ESCOLA E FACULDADE DE TECNOLOGIA SENAI ROBERTO MANGE

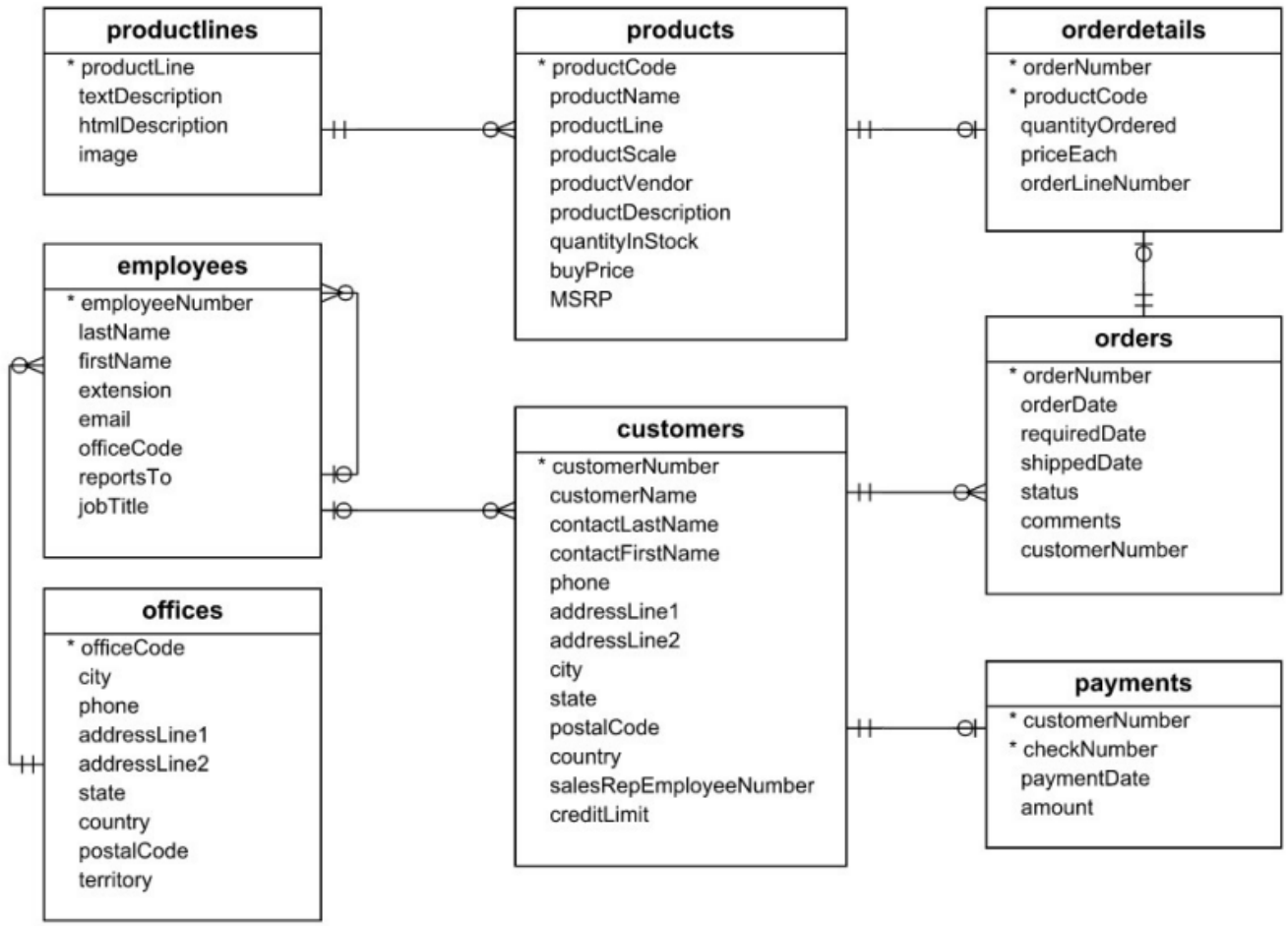
DESENVOLVIMENTO DE SISTEMAS

JULHO DE 2022 — SENAI-SP

SENAI SÃO PAULO

SQL – STRUCTURED QUERY LANGUAGE

CORREÇÃO EXERCÍCIO 1 - WAREHOUSE



- Criar tabelas conforme diagrama;
- Inserir dados fictícios em todas as tabelas;
- Realizar o select com inner join;
- Trabalhe com o git;

