INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO - CAMPUS SÃO PAULO

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JUAN RIBEIRO RODRIGUES - SP309300X

**PROJETO 1**

Banco de Dados

SÃO PAULO-SP

2023

JUAN RIBEIRO RODRIGUES - SP309300X

**PROJETO 1**

Banco de Dados

Trabalho de Banco de Dados 2 apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus São Paulo, com objetivo de ser usado na avaliação semestral do curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

Orientador(a):Jose Oscar machado

SÃO PAULO-SP

2023

Sumário

[1.Introdução ao SGBD MySQL 4](#_gjdgxs)

[1.1 Ativando o Serviço 4](#_30j0zll)

[1.2 Acessando o MySQL 4](#_1fob9te)

[1.3 Guia básico para usar MySQL em qualquer sistema 4](#_3znysh7)

[1.4 Interface do MySQL 6](#_2et92p0)

[2- Tipos de DADOS NO SGBD MySQL 10](#_tyjcwt)

[1. INT 10](#_3dy6vkm)

[2. VARCHAR: 10](#_1t3h5sf)

[3. TEXT: 10](#_4d34og8)

[4. DATE: 10](#_2s8eyo1)

[5. TIME: 10](#_17dp8vu)

[6. DATETIME: 11](#_3rdcrjn)

[2.2 - Saber os tamanhos dos dados em MySQL é importante por vários motivos: 11](#_26in1rg)

[2.3 - Exemplos de tamanho de dados. 12](#_lnxbz9)

[2.4 - Diferença entre as principais variáveis: CHAR X VARCHAR 13](#_35nkun2)

[4-Idealização de um banco de dados – Trabalho 1 13](#_1ksv4uv)

[4.2 - Modelo conceitual 14](#_44sinio)

[4.3- Modelo Lógico 15](#_2jxsxqh)

[Modelo Lógico 17](#_z337ya)

[4.4 - Banco de dados 18](#_3j2qqm3)

[4.4.1 - Uso do banco – inserindo dados as tabelas 20](#_1y810tw)

[4.4.2 - Consultas mais importantes básicas e transações 20](#_4i7ojhp)

[4.4.2 - Tipos de consultas gerais 20](#_2xcytpi)

[4.4.3 -Alguns exemplos de funções comuns no MySQL: 21](#_1ci93xb)

[FUNÇÕES MATEMÁTICAS: 21](#_3whwml4)

[2. FUNÇÕES DE DATA/HORA: 21](#_2bn6wsx)

[3. FUNÇÕES DE TEXTO: 22](#_qsh70q)

[4.5 - Exemplos de Consultas 22](#_3as4poj)

[Utilitários do MySQL: Ferramentas para gerenciamento 23](#_1pxezwc)

[eficiente de banco de dados 23](#_49x2ik5)

# **1.Introdução ao SGBD MySQL**

O MySQL é um sistema de gerenciamento de banco de dados, que utiliza a linguagem SQL como interface. É atualmente um dos sistemas de gerenciamento de bancos de dados mais populares da Oracle Corporation, com mais de 10 milhões de instalações pelo mundo.

**1.1 Ativando o Serviço** - Para acessar o MySQL no laboratório do IFSP vá na aba de pesquisa digite serviços, quando abrir uma pequena interface procure por MySQL e selecione ele, em seguida, aperte para ativar o serviço.

**1.2 Acessando o MySQL** – Pesquise MySQL na aba de pesquisa abra o MySQL workbench, ele irá abrir uma aba com os bancos de dados, na maioria dos laboratórios o login do SGBD é: Estudante e a senha Estudante, esse é um padrão adotado pelo IFSP no dia 20/03/2023, onde eu escrevo esse documento, caso seja trocado o padrão é necessário consultar os professores.

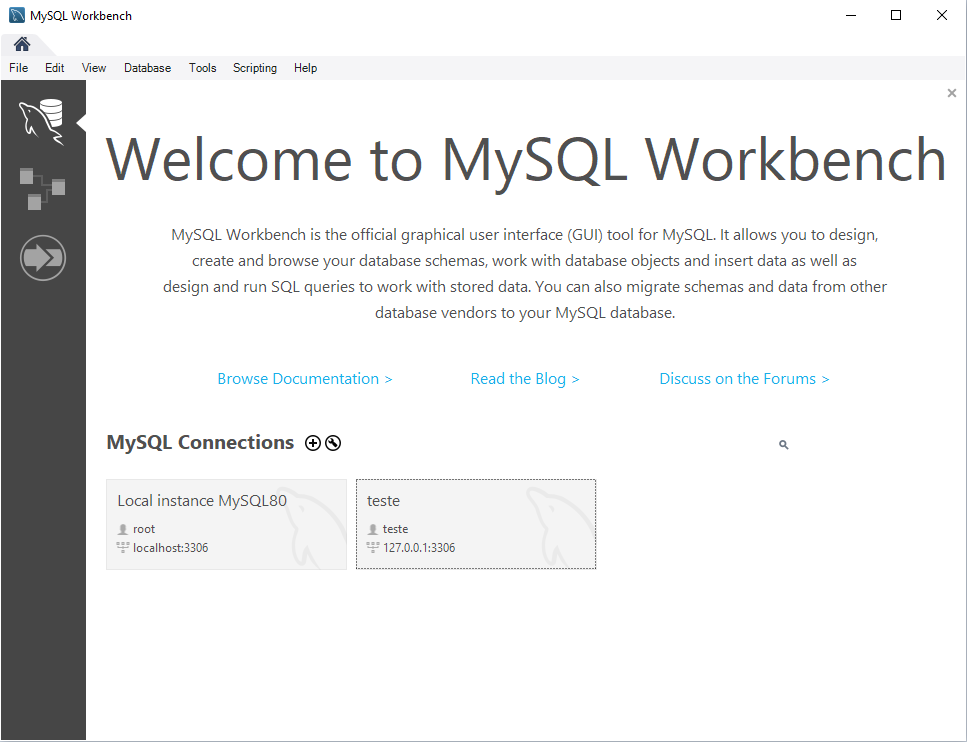
**1.3 Guia básico para usar MySQL em qualquer sistema**

