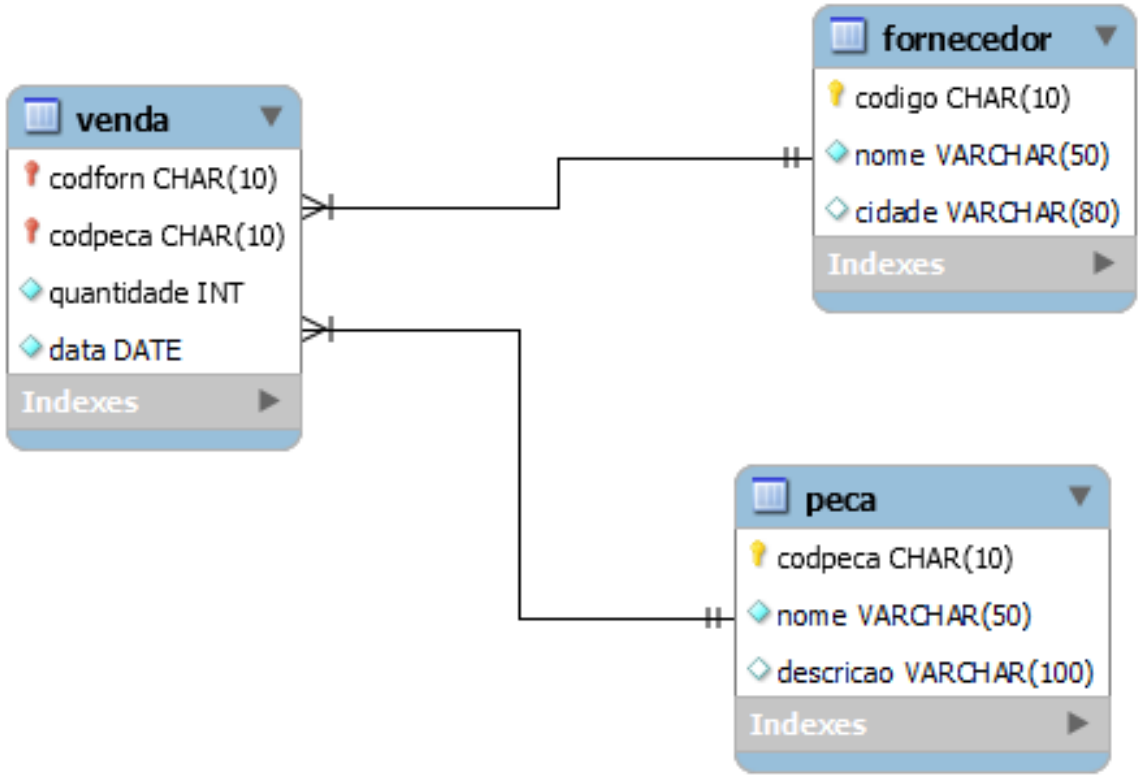
Aquecimento – Exercício 1

Exercício

Defina as tabelas abaixo usando SQL

- **FORNECEDOR** (**codigo**, nome, cidade)
- **VENDA** (**codForn**, **codPeca**, quantidade, dataVenda)
- **PECA** (**codPeca**, nome, descricao)

Aula02


venda
codforn CHAR(10)
codpeca CHAR(10)
quantidade INT
data DATE
Indexes
fornecedor
codigo CHAR(10)
nome VARCHAR(50)
cidade VARCHAR(80)
Indexes
peca
codpeca CHAR(10)
nome VARCHAR(50)
descricao VARCHAR(100)
Indexes

Aula02

```sql
DROP DATABASE IF EXISTS aula02;

CREATE DATABASE IF NOT EXISTS aula02;

USE aula02;
```

