Agenda 12: Modelo Físico de BD - Linguagem SQL

Conceitos trabalhados:

Sumário

Momento de Reflexão

Olá, estudante! :)

Boas-vindas à última agenda do módulo 1!

Você já percebeu que bancos de dados se tornaram componentes essenciais e indispensáveis em nosso dia a dia. Sem eles não seria possível fazer atividades corriqueiras como depósitos, saques ou movimentações bancárias. BD's são fundamentais em qualquer sistema informatizado, e permitem que você possa reservar aquela pousada na praia, pesquisar livros em bibliotecas, fazer compras em supermercados físicos e pela internet, e adquirir passagens aéreas. Todas essas atividades são exemplos clássicos de aplicações que utilizam bancos de dados para **manipular textos, valores, imagens** etc.

Até este momento do curso, você já sabe como montar um Diagrama ER e como fazer seu mapeamento, chegando em um formato mais próximo das tabelas que efetivamente irão compor um banco de dados. Mas, **como criar as tabelas** de fato? E, além disso, **como é possível manipular** esses dados?

Nesta agenda, você se dedicará à criação do modelo físico utilizando a linguagem **SQL ANSI**.

A **Linguagem de Consulta Estruturada**, em inglês *Structured Query Language* (SQL), nos permite solicitar uma informação ao SGBD e receber uma resposta, além de realizar modificações no banco de dados.

Preparado para mais essa etapa?

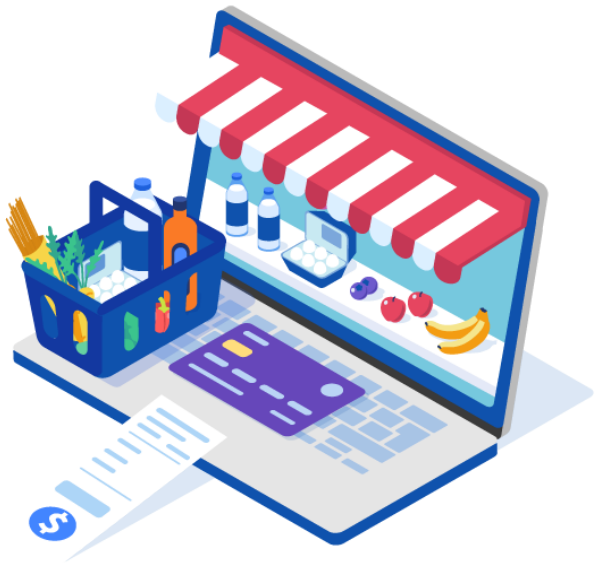
Por que Aprender?

Assim como é fundamental aprender **algoritmos para criar programas**, da mesma forma é fundamental você aprender a **Linguagem SQL para a manipular e recuperar dados**. Com ela, você pode acessar o banco de dados sem se preocupar com a linguagem de programação utilizada para desenvolver o sistema ou com o hardware utilizado. Essa linguagem é **padrão para todos os bancos de dados relacionais** e possui uma sintaxe simples, podendo recuperar dados de um banco local ou mesmo remoto.

Nesta agenda você aprenderá a utilizar ambientes e linguagens para manipulação de dados nos diversos modelos de SGBD (Sistema Gerenciador de Banco de Dados).

Para Começar o Assunto

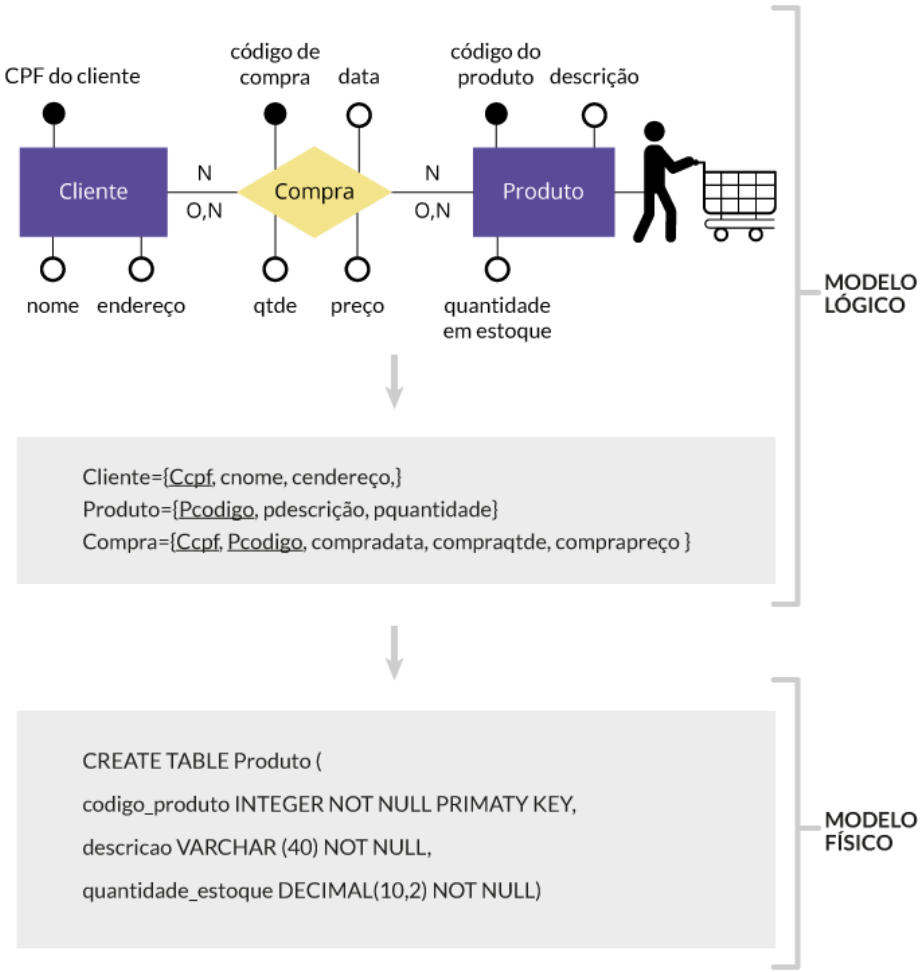
Você se lembra do minimercado do Adriano, o qual conheceu na agenda anterior?



Para abrir um minimercado em seu bairro, Adriano contratou um desenvolvedor de sistemas, Ferrari, com o objetivo de implementar um software para controle de clientes, vendas, produtos e outros dados do negócio. Ferrari fez o projeto conceitual: analisou os requisitos por meio de entrevistas e compreendeu o funcionamento do minimercado; depois, classificou e organizou os requisitos levantados e agrupo-os em entidades e relacionamentos.

Então, em conjunto com Adriano, definiram o SGBD a ser utilizado pelo banco de dados, o MySQL. A partir dessa decisão, Ferrari desenvolveu o projeto lógico, estabelecendo os atributos de cada entidade, além de seu tipo, tamanho e obrigatoriedade ou não de cada atributo. Ainda, avaliou se o atributo era chave primária ou estrangeira e se garantia a unicidade das entidades. Depois, avançou para o mapeamento do diagrama, transpondo o DER em um formato textual, mais próximo do que será utilizado no modelo físico. Nesse mapeamento, cada linha de texto representa uma tabela que existirá no banco de dados.

Agora, Ferrari precisa **desenvolver o projeto físico do minimercado** de Adriano. Para isso, deve traduzir todas as informações do modelo lógico para a linguagem SQL.



Nessa etapa, Ferrari **criará as tabelas** do banco de dados e realizará as **rotinas de inserção, exclusão, atualização e consulta dos dados** dessas tabelas. Assim que terminar esse processo, Adriano conseguirá cadastrar os produtos do mercado, atualizar os dados dos seus clientes e fornecedores, controlar e consultar os produtos que tem em estoque e as compras dos seus clientes, enfim, fazer toda a movimentação dos dados necessários para organização e controle das informações diárias do seu estabelecimento comercial.

Para conhecer a linguagem SQL e entender a sintaxe dos comandos utilizados na criação desses códigos, veja os materiais a seguir!

Mergulhando no Tema

Nas agendas anteriores, você viu que o Modelo Entidade-Relacionamento é um modelo conceitual usado para identificar como as **entidades** (pessoas, objetos ou conceitos) com seus **atributos** (propriedades e características) se relacionam entre si dentro de um sistema (**relacionamento**). Por meio de um conjunto definido de símbolos, a estrutura do BD que se pretende desenvolver é representada no Diagrama Entidade Relacionamento (DER). O modelo lógico é fortemente dependente do ambiente onde será implementado, o Sistema Gerenciador de Banco de Dados. Portanto, a construção do modelo depende da tecnologia a ser adotada (relacional, redes ou orientado a objetos). Você também viu que é necessário fazer o **mapeamento** do modelo lógico para facilitar a construção do modelo físico, seguindo algumas regras. Uma vez aplicadas essas regras, é possível implementar o Banco de Dados, ou seja, **desenvolver o Projeto Físico**. Para projetos de **BD's do tipo relacional são usadas instruções da Linguagem SQL** e, nesse material, o SGBD utilizado é o MySQL.

Instalando o MySQL

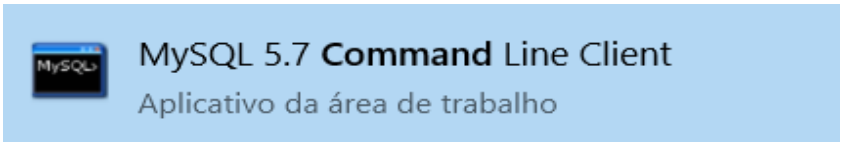
Para começar seu modelo físico, você precisará fazer o download e a instalação do **SGBD MySQL**. Nos seus estudos, recomendamos que você utilize a ferramenta gráfica **Workbench**.

O vídeo a seguir explica como fazer esse processo. Você não precisa assisti-lo por completo neste momento, acompanhe somente a instalação do MySQL. Veja a parte sobre a utilização do Workbenck quando achar necessário.

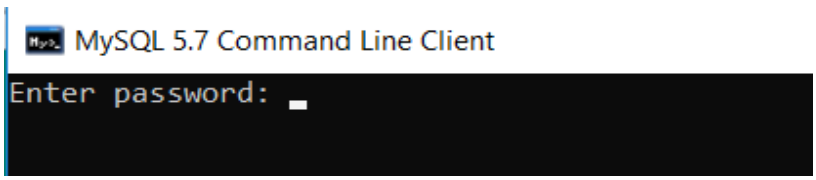
Para criação do modelo físico no MySQL, será utilizada a linguagem SQL (Structured Query Language, Linguagem de Consulta Estruturada). Conheça a origem dessa linguagem no artigo Entendendo a Linguagem SQL.

É importante que fique claro que existem vários SGBDs disponíveis, assim como interfaces gráficas que facilitam sua utilização. Nesse material você utilizará o SGBD MySQL e a interface Workbench, mas o foco é o **conhecimento da linguagem SQL**, que é utilizada em diversos outros sistemas gerenciadores. A ferramenta gráfica facilita muito a implementação de um banco de dados, mas os SGBDs também possuem interface textual, possibilitando ao desenvolvedor implementar o banco de dados por meio de comandos em SQL, o que reforça a importância do estudo da linguagem.