1. Instale o MySQL: Antes de começar a usá-lo, você precisa instalar o software MySQL em seu computador ou servidor. Existem várias opções disponíveis, incluindo instalar o MySQL Server localmente ou usar um serviço de hospedagem na web que inclui o MySQL.
2. Inicie o servidor MySQL: Depois de instalar o MySQL, você precisará iniciar o servidor MySQL. Isso pode ser feito manualmente ou automaticamente, dependendo do seu sistema operacional e da instalação do MySQL.
3. Conecte-se ao servidor MySQL: Depois de iniciar o servidor MySQL, você precisará se conectar a ele usando um cliente MySQL. O cliente MySQL é uma ferramenta que permite que você se conecte ao servidor MySQL e execute comandos.
4. Crie um banco de dados: Depois de se conectar ao servidor MySQL, você pode criar um novo banco de dados usando o comando "CREATE DATABASE".
5. Crie tabelas: Depois de criar um banco de dados, você pode criar tabelas dentro do banco de dados. As tabelas são usadas para armazenar os dados. Você pode criar uma tabela usando o comando "CREATE TABLE".
6. Insira dados na tabela: Depois de criar uma tabela, você pode inserir dados na tabela usando o comando "INSERT INTO".
7. Recupere dados da tabela: Depois de inserir dados na tabela, você pode recuperar esses dados usando o comando "SELECT".
8. Atualize e exclua dados da tabela: Depois de inserir dados na tabela, você pode atualizá-los ou excluí-los usando os comandos "UPDATE" e "DELETE", respectivamente.

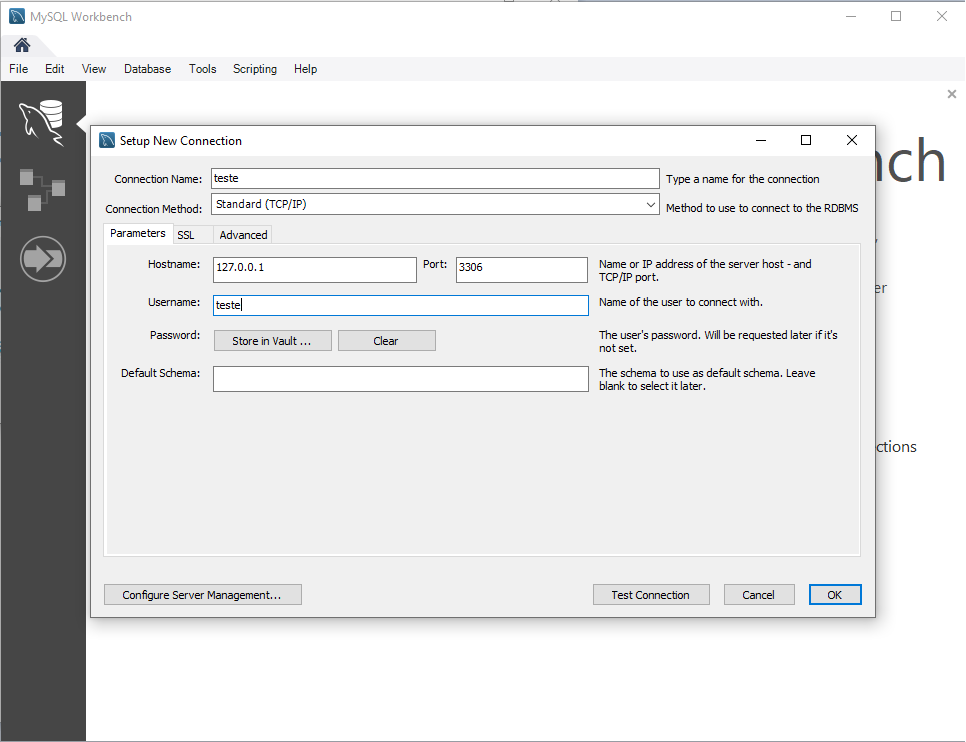
Essas são apenas algumas das funcionalidades básicas do MySQL. Há muito mais recursos avançados que podem ser usados para gerenciar e manipular dados em bancos de dados MySQL.

## **1.4 Interface do MySQL**

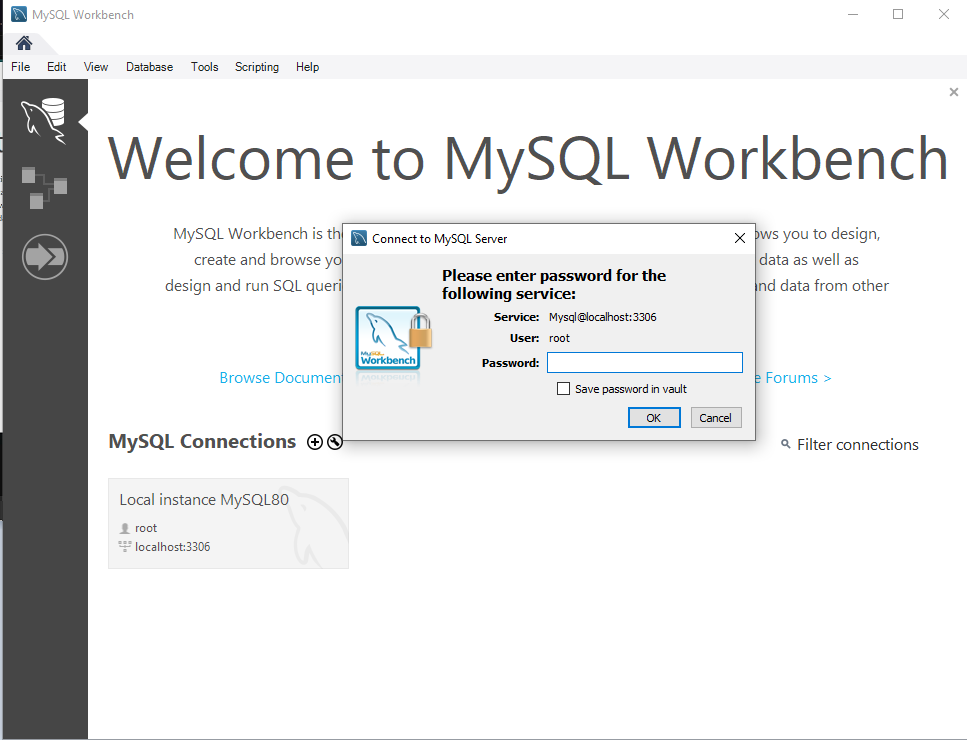
Tela inicial: local instance MySQL80 e teste, são conexão já criadas. Para criar uma vai no MYSQL Connections e aperte no +.



