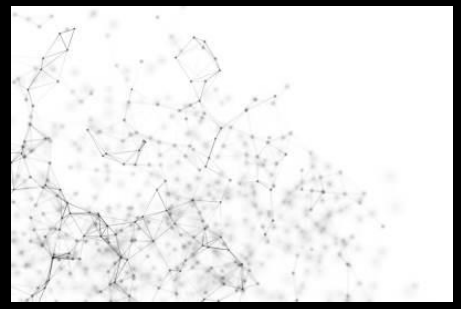
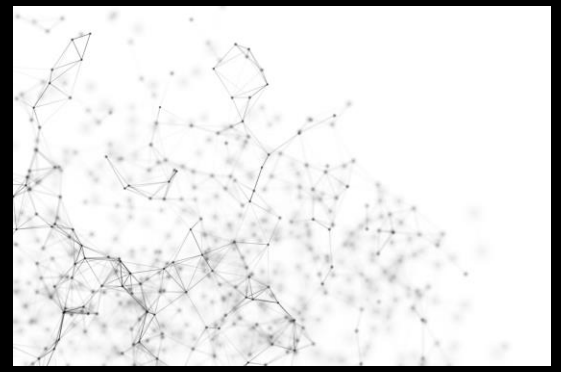




UC6- AULA 4

CRIAR E MANTER BANCO DE DADOS

PROF. CALEBE PEREIRA LEMOS
EMAIL: CALEBE.PEREIRA@UFMS.BR

SQL

LINGUAGEM

Structured Query Language - Linguagem de Consulta Estruturada

- A linguagem SQL surgiu em meados da **década de 70**, sendo resultado de **um estudo de E. F. Codd**, membro do laboratório de **pesquisa da IBM, na Califórnia**. Este estudo tinha foco em **desenvolver uma linguagem com foco no modelo relacional**. O primeiro sistema de BD baseado em SQL tornou-se comercial no final dos anos 70. O sucesso da linguagem SQL foi tão grande que obrigou o **ANSI (American National Standarts Institute), a padronizar as implementações da linguagem**, assim, nos dias de hoje, a maior parte de BD's seguem criteriosamente esta padronização.

SQL

- A programação SQL pode ser usada para analisar ou executar tarefas em tabelas. A sintaxe da linguagem, é baseada no inglês.
- As 4 operações básicas utilizadas em BD, são conhecidas como **CRUD** (Create, Read, Update e Delete). Uma instrução é composta por uma sequência de termos (tokens), formados por comandos, cláusulas e operadores, terminados por um ponto-e-vírgula (";").

Sintaxe – Exemplo:

- Uma instrução SQL simples para criar um Banco de Dados, segue a seguinte estrutura:
- CREATE DATABASE "nome_do_banco";
- Agora uma instrução para selecionar uma lista de dados:
- SELECT "nome_coluna" FROM "nome_tabela";
- INSERT INTO "nome_tabela“ ("nome_coluna“, "nome_coluna“, ...) VALUES (valor1, valor2, ...);

Comandos:

- `SELECT` : busca linhas em tabelas de acordo com um critério definido dentro da chamada cláusula de `WHERE`
- `INSERT` : insere novas linhas na tabela. no nosso caso, colocaria novas notas fiscais dado os argumentos que são passados para o `INSERT` . Por exemplo, no nosso caso: `INSERT INTO nf (titulo, pagamento, valor) VALUES 'canetas', '2019-07-15', 150` .
- `UPDATE` : atualiza linhas do banco de dados de acordo com um critério de `WHERE` , como mudar o CPF
- `DELETE` : remove linhas da tabela de acordo com um critério.

Cláusulas:

Cláusulas	
Cláusula	**Descripción**
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

De onde

Especificar critérios

Agrupar registros

Condições para grupos

Ordenar

Operadores:

Operador
AND
OR
NOT

Operador	
<	
>	
<>	
<=	
>=	
=	
BETWEEN	Entre
LIKE	Comparar String
In	Comparar com conjunto de valores

DDL - Data Definition Language

DDL - Linguagem de definição de dados: É um conjunto de instruções usado para criar e modificar as estruturas dos objetos armazenados no banco de dados.

CREATE: Use instruções CREATE para definir novas entidades. Use CREATE TABLE para adicionar uma nova tabela em um banco de dados.

ALTER: Use as instruções ALTER para modificar a definição de entidades existentes. Use ALTER TABLE para adicionar uma nova coluna a uma tabela ou use ALTER DATABASE para definir opções do banco de dados.

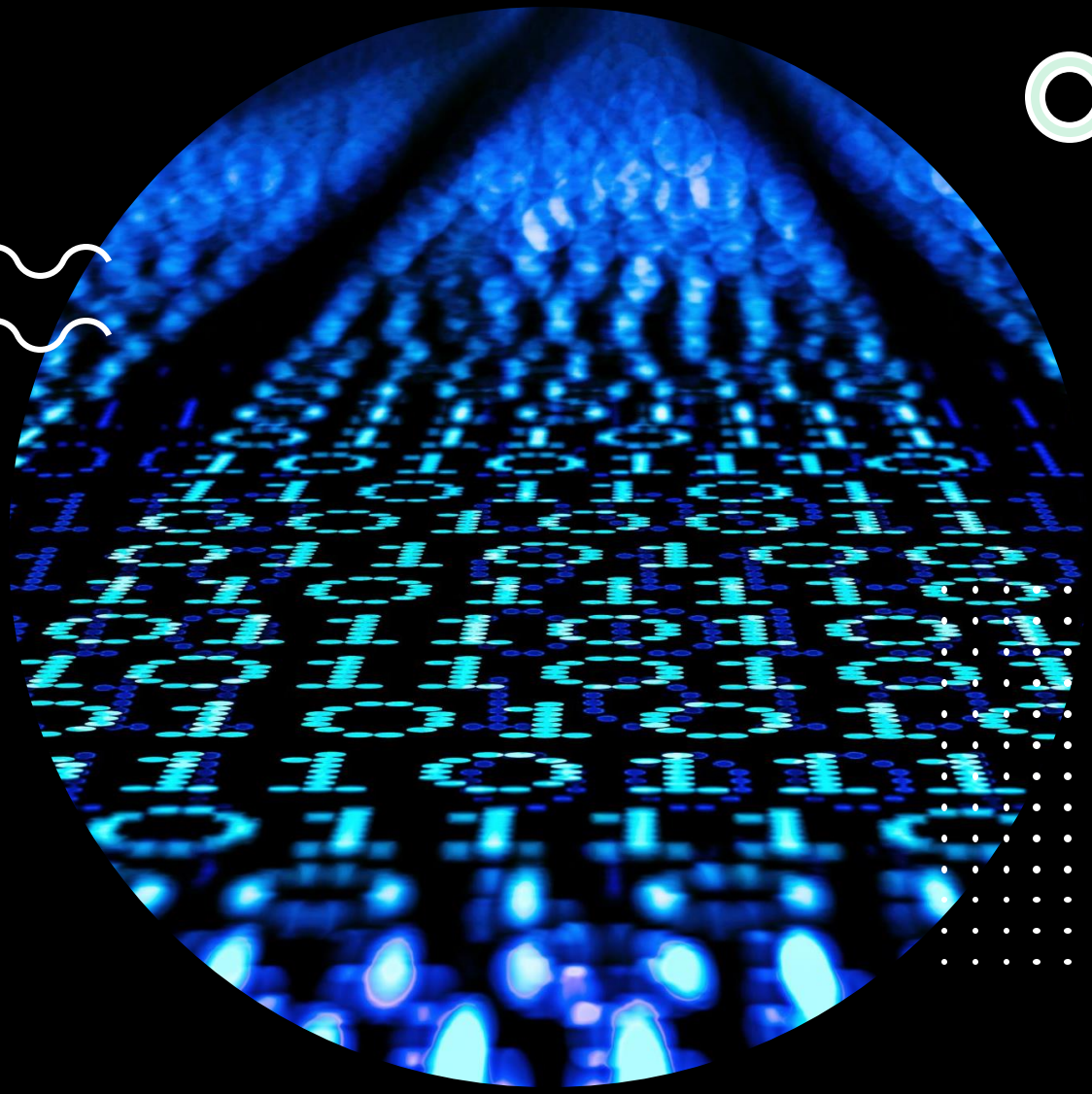
DROP: Use instruções DROP para remover entidades existentes. Use DROP TABLE para remover uma tabela de um banco de dados.