Fornecedor

```sql
CREATE TABLE fornecedor
  (
     codigo CHAR(10),
     nome   VARCHAR(50) NOT NULL,
     cidade VARCHAR(80),
     PRIMARY KEY(codigo)
  );
```

```sql
CREATE TABLE peca
  (
     codpeca   CHAR(10),
     nome      VARCHAR(50) NOT NULL,
     descricao VARCHAR(100),
     PRIMARY KEY (codpeca)
  );
```

```sql
CREATE TABLE venda
  (
     codforn    CHAR(10),
     codpeca    CHAR(10),
     quantidade INT NOT NULL,
     data       DATE NOT NULL,
     PRIMARY KEY (codforn, codpeca),
     FOREIGN KEY (codforn) REFERENCES fornecedor(codigo) ON DELETE RESTRICT ON UPDATE CASCADE,
     FOREIGN KEY (codpeca) REFERENCES peca(codpeca) ON DELETE RESTRICT ON UPDATE CASCADE
  );
```

Aquecimento – Exercício 2

Empregado

```sql
CREATE TABLE empregado
  (
     matricula  CHAR(9),
     nome       VARCHAR(15) NOT NULL,
     datanasc   DATE,
     endereco   VARCHAR(30),
     sexo       CHAR(1),
     salario    NUMERIC(10, 2),
     supervisor CHAR(9),
     depto      INT(11) NOT NULL
  );
```

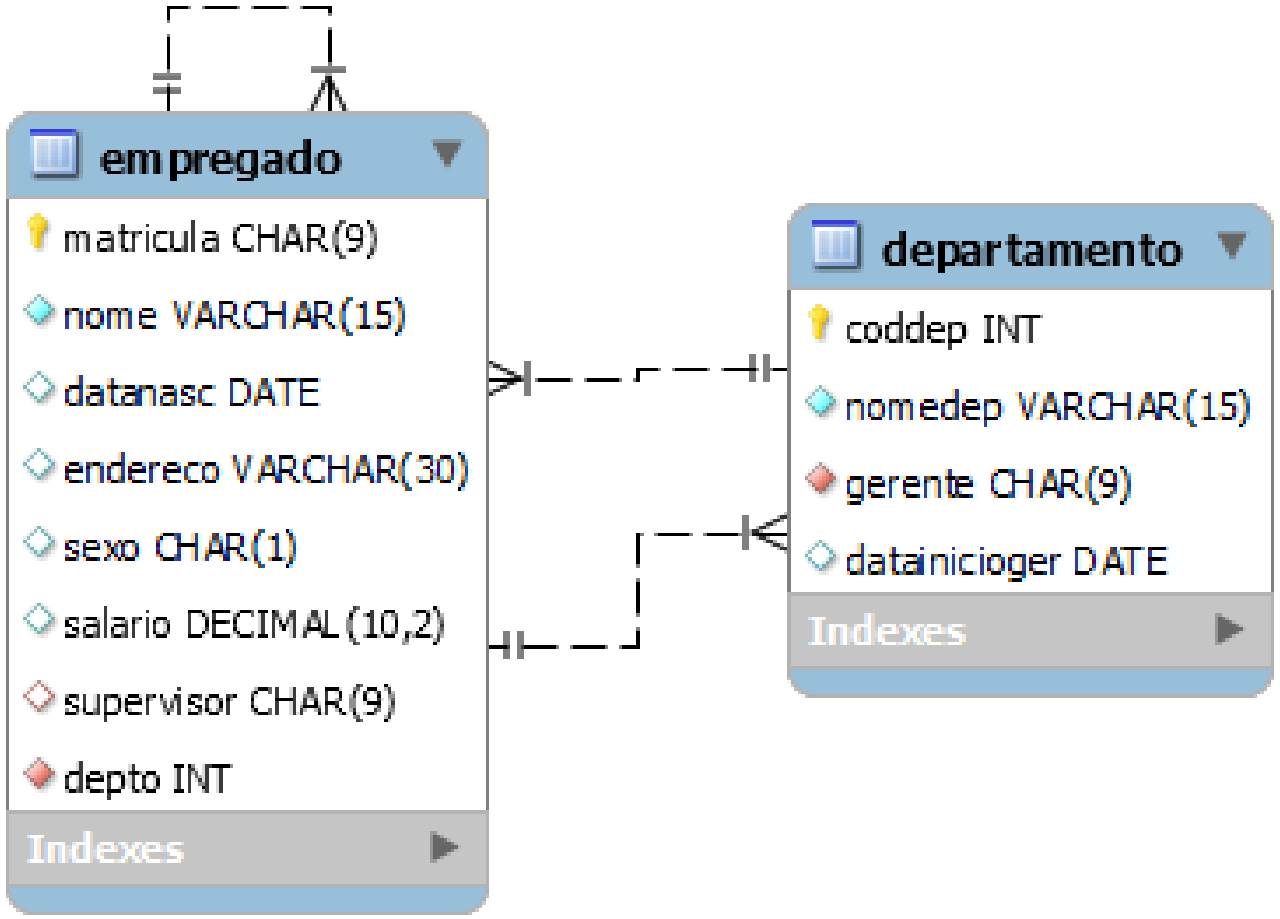
Departamento

```sql
CREATE TABLE departamento
  (
     coddep          INT(11),
     nomedep         VARCHAR(15) NOT NULL,
     gerente         CHAR(9) NOT NULL,
     datainicioger DATE
  );
```