Aba depois de clicar no +. é necessário apenas colocar o usuário e a senha para criar a conexão.



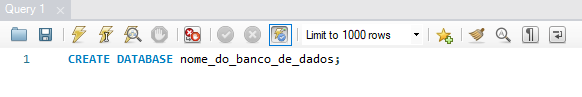
Depois de criar a conexão, é só clicar em cima do “local instance MySQL80”, que no caso foi estabelecido como nome da minha conexão. A seguir ele vai abrir essa seguinte aba, sendo necessário só colocar a senha para se conectar.

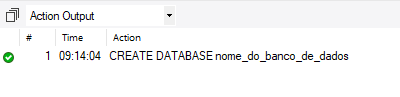


Depois de entrar no cliente MySQL, você pode criar um novo banco de dados usando o seguinte comando SQL:

CREATE DATABASE nome\_do\_banco\_de\_dados;

Para executar o comando é necessário selecionar a linha(s) e aperta no terceiro ícone(raio).

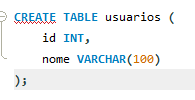




Em seguida é necessário criar as tabelas. você pode criar uma tabela usando o comando **CREATE TABLE**. Você precisa especificar o nome da tabela, bem como as colunas e seus tipos de dados. Por exemplo, vamos criar uma tabela chamada "usuarios" com duas colunas, "id" (número inteiro) e "nome" (texto):



Para usar a tabela é necessário o comando:





# **2- Tipos de DADOS NO SGBD MySQL**

O MySQL oferece uma variedade de tipos de dados para atender a diferentes necessidades de armazenamento de dados. A seguir, estão alguns dos tipos de dados mais comuns do MySQL com uma breve descrição e exemplos:

1. **INT:** O tipo de dados INT é usado para armazenar números inteiros. Ele pode ser especificado com um tamanho opcional para determinar o intervalo de valores que ele pode armazenar. Por exemplo, INT(11) pode armazenar valores de -2147483648 a 2147483647. Exemplo: CREATE TABLE users (id INT(11), nome VARCHAR(50));
2. **VARCHAR:** O tipo de dados VARCHAR é usado para armazenar cadeias de caracteres de tamanho variável. Ele requer um comprimento máximo que deve ser especificado ao criar a tabela. Exemplo: CREATE TABLE users (id INT(11), nome VARCHAR(50));
3. **TEXT:** O tipo de dados TEXT é usado para armazenar grandes blocos de texto, com um comprimento máximo de 65.535 caracteres. Exemplo: CREATE TABLE posts (id INT(11), title VARCHAR(255), content TEXT);
4. **DATE:** O tipo de dados DATE é usado para armazenar datas no formato 'yyyy-mm-dd'. Exemplo: CREATE TABLE users (id INT(11), nome VARCHAR(50), data\_nascimento DATE);
5. **TIME:** O tipo de dados TIME é usado para armazenar horários no formato 'hh:mm:ss'. Exemplo: CREATE TABLE events (id INT(11), nome VARCHAR(50), horario TIME);
6. **DATETIME:** O tipo de dados DATETIME é usado para armazenar data e horário juntos no formato 'yyyy-mm-dd hh:mm:ss'. Exemplo: CREATE TABLE events (id INT(11), nome VARCHAR(50), datahora DATETIME);

Esses são apenas alguns exemplos dos tipos de dados disponíveis no MySQL. O MySQL também suporta tipos de dados adicionais, como FLOAT, DOUBLE, DECIMAL, BOOLEAN, entre outros. A escolha do tipo de dados adequado dependerá das necessidades específicas do projeto.

## **2.2 - Saber os tamanhos dos dados em MySQL é importante por vários motivos:**

1. **Otimização de desempenho:** Quando você sabe o tamanho dos dados que está armazenando em um banco de dados MySQL, pode otimizar o desempenho do banco de dados. Isso inclui o tamanho das colunas, o tamanho das tabelas e o número de registros. Ao otimizar o tamanho dos dados, você pode reduzir o tempo necessário para executar consultas e operações no banco de dados.
2. **Economia de espaço de armazenamento:** Ao saber o tamanho dos dados que está armazenando, você pode economizar espaço de armazenamento. Isso é importante porque o espaço de armazenamento pode ser caro, especialmente em servidores com grande volume de dados. Ao economizar espaço de armazenamento, você pode reduzir o custo de manutenção do banco de dados.
3. **Evitar erros de estouro de dados:** Se você definir o tamanho máximo para uma coluna ou tabela no MySQL e os dados inseridos forem maiores do que o tamanho máximo, ocorrerá um erro de estouro de dados. Saber o tamanho dos dados que está armazenando ajuda a evitar esses erros, garantindo que os dados inseridos estejam dentro dos limites definidos.
4. **Melhoria na segurança do banco de dados:** Ao saber o tamanho dos dados que está armazenando, você pode definir limites de tamanho para prevenir ataques de injeção de dados. Isso é importante porque ataques de injeção de dados podem ser usados para acessar ou alterar dados no banco de dados, comprometendo a segurança do banco de dados.