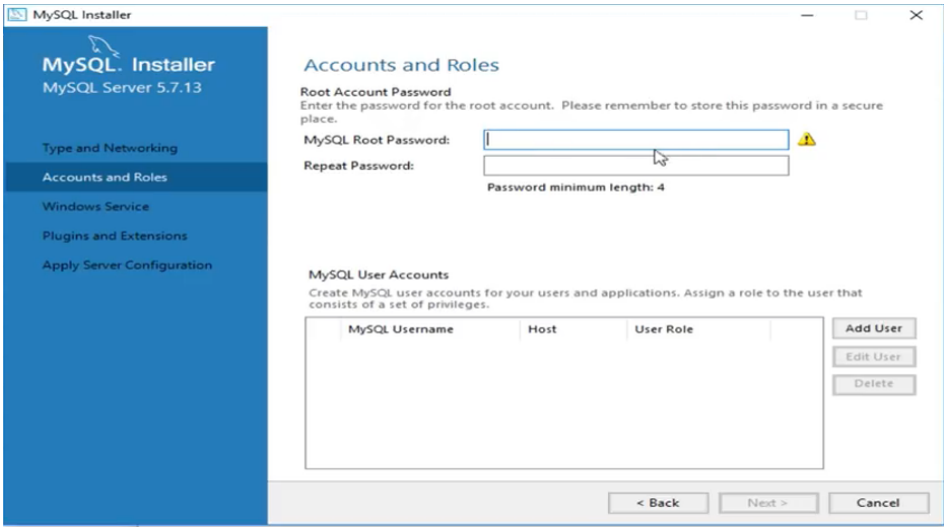
No MySQL podemos acessar a linguagem SQL pelo item:



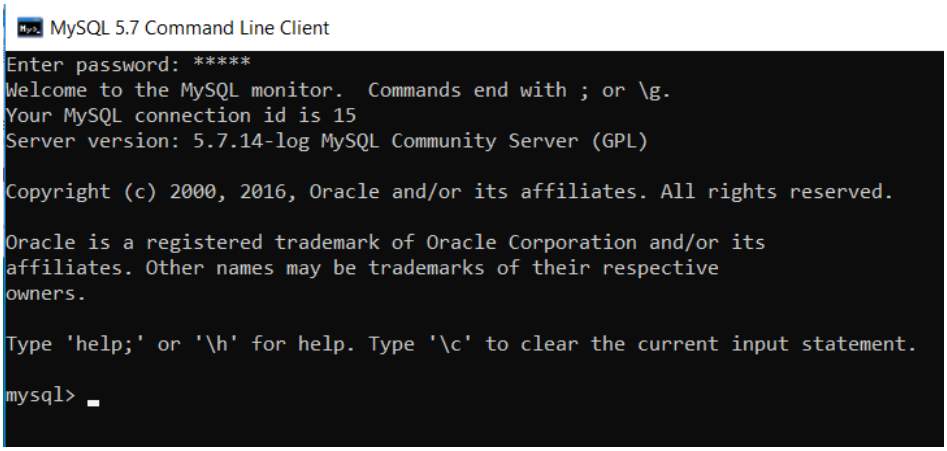
Será apresentada a seguinte interface:



A imagem a seguir apresenta a interface na qual a senha é definida no momento da instalação do MySQL, por isso é importante que você assista ao vídeo indicado anteriormente. O processo de instalação de um SGDB não deve ser rápido, ele exige atenção aos detalhes. Caso isso não aconteça, somente uma desinstalação e uma nova instalação do SGBD poderão resolver problemas futuros.



Informe a senha definida na instalação do MySQL e tecle <ENTER>:



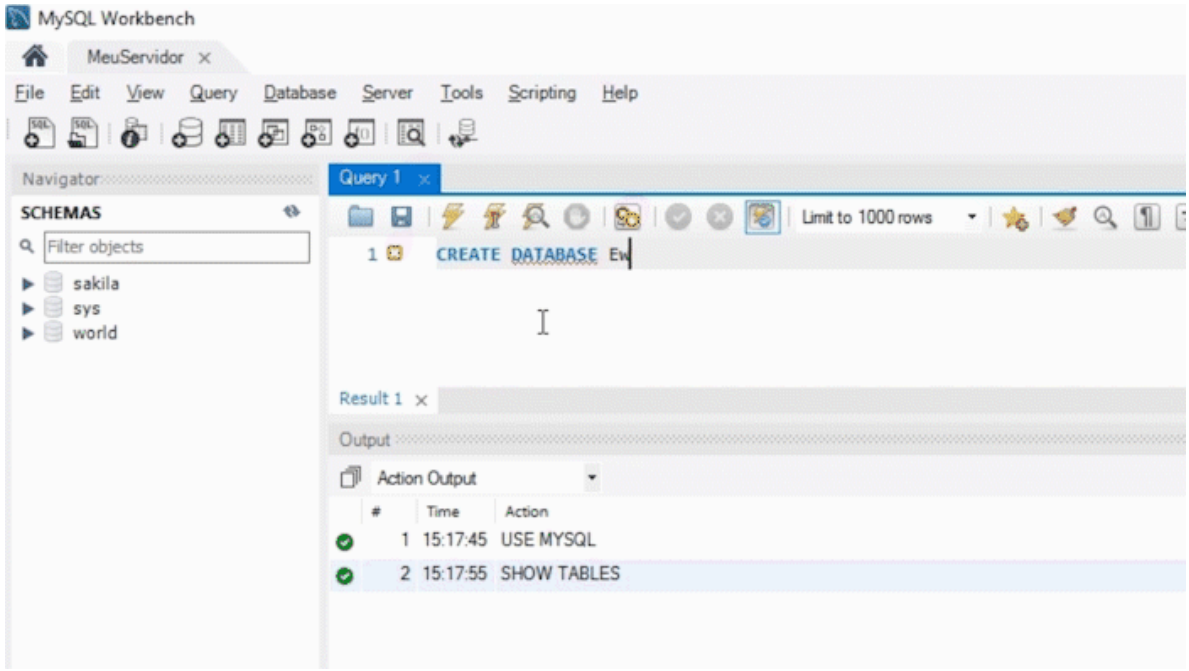
Tudo pronto! Vamos começar?!

Criando o modelo físico

A partir de agora, conheça os principais comandos utilizados para a criação de um banco de dados utilizando a linguagem SQL.

Criação do Database

O MySQL já vem com alguns bancos de dados instalados, para saber, digite: ***show databases***;



Você observou que ao final do comando foi utilizado o **; (ponto e vírgula)**? Assim como em outras linguagens, sua função no MySQL é **finalizar a linha de comando** que deverá ser executada após teclar <ENTER>. Você utilizará muito esse elemento!

A partir de agora, iremos apresentar os principais comandos utilizados para a criação do modelo físico. Para melhor entendimento, antes de cada comando será apresentada sua sintaxe. Em linguagem de programação, quando falamos de **sintaxe** nos referimos à **forma de escrever o código fonte** (palavras reservadas, comandos e recursos diversos). Os conteúdos entre os símbolos <> ou [ ] encontrados na sintaxe significam que os mesmos devem ser **substituídos** ou são **opcionais**, respectivamente.

Vamos aos principais comandos!

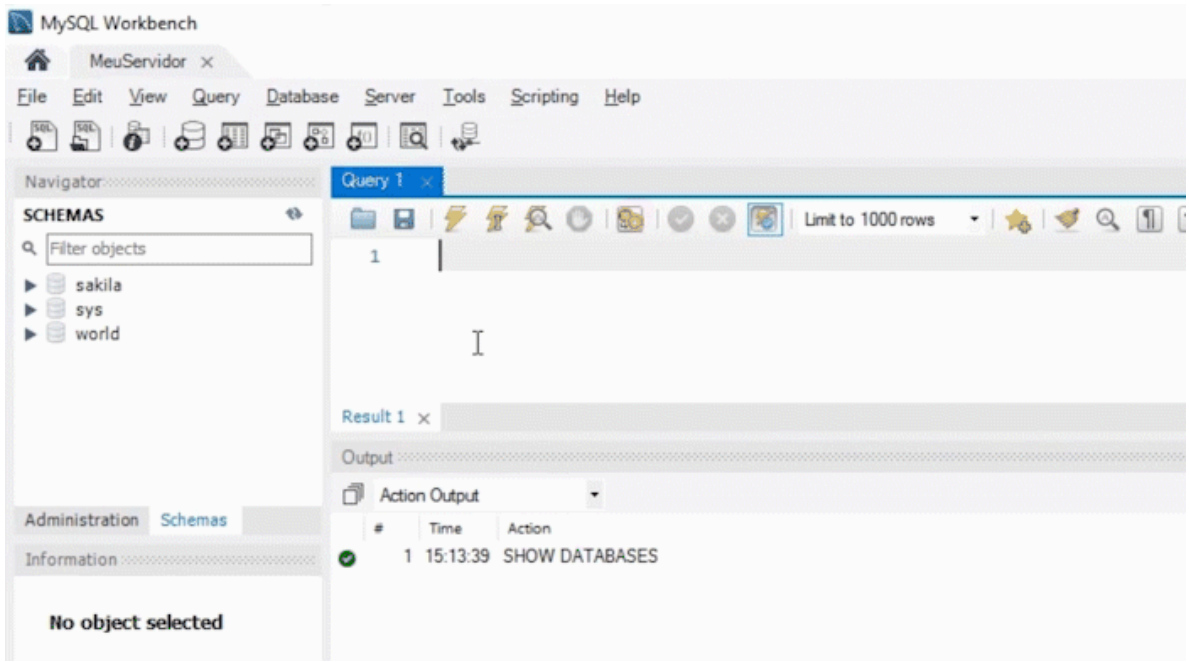
1. Para **selecionar um banco de dados** utilize o comando ***use*:**

Sintaxe:

use <nome_banco_de_dados>;

Exemplo:

use mysql;



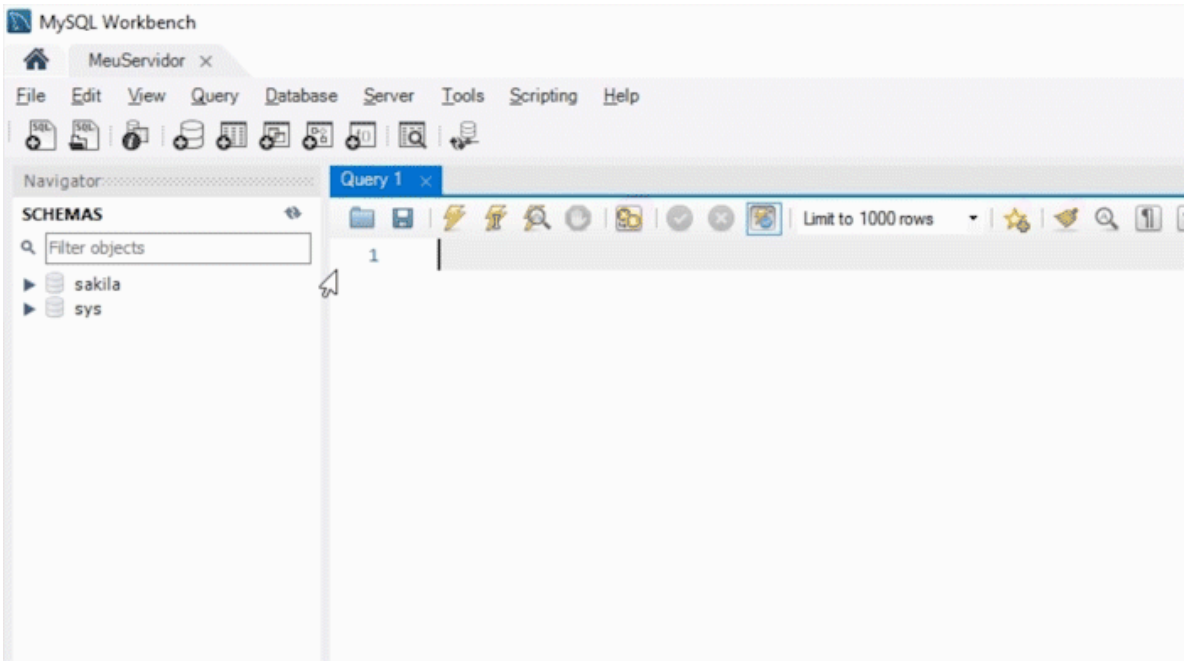
Importante:

Você já deve ter percebido que estamos utilizando somente caracteres em minúsculo, isso é importante pelo recurso **CASE SENSITIVE**, que **difere letras maiúsculas de minúsculas**. Nos computadores com S.O. (Sistema Operacional) Windows por padrão não ocorre a diferenciação, mas em computadores com S.O. Linux você poderá ter alguns problemas de adaptação: normalmente seus filesystems são CASE SENSITIVE, ou seja, essa diferenciação está presente. Existe uma forma de ajustar isso, mas o melhor é se adaptar ao tipo de plataforma que está utilizando, portanto, se você digitar as letras somente em minúsculo não terá problema!