Empregado

```
ALTER TABLE empregado
  ADD CONSTRAINT emppk PRIMARY KEY (matricula);

ALTER TABLE departamento
  ADD CONSTRAINT deppk PRIMARY KEY(coddep);

ALTER TABLE empregado
  ADD CONSTRAINT empsuperfk FOREIGN KEY (supervisor) REFERENCES empregado (
  matricula) ON DELETE RESTRICT ON UPDATE CASCADE;

ALTER TABLE empregado
  ADD CONSTRAINT empdep FOREIGN KEY (depto) REFERENCES departamento(coddep);

ALTER TABLE departamento
  ADD CONSTRAINT depnome UNIQUE (nomedep);

ALTER TABLE departamento
  ADD CONSTRAINT depger FOREIGN KEY (gerente) REFERENCES empregado(matricula);
```

Aula02


empregado
matricula CHAR(9)
nome VARCHAR(15)
datanasc DATE
endereco VARCHAR(30)
sexo CHAR(1)
salario DECIMAL(10,2)
supervisor CHAR(9)
depto INT
Indexes
departamento
coddep INT
nomedep VARCHAR(15)
gerente CHAR(9)
datainicioger DATE
Indexes

Índice

Comandos SQL - DDL

Especificando índices em SQL

- SQL possui comandos para criar e remover índices em atributos de relações base (faz parte da SQL DDL)

- Um índice é uma estrutura de acesso físico que é especificado em um ou mais atributos de um arquivo, permitindo um acesso mais eficiente aos dados

- Se os atributos usados nas condições de seleção e junção de uma query são indexados, o tempo de execução da query é melhorado

Comandos SQL - DDL

Ex.: Criar um índice no atributo nome da relação Empregado

```
CREATE INDEX nomeEmpIndex ON Empregado(nome)
```

O default é ordem ascendente, se quisermos uma ordem descendente adicionamos a palavra chave DESC depois do nome do atributo

Comandos SQL - DDL

Para especificar a restrição de chave usamos a palavra UNIQUE

```
CREATE UNIQUE INDEX matrEmpIndex ON Empregado (matricula)
```

Para eliminarmos um índice usamos o comando DROP

- Ex. `DROP INDEX nome-indice`

Se desejar acompanhar os próximos exemplos...

Se desejar acompanhar os próximos exemplos...

```sql
CREATE TABLE teste
  (
     codigo INT,
     nome   CHAR(15),
     email  VARCHAR(30),
     PRIMARY KEY(codigo)
  );
```

Alteração de tabelas (modificação da estrutura de tabelas)

Quando notamos que as necessidades da aplicação mudaram ou que foi cometido um erro, podemos modificar a estrutura das tabelas já criadas.

Podemos incluir ou excluir colunas, restrições, modificar nome de coluna ou da própria tabela.