DESAFIO:

- Após adicionar os dados fictícios, faça um select na tabela orders buscando por um produto específico
- Após adicionar os dados fictícios, faça um select na tabela customers para exibir apenas os clientes com mais de uma ordem

SENAI SÃO PAULO

CORREÇÃO EXERCÍCIO 1 - WAREHOUSE

```sql
create database warehouse;

create table productLines(
    id bigint not null auto_increment,
    textDescription varchar(200) not null,
    htmlDescription varchar(200) not null,
    image varchar(1000),
    primary key(id)
);

insert into productLines
(textDescription,htmlDescription,image) values
("Limpeza", "<h1>Limpeza</h1>", "http://google.com"),
("Frutas", "<h1>Frutas</h1>", "http://google.com");
```

```sql
create table products(
    id bigint not null auto_increment,
    name varchar(200) not null,
    productLineFK bigint not null,
    scale float,
    vendor varchar(200) not null,
    description varchar(200) not null,
    quantity float not null,
    price float not null,
    msrp float,
    primary key(id),
    foreign key(productLineFK) references productLines(id)
);

insert into products
(name, productLineFK, scale, vendor, description, quantity, price, msrp)
values
("Sabonete",1,1.2,"Dove","Lavanda",150,2.75,2);
```

CORREÇÃO EXERCÍCIO 1 - WAREHOUSE

```sql
create table office(
    id bigint not null auto_increment,
    city varchar(150) not null,
    phone varchar(15) not null,
    addressLine1 varchar(150) not null,
    addressLine2 varchar(150),
    state varchar(100) not null,
    country varchar(100) not null,
    postalCode varchar(100),
    primary key(id)
);

insert into office
(city, phone, addressLine1,state,country)
values
("Americana", "5548884455", "Rua Santo Antônio, 235",
"SP", "Brasil");
```

```sql
create table employee(
    id bigint not null auto_increment,
    lastName varchar(100) not null,
    firstName varchar(100) not null,
    email varchar(150) not null,
    jobTitle varchar(150),
    officeFK bigint not null,
    primary key(id),
    foreign key(officeFK) references office(id)
);

insert into employee
(lastName, firstName, email, jobTitle, officeFK)
values
("Castanho", "Joana", "joana@castanho", "Promotora", 1);
```

CORREÇÃO EXERCÍCIO 1 - WAREHOUSE

```sql
create table customers(
    id bigint not null auto_increment,
    name varchar(150) not null,
    contactLastName varchar(100) not null,
    contactFirstName varchar(100) not null,
    phone varchar(15) not null,
    addressLine1 varchar(150) not null,
    addressLine2 varchar(150),
    city varchar(150) not null,
    state varchar(100) not null,
    country varchar(100) not null,
    creditLimit float not null,
    employeeFK bigint not null,
    primary key(id),
    foreign key(employeeFK) references employee(id)
);
```

```sql
create table payments(
    id bigint not null auto_increment,
    customerFK bigint not null,
    paymentDate datetime not null default now(),
    amount float,
    primary key(id),
    foreign key(customerFK) references customers(id)
);

create table orders(
    id bigint not null auto_increment,
    orderDate datetime not null default now(),
    requiredDate datetime,
    shippedDate datetime,
    status varchar(1) not null default 'P',
    comments varchar(200),
    customerFk bigint not null,
    primary key(id),
    foreign key(customerFK) references customers(id)
);
```

SENAI SÃO PAULO

CORREÇÃO EXERCÍCIO 1 - WAREHOUSE

```
create table orderDetails(
    id bigint not null auto_increment,
    orderFK bigint not null,
    productFK bigint not null,
    quantityOrdered float not null,
    price float not null,
    primary key(id),
    foreign key(orderFK) references orders(id),
    foreign key(productFK) references products(id)
);
```

- Realizar o select com inner join:

```
select *from customers c
join employee e on c.employeeFK = e.id
join office o on o.id = e.officeFK
join payments p on p.customerFK = c.id
join orders od on od.customerFk = c.id
join orderdetails odt on odt.orderFK = od.id
join products pr on pr.id = odt.productFK
join productlines pl on pl.id = pr.productLineFK;
```

DESAFIO:

- Após adicionar os dados fictícios, faça um select na tabela orders buscando por um produto específico:

```
select *from orders od
join orderdetails odt on odt.orderFK = od.id
join products pr on pr.id = odt.productFK
where pr.name = 'Sabonete';
```