2. Para **listar todas as tabelas** de um banco de dados utilize o comando: ***show tables*;**

Sintaxe:

show tables;



3. Para **criar um banco de dados,** utilize o comando ***create database*** ou ***create schema*****:**

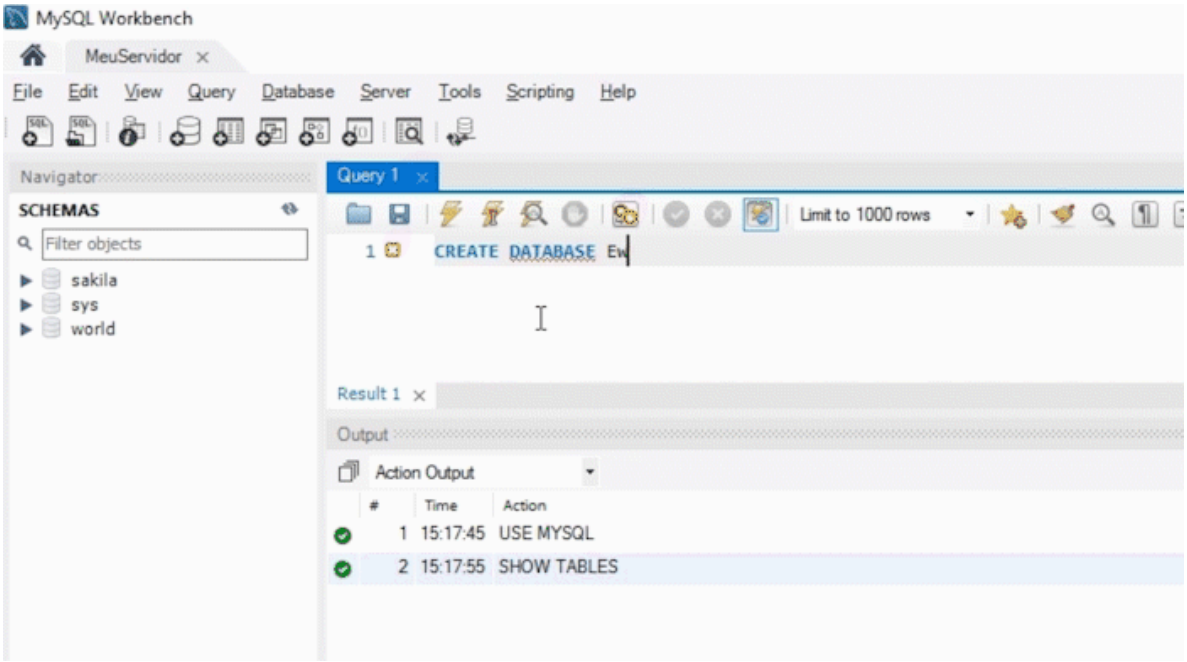
Sintaxe:

create database <nome_banco_de_dados>;

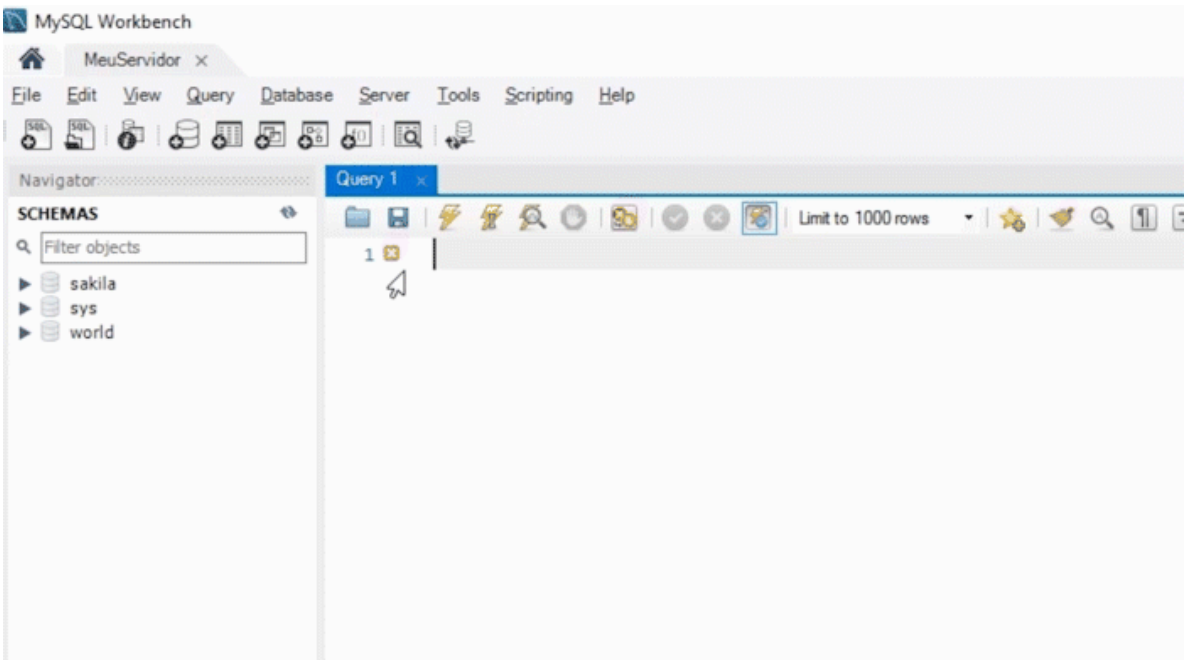
create schema <nome_banco_de_dados>;

Exemplo:

create database escola;



E então utilize o comando ***show databases*** para **visualizar o banco de dados** criado:

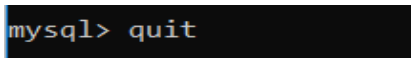


4. Para **sair da linha de comando** do MySQL, utilize o comando: ***quit***;

Sintaxe:

quit

Nota: nesse caso **não é necessário a utilização do ;**



Não se preocupe em guardar todos os comandos! Você está iniciando nesse universo e, à medida em que for utilizando-os, seu uso ficará mais intuitivo. Além disso, a ferramenta gráfica Workbench poderá te ajudar no desenvolvimento de códigos com mais clareza. Nos exemplos que você verá adiante, os comandos serão utilizados de maneira objetiva e bem simples, evitando variações complexas.

Criação de tabela

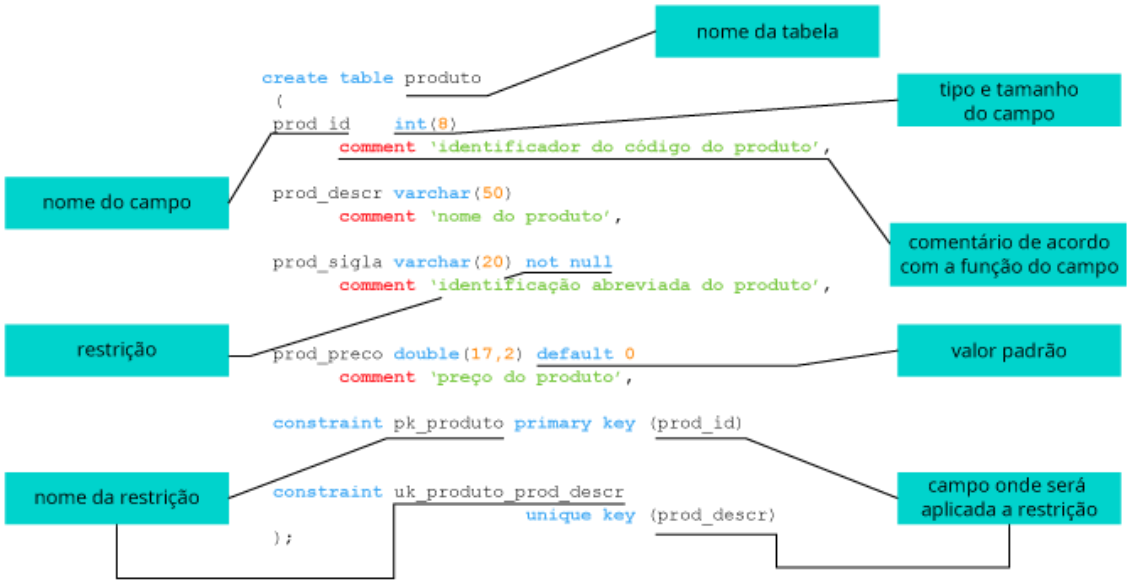
Agora que você já sabe como criar o banco, é momento de criar as **tabelas**. Para isso, será utilizada uma parte do SQL chamada DDL (Linguagem de Definição de Dados, do inglês, *Data Definition Language*) que possui comandos para **definição** e **alteração** de estruturas do banco de dados. Vamos lá?

1. Para **criar uma tabela**, utilize o comando ***create table***:

Veja a sintaxe:

```
create table <nome_da_tabela>(
campo_1 tipo(tamanho) [padrão] [restrição]
[comentário],
.
.
campo_n tipo tamanho) [padrão] [restrição]
[comentário],
[constraint <nome da restrição> primary key <campo(s) da tabela)>],
[constraint <nome da restrição> foreign key (<campo da tabela>)
references <tabela de origem> (<campo origem>)],
[constraint <nome da restrição> unique key (<campo(s) da tabela)]);
```

A imagem a seguir explica a composição da sintaxe da claúsula ***create table***.



Você reparou que no comando acima existe o termo ***unique key*** **(chave única)**? É importante que você não a confunda com chave primária (***primary key***). A chave única é utilizada quando necessitamos definir que um campo (ou um conjunto de campos) **não pode ter seu conteúdo repetido**, mas sem que esse campo seja definido como chave primária.

Agora, veja o exemplo a seguir:

```
create table funcionario (
    fnumero int(10) unsigned auto_increment
                comment 'identificador do funcionario',
    fnome varchar(80) not null
                comment 'nome do funcionário',
 endereco varchar(80) not null
                comment 'endereço do funcionário',
    salario double(10,2) default 0
                comment 'quantidade do produto em estoque',
    supernumero int(10)
                comment 'identificador do funcionário supervisor',
         dnumero int(5) not null
                comment 'identificador do departamento',
    constraint pk_funcionario primary key (fnumero)
);
```

Você pode estar se perguntando: onde estão as chaves estrangeiras da tabela funcionário? Não se preocupe, elas serão implementadas mais à frente. Ainda nesse primeiro exemplo, você pode perceber que foram inseridas algumas cláusulas como: ***unsigned, auto_increment, not null, default***. Veja seus significados:

AUTO_INCREMENT - permite que um número único seja gerado quando um novo registro é inserido em uma tabela. Em MySQL, a palavra AUTO_INCREMENT, inicia com o valor 1, e se incrementa de 1 em 1.