Tudo isso pode ser feito através do comando **`ALTER TABLE`**

Alteração de tabelas (modificação da estrutura de tabelas)

ADD <campo> <tipo>

- Insere novo campo

DROP <campo>

- Remove determinado campo

MODIFY <campo><tipo>

- Modifica o tipo de determinado campo

Alteração de tabelas (modificação da estrutura de tabelas)

Inserir na tabela teste o campo endereço após o campo nome.

```
ALTER TABLE teste
     ADD endereco CHAR(50)
     AFTER nome;
```

Observe as alterações com `DESCRIBE teste;`

Obs. Para inserir antes de todos os outros campos use **`FIRST`**

Alteração de tabelas (modificação da estrutura de tabelas)

Inserir na tabela teste o campo nascimento que conterá a data de nascimento dos cadastrados.

```
ALTER TABLE teste ADD nascimento DATE;
```

A nova coluna não pode possuir a restrição de não-nulo, porque a coluna inicialmente deve conter valores nulos. Porém, a restrição de não-nulo pode ser adicionada posteriormente.

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

O campo email foi criado com limite de 30 caracteres. Observe isso com o comando describe teste;

Trocar para 40 caracteres .

```
ALTER TABLE teste MODIFY email CHAR(40);
```

Execute o `DESCRIBE teste;` novamente para observar a alteração.

Alteração de tabelas (modificação da estrutura de tabelas)

Trocar no nome da coluna email por e_mail .

```
ALTER TABLE teste CHANGE email e_mail CHAR(30);
```

Execute o `DESCRIBE teste;` para observar a alteração.

Alteração de tabelas (modificação da estrutura de tabelas)

Abaixo vemos como excluir o campo codigo da tabela Teste:

```
ALTER TABLE teste DROP codigo;
```

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

Definir o campo **nome** como chave da tabela Teste:

```
ALTER TABLE teste ADD PRIMARY KEY (nome);
```

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

Excluir a chave primária, mas não a coluna

```
ALTER TABLE teste DROP PRIMARY KEY;
```

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

Alterar o nome da tabela teste para teste2

```
ALTER TABLE teste RENAME TO teste2;
```

Observe a alteração com **SHOW TABLES;**

Alteração de tabelas (modificação da estrutura de tabelas)

Excluir a tabela. Todos os dados e definições da tabela são removidos.

```
DROP TABLE teste2;
```

Observe a alteração com **SHOW TABLES;**

Atividade

ATIVIDADE - Crie a modelagem lógica, a física e insira dados:

EMPREGADO (matricula, nome, endereco, salario, matriculaBoss, codigoDepartamento)