DQL – Data Query Language

DQL - Linguagem de consulta de dados: É um conjunto de instruções responsável por realizar consultas aos dados armazenados no BD.

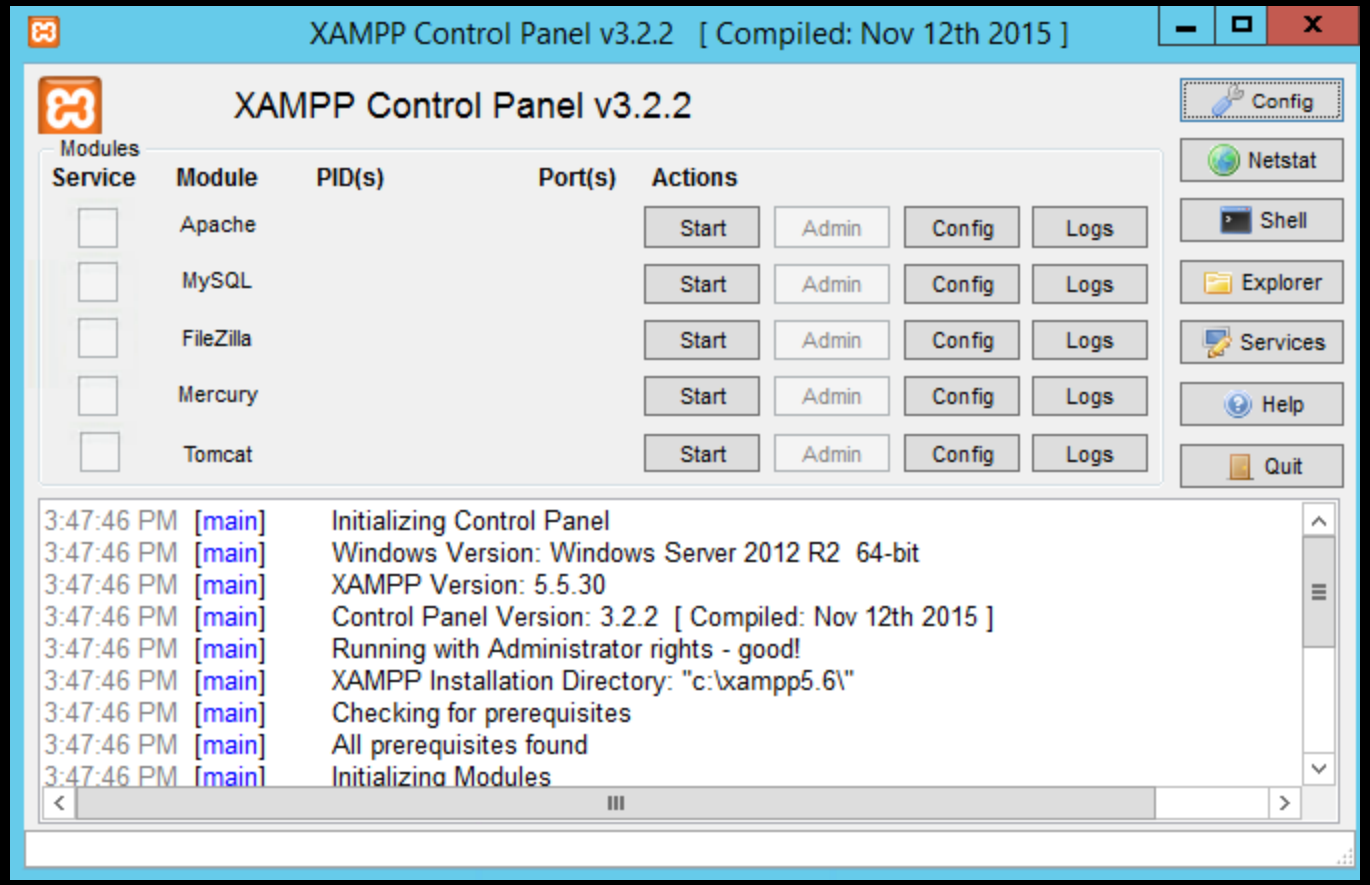
SELECT: Esse comando é um dos mais importantes da SQL, pois é ele quem possibilita a consulta a dados de uma tabela. De modo geral, o Select recupera dados de determinado lugar. Os dados recuperados pelo Select são armazenados em uma nova tabela, chamada conjunto de resultados.

LET'S BORA PRATICAR!



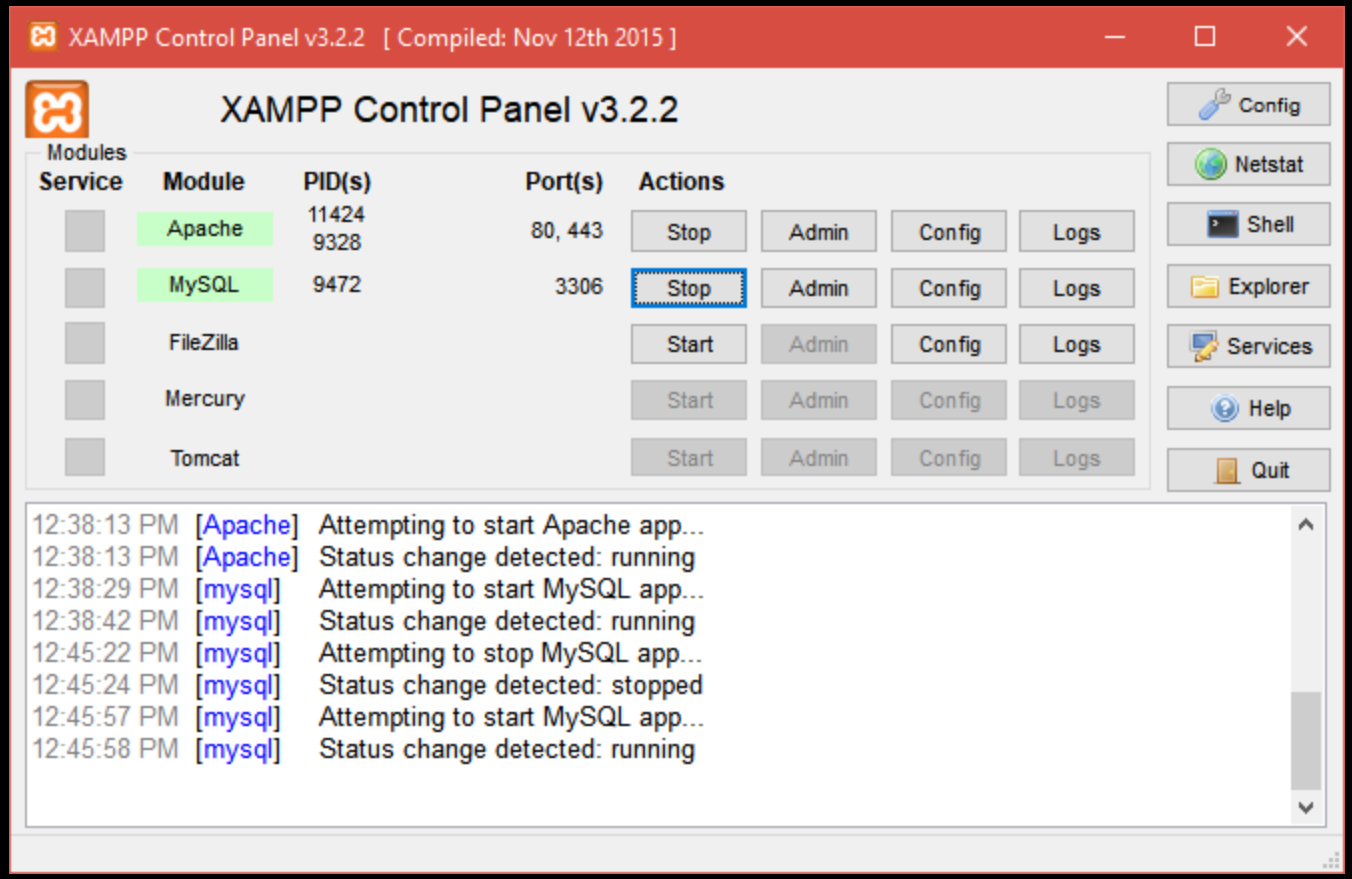
1. XAMPP

Abra o XAMPP.

