



# Banco de Dados de Alta Performance

1ª  
Edição



# BANCO DE DADOS

Autoria:  
Geraldo Barbosa do Amarante



# EXPEDIENTE

**REITOR:**

PROF. CLÁUDIO FERREIRA BASTOS

**PRÓ-REITOR ADMINISTRATIVO FINANCEIRO:**

PROF. RAFAEL RABELO BASTOS

**PRÓ-REITOR DE RELAÇÕES INSTITUCIONAIS:**

PROF. CLÁUDIO RABELO BASTOS

**PRÓ-REITOR ACADÉMICO:**

PROF. HERBERT GOMES MARTINS

**DIREÇÃO EAD:**

PROF. RICARDO ZAMBRANO JÚNIOR

**COORDENAÇÃO EAD:**

PROFA. LUCIANA RODRIGUES RAMOS

**FICHA TÉCNICA****AUTORIA:**

GERALDO BARBOSA DO AMARANTE

**SUPERVISÃO DE PRODUÇÃO EAD:**

FRANCISCO CLEUSON DO NASCIMENTO ALVES

**DESIGN INSTRUACIONAL:**

ANTONIO CARLOS VIEIRA

**PROJETO GRÁFICO E CAPA:**

FRANCISCO ERBÍNIO ALVES RODRIGUES

**DIAGRAMAÇÃO E TRATAMENTO DE IMAGENS:**

ISMAEL RAMOS MARTINS

**REVISÃO TEXTUAL:**

ANTONIO CARLOS VIEIRA

**FICHA CATALOGRÁFICA****CATALOGAÇÃO NA PUBLICAÇÃO****BIBLIOTECA CENTRO UNIVERSITÁRIO ATENEU**

AMARANTE, Geraldo Barbosa do. Banco de dados. Geraldo Barbosa do Amarante. - Fortaleza: Centro Universitário Ateneu, 2020.

172 p.

ISBN: 978-65-88268-36-0

1. Sistemas Gerenciadores de Banco de Dados (SGBD). 2. Linguagem SQL. 3. Técnicas avançadas de dados. 4. Bancos de dados em alta disponibilidade e alta performance. Centro Universitário Ateneu. II. Título.

Todos os direitos reservados. Nenhuma parte desta publicação pode ser reproduzida, total ou parcialmente, por quaisquer métodos ou processos, sejam eles eletrônicos, mecânicos, de cópia fotostática ou outros, sem a autorização escrita do possuidor da propriedade literária. Os pedidos para tal autorização, especificando a extensão do que se deseja reproduzir e o seu objetivo, deverão ser dirigidos à Reitoria.



# SEJA BEM-VINDO!

Caro estudante, este é o material da disciplina *Banco de dados*, que pretende ser o guia para o estudo e entendimento da tecnologia de banco de dados, tecnologia esta que é a base para todo e qualquer sistema computadorizado. De nada adiantaria sistemas sofisticados de software se não tivéssemos onde guardar as informações produzidas por eles, de uma forma segura e disponível.

Podemos dizer que o banco de dados é a “alma da empresa”, não importando o seu porte no mundo dos negócios, seja uma empresa gigantesca, como a Amazon ou a Google, ou uma microempresa familiar, a questão da tomada de decisões baseada em dados independe do seu tamanho, é vital para a sua sobrevivência e crescimento no mercado.

A revista *The Economist*, em 2017, trouxe uma frase que transmite bem o quanto é importante gerir os dados da empresa, a frase dizia que “*o recurso mais valioso do mundo não é o petróleo, mas sim os dados*”. Isto significa que é crucial as empresas manterem as informações de qualidade sobre os seus clientes, fornecedores, suas transações contábeis, processos etc. de forma segura e disponível.