Em resumo, saber os tamanhos dos dados em MySQL é importante para otimizar o desempenho do banco de dados, economizar espaço de armazenamento, evitar erros de estouro de dados e melhorar a segurança do banco de dados.

## **2.3 - Exemplos de tamanho de dados.**

| SGBD | variável | Tipo | Descrição | Sintaxe |
| --- | --- | --- | --- | --- |
| MySQL | char | String | Tamanho pré-estabelecido sendo mais fácil de manipular e mais rápido em comparação ao varchar | UF char(2)  “S “  “RJ” |
| MySQL | Varchar | String | Usado quando não exatamente sabe o tamanho exato de um digito. | UF Varchar (2)  “S”  “RJ” |
| MySQL | Text | String | É aceito qualquer digito, cada um valendo 1 byte por caractere, equivalente a 2.147.483.647 bytes. | TEXT (500) |
| MySQL | BigInt | Numérico | Ocupa 8 bytes, aceita valores entre e -2^63 e 2^63-1. | BIGINT UNSIGNED |

| SGBD | Variável | Tipo | Descrição | Sintaxe |
| --- | --- | --- | --- | --- |
| MySQL | Smallint | Numérico | Ocupa 2 bytes, aceita valores entre e -32768 até 32767 | SMALLINT |
| MySQL | Tinyint | Numérico | Ocupa 1 bytes, aceita valores entre e 0 até 250 | TINYINT |
| MySQL | Bit | Numérico | Os seus valores podem ser apenas nulo, 0 e 1. | BIT(8)  BOOL  BIT |

## **2.4 - Diferença entre as principais variáveis: CHAR X VARCHAR**

| **Value** | **CHAR(4)** | **Storage Required** | | |
| --- | --- | --- | --- | --- |
|  |  | PostgreSQL | MariaDB | MySQL |
| '’’ | ' ' | 4 bytes | 4 bytes | 4 bytes |
| 'ab' | 'ab ' | 4 bytes | 4 bytes | 4 bytes |
| 'abcd' | 'abcd' | 4 bytes | 4 bytes | 4 bytes |
| 'abcdefgh' | 'abcd' | 4 bytes | 4 bytes | 4 bytes |
| **Value** | **VARCHAR(4)** | **Storage Required** | | |
|  |  | PostgreSQL | MariaDB | MySQL |
| ‘’ | ' ' | 1 bytes | 1 bytes | 1 byte |
| 'ab ' | 'ab' | 3 bytes | 3 bytes | 3 bytes |
| 'abcd' | 'abcd' | 5 bytes | 5 bytes | 5 bytes |
| 'abcdefgh' | 'abcd' | 5 bytes | 5 bytes | 5 bytes |
|  |  |  |  |  |

# **4-Idealização de um banco de dados – Trabalho 1**

**4.1**- A finalidade desse banco de dados é gerenciar informações sobre as encomendas, suas distribuições, os emissores e as distribuidoras envolvidos no processo de entrega. As tabelas são relacionadas entre si através de chaves estrangeiras, permitindo consultas e análises dos dados de forma integrada.

**Principais entidades:**

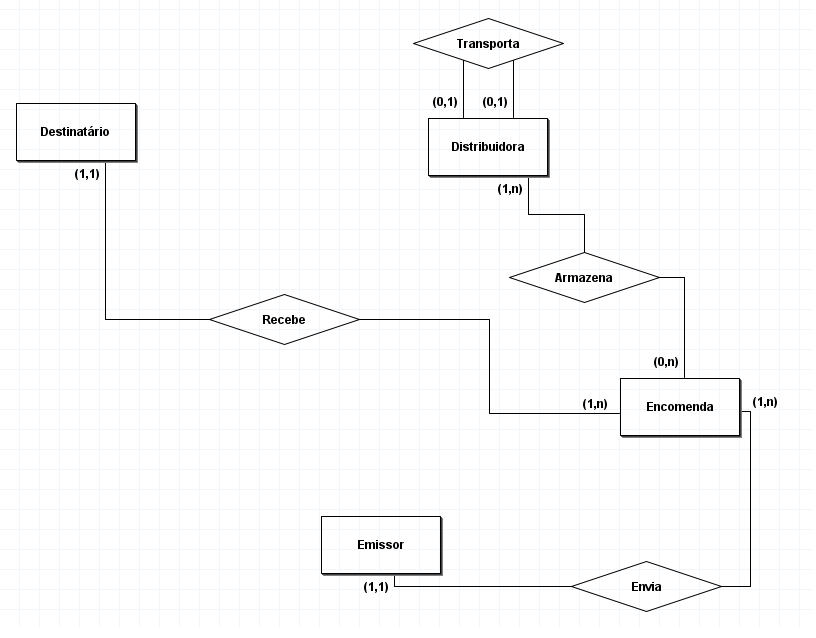
* Encomenda: contém informações sobre as encomendas, como o nome do destinatário, o volume, o peso e a situação. Também possui uma chave estrangeira que faz referência à tabela Distribuição.
* Distribuição: contém informações sobre as distribuições das encomendas, como a situação e as distribuidoras atual e de destino. Também possui duas chaves estrangeiras que fazem referência à tabela Distribuidora.
* Emissor: contém informações sobre a empresa emissora da encomenda, como o nome, o endereço e o número de telefone. Também possui uma chave estrangeira que faz referência à tabela Encomenda.
* Distribuidora: contém informações sobre as distribuidoras de encomendas, como o nome, o endereço e o horário de chegada e saída.