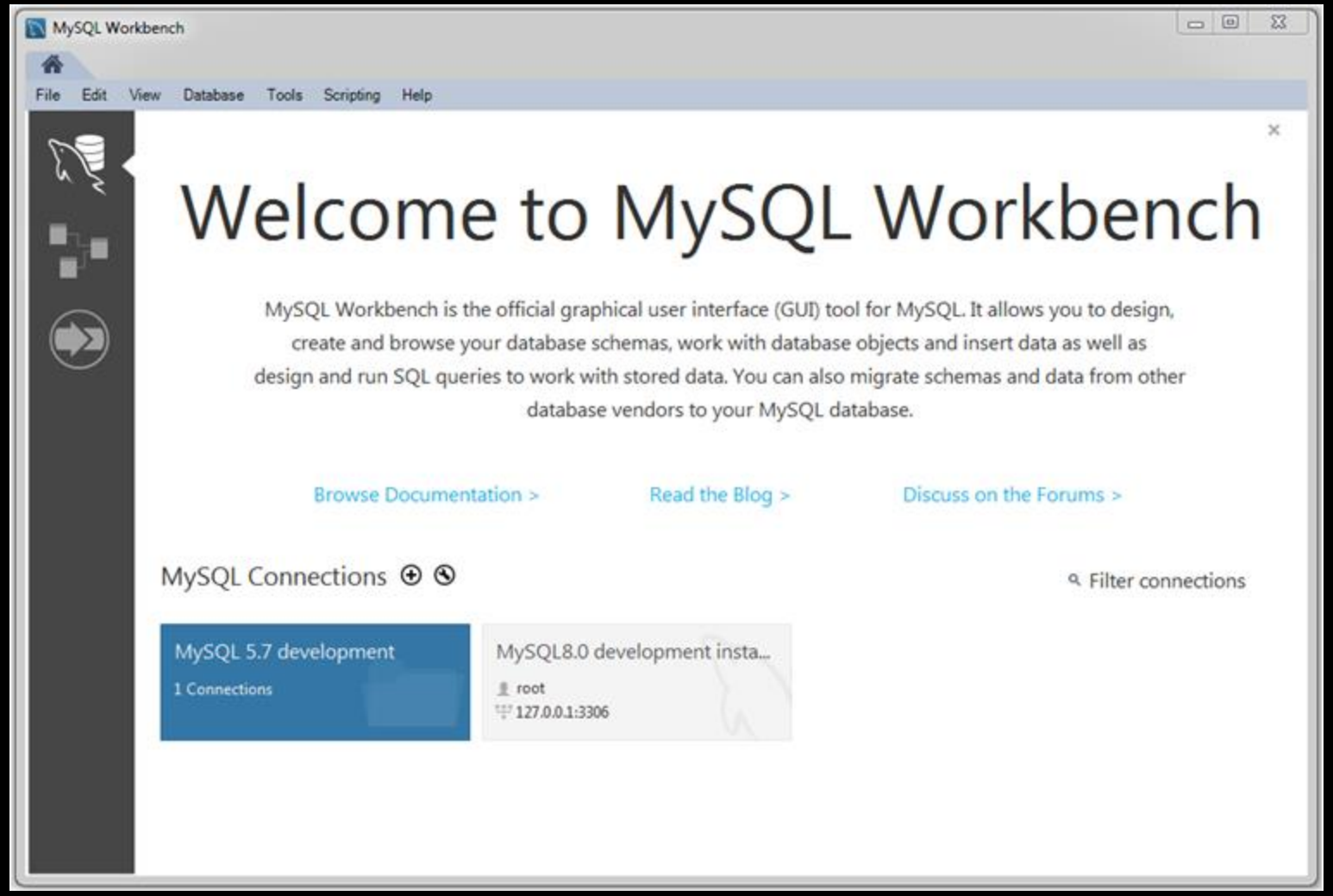


2. XAMPP: Start Apache and MySQL

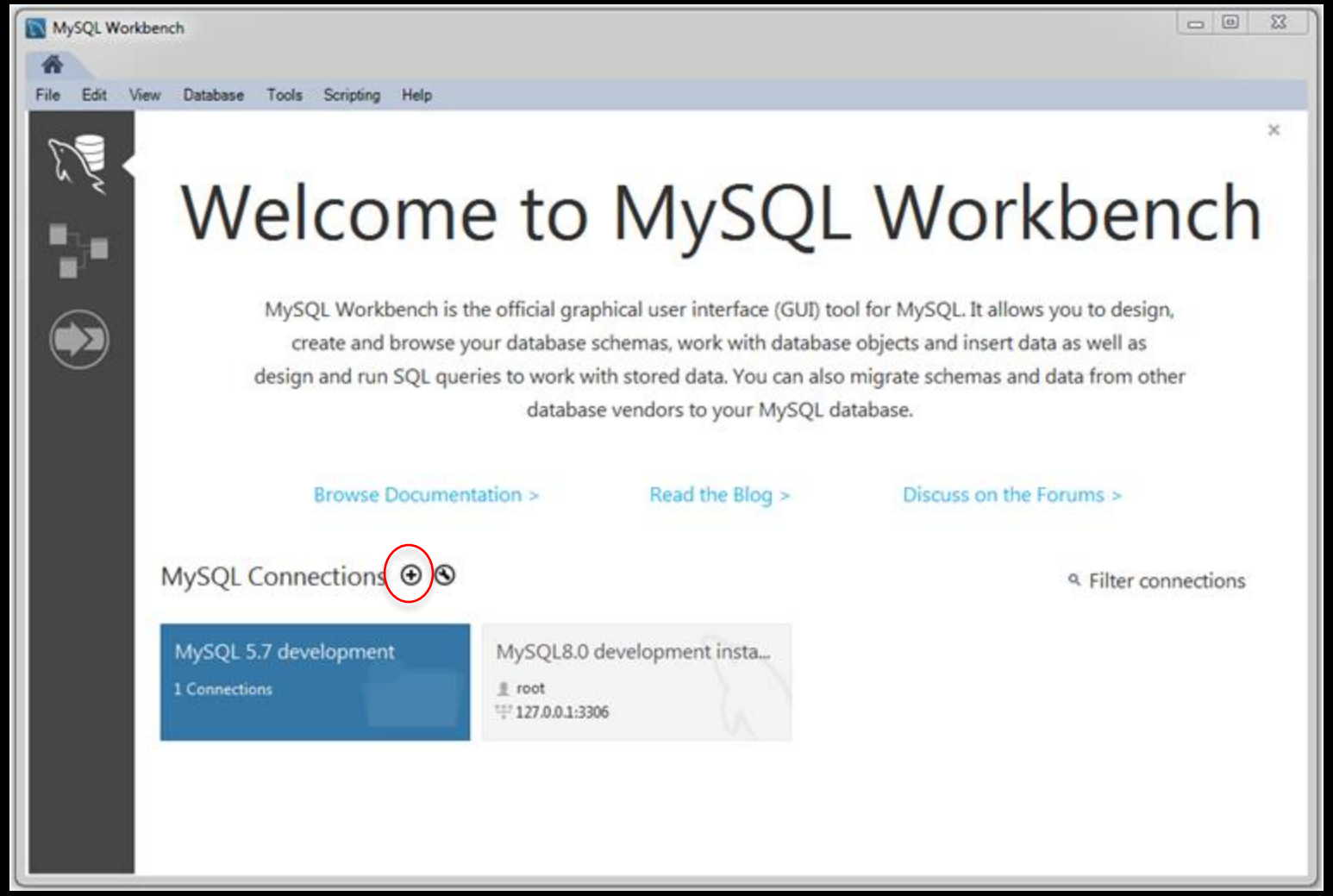
Observe as portas (Port).