- Após adicionar os dados fictícios, faça um select na tabela customers para exibir apenas os clientes com mais de uma ordem

```
select c.name, count(*) as total
from customers c
join orders od on od.customerFk = c.id
group by c.name
having total > 10
order by c.name asc;
```

SENAI SÃO PAULO

SQL – STRUCTURED QUERY LANGUAGE

CORREÇÃO EXERCÍCIO 2 - COMPANY

- Crie uma aplicação para gerenciar empregados de uma empresa.
- Nesta empresa deve-se armazenar os funcionários com sua data de nascimento, cpf, nome, gênero (permitido M ou F), além data de contratação.
- Atente-se que cada funcionário pode assumir diferentes cargos ao longo de sua jornada profissional, sendo necessário este registro ao longo da vida profissional do funcionário bem como qual cargo assumiu e por qual período;
- É necessário registrar o salário bruto do funcionário, compreendendo que o mesmo também terá inúmeros salários ao longo de sua permanência na empresa e é necessário o histórico disso bem como cada período;
- Nesta empresa as áreas são divididas em departamentos, onde são associados quais os empregados de cada departamento;
- Considere que como um funcionário pode ter diferentes cargos ao longo de sua vida na empresa, o mesmo pode passar por diferentes departamentos e é necessário que se mantenha este histórico e por qual período;
- Cada departamento tem um gerente (que também não deixa de ser um empregado) mas também considere que um departamento pode passar por diferentes gerentes e deve-se obter um histórico disso bem como qual período;
- Crie as tabelas, insira os dados fictícios e faça as consultas seguintes (próximo slide):

SQL – STRUCTURED QUERY LANGUAGE

CORREÇÃO EXERCÍCIO 2 - COMPANY

- Busque os funcionários com salários maior que 5000;
- Busque o funcionário com maior salário;
- Busque todos os funcionários que foram de um respectivo gerente;
- Busque todos os cargos ocupados por um respectivo funcionário;
- Busque todos os gerentes que passaram por um determinado departamento;
- Selecionar as tabelas com join;

- **DESAFIO:**
- Faça uma busca para mostrar todos os valores de salários já cadastrados sem repetição de valores no resultado de busca;
- Encontre quais os gerentes que possuem atualmente mais que 5 funcionários sob seu comando;

CORREÇÃO EXERCÍCIO 2 - COMPANY

```sql
create database company_ex2;
use company_ex2;

create table jobs(
    id bigint not null auto_increment,
    name varchar(150) not null,
    primary key(id)
);

insert into jobs(name)
values
("Dev Frontend Pl"), ("Dev Frontend Sr"),
("Dev Backend Jr"), ("Dev Backend Pl"),
("Dev Backend Sr");

create table employees(
    id bigint not null auto_increment,
    birthDate date not null,
    cpf varchar(11) not null,
    name varchar(150) not null,
    gender enum('M','F') not null,
    hiredDate date not null,
    primary key(id)
);

insert into employees (birthDate, cpf, name, gender, hiredDate)
values
("2000-01-01", "1234567889", "Joãozinho", "M", "2019-01-02"),
("1995-02-01", "1234233889", "Maria", "F", "2018-02-02"),
("2001-04-01", "1234233889", "Larissa", "F", "2018-12-02"),
("2002-03-01", "1234233855", "Matheus", "M", "2022-12-02");

create table job_employee(
    id bigint not null auto_increment,
    employeeFK bigint not null,
    jobFK bigint not null,
    startDate date not null,
    endDate date,
    primary key(id),
    foreign key(employeeFK) references employees(id),
    foreign key(jobFK) references jobs(id)
);
```

SENAI SÃO PAULO

CORREÇÃO EXERCÍCIO 2 - COMPANY

```
insert into job_employee (employeeFK, jobFK,
startDate, endDate)
values
(1, 1, "2020-01-01", null),
(2, 1, "2021-01-01", "2022-02-02"),
(2, 2, "2022-02-03", null),
(3, 2, "2002-02-03", "2012-01-01"),
(3, 3, "2012-01-02", null);
```

```
create table salary(
    id bigint not null auto_increment,
    employeeFK bigint not null,
    amount float not null,
    startDate date not null,
    endDate date,
    primary key(id),
    foreign key(employeeFK) references employees(id)
);
```

```
insert into salary (employeeFK, amount, startDate, endDate)
values
(1, 2500, "2022-01-01", "2022-05-05"),
(1, 2900, "2022-05-06", null),
(2, 9000, "2002-05-06", null),
(3, 4500, "2020-01-01", null),
(4, 10500, "2023-01-01", null);
```

```
create table departments(
    id bigint not null auto_increment,
    name varchar(150) not null,
    primary key(id)
);
```

```
insert into departments (name)
values ("Suporte"), ("Desenvolv. Web"),
("Desenvolv. Mobile");
```