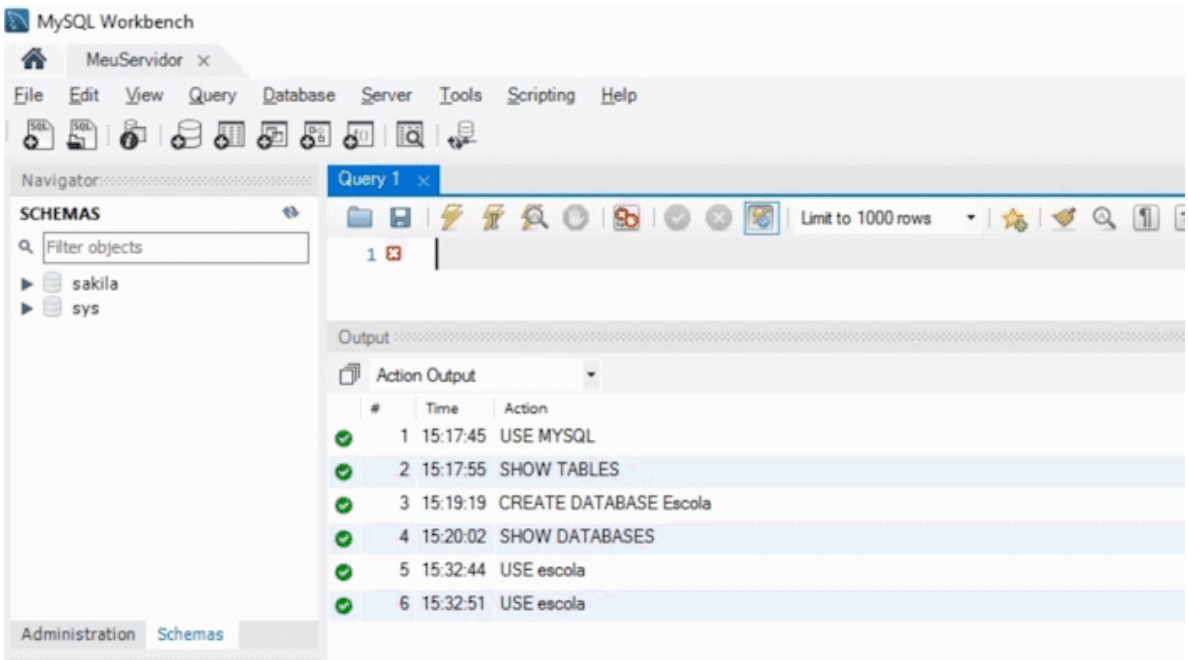
DEFAULT - Define um valor padrão que é adicionado quando nenhum outro valor é passado.

NOT NULL - Cada linha deve conter um valor para essa coluna, valores nulos não são permitidos.

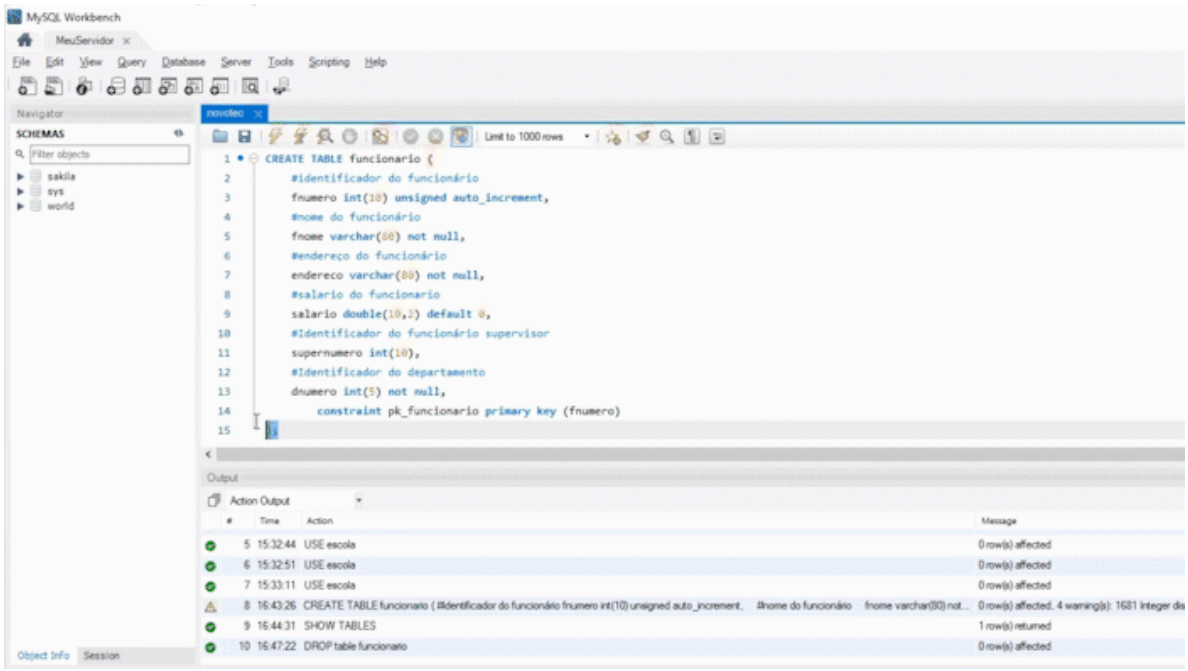
UNSIGNED - Usado para tipos numéricos, limita os dados armazenados a números positivos e zero. Por exemplo, quando queremos bloquear inserção de valores negativos em uma coluna utilizamos o parâmetro UNSIGNED.

Vale ressaltar ainda a cláusula **COMMENT**, que permite que o desenvolvedor comente os campos da tabela. Veja esse exemplo:

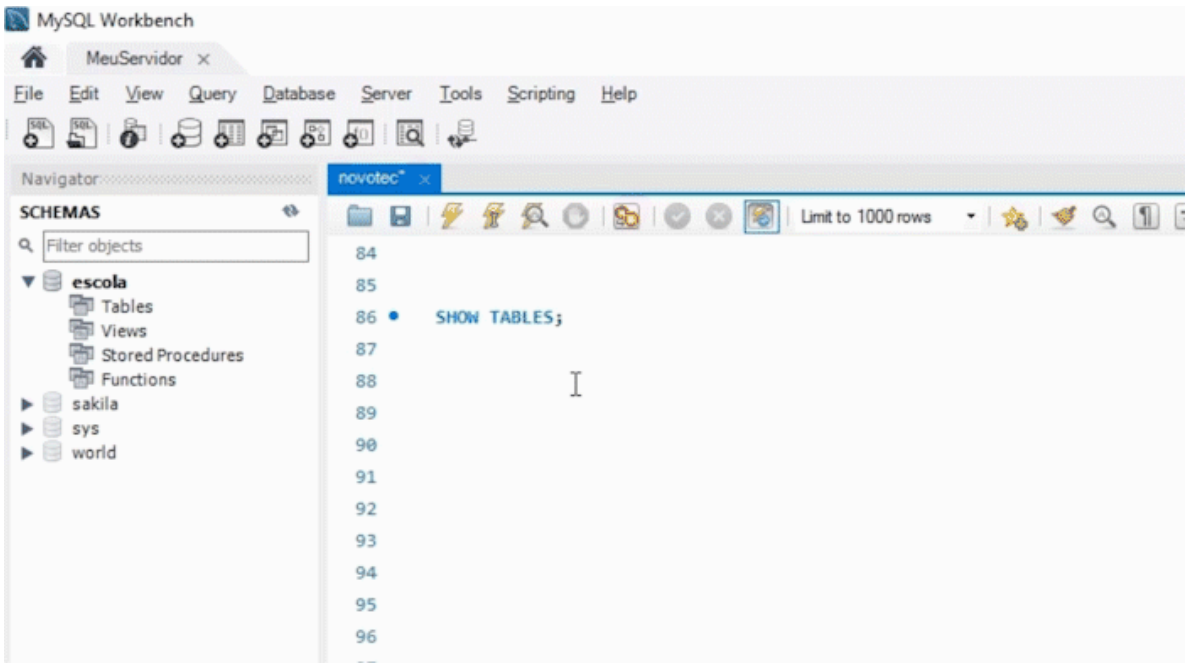
Na base de dados escola, digite os comandos a seguir, na linha de comando do MySQL:



Após a seleção do banco de dados, digite o comando para criação da tabela **funcionário**:



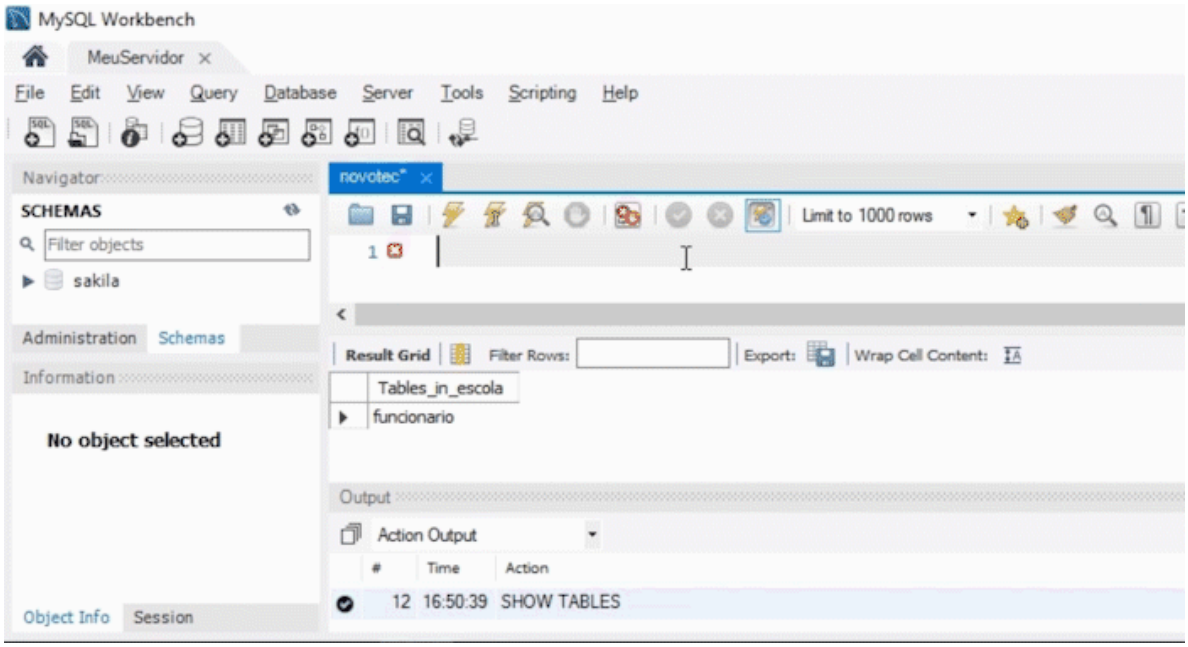
Agora execute o comando "***show tables***" para verificar a criação da tabela:



2. Para obter **maiores detalhes** da tabela utilize o comando ***describe***.

Veja a sintaxe:

describe funcionario;