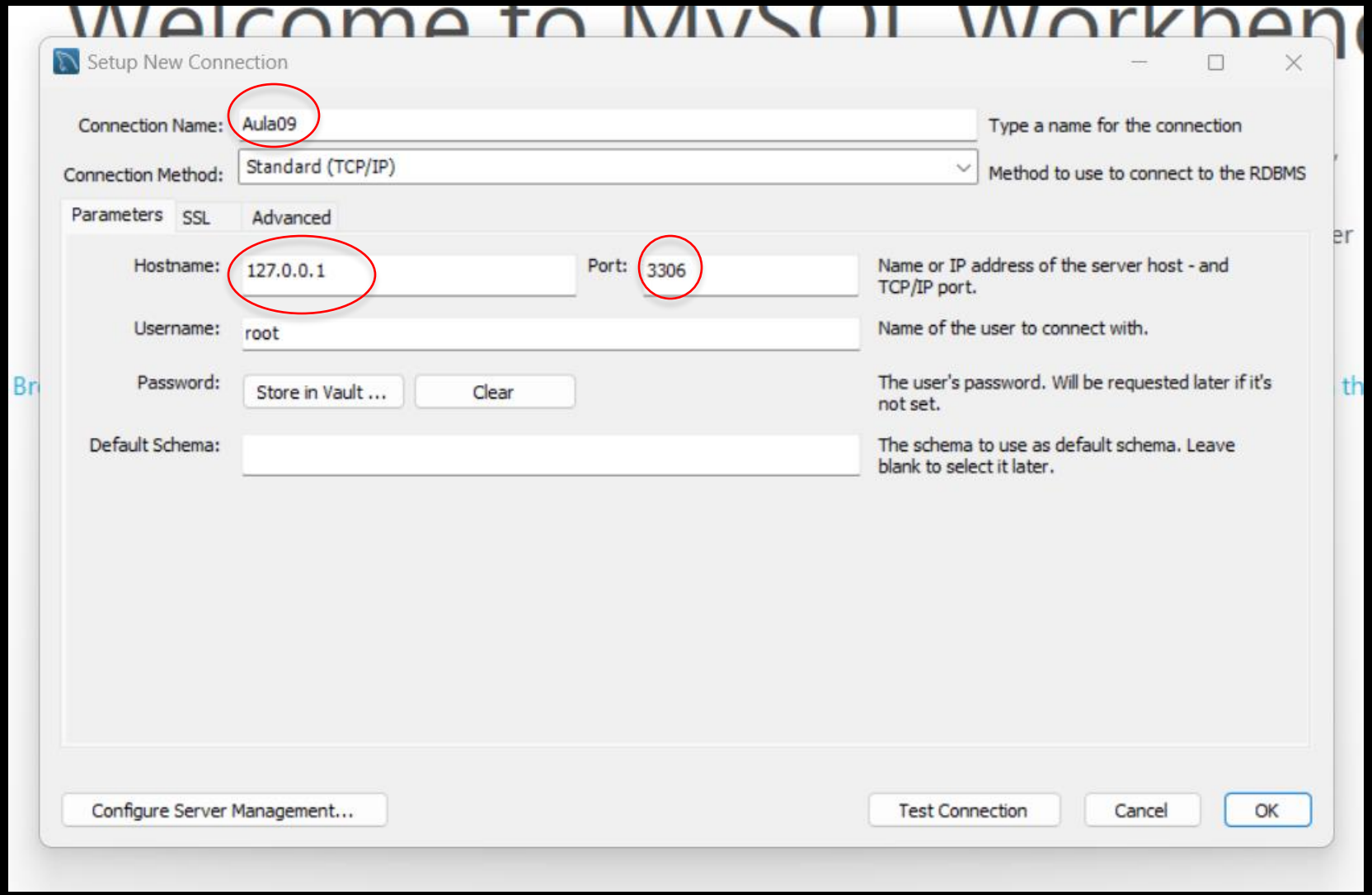
1. MySQL Workbench:



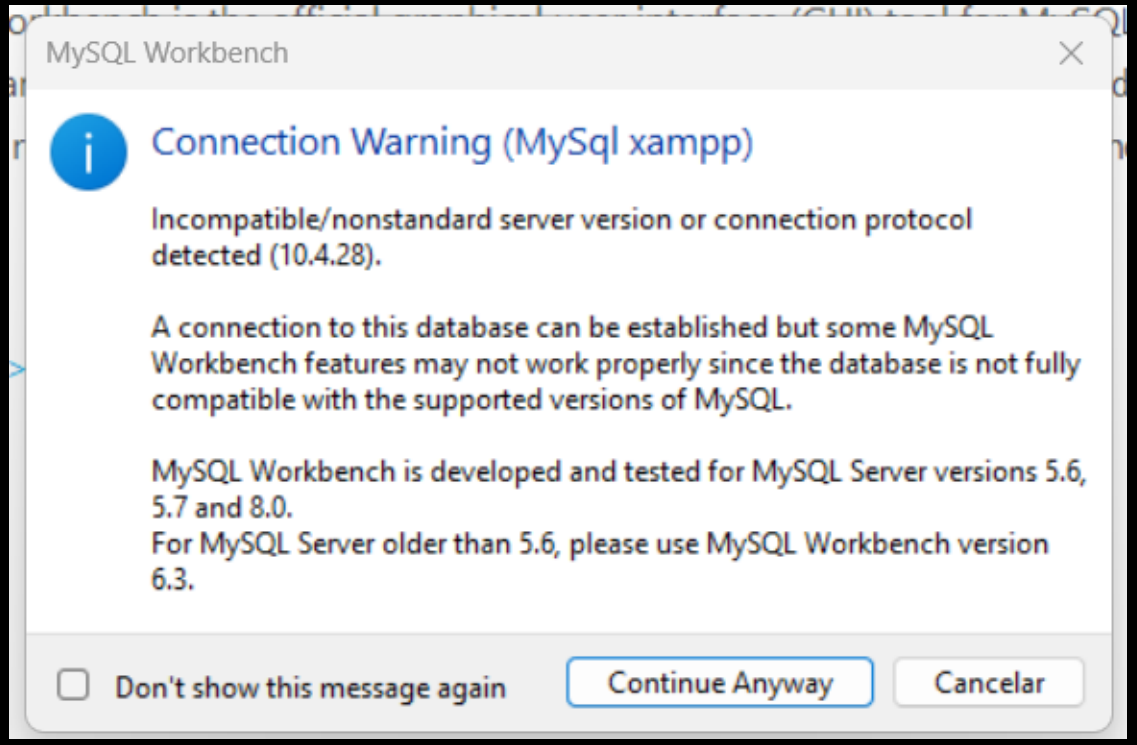
2. MySQL Workbench: Criar Conexão



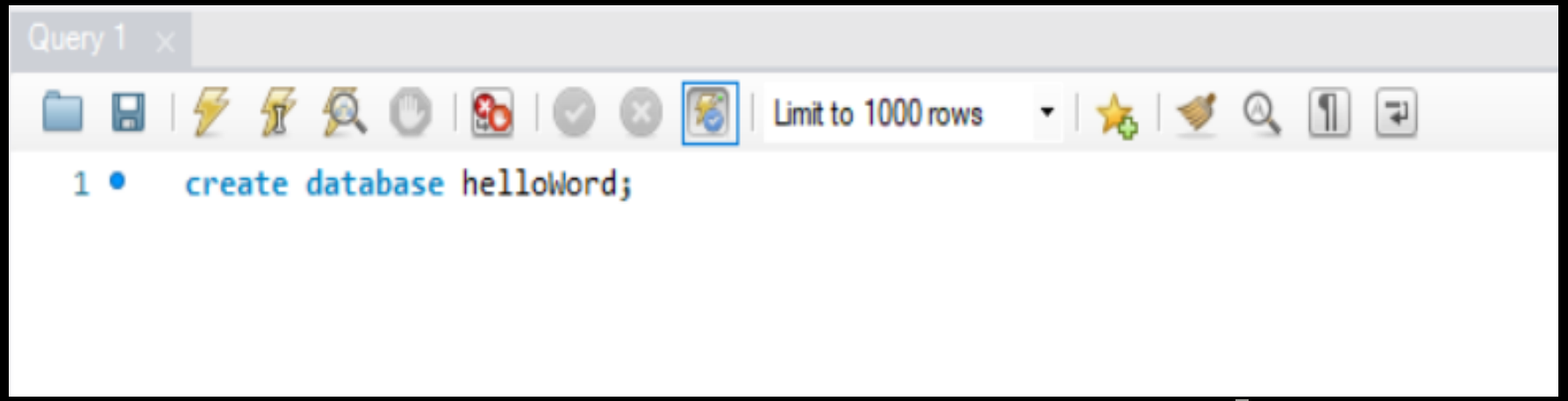
3. MySQL Workbench: Criar Conexão

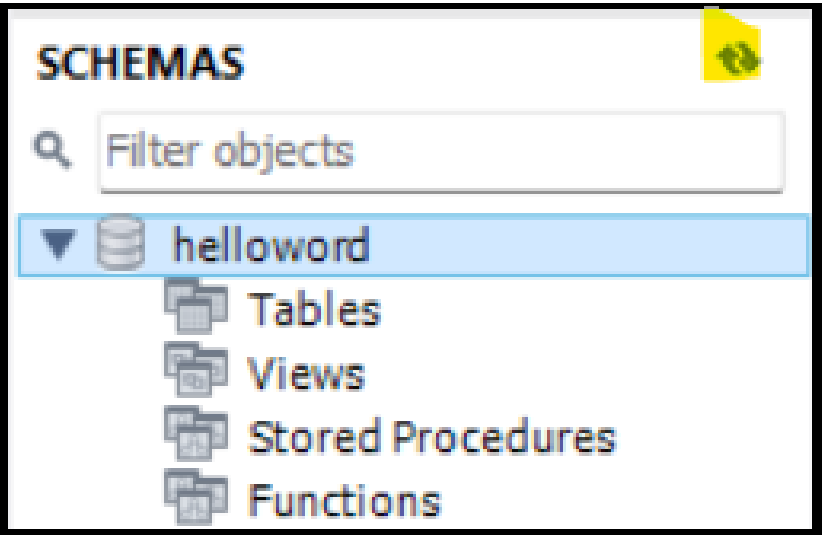


4. MySQL Workbench: Criar Conexão



4. Vamos realizar um teste:





1 SQL. Criar um banco:

- CREATE DATABASE cadastro;
- SHOW DATABASES; #outra opção para visualizar os bancos.
- SHOW TABLES; #mostrar as tabelas.
- USE “nome do banco”; #comando para escolher o BD.