SENAI SÃO PAULO

CORREÇÃO EXERCÍCIO 2 - COMPANY

```
create table department_employee(
    id bigint not null auto_increment,
    departmentFK bigint not null,
    employeeFK bigint not null,
    startDate date not null,
    endDate date,
    primary key(id),
    foreign key(departmentFK) references departments(id),
    foreign key(employeeFK) references employees(id)
);

insert into department_employee
(departmentFK, employeeFK, startDate, endDate)
values (1, 1, '2022-01-01', null),
(1, 2, '2002-01-01', '2021-01-01'),
(1, 2, '2021-01-02', null),
(1, 3, '2020-01-01', null);
```

```
create table managers(
    id bigint not null auto_increment,
    employeeFK bigint not null,
    departmentFK bigint not null,
    startDate date not null,
    endDate date,
    primary key(id),
    foreign key(employeeFK) references employees(id),
    foreign key(departmentFK) references departments(id)
);

insert into managers (employeeFK, departmentFK,
startDate, endDate)
values(5,2,'2015-01-01',null);
```

CORREÇÃO EXERCÍCIO 2 - COMPANY

- Busque os funcionários com salários maior que 5000;

```
select *from employees e
join salary s on e.id = s.employeeFK
where s.amount > 5000;
```

- Busque o funcionário com maior salário;

```
select * from employees e
join salary s on e.id = s.employeeFK
where s.amount in (
    select max(amount) from salary
);
```

- Busque todos os funcionários que foram de um respectivo gerente;

```
select *from employees e
join department_employee de on de.employeeFK = e.id
join departments d on d.id = de.departmentFK
join managers m on m.departmentFK = d.id
where m.employeeFK = 4;
```

- Busque todos os cargos ocupados por um respectivo funcionário;

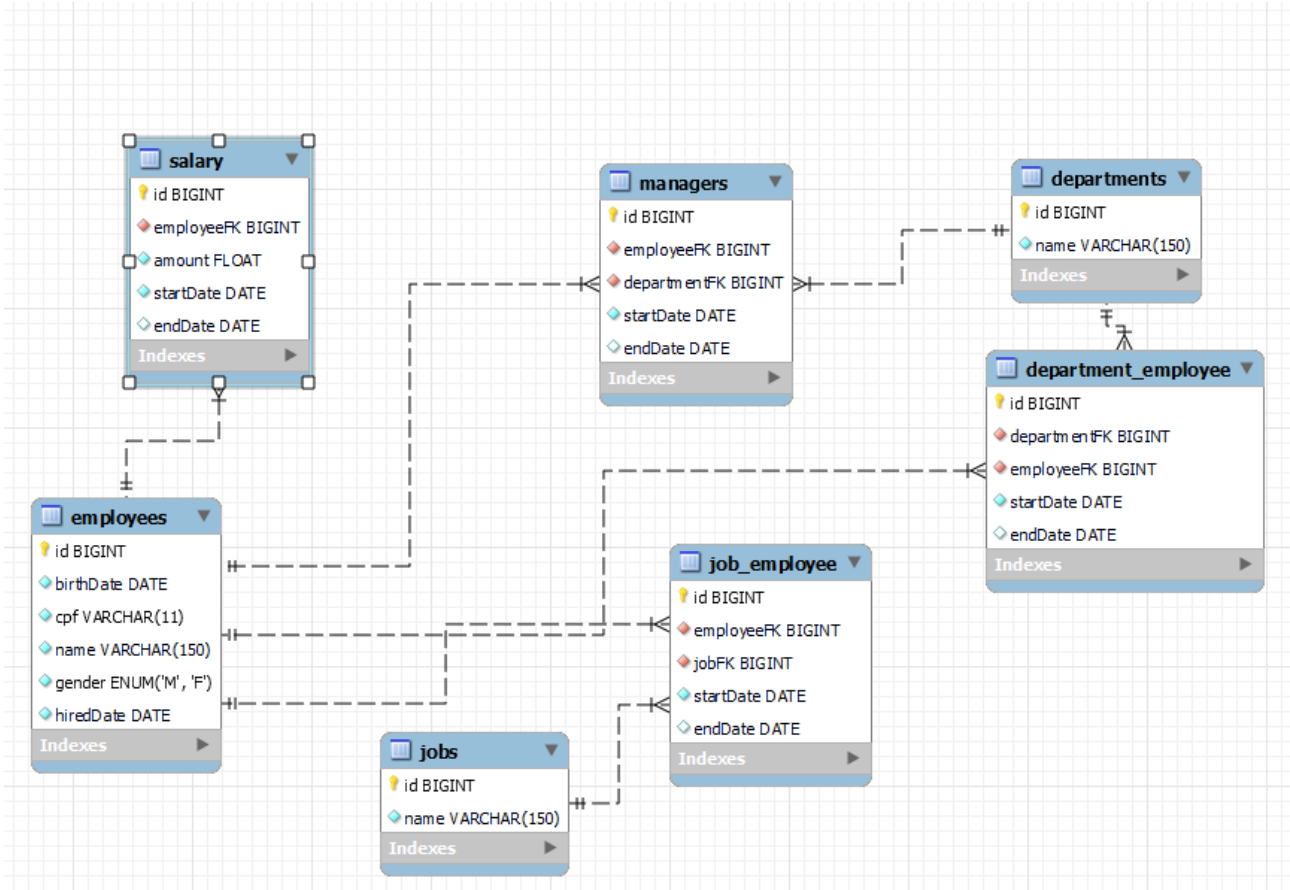
```
select *from job_employee je
join employees e on e.id = je.employeeFK
join jobs j on j.id = je.jobFK
where e.name = "Maria";
```