Este livro está dividido em quatro unidades. A primeira unidade tem o objetivo de dar uma visão geral dos sistemas gerenciadores de banco de dados (SGBD), fornecendo uma base conceitual. Também discutiremos o modelo relacional que é mais amplamente utilizado na maioria dos sistemas gerenciadores de banco de dados comerciais. Veremos também tópicos relacionados a um projeto de um pequeno banco de dados a ser implementado nas unidades subsequentes. Na segunda unidade, aprenderemos a instalar e configurar o SGBD MySQL, assim como a ferramenta MySQL Workbench para que seja possível desenvolver uma aplicação prática dos conceitos aprendidos utilizando a linguagem SQL. Na terceira unidade, aprofundaremos mais nas características e recursos do SGBD MySQL, para aprendermos a tornar os nossos bancos de dados mais eficientes e com alto desempenho. Na quarta e última unidade, discutiremos assuntos relacionados à distribuição de banco de dados, seu monitoramento e como melhorar o desempenho desta estrutura.

Bons estudos!

Estes ícones aparecerão em sua trilha de aprendizagem e significam:



### **ANOTAÇÕES**

Espaço para anotar suas ideias.



### **MATERIAL COMPLEMENTAR**

Texto ou mídias complementares ao assunto da aula.



### **CONECTE-SE**

Convocar o estudante para interagir no fórum tira-dúvidas.



### **MEMORIZE**

Tópico ou fato importante de lembrar.



### **CURIOSIDADE**

Informação curiosa relacionada ao conteúdo.



### **OBJETIVOS DE APRENDIZAGEM**

Objetivos de estudo do capítulo ou unidade.



### **EXERCÍCIO RESOLVIDO**

Atividade explicativa para guiar o estudante.



### **PRATIQUE**

Exercícios para fixar os conteúdos.



### **FIQUE ATENTO**

Informação complementar ao texto principal.



### **REFERÊNCIAS**

Fontes de pesquisa citadas no texto.



### **LINK WEB**

Indicação de sites.



### **RELEMBRE**

Resumo do conteúdo estudado.



# SUMÁRIO

## 01

### SISTEMAS GERENCIADORES DE BANCO DE DADOS (SGBD)

1. Visão geral sobre sistemas gerenciadores de banco de dados .....	8
2. Tipos de banco de dados .....	14
3. O modelo relacional.....	14
4. Projeto de banco de dados - normalização .....	20
5. As formas normais mais utilizadas.....	23
6. O dicionário de dados.....	32
7. Aplicando a normalização .....	37
8. O diagrama de entidades e relacionamentos.....	42
Referências .....	47

### O SISTEMA GERENCIADOR DE BANCO DE DADOS MYSQL E A LINGUAGEM SQL

1. O sistema gerenciador de banco de dados MySQL .....	50
1.1. Instalação do MySQL .....	51
1.2. Configuração .....	53
2. A linguagem SQL ( <i>structure query language</i> ) .....	59
2.1. Utilizando o MySQL para a criação e manipulação de banco de dados .....	61
2.2. Criação das tabelas do banco de dados .....	65
2.3. Criação das chaves primárias .....	68
2.4. Implementando a propriedade de AUTO INCREMENTO .....	69
2.5. Implementando os relacionamentos entre as tabelas .....	71
2.6. Manipulação dos dados.....	75
3. Tabelas temporárias.....	84
3.1. Vantagens das tabelas temporárias .....	85
3.2. Criação de tabelas temporárias.....	85
3.3. Criação de tabelas temporárias com informações vindas de um comando <i>SELECT</i> .....	86
3.4. Exclusão das tabelas temporárias.....	86
Referências .....	90

## 02

# 03

## TÉCNICAS AVANÇADAS DE DADOS

1. Particionamento.....	94
1.1. Particionamento tipo <i>RANGE</i> .....	95
1.2. Particionamento tipo <i>LIST</i> .....	97
1.3. Particionamento tipo <i>COLUMN</i> .....	98
1.4. Particionamento do tipo <i>HASH</i> .....	102
1.5. Particionamento do tipo <i>KEY</i> .....	103
2. Indexação .....	104
2.1. Classificação dos índices conforme sua estrutura .....	106
2.2. Criação de índices no MySQL .....	106
3. Objetos avançados de banco de dados .....	107
3.1. Visões ( <i>Views</i> ) .....	107
3.2. Funções ( <i>Functions</i> ) .....	112
3.3. Gatilhos ( <i>Triggers</i> ) .....	116
4. Procedimentos armazenados <i>(stored procedures)</i> .....	121
Referências .....	129

