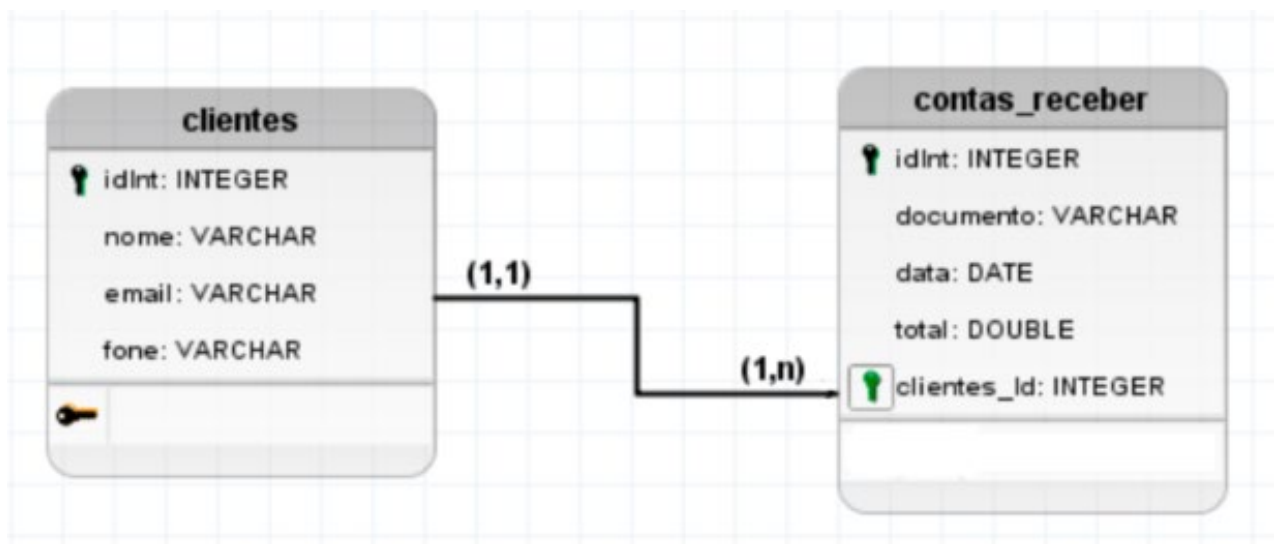


Prática



1) Criar database empresa:

```
CREATE DATABASE empresa;
```

2) Criar a tabela clientes:

```
CREATE TABLE clientes (  
  id_cliente INT NOT NULL AUTO_INCREMENT,  
  nome VARCHAR (50) NOT NULL,  
  sobrenome VARCHAR (50) NOT NULL,  
  email VARCHAR (40) NOT NULL,  
  endereço VARCHAR (100) NOT NULL,  
  bairro VARCHAR (40) NOT NULL,  
  cidade VARCHAR (40) NOT NULL,  
  estado CHAR (2) NOT NULL,  
  pais VARCHAR (50) NOT NULL,  
  fone VARCHAR (20) NULL,  
  sexo CHAR (2) NOT NULL,  
  PRIMARY KEY (id_cliente)
```



);

3) Inserir registros na tabela clientes:

```
INSERT INTO clientes (nome, sobrenome, email, endereço, bairro, cidade, estado, pais, fone, sexo)
VALUES
('Jair', 'Ferreira', 'jair@gmail.com', 'Rua Boituva', 'Ipiranga', 'São Paulo', 'SP', 'Brasil', ' (18)97441245', 'M'),
('Maria', 'Aparecida', 'maria@gmail.com', 'Rua da Graças', 'Jardins', 'São Paulo', 'SP', 'Brasil', ' (11)9448124', 'F'),
('Celso', 'Castro', 'castro@yahoo.com', 'Rua do Som', 'Mooca', 'São Paulo', 'SP', 'Brasil', ' (11)9448111', 'M'),
('Joyce', 'Rodrigues', 'joyce@yahoo.com', 'Rua Fagundes', 'Ipiranga', 'São Paulo', 'SP', 'Brasil', ' (11)9558111', 'F'),
('Juliana', 'Rodrigues', 'ju@yahoo.com', 'Rua Fagundes', 'Ipiranga', 'São Paulo', 'SP', 'Brasil', ' (11)9668111', 'F');
```

4) Criar tabela contas_receber

```
CREATE TABLE contas_receber (
id_receber INT NOT NULL AUTO_INCREMENT,
id_cliente_r INT (11) DEFAULT NULL,
documento    varchar(100) DEFAULT NULL,
data_prevista date NOT NULL,
valor_receber double(20,0) NOT NULL,
PRIMARY KEY (id_receber),
FOREIGN KEY (id_cliente_r) REFERENCES clientes (id_cliente)
);
```