CORREÇÃO EXERCÍCIO 2 - COMPANY

- Busque todos os gerentes que passaram por um determinado departamento;

```
select *from managers m
join departments d on d.id = m.departmentFK
where d.name = "Suporte";
```

CORREÇÃO EXERCÍCIO 2 - COMPANY



SENAI SÃO PAULO

SQL – STRUCTURED QUERY LANGUAGE

CORREÇÃO EXERCÍCIO 3 – E-MAIL

- Crie uma aplicação que basicamente é uma provedora de e-mail;
- Cada e-mail possuí obrigatóriamente um assunto, pode conter ou não um corpo de texto do e-mail e pode conter ou não um ou mais anexos que devem ser salvos via links pois serão salvos via serviço de storage apropriado;
- Cada anexo relatado tem portanto um link de acesso, o nome do arquivo e também o tamanho do arquivo;
- Neste provedor de e-mail somente é possível enviar como destinatários usuários já existentes, compreendendo que pode-se existir mais de um destinatário;
- Além disso, é necessário salvar a data e hora de envio deste e-mail e qual o status do mesmo no servidor, isto é, se está em rascunho, enviando ou enviado;
- Criar as tabelas, os relacionamentos e inserir dados fictícios;
- Criar consultas:

- Procurar todos os e-mails não enviados na data de ontem;
- Procurar todos os e-mails com mais de 4 destinatários que não contenham anexo;
- Contar todos os e-mails com status enviando no servidor;
- Selecionar todas as tabelas com join;

CORREÇÃO EXERCÍCIO 3 – E-MAIL

```sql
create database emailServer;
use emailServer;


create table users (
    id bigint not null auto_increment,
    name varchar(150) not null,
    emailAddress varchar(100) not null,
    primary key(id)
);

create table email (
    id bigint not null auto_increment,
    subject varchar(150) not null,
    body varchar(5000),
    sentDate datetime not null default now(),
    status enum ('Draft', 'Sending', 'Sent') not null,
    userFK bigint not null,
    primary key(id),
    foreign key(userFK) references users(id)
);
```

```sql
create table attachment(
    id bigint not null auto_increment,
    emailFK bigint not null,
    link varchar(1000) not null,
    fileName varchar(200) not null,
    size long not null,
    primary key(id),
    foreign key(emailFK) references email(id)
);

create table users_destination(
    id bigint not null auto_increment,
    emailFK bigint not null,
    userFK bigint not null,
    primary key(id),
    foreign key(emailFK) references email(id),
    foreign key(userFK) references users(id)
);
```

SENAI SÃO PAULO

CORREÇÃO EXERCÍCIO 3 – E-MAIL