Alguns dos principais pontos desse banco de dados são:

* Ele permite armazenar informações de clientes, endereços e pedidos em um único lugar
* Ele usa chaves estrangeiras para relacionar as tabelas de clientes, endereços e pedidos, garantindo integridade referencial
* Ele pode ser consultado para obter informações como o histórico de pedidos de um determinado cliente, o endereço de entrega de um pedido específico, entre outras informações relevantes para o negócio.

## **4.2 - Modelo conceitual**

Este banco de dados modela um sistema de encomendas, onde uma encomenda é emitida por um emissor, é destinada a um destinatário e deve passar por uma ou mais distribuidoras até chegar ao seu destino final.



## **4.3- Modelo Lógico**

Esse banco de dados consiste em cinco tabelas relacionadas que armazenam informações sobre um sistema de entrega de encomendas.

A tabela "Encomenda" armazena informações sobre uma encomenda, como o nome e sobrenome do destinatário, volume, peso e situação da encomenda. Além disso, essa tabela tem uma chave estrangeira "IdDistribuicao" que se relaciona com a tabela "Distribuicao".

A tabela "Distribuicao" armazena informações sobre o trajeto da encomenda, incluindo a distribuidora atual, a distribuidora de destino e a situação da distribuição. Essa tabela também tem duas chaves estrangeiras "IdDistribuidoraAtual" e "IdDistribuidoraDestino" que se relacionam com a tabela "Distribuidora".

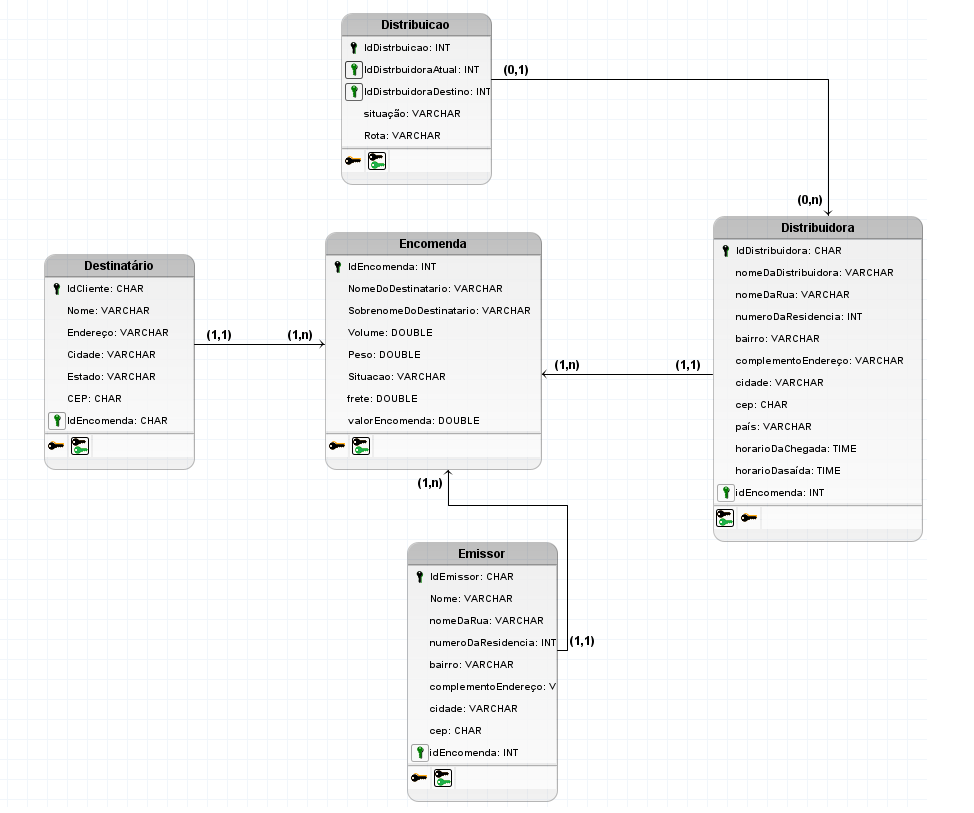
A tabela "Emissor" armazena informações sobre o emissor da encomenda, como nome, endereço e a chave estrangeira "IdEncomenda" que se relaciona com a tabela "Encomenda".

A tabela "Distribuidora" armazena informações sobre a distribuidora, como o nome, endereço e horário de chegada e saída. Essa tabela é relacionada com as tabelas "Distribuicao" e "Emissor" através de suas chaves estrangeiras.

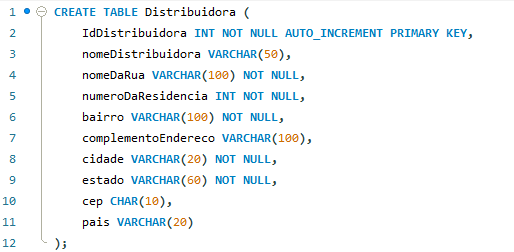
A tabela "Destinatario" armazena informações sobre o destinatário da encomenda, como nome, sobrenome, e-mail, telefone e a chave estrangeira "IdEncomenda" que se relaciona com a tabela "Encomenda".

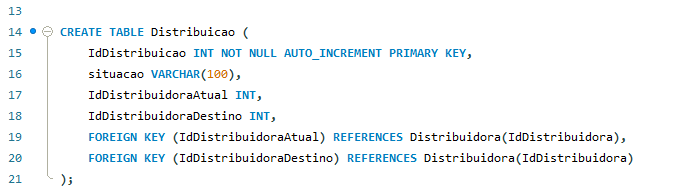
Essas tabelas em conjunto permitem armazenar informações completas sobre uma encomenda e seus envolvidos, permitindo rastrear seu trajeto e situação.