2 SQL. Criar tabelas:

- CREATE TABLE "nome_tabela“
("coluna 1" "tipo_dados_para_coluna_1" restricoes,
"coluna 2" "tipo_dados_para_coluna_2" restricoes,
restricoes extras
...);

CONSTRAINTS – Restrições

As restrições no SQL são regras predefinidas e restrições que são aplicadas em uma única coluna ou várias colunas, em relação aos valores permitidos nas colunas, para manter a integridade, precisão e confiabilidade dos dados dessa coluna.

NOT NULL;

DEFAULT;

UNIQUE;

AUTO_INCREMENT;

PRIMARY KEY;

FOREIGN KEY;

3 SQL. Criar tabelas e adicionar restrições:

```
CREATE TABLE pessoa (
 id INT NOT NULL,
 nome VARCHAR (100) NOT NULL,
 cpf CHAR(11) NOT NULL UNIQUE,
 idade INT NOT NULL,
 PRIMARY KEY (id)
 );
```

4 SQL. Restrições: (chave estrangeira)

```
create table venda(
id int NOT null primary key,
produto varchar(55),
id_pessoa int NOT NULL,
foreign key (id_pessoa) references pessoa(id)
);
```

5 SQL. Descrever tabela:

DESCRIBE pessoa;

	Field	Type	Null	Key	Default	Extra
▶	id	int(11)	NO	PRI	NULL	
	cpf	varchar(11)	YES		NULL	
	data_nasc	date	NO		NULL	

6 SQL. Descrever tabela:

DESCRIBE venda;

	Field	Type	Null	Key	Default	Extra
▶	id	int(11)	NO	PRI	NULL	
	produto	varchar(55)	YES		NULL	
	id_pessoa	int(11)	NO	MUL	NULL	

Chaves:

PRI: primária;

UNI: única;

MUL: multivalorado – são permitidas múltiplas ocorrências do mesmo valor.

EXERCÍCIOS

LET'S BORA PRATICAR!

1.

- Crie uma tabela chamada "Clientes" com as seguintes colunas:

- ID (chave primária)
- Nome
- Sobrenome
- Email
- Data de Nascimento

2.

- Crie uma tabela chamada "Produtos" com as seguintes colunas:

- ID (chave primária)
- Nome do Produto
- Preço
- Descrição
- Categoria

3.

- Crie uma tabela chamada "Pedidos" com as seguintes colunas:

- ID (chave primária)
- Data do Pedido
- ID do Cliente (chave estrangeira referenciando a tabela Clientes)
- Status do Pedido (por exemplo, "Em andamento", "Concluído", "Cancelado")

4.

- Crie uma tabela chamada "ItensPedido" para rastrear os itens em cada pedido com as seguintes colunas:

- ID (chave primária)
- ID do Pedido (chave estrangeira referenciando a tabela Pedidos)
- ID do Produto (chave estrangeira referenciando a tabela Produtos)
- Quantidade

5.

- Crie uma tabela chamada "Categorias" para categorizar os produtos com as seguintes colunas:

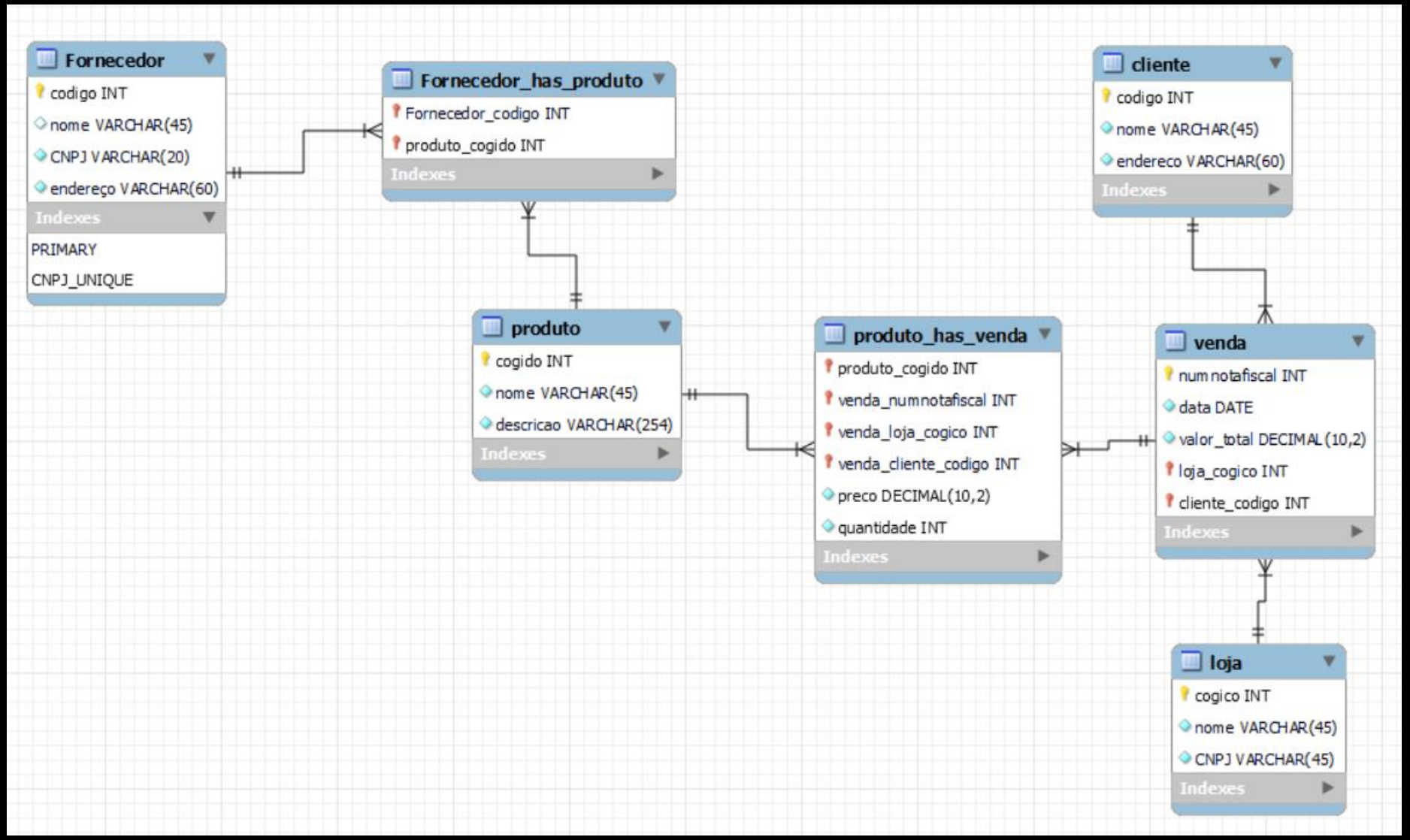
- ID (chave primária)
- Nome da Categoria

6.