No próximo exemplo, utilizaremos o conceito de **chave estrangeira**, relacionando as tabelas **departamento** e **funcionário**, e de **chave única**, não permitindo que o nome do departamento seja repetido, mesmo não sendo chave primária. Veja só:

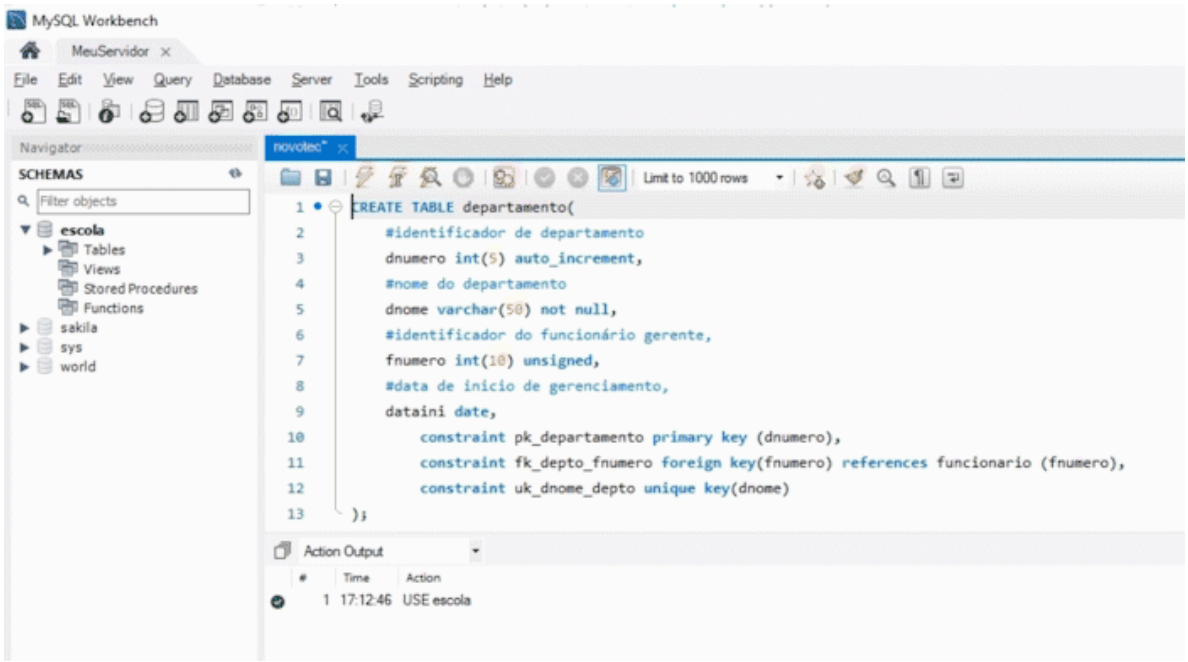
Exemplo 2:

```
create table departamento (
    dnumero int(5) auto_increment
        comment 'identificador do departamento',
    dnome varchar(50) not null
        comment 'nome do departamento',
    fnumero int(10) unsigned

        comment 'identificador do funcionário gerente',
    datainí date
        comment 'data de início do gerenciamento',
    constraint pk_departamento primary key (dnumero),

    constraint fk_depto_fnumero

        foreign key (fnumero)

        references funcionario (fnumero),

 constraint uk_dnome_depto
    unique key (dnome)

);
```

Campo da tabela definido como chave estrangeira

Tabela e campo de origem da chave estrangeira

Lembre-se de que uma **chave estrangeira deve ter o mesmo tipo e tamanho do campo da tabela de origem**. No exemplo anterior, o campo fnumero da tabela departamento, tem o mesmo tipo e tamanho do campo fnumero da tabela funcionário.



Exclusão da Tabela

Caso você precise **excluir uma tabela**, deverá utilizar o comando ***drop table***, conforme a sintaxe a seguir:

Sintaxe:

drop table <nome_da_tabela>;

O exemplo a seguir explica a composição da sintaxe da claúsula drop table, quando da exclusão de uma tabela da base de dados.

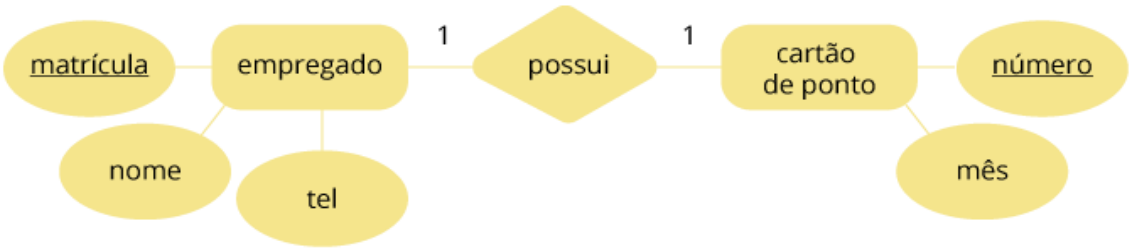
Exemplo:

Aqui também vale lembrá-lo que a execução do comando ***drop table***, sem nenhum critério ou análise mais profunda, pode causar a perda permanente de dados. No exemplo acima, todos os registros da tabela funcionário serão perdidos.

Praticando a criação de BD

Agora que você já conhece os principais comandos, é interessante que pratique criando um modelo físico. Sugerimos que você implemente as **estruturas mapeadas na situação A**, que você conheceu na unidade anterior.

Situação A:



Empregado – EmpMatrícula, EmpNome, EmpTelefone

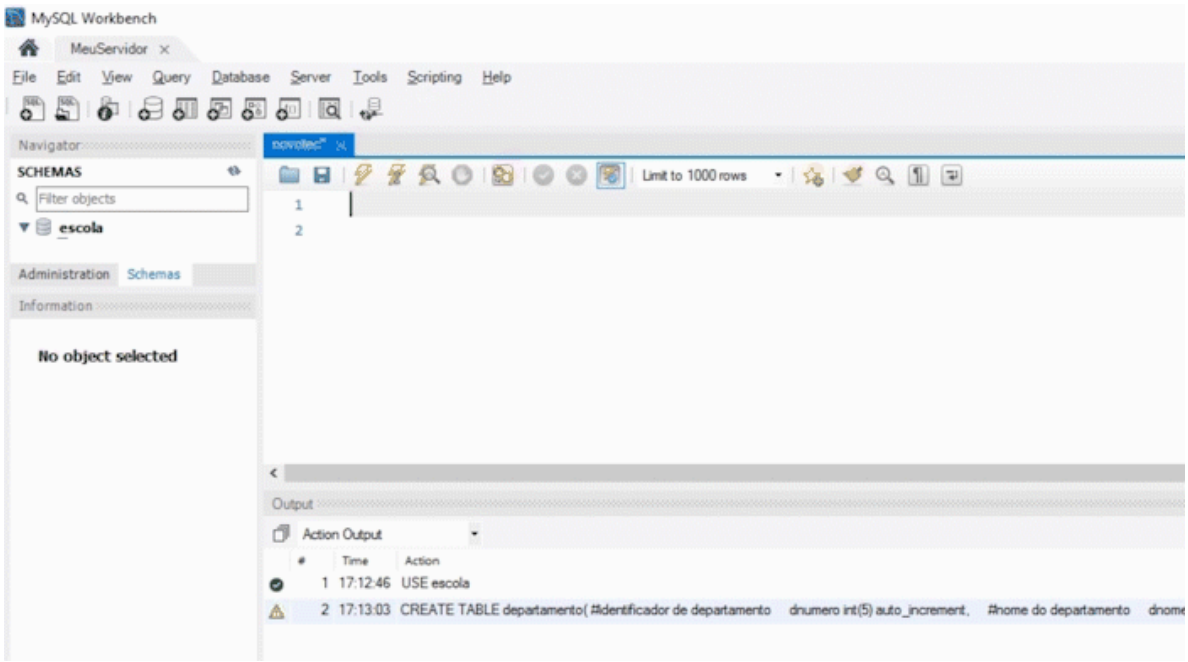
Cartão de Ponto – CpNúmero, CpMês, EmpMatrícula

Manipulando o modelo físico

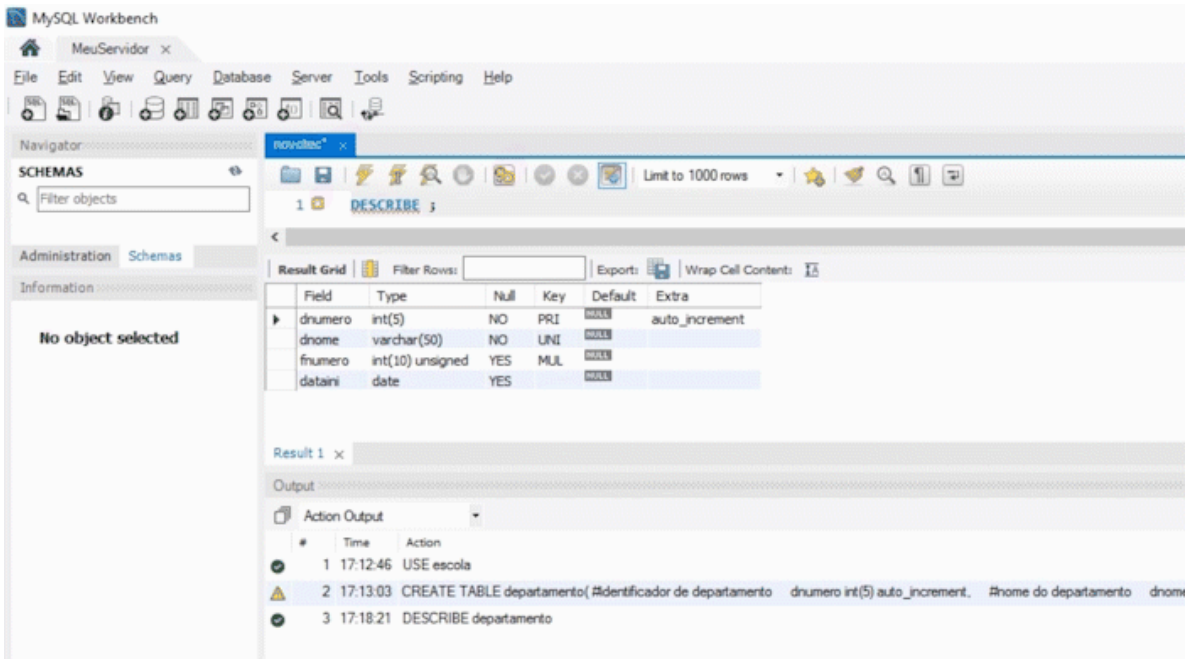
Muito bem, agora você já consegue criar um banco de dados e suas tabelas. Mas você também precisa saber manipular e consultar as informações nesse banco, não é mesmo? Agora, você conhecerá outras duas partes da linguagem da SQL: as chamadas **DML** e **DQL**. Juntas, elas possuem comandos utilizados para **inserir**, **alterar**, **excluir** e **consultar registros** nas estruturas do banco de dados.