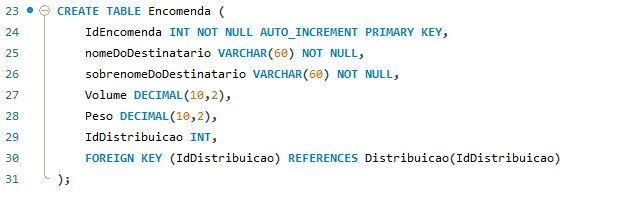
## **Modelo Lógico**

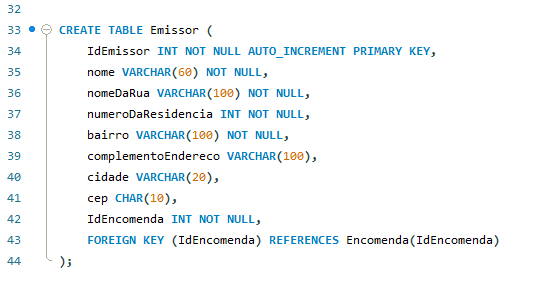


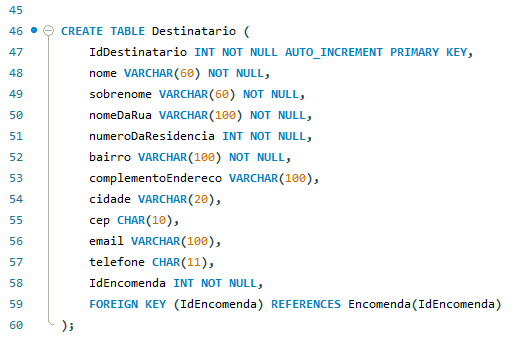
## **4.4 - Banco de dados**







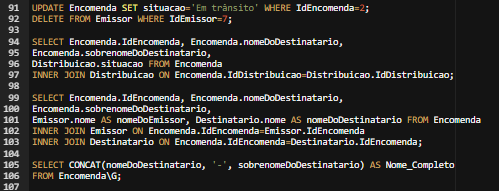




### **4.4.1 - Uso do banco – inserindo dados as tabelas**



### **4.4.2 - Consultas mais importantes básicas e transações**



### **4.4.2 - Tipos de consultas gerais**

1. Existem várias consultas que podem ser feitas em SQL, mas algumas das mais comuns e importantes são:
2. **SELECT** - Essa é a consulta mais utilizada em SQL, ela é usada para selecionar colunas e registros de uma ou mais tabelas.
3. **WHERE** - A cláusula WHERE é usada para filtrar os registros que atendem a uma determinada condição.
4. **JOIN** - O JOIN é usado para combinar dados de duas ou mais tabelas com base em uma coluna em comum.
5. **ORDER BY** - Essa cláusula é usada para classificar os resultados em ordem crescente ou decrescente com base em uma ou mais colunas.
6. **GROUP BY** - A cláusula GROUP BY é usada para agrupar os resultados com base em uma ou mais colunas e aplicar uma função de agregação, como SUM, COUNT ou AVG.
7. **HAVING** - A cláusula HAVING é usada para filtrar os resultados de uma consulta que contém uma cláusula GROUP BY.
8. **DISTINCT** - A palavra-chave DISTINCT é usada para retornar valores exclusivos de uma determinada coluna.
9. **UNION** - O UNION é usado para combinar os resultados de duas ou mais consultas SELECT em uma única tabela.

### **4.4.3 -Alguns exemplos de funções comuns no MySQL:**

#### **FUNÇÕES MATEMÁTICAS:**

* ABS(x): Retorna o valor absoluto de x.
* CEILING(x): Retorna o menor número inteiro que é maior ou igual a x.
* FLOOR(x): Retorna o maior número inteiro que é menor ou igual a x.
* ROUND(x, d): Arredonda o valor de x para d casas decimais.

#### **2. FUNÇÕES DE DATA/HORA:**

* NOW(): Retorna a data e hora atual do sistema.
* DATE\_FORMAT(date, format): Retorna uma string formatada com base na data especificada. Por exemplo, DATE\_FORMAT('2023-05-09', '%d/%m/%Y') retornaria '09/05/2023'.
* YEAR(date): Retorna o ano da data especificada.
* MONTH(date): Retorna o mês (como um número) da data especificada.
* DAY(date): Retorna o dia do mês da data especificada.

#### **3. FUNÇÕES DE TEXTO:**

* CONCAT(str1, str2): Concatena duas strings.
* LENGTH(str): Retorna o comprimento (em caracteres) da string especificada.
* UPPER(str): Converte a string especificada em letras maiúsculas.
* LOWER(str): Converte a string especificada em letras minúsculas.
* SUBSTRING(str, start, length): Retorna uma parte da string especificada, começando na posição start e com o comprimento especificado em length.

## **4.5 - Exemplos de Consultas**

1. Qual é a situação de todas as encomendas**?** - (FROM e SELECT)

**Resposta:** SELECT situacao FROM Encomenda;

**2.**Quais são as encomendas que estão com a situação "pendente"?

Resposta:

SELECT \*

FROM Encomenda

WHERE situacao = 'pendente';

**3.**Selecionar todas as encomendas com volume acima de 8:

**Resposta**:

SELECT \*

FROM Encomenda

WHERE Volume > 8;

**4.**Selecionar todas as encomendas com volume superior a 10 e peso inferior a 5:

**Resposta:**

SELECT \*

FROM Encomenda

WHERE Volume > 10

AND Peso < 5;

**5.** Selecionar a quantidade de encomendas por “situacao”:

**Resposta:**

SELECT situacao

COUNT (\*) AS Quantidade

FROM Encomenda

GROUP BY situacao;

**6.** Consulta para solicitar o peso médio das encomendas:

**Resposta:**

SELECT

AVG(Peso)

AS PesoMedio

FROM Encomenda;

**7.** Consulte o endereço das distribuidoras com CONCAT:

SELECT

CONCAT(nomeDaRua, ', ', numeroDaResidencia, ', ', bairro, ', ', cidade, ', ', estado)

AS endereco, nomeDistribuidora

FROM Distribuidora;

# **5.Utilitários do MySQL: Ferramentas para gerenciamento**

# **eficiente de banco de dados**

O MySQL é um dos sistemas de gerenciamento de banco de dados mais populares do mundo. Além do servidor MySQL em si, o ecossistema MySQL também oferece uma variedade de utilitários poderosos para auxiliar os administradores de banco de dados no gerenciamento eficiente de seus sistemas. Esses utilitários fornecem recursos adicionais, como backup, restauração, importação/exportação de dados e otimização de consultas. Neste trabalho, exploraremos alguns dos utilitários mais comuns do MySQL e discutiremos suas funcionalidades e benefícios.