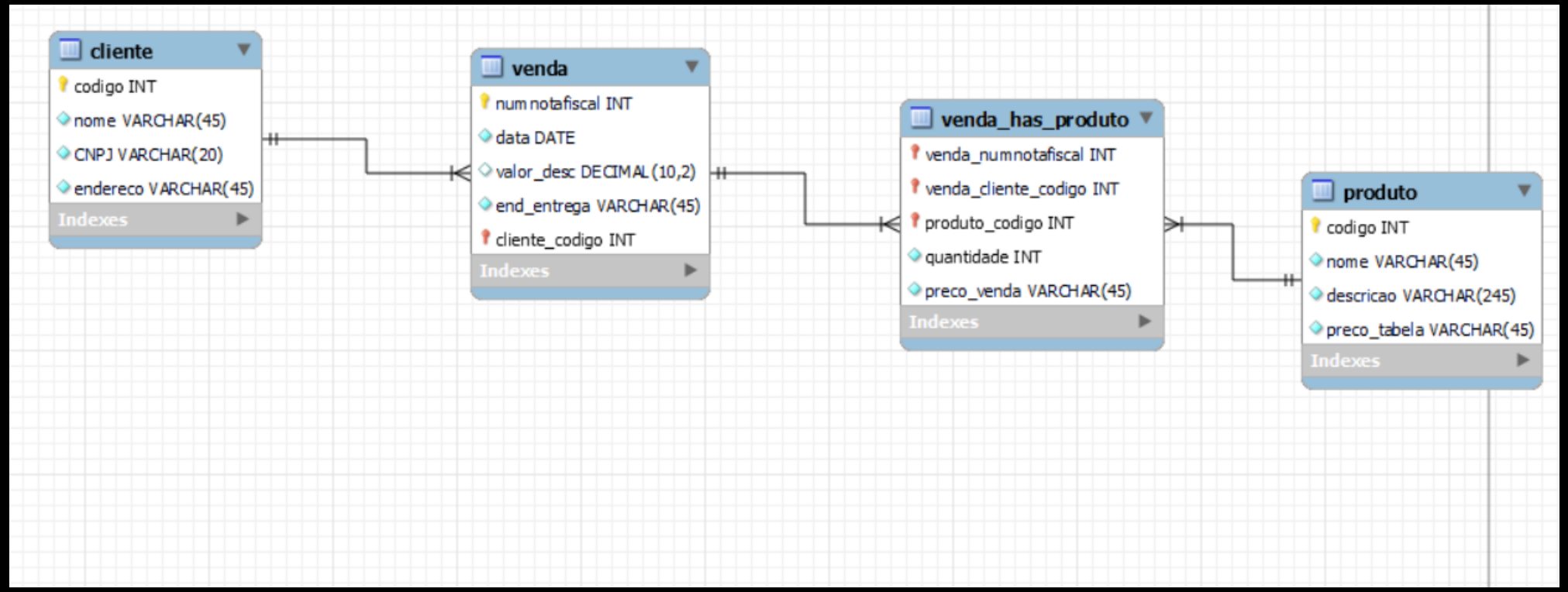
- Crie uma tabela chamada "Funcionários" com as seguintes colunas:

- ID (chave primária)
- Nome
- Sobrenome
- Cargo
- Data de Contratação
- Salário

7. Crie o banco do modelo abaixo:



8. Crie o banco do modelo abaixo:



Renomear tabela:

Utilizar o comando RENAME

```
rename table clientes to cliente;
```

ALTER: (renomear atributos)

Utilizar o comando ALTER e CHANGE

```
alter table cliente
change data_nasc dataNasc date not null;
```

ALTER: (adicionar atributos (coluna))

Utilizar o comando ALTER e ADD

```
alter table cliente add column cpf varchar(11) not null unique;
```

ALTER: (apagar coluna)

Utilizar o comando ALTER e DROP

```
ALTER TABLE pessoa DROP COLUMN estado;
```

ALTER: (adicionar atributos (ex: chave))

Utilizar o comando ALTER e ADD

```
alter table pedidos add column id_produto int not null;
alter table pedidos add foreign key(id_produto)
references produtos(id);
```

ALTER: (remover atributos (ex: chave))

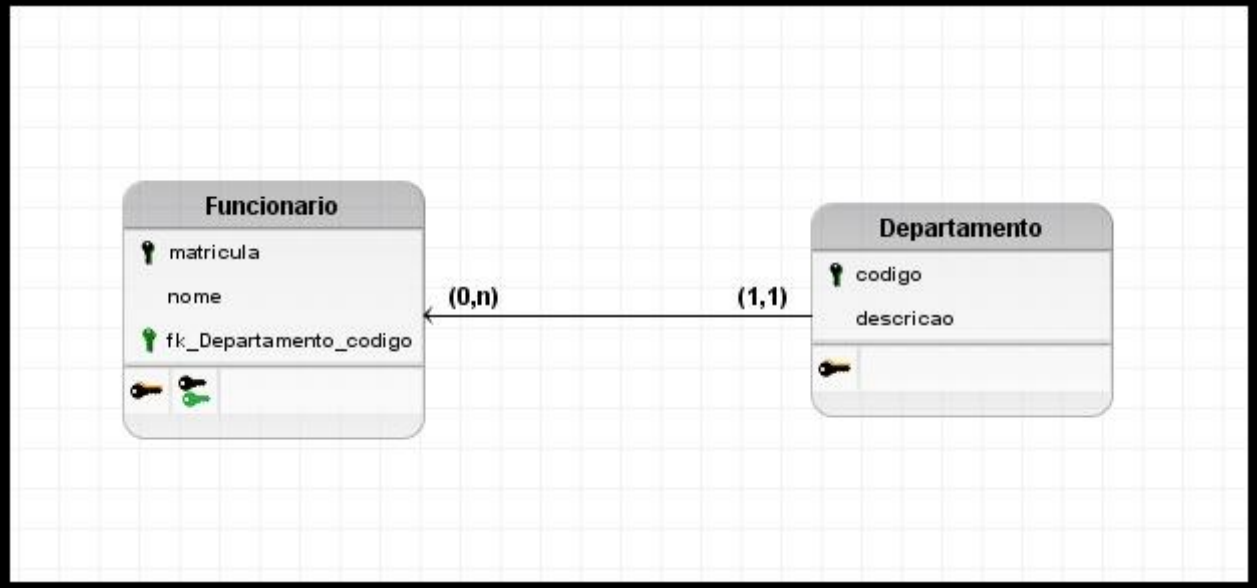
Utilizar o comando ALTER e DROP

```
ALTER TABLE pessoa DROP FOREIGN KEY NOME_FOREIGN_KEY;
ALTER TABLE pessoa DROP INDEX NOME_INDEX;
ALTER TABLE pessoa DROP COLUMN estado;
```

Relacionamentos

1:N

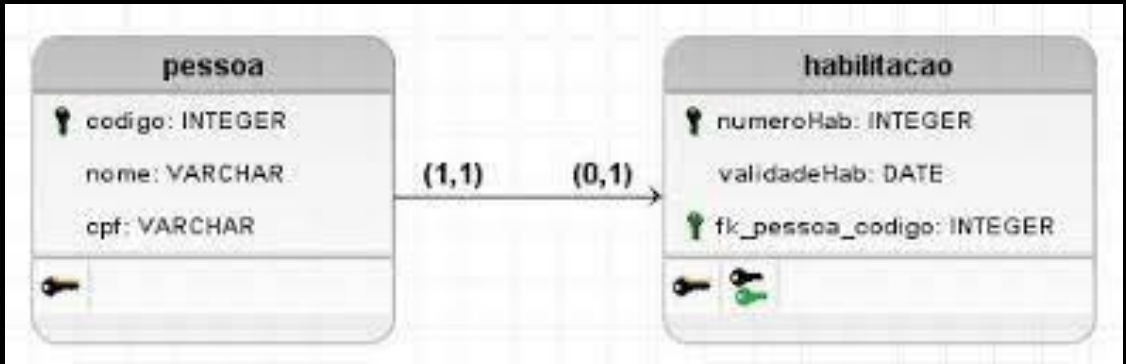
Cria-se uma chave estrangeira Na tabela “lado N”, que aponta para chave primária da tabela “lado 1”.