5) Inserir registros na tabela contas_receber

```
INSERT INTO contas_receber (id_cliente_r, documento, data_prevista, valor_receber) VALUES (3, 125456, '2020-10-07', 200), (5, 135460, '2020-10-08', 1050), (1, 185470, '2020-10-20', 1080), (1, 185470, '2020-10-20', 1080), (3, 195495, '2020-09-23', 850);
```



SQL Inner Join

Síntaxe Inner Join:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```


```
SELECT nome, sobrenome, email, cidade, documento, data_prevista, valor_receber
FROM clientes
INNER JOIN contas_receber
ON clientes.id_cliente = contas_receber.id_cliente_r;
```

SQL Left Join

Síntaxe de LEFT JOIN

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT clientes.nome, contas_receber.id_receber
FROM clientes
LEFT JOIN contas_receber
ON clientes.id_cliente = contas_receber.id_cliente_r;
```



```
SELECT clientes.nome, contas_receber.id_receber, contas_receber.data_prevista  
FROM clientes  
LEFT JOIN contas_receber  
ON clientes.id_cliente = contas_receber.id_cliente_r  
ORDER BY contas_receber.data_prevista;
```

SQL RIGHT JOIN

A palavra-chave RIGHT JOIN retorna todos os registros da tabela da direita (tabela2) e os registros correspondentes da tabela da esquerda (tabela1). O resultado é NULL do lado esquerdo, quando não há correspondência.

```
SELECT contas_receber.*, clientes.nome  
FROM contas_receber  
RIGHT JOIN clientes  
ON contas_receber.id_cliente_r = clientes.id_cliente;
```

=====

Um exemplo típico onde é necessário realizar mais de um **JOIN** ocorre quando você precisa consultar dados que estão distribuídos em múltiplas tabelas. Vamos supor que temos três tabelas em um banco de dados de uma empresa: **clientes**, **contas_receber** e **pagamentos**.

Aqui está a estrutura de exemplo:

Tabelas

- **clientes**: Contém informações dos clientes.
- **contas_receber**: Armazena as contas que os clientes devem pagar.
- **pagamentos**: Registra os pagamentos efetuados por esses clientes.

Estrutura das tabelas:



CÓDIGO SQL

```
CREATE TABLE clientes (
```

```
    id_cliente INT NOT NULL AUTO_INCREMENT,
```

```
    nome VARCHAR(50) NOT NULL,
```

```
    sobrenome VARCHAR(50) NOT NULL,
```

```
    email VARCHAR(40) NOT NULL,
```

```
    PRIMARY KEY (id_cliente)
```

```
);
```

```
CREATE TABLE contas_receber (
```

```
    id_receber INT NOT NULL AUTO_INCREMENT,
```

```
    id_cliente_r INT,
```

```
    valor_receber DOUBLE NOT NULL,
```

```
    data_prevista DATE NOT NULL,
```

```
    PRIMARY KEY (id_receber),
```

```
    FOREIGN KEY (id_cliente_r) REFERENCES clientes(id_cliente)
```

```
);
```

```
CREATE TABLE pagamentos (
```

```
    id_pagamento INT NOT NULL AUTO_INCREMENT,
```

```
    id_receber_p INT,
```

```
    valor_pago DOUBLE NOT NULL,
```

```
    data_pagamento DATE NOT NULL,
```

```
    PRIMARY KEY (id_pagamento),
```

```
    FOREIGN KEY (id_receber_p) REFERENCES contas_receber(id_receber)
```

```
);
```



Exemplo de múltiplos JOINS

O objetivo dessa consulta é trazer o nome do cliente, o valor da conta a receber e o valor pago (se houver pagamento) utilizando **INNER JOIN** e **LEFT JOIN**:

sql

//CÓDIGO

SELECT

 clientes.nome,
 contas_receber.valor_receber,
 pagamentos.valor_pago,
 pagamentos.data_pagamento

FROM

 clientes

INNER JOIN

 contas_receber ON clientes.id_cliente = contas_receber.id_cliente_r

LEFT JOIN

 pagamentos ON contas_receber.id_receber = pagamentos.id_receber_p

ORDER BY

 clientes.nome;

Explicação:

1. **INNER JOIN (clientes e contas_receber)**: Esta junção garante que somente os clientes que possuem contas a receber sejam incluídos no resultado.
2. **LEFT JOIN (contas_receber e pagamentos)**: O **LEFT JOIN** é utilizado para que todas as contas apareçam, mesmo aquelas que ainda não têm pagamento associado. Se a conta não tiver sido paga, as colunas referentes ao pagamento (como valor_pago e data_pagamento) serão **NULL**.

=====



CONCAT

```
SELECT CONCAT(nome,' ',sobrenome) AS Nome, email, fone AS Telefone FROM clientes;
```