**MySQLdump:** O MySQLdump é uma ferramenta utilizada para realizar backups lógicos do banco de dados MySQL. Ele permite extrair a estrutura do banco de dados, bem como os dados contidos nele, em um formato legível e facilmente importável. O MySQLdump é útil para criar cópias de backup, migrar dados entre servidores e replicar bancos de dados. Ele também suporta opções avançadas, como backups incrementais e paralelos, o que o torna uma escolha popular entre os administradores de banco de dados.

**MySQLimport e MySQLexport:** Essas ferramentas permitem importar e exportar dados em massa para o banco de dados MySQL, respectivamente. O MySQLimport permite carregar dados de arquivos externos, como arquivos CSV (valores separados por vírgula) ou SQL, diretamente para tabelas do banco de dados. O MySQLexport, por outro lado, extrai dados de tabelas do banco de dados em um formato externo. Essas ferramentas são úteis para carregar grandes quantidades de dados ou para transferir dados entre diferentes bancos de dados.

**MySQLbinlog:** O MySQLbinlog é uma ferramenta que permite examinar o log binário do MySQL. O log binário contém todas as alterações que ocorreram no banco de dados, permitindo a recuperação ponto a ponto e a replicação do banco de dados. O MySQLbinlog pode ser usado para visualizar o log binário, filtrar eventos específicos, extrair consultas SQL do log e até mesmo reproduzir eventos em outro servidor. É uma ferramenta essencial para rastrear e analisar as mudanças ocorridas no banco de dados.

**MySQLcheck:** O MySQLcheck é uma ferramenta de verificação de integridade e reparo de tabelas do MySQL. Ele analisa as tabelas em busca de erros, corrupção ou inconsistências e tenta repará-las, se possível. Essa ferramenta é útil para manter a saúde do banco de dados, identificando e solucionando problemas de tabelas que possam afetar a integridade dos dados.

**MySQLtuner:** O MySQLtuner é um utilitário de código aberto que analisa e fornece recomendações para otimizar a configuração do servidor MySQL. Ele examina estatísticas, variáveis de configuração e o uso geral do servidor para sugerir ajustes que podem melhorar o desempenho e a estabilidade do banco de dados. O MySQLtuner é uma ferramenta valiosa para garantir que o servidor MySQL esteja configurado de forma otimizada para atender às necessidades específicas do sistema.

Os utilitários do MySQL desempenham um papel fundamental no gerenciamento eficiente de bancos de dados MySQL. Desde backups e restaurações até a otimização do desempenho do servidor, essas ferramentas oferecem recursos adicionais que facilitam o trabalho dos administradores de banco de dados. Ao utilizar esses utilitários, os profissionais podem automatizar tarefas, melhorar a confiabilidade dos dados e otimizar o desempenho do banco de dados, resultando em um ambiente de banco de dados mais eficiente e robusto.

# **6.Procedures**

"procedimentos armazenados" (stored procedures) do MySQL. Os procedimentos armazenados são uma forma de agrupar instruções SQL em uma unidade lógica, que pode ser chamada e executada posteriormente. Eles são usados para encapsular lógica de negócios complexa e repetitiva no banco de dados.

Um procedimento armazenado é escrito em uma linguagem de programação específica para o banco de dados, como o MySQL, usando a sintaxe do MySQL. Ele pode receber parâmetros de entrada, executar consultas e atualizações de dados, e retornar resultados.

Os procedimentos armazenados no MySQL têm várias vantagens, incluindo:

1. **Reutilização de código:** Os procedimentos armazenados podem ser chamados de várias partes do aplicativo, evitando a duplicação de código e promovendo a reutilização.
2. **Desempenho:** Os procedimentos armazenados são compilados e armazenados em cache, o que pode resultar em melhor desempenho, especialmente para tarefas complexas que são executadas repetidamente.
3. **Segurança:** Os procedimentos armazenados podem ajudar a restringir o acesso direto às tabelas e dados subjacentes, permitindo que o controle de acesso seja aplicado à lógica encapsulada no procedimento.
4. **Manutenção:** Com a lógica de negócios encapsulada em procedimentos armazenados, as atualizações e modificações podem ser feitas centralmente, facilitando a manutenção do código.

**Exemplo:** 

**Para chamar a PROCEDURE:**

CALL ConsultarEncomendasPorDistribuidora('São-Paulo');