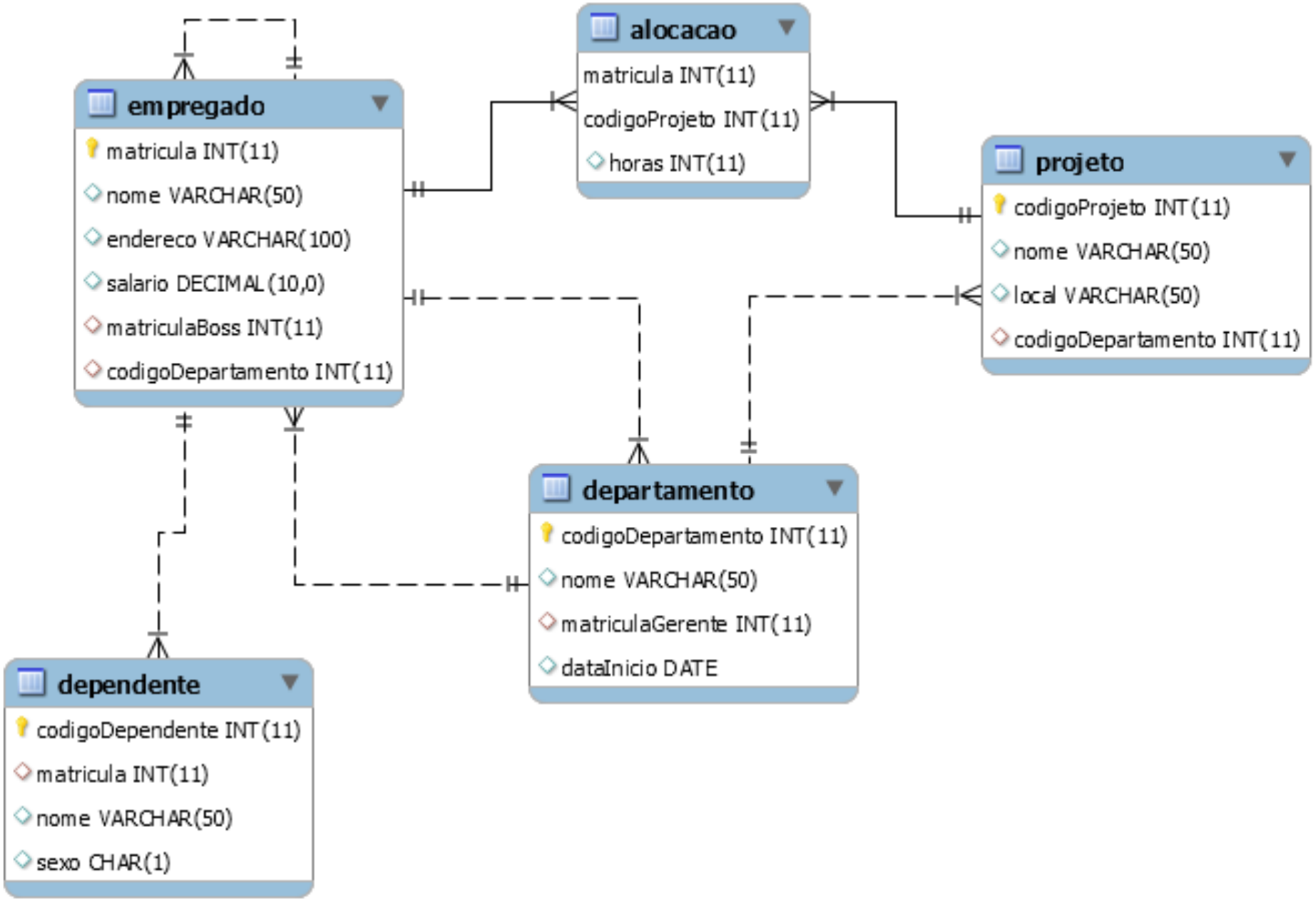
DEPARTAMENTO (codigoDepartamento, nome, matriculaGerente, dataInicio)

PROJETO (codigoProjeto, nome, local, codigoDepartamento)

ALOCACAO (matricula, codigoProjeto, horas)

DEPENDENTE (codigoDependente, matricula, nome, sexo)

Modelo lógico



Modelo físico

```sql
-- Aula02
DROP DATABASE IF EXISTS aula02;
CREATE DATABASE IF NOT EXISTS aula02;
USE aula02;

-- Criar tabela EMPREGADO (matricula, nome, endereco, salario, matriculaBoss, codigoDepartamento)
CREATE TABLE empregado (
  matricula INT,
  nome VARCHAR(50),
  endereco VARCHAR(100),
  salario DECIMAL(10.2),
  matriculaBoss INT,
  codigoDepartamento INT
) ENGINE = INNODB;

-- Criar tabela DEPARTAMENTO (codigoDepartamento, nome, matriculaGerente, dataInicio)
CREATE TABLE departamento (
  codigoDepartamento INT,
  nome VARCHAR(50),
  matriculaGerente INT,
  dataInicio DATE
) ENGINE = INNODB;
```

Modelo físico

```sql
-- Criar tabela DEPARTAMENTO (codigoDepartamento, nome, matriculaGerente, dataInicio)
CREATE TABLE departamento (
  codigoDepartamento INT,
  nome VARCHAR(50),
  matriculaGerente INT,
  dataInicio DATE
) ENGINE = INNODB;

-- Criar tabela PROJETO (codigoProjeto, nome, local, codigoDepartamento)
CREATE TABLE projeto (
  codigoProjeto INT,
  nome VARCHAR(50),
  local VARCHAR(50),
  codigoDepartamento INT
) ENGINE = INNODB;

-- Criar tabela ALOCACAO (matricula, codigoProjeto, horas)
CREATE TABLE alocacao (
  matricula INT,
  codigoProjeto INT,
  horas INT
) ENGINE = INNODB;

-- Criar tabela DEPENDENTE (codigoDependente, matricula, nome, sexo)
CREATE TABLE dependente (
  codigoDependente INT,
  matricula INT,
  nome VARCHAR(50),
  sexo CHAR(1)
) ENGINE = INNODB;
```