Relacionamentos

1:1

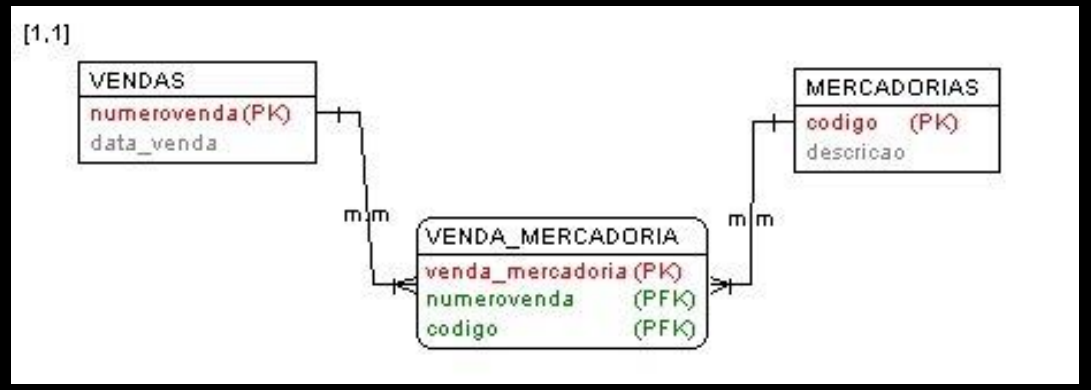
Escolhe-se o lado dominante
e cria-se uma chave estrangeira
que aponta para chave primária
da outra tabela.



Relacionamentos

N:N

Cria-se uma nova tabela (relacionamento, adicionando uma chave estrangeira apontando para o id de cada tabela envolvida.



INSERT: (adicionar valores)

Utilizar o comando INSERT INTO

```
insert into cliente(id, nome, sobrenome, email, dataNasc, cpf)
values(null, "calebe", "lemos", "test@gmail.com", "2000-02-21", "00000000000");
```

SQL. Select:

Sintaxe: SELECT "nome_coluna" FROM "nome_tabela";

SELECT * FROM pessoa;

SELECT nome FROM pessoa;

SELECT nome, cpf FROM pessoa;

SELECT nome, cpf, idade FROM pessoa;

SELECT: (tabela cliente)

Utilizar o comando SELECT ... FROM

```
select * from cliente;
```

id	nome	sobrenome	email	dataNasc	cpf
4	calebe	lemos	test@gmail.com	2000-02-21	00000000000
NULL	NULL	NULL	NULL	NULL	NULL

```
select email from cliente;
```

email
test@gmail.com

DELETE: (tabela cliente)

Utilizar o comando DELETE FROM

```
delete from cliente where id=3;
```

UPDATE: (alterar dado da tabela)

Utilizar o comando UPDATE e SET

```
update cliente set email = "novo@gmail.com" where id = 4;
```

Exercício

Crie uma tabela chamada "Produtos" com os seguintes campos:

ID (chave primária, auto-incremento)
Nome (varchar, 50 caracteres)
Descrição (text)
Preço (decimal, 10, 2)
Quantidade (int)

Exercício

Adicione três produtos à tabela "Produtos" com os seguintes dados:

Produto 1: Nome = "Camiseta", Descrição = "Camiseta branca de algodão", Preço = 19.99, Quantidade = 100

Produto 2: Nome = "Tênis", Descrição = "Tênis esportivo preto", Preço = 49.99, Quantidade = 50

Produto 3: Nome = "Celular", Descrição = "Smartphone com tela OLED", Preço = 399.99, Quantidade = 20

Exercício

Escreva uma consulta SQL para recuperar todos os produtos da tabela “Produtos” e uma consulta para recuperar apenas preços maiores que 20.00.

Exercício

Atualize o preço do "Produto 1" para 24.99 e a quantidade para 120 unidades.

Exclua o "Produto 3" da tabela "Produtos".

EXERCÍCIOS

LET'S BORA PRATICAR!

Exercício 1 – Bora pratica os CRUDs

Crie uma tabela chamada "Clientes" com as colunas "ID" (chave primária), "Nome", "Email" e "Telefone". Logo após, insira 5 clientes ao banco.

('João Silva', 'joao@email.com', '23456-7890’)

('Maria Santos', 'maria@email.com', ‘87654-3210’)

(‘Amarildo Carlos Pereira', ‘acarlos@email.com', ‘75555-5555’)

('Ana Rodrigues', 'ana@email.com', '77123-4567’)

('Pedro Oliveira', 'pedro@email.com', '33888-9999')

Exercício 1 –

Selecione todos os clientes cujo nome comece com a letra “A”.

Para denotar uma sequência de caracteres qualquer, utilizamos o operador %. Por exemplo, se a ideia fosse selecionar todos CPFs iniciados com 867, a condição seria: “867%”.

Exercício 1 –

Atualize o número de telefone do cliente João Silva para ‘99867-0898’.

Exercício 1 –

Exclua todos clientes com número de telefone que possui inicial 7.

Exercício 2: Crie as seguintes tabelas:

Tabela "Medicamentos":

Colunas: MedicamentoID (Chave Primária), Nome, Fabricante, Preço, QuantidadeEmEstoque

Tabela "Clientes":

Colunas: ClienteID (Chave Primária), Nome, Endereço, Telefone

Tabela "Vendas":

Colunas: VendaID (Chave Primária), ClienteID (Chave Estrangeira), DataDaVenda

Tabela "ItensDeVenda":

Colunas: ItemID (Chave Primária), VendaID (Chave Estrangeira), MedicamentoID (Chave Estrangeira), Quantidade, PrecoTotal

Exercício 2

Insira dados fictícios nas tabelas. Aqui estão alguns exemplos:

Medicamentos:

('Paracetamol', 'Farmaco', 5.99, 100)

('Amoxicilina', 'PharmaCorp', 8.50, 50)

('Omeprazol', 'MegaMed', 7.25, 80)

Clientes:

('Maria Silva', 'Rua A, 123', '555-1234’)

('João Santos', 'Av. B, 456', '555-5678')

Exercício 2

Insira dados fictícios nas tabelas. Aqui estão alguns exemplos:

Vendas:

(1, ‘2023-10-10’)

(2, '2023-10-11’)

Itens da Venda:

(1, 1, 2, 11.98)

(1, 2, 3, 25.50)

Exercício 2

Atualizar o preço do paracetamol para R$6.50, atualizar o endereço do João Santos para Rua X, 109 e atualizar a quantidade de Omeprazol para 100 unidades.

Exercício 2

Remover a venda com ID=2 e os seus respectivos itens.

(NÃO REALIZAR ESSE EXERCÍCIO!!!!)

Exercício 2

Faça a consulta que retorne todos medicamentos cadastrados no banco.

Exercício 2

Faça uma consulta que retorne todos medicamentos cadastrados que são das fabricantes PharmaCorp ou MegaMed.

Exercício 2

Realize o cadastro dos medicamentos abaixo. Logo após, faça uma consulta que retorne apenas medicamentos abaixo de R$6.00 e com mais de 50 unidades.

- Medicamento: Loratadina
  - Fabricante: AllergiPharma
  - Preço: 4.50
  - Quantidade: 120 unidades
- Medicamento: Ibuprofeno
  - Fabricante: MedicaGen
  - Preço: 5.50
  - Quantidade: 80 unidades

Vamos praticar algumas consultas:

Ordenação:

Vamos selecionar os produtos em ordem crescente e decrescente.

```
SELECT coluna1, coluna2
FROM nome_da_tabela
ORDER BY coluna_a_ordenar ASC/DESC;
```

Vamos utilizar as funções de MAX e MIN.

```
SELECT MAX(coluna)
FROM nome_da_tabela;
```

Vamos praticar algumas consultas:

Para trazer o maior ou menor preço:

```
select max(Preço) from medicamentos;
```

Para o menor, é só substituir a função para min().

Mas, e para trazer o nome do medicamento com maior ou menor preço?

```
select * from medicamentos where Preço = (select max(Preço) from medicamentos);
```

Vamos praticar algumas consultas:

COUNT, AVG e SUM

Count, é utilizada para contagem de registros.

AVG, usada para calcular a média dos valores.

SUM, usada para somar valores de uma coluna.

```
SELECT COUNT(coluna) FROM nome_da_tabela;

SELECT AVG(coluna) FROM nome_da_tabela;

SELECT SUM(coluna) FROM nome_da_tabela;
```

Vamos praticar algumas consultas:

Selecionar valores dentro de um intervalo:

```
SELECT coluna(s) FROM nome_da_tabela
WHERE coluna BETWEEN valor_inicial AND valor_final;
```

JOINS:

INNER JOIN

LEFT JOIN

RIGHT JOIN

FULL JOIN

Inner Join / Join

As cláusulas Join, são usadas para combinar dados de tabelas. Por exemplo:

Se eu tenho uma tabela de vendas, que possui clientes relacionados a cada venda, para saber informação completa, preciso relacionar as tabelas e assim obter todas as informações sobre a venda, como data, produto, cliente envolvido, dados do cliente, entre outras.