Para demonstrar a utilização da SQL, pense no banco de dados de uma escola. Nele, existem as tabelas **funcionário** e **departamento**, que possuem as seguintes estruturas:

describe departamento;

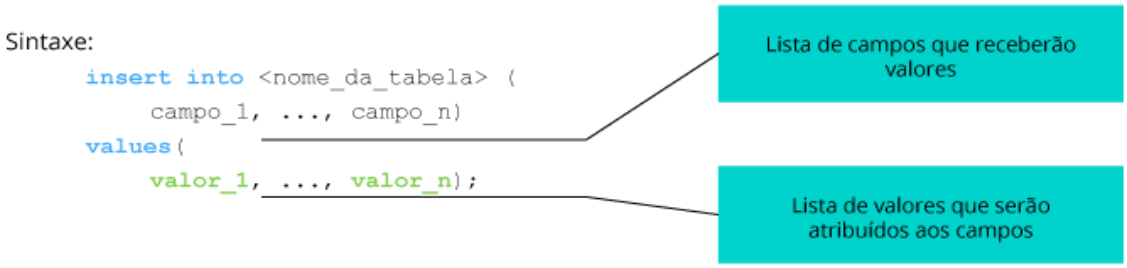


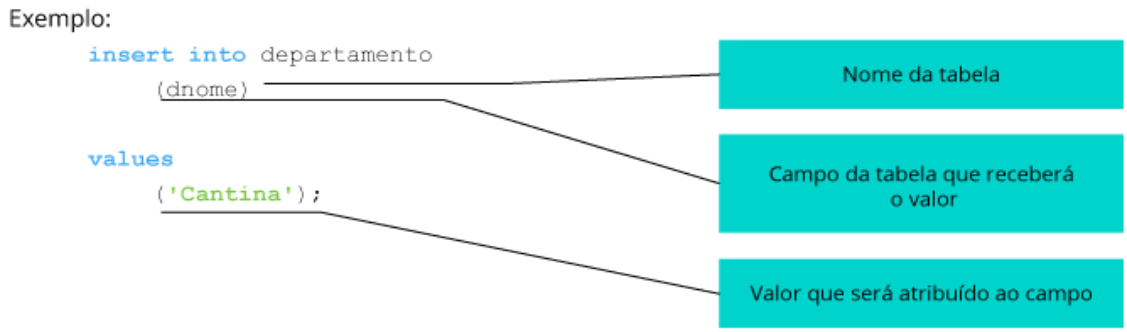
describe funcionario;



Como você já viu no início desse material, o comando ***describe*** descreve as informações da estrutura da tabela. Vamos aos comandos para **inclusão de dados**?

1. Para **inserir registros** em uma tabela utilize o comando ***insert into***:



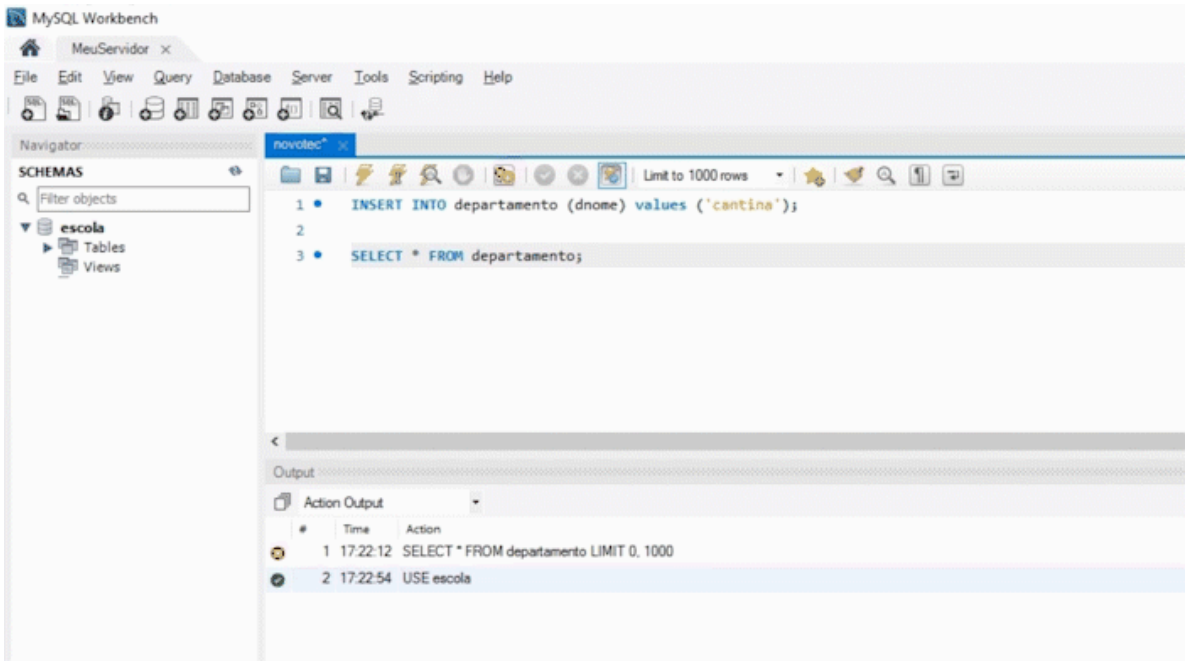


O valor do campo **dnumero não necessita ser definido** porque na criação da estrutura departamento foi definido pela cláusula ***auto_increment*** para que o conteúdo seja gerado automaticamente.

2. Para **verificar se um registro foi incluído**, utilize o comando ***select * from***:

*select * from departamento;*

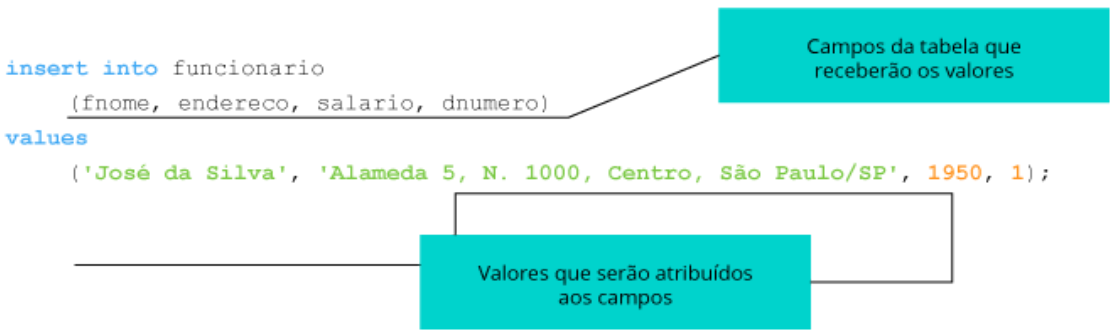
Esse comando listará todos os registros existentes na tabela departamento.



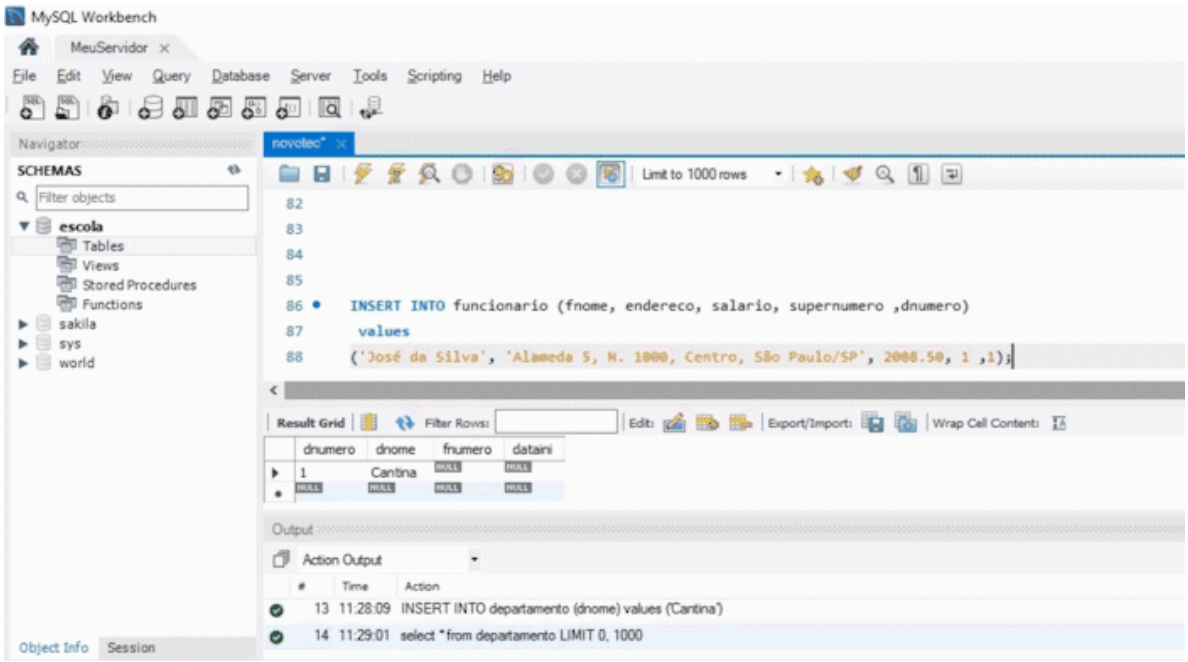
Obs: quando um campo não tem valor definido, é apresentado o valor NULL. Isso porque na estrutura departamento os campos fnumero e dataini não são obrigatórios.

Vendo mais um exemplo

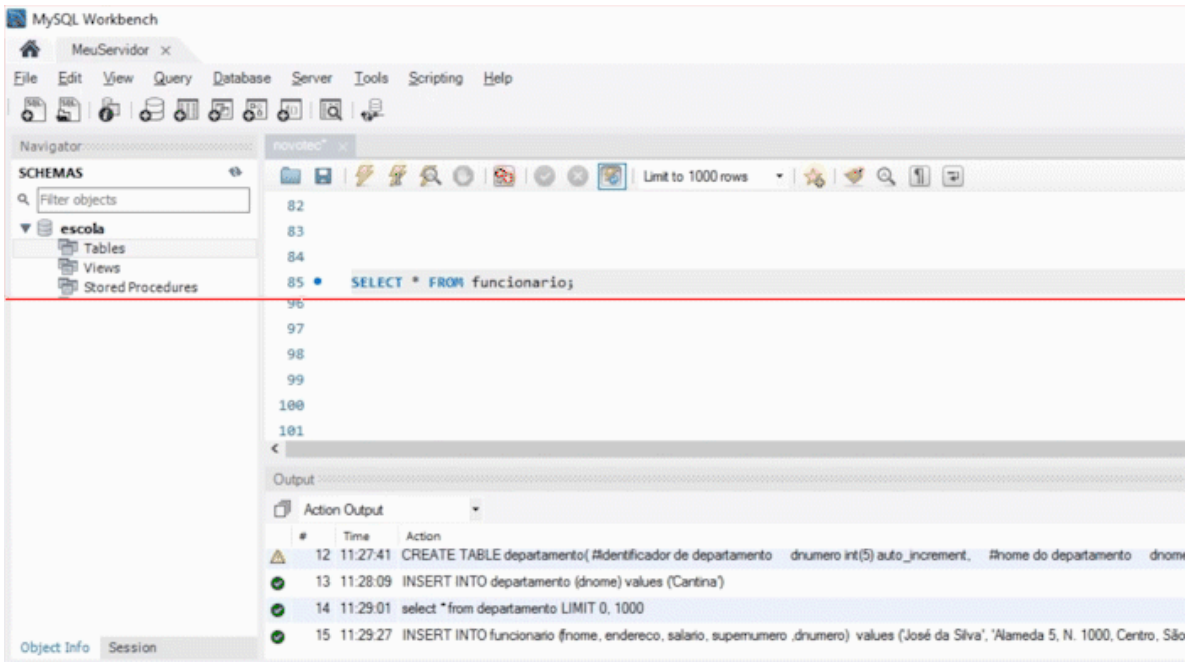
Agora veja mais um exemplo com a tabela **funcionário**, inserindo um registro de uma pessoa que trabalha no departamento **Cantina** e que não possui supervisor:



Perceba que utilizamos a vírgula para separar os **campos** e seus respectivos **valores**, que serão atribuídos às colunas de acordo com sua **ordem**, ou seja, a primeira coluna receberá o conteúdo do primeiro valor e assim sucessivamente.