# **7.Triggers**

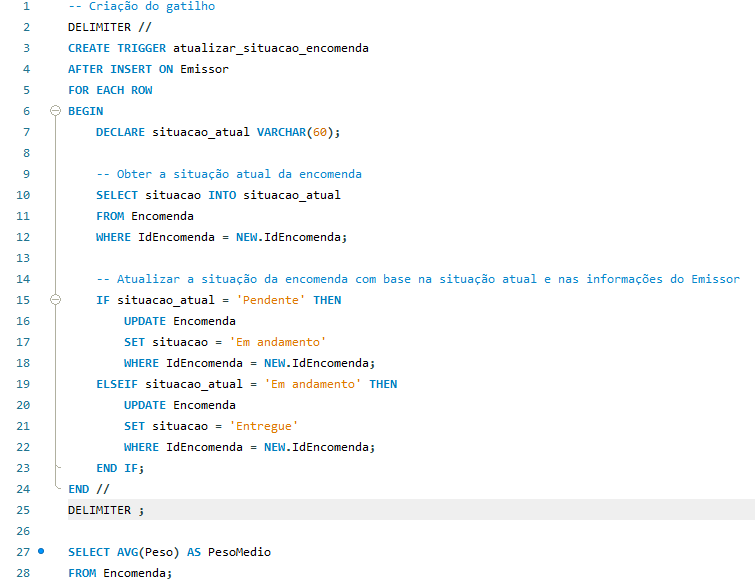
Os triggers (gatilhos) são objetos em bancos de dados que são acionados automaticamente em resposta a determinados eventos ou ações que ocorrem em uma tabela específica. Quando esses eventos são desencadeados, o trigger é executado, permitindo que você defina um conjunto de ações personalizadas para manipular os dados relacionados.

No MySQL, os triggers são escritos em SQL e podem ser definidos para responder a diferentes tipos de eventos, como inserções, atualizações ou exclusões de linhas em uma tabela. Cada trigger está associado a uma tabela específica e é disparado antes ou depois de um evento ocorrer.

Aqui estão alguns pontos importantes a serem considerados sobre os triggers no MySQL:

* Eventos disparadores: Os triggers podem ser acionados por diferentes tipos de eventos, como:
* **BEFORE INSERT:** Disparado antes de uma nova linha ser inserida na tabela.
* **AFTER INSERT:** Disparado depois de uma nova linha ser inserida na tabela.
* **BEFORE UPDATE:** Disparado antes de uma linha existente ser atualizada.
* **AFTER UPDATE:** Disparado depois de uma linha existente ser atualizada.
* **BEFORE DELETE:** Disparado antes de uma linha ser excluída da tabela.
* **AFTER DELETE:** Disparado depois de uma linha ser excluída da tabela.
* **Lógica personalizada:** Os triggers permitem que você defina uma lógica personalizada para manipular os dados relacionados ao evento disparado. Por exemplo, você pode usar um trigger AFTER INSERT para atualizar automaticamente outras tabelas com base nos dados inseridos na tabela principal.
* **Restrições e validações:** Os triggers também podem ser usados para impor restrições e validações adicionais nos dados. Por exemplo, você pode criar um trigger BEFORE INSERT para verificar se os dados inseridos atendem a determinados critérios e, se não atenderem, impedir a inserção.
* **Recursividade:** No MySQL, os triggers podem ser recursivos, o que significa que eles podem acionar outros triggers, inclusive na mesma tabela. É importante ter cuidado ao usar triggers recursivos para evitar loops infinitos ou comportamentos indesejados.
* **Gerenciamento e manutenção:** Os triggers são objetos armazenados no banco de dados e podem ser criados, modificados ou excluídos usando comandos SQL específicos, como CREATE TRIGGER, ALTER TRIGGER e DROP TRIGGER. Além disso, você pode consultar a definição e o status de um trigger usando consultas de metadados.

Os triggers são poderosas ferramentas no MySQL para automatizar ações e impor regras e restrições nos dados. No entanto, é importante usá-los com cautela e compreender completamente o impacto que eles podem ter no desempenho e na integridade dos dados do banco de dados.



# **7.NoSQL**

NoSQL, também conhecido como "Not Only SQL", é um termo utilizado para descrever um conjunto de tecnologias e abordagens de bancos de dados que diferem dos bancos de dados relacionais tradicionais (SQL). O NoSQL se baseia em modelos de dados não relacionais, projetados para lidar com grandes volumes de dados, escalabilidade horizontal e requisitos de desempenho.

Ao contrário dos bancos de dados relacionais, que armazenam dados em tabelas com esquemas definidos, o NoSQL adota diferentes modelos de dados, como:

* **Modelos de Documentos:** Nesse modelo, os dados são armazenados em documentos semiestruturados, geralmente usando formatos como JSON ou XML. Cada documento pode ter uma estrutura flexível, permitindo a inclusão de campos diferentes para cada documento.
* **Modelos de Grafos:** Esse modelo é adequado para representar relacionamentos complexos entre os dados. Os dados são armazenados como nós e arestas de um grafo, permitindo consultas eficientes de relacionamentos entre entidades.
* **Modelos de Colunas Largas:** Também conhecidos como bancos de dados de colunas, esses modelos armazenam os dados em formato de colunas, em vez de linhas. Isso permite consultas rápidas e eficientes em grandes volumes de dados.
* **Modelos de Chave-Valor:** Nesse modelo, os dados são armazenados como pares de chave-valor simples. É a forma mais simples de armazenamento de dados, oferecendo alta escalabilidade e desempenho, mas com recursos limitados de consulta e indexação.

As tecnologias NoSQL são frequentemente escolhidas quando há necessidade de lidar com dados não estruturados ou semiestruturados, alta escalabilidade horizontal (distribuição em vários servidores) e requisitos de desempenho em tempo real. Elas são amplamente utilizadas em aplicativos da web, redes sociais, análise de big data e outras áreas onde a flexibilidade e a velocidade são essenciais.

Alguns exemplos populares de bancos de dados NoSQL incluem MongoDB, Cassandra, Redis, Neo4j e Couchbase. Cada um desses bancos de dados NoSQL possui suas próprias características, pontos fortes e casos de uso adequados.

É importante ressaltar que o NoSQL não substitui completamente os bancos de dados relacionais. Em muitos cenários, ambos os tipos de bancos de dados podem ser usados em conjunto, aproveitando as vantagens específicas de cada um para atender às necessidades de um sistema de aplicação.

# **8.Referência bibliográfica**

MySQL. (2021). MySQL :: The world's most popular open source database. Recuperado em 10 de junho de 2023, de [https://www.mysql.com](https://www.mysql.com/)