Modelo físico

```
47  -- Adicionar chaves primárias
48
49  /* Chave primária da tabela EMPREGADO */
50  ALTER TABLE empregado
51      ADD CONSTRAINT empPK
52      PRIMARY KEY (matricula);
53
54  /* Chave primária da tabela DEPARTAMENTO */
55  ALTER TABLE departamento
56      ADD CONSTRAINT depPK
57      PRIMARY KEY (codigoDepartamento);
58
59  /* Chave primária da tabela PROJETO */
60  ALTER TABLE projeto
61      ADD CONSTRAINT projPK
62      PRIMARY KEY (codigoProjeto);
63
64  /* Chave primária da tabela ALOCACAO
65  (é uma chave composta por dois campos: matricula e codigoProjeto */
66  ALTER TABLE alocacao
67      ADD CONSTRAINT alocPK
68      PRIMARY KEY (matricula, codigoProjeto);
69
70  /* Chave primária da tabela DEPENDENTE */
71  ALTER TABLE DEPENDENTE
72      ADD CONSTRAINT depPK
73      PRIMARY KEY (codigoDependente);
```

Modelo físico

```
75  -- Adicionar constraints da tabela EMPREGADO
76
77  /* Autorelacionamento entre o campo matriculaBoss e matricula */
78  ALTER TABLE empregado
79      ADD CONSTRAINT empBossFK
80      FOREIGN KEY (matriculaBoss) REFERENCES empregado (matricula);
81
82  /* Relacionamento entre as tabelas EMPREGADO e DEPARTAMENTO
83     para indicar a qual departamento o empregado pertence */
84  ALTER TABLE empregado
85      ADD CONSTRAINT empDepFK
86      FOREIGN KEY (codigoDepartamento) REFERENCES departamento (codigoDepartamento);
87
88  -- Adicionar constraints da tabela DEPARTAMENTO
89
90  /* Relacionamento entre as tabelas DEPARTAMENTO e EMPREGADO
91     para indicar qual empregado gerencia o departamento */
92  ALTER TABLE departamento
93      ADD CONSTRAINT depGerFK
94      FOREIGN KEY (matriculaGerente) REFERENCES empregado (matricula);
95
```

Modelo físico

```
96   -- Adicionar constraints da tabela PROJETO
97
98   /* Relacionamento entre as tabelas PROJETO e DEPARTAMENTO
99      para indicar a qual departamento o projeto pertence */
100  ALTER TABLE projeto
101      ADD CONSTRAINT projDepFK
102      FOREIGN KEY (codigoDepartamento) REFERENCES departamento (codigoDepartamento);
103
104  -- Adicionar constraints da tabela ALOCACAO
105
106  /* Relacionamento entre as tabelas ALOCACAO e EMPREGADO
107     para indicar qual empregado está alocado no projeto */
108  ALTER TABLE alocacao
109      ADD CONSTRAINT alocEmpFK
110      FOREIGN KEY (matricula) REFERENCES empregado (matricula);
111
112  /* Relacionamento entre as tabelas ALOCACAO e PROJETO
113     para indicar qual PROJETO faz parte da alocação */
114  ALTER TABLE alocacao
115      ADD CONSTRAINT alocProjFK
116      FOREIGN KEY (codigoProjeto) REFERENCES PROJETO (codigoProjeto);
117
```

Modelo físico

```
118   -- Adicionar constraints da tabela DEPENDENTE
119
120   /* Relacionamento entre as tabelas DEPENDENTE e EMPREGADO
121      para indicar a qual empregado o dependente pertence */
122   ALTER TABLE DEPENDENTE
123       ADD CONSTRAINT depEmpFK
124       FOREIGN KEY (matricula) REFERENCES empregado (matricula);
125
```

Modelo físico

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_aula02 |
+------------------+
| alocacao         |
| departamento     |
| dependente       |
| empregado        |
| projeto          |
+------------------+
5 rows in set (0.00 sec)
```