Agora, execute o comando "***Select***" para verificar se os dados foram incluídos.

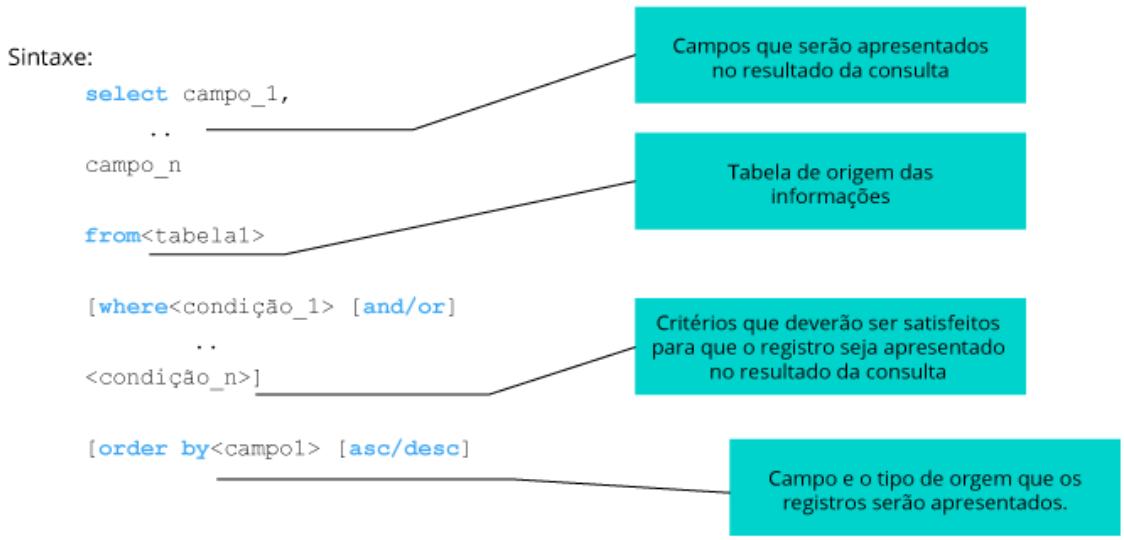


Consultas ao Banco de Dados

Legal, agora você já sabe criar as tabelas e inserir os dados dentro dela. Mas, como fazer para **consultar** os dados dentro das tabelas?

Para **consultar** o banco, utilizaremos o comando ***select***.

Com certeza, esse é um dos comandos mais utilizados do SQL. Ele faz parte de uma outra divisão da linguagem SQL, a **DQL** (Data Query Language), Linguagem de Consulta de Dados, e é utilizado quando necessitamos buscar informações no banco de dados. Vamos utilizar primeiramente uma sintaxe bem simples:



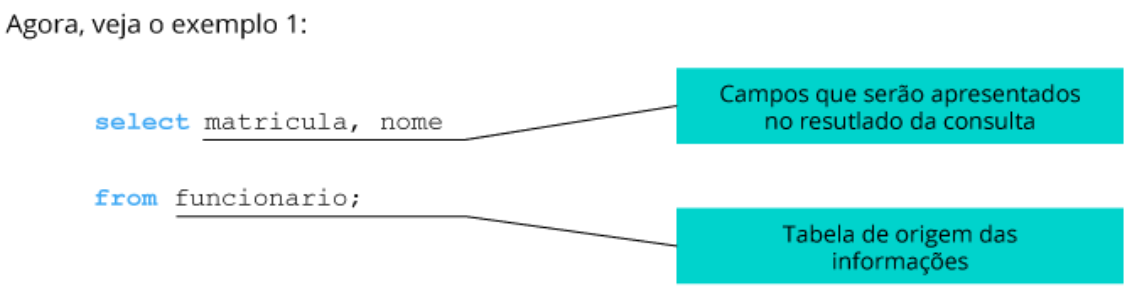
Fique atento:

ASC : significa que os resultados serão apresentados por ordem **ascendente ou crescente**.

DESC: significa que os resultados serão apresentados por ordem **descendente ou decrescente**.

Agora, conheça algumas formas de trabalhar com o comando select:

1. Selecionando todos os campos de uma tabela



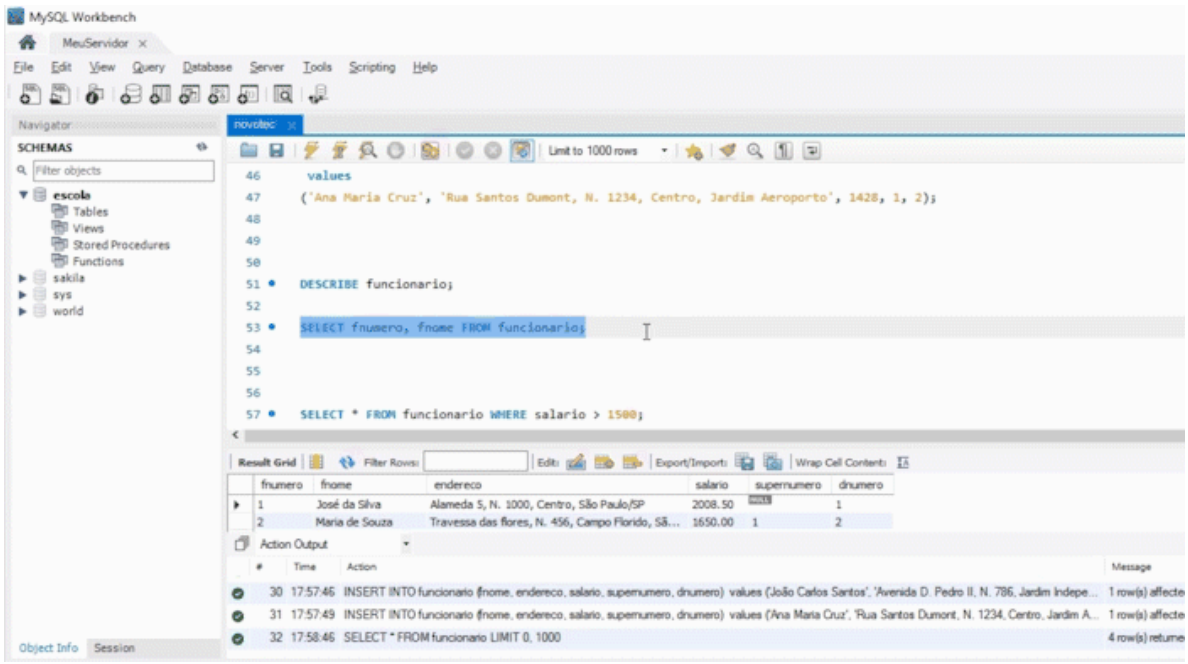
Aqui estamos selecionando apenas os campos *matricula* e *nome* da tabela *funcionário*. Caso você queira selecionar **todos os campos** da tabela funcionário, use o símbolo de **asterisco *****. Veja o

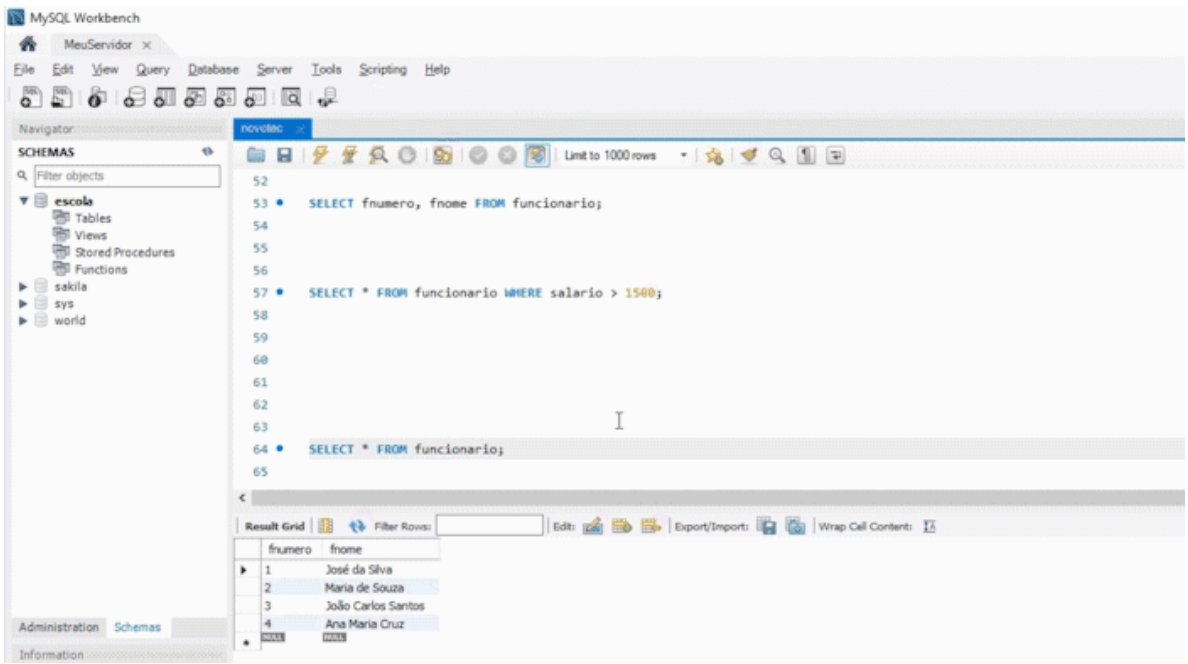
exemplo 2:

*select **

from funcionario;

Observe a diferença entre os resultados das duas consultas:



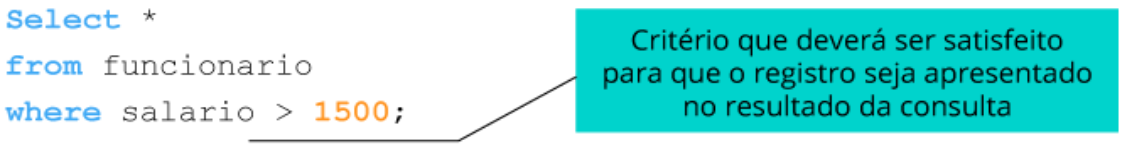


Após a linha de comando do **exemplo 2**, que utilizou **o ***, é possível ver todos os campos no resultado da consulta. O mesmo não ocorreu após a linha de comando do **exemplo 1**: perceba que somente os campos matrícula e nome foram selecionados