Inner Join / Join - Sintaxe

Inner Join: Retorna apenas os registros que tem correspondência nas duas tabelas. Podemos utilizar apenas Join também.

```
SELECT tabela1.coluna1, tabela2.coluna2
FROM tabela1
INNER JOIN tabela2 ON tabela1.chave = tabela2.chave;
```

```
SELECT
   clientes.Coluna1,
   clientes.Coluna2,
   pedidos.Coluna3
FROM
   Clientes
JOIN
   Pedidos ON clientes.ID_Cliente = pedidos.ID_Cliente;
```

Detalhes Inner Join

SELECT: Especifique as colunas que deseja realizar a consulta.

FROM: Indica a tabela principal da consulta.

INNER JOIN OU JOIN: Especifica a tabela que deseja combinar.

ON: Define a condição de junção.

```
SELECT tabela1.coluna1, tabela2.coluna2
FROM tabela1
INNER JOIN tabela2 ON tabela1.chave = tabela2.chave;
```

Exemplo Inner Join

Quero retornar todos clientes que efetuaram compras e a data correspondente à compra.

```
select clientes.Nome, vendas.DataDaVenda from clientes join vendas on
clientes.ChavePrimariaID = vendas.ChaveEstrangeiraClienteID;
```

Delete x Truncate

Delete é usado para excluir registros específicos, usando a cláusula Where, já o Truncate, é usado para excluir todos os registros da tabela (se utilizar delete sem Where, acontece a mesma coisa, porém os ID's não serão zerados).

```
truncate table itensdevenda;
```

```
delete from itensdevenda where ItemID = 3;
```

Exemplo Join

Agora, vamos retornar todas as compras realizadas com as seguintes informações: nome do comprador, data da compra, nome do produto e preço total da compra. A ideia da consulta é buscar todas as compras que os clientes estão envolvidos.

```
select clientes.Nome, vendas.DataDaVenda, itensdevenda.PrecoTotal, itensdevenda.Quantidade,medicamentos.Nome from clientes join vendas on clientes.ClienteID = vendas.ClienteID join itensdevenda on vendas.VendaID = itensdevenda.VendaID join medicamentos on itensdevenda.MedicamentoID = medicamentos.MedicamentoID;
```

Adicionar condição ao Join

Podemos adicionar condições ao Join também. Para isso, é só utilizar a cláusula Where. Por exemplo, para retornar vendas com valores acima de R$20.00:

```
select clientes.Nome, vendas.DataDaVenda, itensdevenda.PrecoTotal, itensdevenda.Quantidade,medicamentos.Nome from clientes join vendas on clientes.ClienteID = vendas.ClienteIDjoin itensdevenda on vendas.VendaID = itensdevenda.VendaID join medicamentos on itensdevenda.MedicamentoID = medicamentos.MedicamentoID Where itensdevenda.PrecoTotal > 20.00;
```

Revisão Join

A cláusula Join permite vincular dados entre tabelas, com base nos valores das colunas em comum entre elas (chave primária e chave estrangeira).

```
SELECT tabela.lista_colunas FROM tabela_pai JOIN tabela_filha ON condição;
```

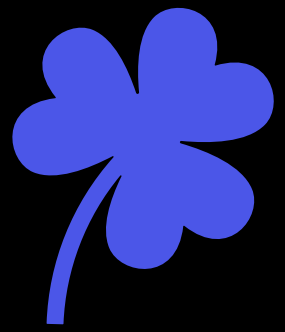
Alias (Apelido)

Útil para renomear temporariamente colunas ou tabelas em uma consulta SQL, o que facilita a compreensão dos dados.

```
SELECT nome, sobrenome, valor AS salário FROM funcionarios;
```

Exercícios: Join

BORA PRATICAR!



Boa sorte!

Exercícios: Join

Você tem duas tabelas, "Funcionários" e "Departamentos." A tabela "Funcionários" contém informações sobre os funcionários de uma empresa, incluindo seu nome, sobrenome e o departamento ao qual estão associados (usando uma coluna "ID_Departamento"). A tabela "Departamentos" contém informações sobre os departamentos, incluindo seus nomes e localizações. Crie uma consulta SQL que retorne o nome completo do funcionário, o nome do departamento ao qual pertence e a localização do departamento.

Exercícios: Join

- Cadastre os seguintes funcionários:

```
(1, 'João', 'Silva', 1),
(2, 'Maria', 'Santos', 2),
(3, 'Pedro', 'Ferreira', 1),
(4, 'Ana', 'Oliveira', 3);
```

- Cadastre os seguintes departamentos:

```
(1, 'Vendas', 'São Paulo’),
(2, 'Marketing', 'Rio de Janeiro’),
(3, 'TI', 'Belo Horizonte');
```

Exercícios: Join - 2

Você tem três tabelas, "Alunos," "Disciplinas" e "Notas." A tabela "Alunos" contém informações sobre os alunos, incluindo seus nomes e números de identificação de aluno. A tabela "Disciplinas" contém informações sobre as disciplinas, incluindo seus nomes e códigos de disciplina. A tabela "Notas" contém as notas dos alunos em cada disciplina, incluindo o número de identificação do aluno (ID_Aluno), o código da disciplina (ID_Disciplina) e a nota. Crie uma consulta SQL que retorne o nome completo do aluno, o nome da disciplina, o código da disciplina e a nota correspondente para cada aluno.

Exercícios: Join - 2

- Cadastre os seguintes alunos e as notas (lado direito):

```
(1, 'Carlos', 'Silva'),
(2, 'Ana', 'Ferreira'),
(3, 'Paulo', 'Santos');
```

```
(1001, 1, 101, 90.5),
(1002, 1, 102, 85.0),
(1003, 2, 101, 88.5),
(1004, 2, 102, 92.0),
(1005, 3, 101, 78.0),
(1006, 3, 103, 96.5);
```

- Cadastre as seguintes disciplinas:

```
(101, 'Matemática', 'MAT101'),
(102, 'História', 'HIS102'),
(103, 'Ciências', 'CIE103');
```

Exercícios: Join - 3

Você tem duas tabelas, "Produtos" e "Categorias." A tabela "Produtos" contém informações sobre os produtos, incluindo seus nomes, preços e as categorias a que pertencem (usando uma coluna "ID_Categoria"). A tabela "Categorias" contém informações sobre as categorias de produtos, incluindo seus nomes e descrições. Crie uma consulta SQL que retorne o nome do produto, o preço, o nome da categoria a que pertence e a descrição da categoria.

Exercícios: Join - 3

- Cadastre os seguintes produtos:

```
(1, 'Camiseta', 20.00, 1),
(2, 'Calça', 30.00, 2),
(3, 'Tênis', 50.00, 1),
(4, 'Bolsa', 40.00, 2);
```

- Cadastre as seguintes categorias:

```
(1, 'Roupas', 'Roupas masculinas e femininas’),
(2, 'Acessórios', 'Acessórios de prata');
```

Exercícios: Join - 4

Você tem três tabelas, “Livros”, “Autores” e “Empréstimos”. A tabela “Livros” contém informações sobre os livros da biblioteca, incluindo título, autor, e quantidade de exemplares. A tabela “Autores” contém informações sobre os autores, incluindo nome e nacionalidade. Já a tabela “Empréstimo”, contém informações sobre os empréstimos realizados, armazenando o nome do cliente, o livro emprestado, a data de empréstimo e data de devolução. Crie uma consulta SQL que retorne informações sobre os livros atualmente emprestados, incluindo o título, o nome do autor, o nome do cliente que fez o empréstimo e a data de empréstimo. Além disso, você quer que a consulta liste apenas os livros atualmente emprestados, ou seja, onde a data de devolução seja nula.

Exercícios: Join - 4

- Cadastre os seguintes livros e autores (lado direito):

```
('Harry Potter and the Sorcerer s Stone', 1, 5)
('1984', 2, 3)
('Pride and Prejudice', 3, 4)
('The Adventures of Huckleberry Finn', 4, 6)
```

```
('J.K. Rowling', 'Reino Unido’)
('George Orwell', 'Reino Unido’)
('Jane Austen', 'Reino Unido’)
('Mark Twain', 'Estados Unidos')
```

- Cadastre as seguintes empréstimos:

```
('João', 1, '2023-10-01', NULL)
(‘Marcela', 3, '2023-10-02', NULL)
('Pedro', 4, '2023-10-03', NULL)
```

```
('Ana', 2, '2023-09-15', '2023-09-30’)
('Carlos', 1, '2023-09-20', '2023-10-05')
```