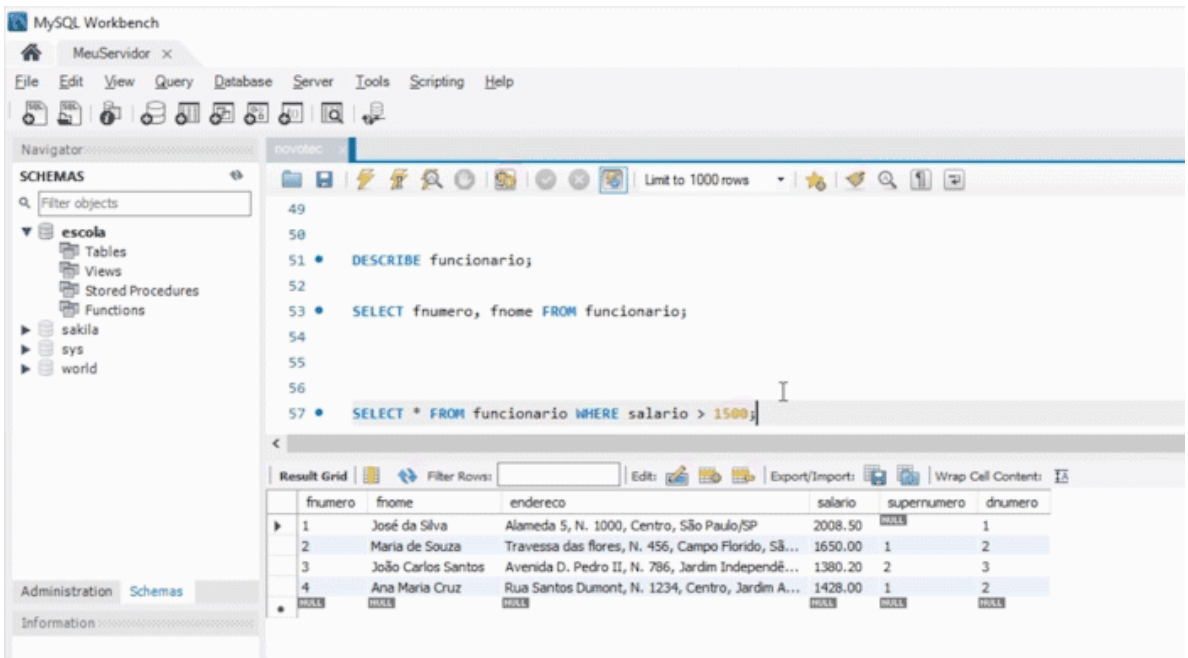
Além dos operadores lógicos, você também pode usar os o**peradores relacionais** na cláusula ***where***. Veja:

Operador	Descrição
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
!= ou <>	Diferente

Agora, observe o exemplo 3:



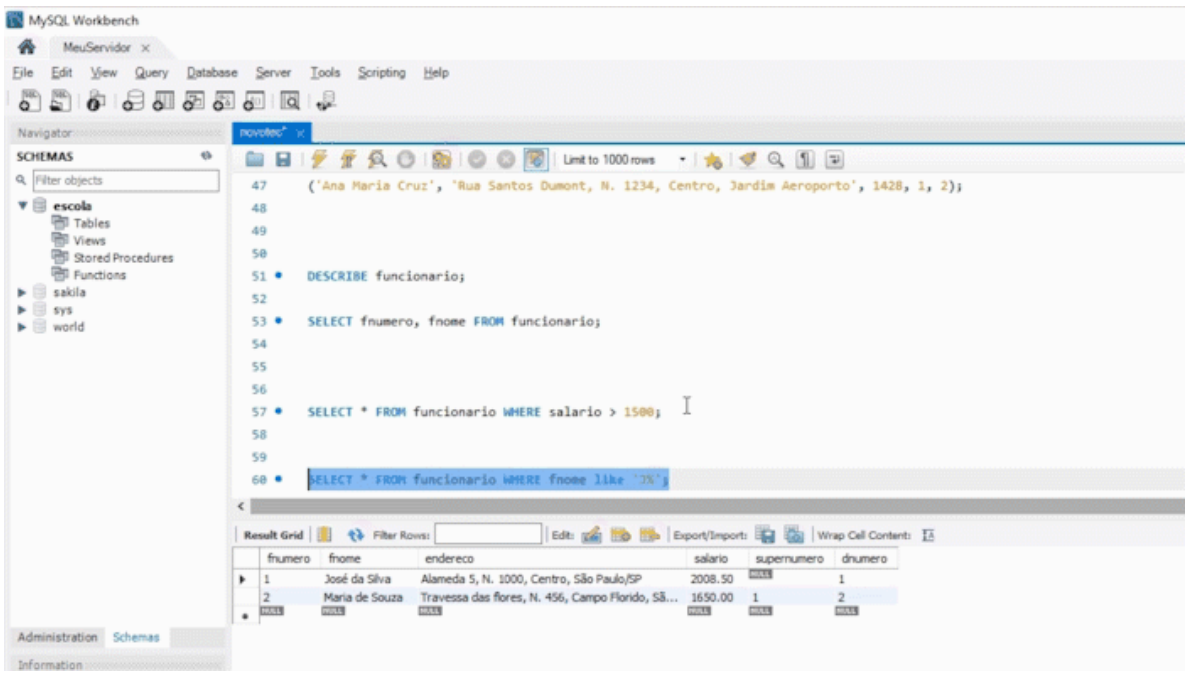
Veja o resultado:



2. Selecionando um conteúdo específico

Um outro operador SQL muito utilizado é o *like*, que permite buscar por uma determinada *string* dentro de um campo com valores textuais.

Veja o exemplo 4:

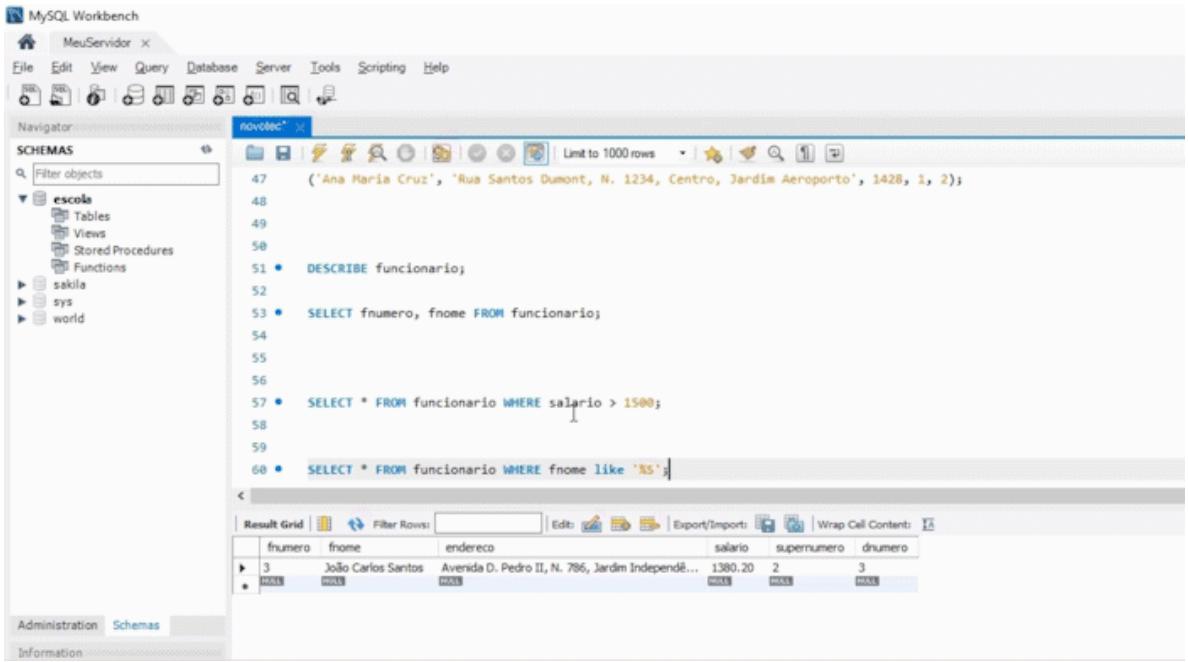


Nesse exemplo, foram selecionados todos os funcionários cujo campo *fnome* ***se inicia com a letra "J"***. O caractere "**%**" é um "**coringa**": significa que não importa quais serão os próximos caracteres. Em outras palavras, nesse exemplo o importante é que o n**ome comece com** "**J**", independente do conteúdo após essa letra. Agora, veja o exemplo 5:

*select **

from funcionario

where fnome like '%S';



Nesse exemplo o que importa é o **final do conteúdo**, nesse caso, que seja "**S**", independente do que está preenchido antes. Por fim, conheça o exemplo 6:

*select **

from funcionario

where nome like '%ri%';

No exemplo 6, o caractere coringa "**%**" foi utilizado duas vezes, **no início e no final**. Isso significa que o importante é **encontrar o nome que contenha as *strings* "ri"** independentemente do que está antes ou depois de "**ri**".

Recapitulando

Neste mergulhando, você viu como criar e manipular um banco de dados utilizando a linguagem SQL. Para sintetizar e retomar os conceitos estudados, assista à videoaula do professor Rogério Silva.

DS - Módulo 2 - Agenda 4 - Linguagem SQL

Ampliando Horizontes

Você acabou de conhecer um pouco mais de banco de dados utilizando o SGBD MySQL e a linguagem SQL. A SQL é uma linguagem muito comum no desenvolvimento de sistemas que manipulam banco de dados e seu entendimento é muito importante para quem deseja seguir nessa área. Em sua carreira, você encontrará vários outros exemplos e, principalmente, recursos a respeito da linguagem SQL que enriquecerão ainda mais seu conhecimento. Veja a seguir materiais que auxiliarão nesse aprofundamento!

Onde encontro mais informações sobre modelos físicos?

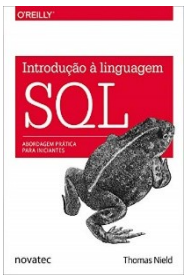
Veja a seguir algumas dicas de livros e artigos que se relacionam com o conteúdo estudado. Esses materiais são muito importantes para você!

Os nove passos do Banco de Dados (Links to an external site.)

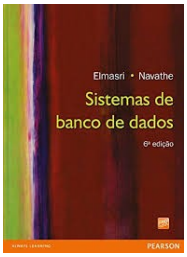
Este artigo, que trata da linguagem básica do sql, foi escrito para iniciantes na linguagem e apresenta breves definições de alguns conceitos estudados na agenda.

SQL Select: Guia para iniciantes (Links to an external site.)

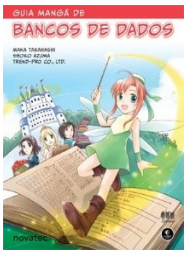
Este artigo foi preparado como um pequeno guia para consulta abordando o comando select.



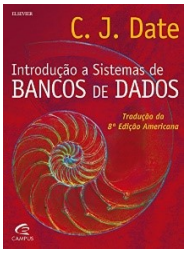
Introdução à linguagem SQL: abordagem prática para iniciantes. NIELD, Thomas. Editora Novatec, 2016.



Sistemas de bancos de dados. ELMASRI, R.; NAVATHE, S. B. 6ª edição. Editora Pearson, 2012.



Guia mangá de banco de dados. TAKAHASHI, M.; AZUMA, S. 2ª edição. Editora Novatec, 2011.



Introdução a Sistema de Banco de Dados. DATE, C. J. 8ª edição. Editora Campus, 2013.

Resumindo o Estudo

Ao longo das últimas três agendas, você percorreu todas as etapas da **modelagem de bancos de dados do tipo relacional**: aprendeu a definir o **modelo conceitual** a partir do levantamento de dados e requisitos na Agenda 10, refinou seu **modelo físico** por meio do mapeamento na Agenda 11 e, nesta agenda, conheceu e exercitou os comandos SQL mais comuns utilizados na criação e manipulação de bancos de dados, trabalhando em seu **modelo físico**. Desse modo, você **concluiu** uma importante etapa em seus estudos e alcançou a conclusão do módulo 1, parabéns!

No próximo módulo você entrará em um novo tema.

Até lá!