## BANCOS DE DADOS EM ALTA DISPONIBILIDADE E ALTA PERFORMANCE

1. A alta disponibilidade e a alta performance.....	132
2. A replicação.....	133
2.1. Teoria da replicação .....	135
2.2. Replicação no MySQL.....	145
3. <i>Cluster</i> .....	147
4. Monitoramento.....	151
4.1. Descrição dos processos de monitoramento e suporte .....	153
4.2. Ferramentas de monitoramento no MySQL ..	161
5. Técnicas e dicas para alta performance de banco de dados .....	166
Referências .....	171

# 04

## Unidade 02

# O SISTEMA GERENCIADOR DE BANCO DE DADOS MYSQL E A LINGUAGEM SQL

### Apresentação

O objetivo principal desta unidade é apresentar a ferramenta MySQL e a linguagem SQL, incentivando o aprendizado desta linguagem através da utilização de uma ferramenta muito utilizada nos meios empresariais, que nos permite criar um banco de dados e gerenciá-lo através do uso da ferramenta apresentada.

Nesta unidade, exploraremos a linguagem SQL através da criação de bancos de dados, da criação de tabelas, da inserção, alteração e exclusão de dados, além da implementação de chaves primárias e chaves estrangeiras para manter a integridade do banco de dados. Estudaremos também as tabelas temporárias, aprendendo a criá-las e utilizá-las e através da prática verificarmos as vantagens do seu uso.



## OBJETIVOS DE APRENDIZAGEM

- Promover a utilização da ferramenta MySQL através da instalação do servidor e da ferramenta cliente, o MySQL Workbench;
- Transformar os conceitos aprendidos em uma aplicação prática, que permita, através do uso da linguagem SQL, transformar os conhecimentos teóricos em um banco de dados com alto desempenho;
- Compreender como a linguagem SQL permite a criação de bancos de dados e tabelas e permite a manipulação de forma eficiente, através da aplicação da teoria de banco de dados na construção dos objetos de dados.

## 1. O SISTEMA GERENCIADOR DE BANCO DE DADOS MYSQL

O **SGBD MySQL** foi inicialmente desenvolvido por uma empresa sueca, cujo nome é MySQL AB e, posteriormente, adquirido pela **Oracle Corporation**. Ele é uma ferramenta de **código aberto** e que fez sucesso em função do seu desempenho, facilidade de uso e confiabilidade, e atualmente é bastante utilizado para aplicativos web. Esta é a escolha mais comum de aplicativos da web para um banco de dados relacional, e muitas empresas de sucesso, como Facebook, Twitter e Wikipedia, o utilizam.

Para utilizarmos o Sistema Gerenciador de Banco de Dados MySQL, teremos que fazer a **instalação** do software servidor e da ferramenta cliente, que será utilizada para que possamos enviar comando SQL ao SGBD. Para a instalação do software servidor (SGBD MySQL) e do software cliente (MySQL Workbench), deveremos, inicialmente, obter o instalador que faz a instalação de todos os softwares necessários no endereço: <https://dev.mysql.com/downloads/mysql/>. O programa instalador é o mysql-installer-web-community-8.0.21.0.msi, após baixá-lo do endereço informado, e ao executar, iniciaremos as fases da instalação que estão descritas a seguir.

● O SGBD MySQL foi inicialmente desenvolvido por uma empresa sueca, cujo nome é MySQL AB... ●

## 1.1. Instalação do MySQL

Os passos da instalação do SGBD MySQL são mostrados a seguir. Ele é uma instalação bem simples, não oferecendo dificuldades. Inicialmente, instalaremos o **servidor MySQL**, ou seja, o SGBD propriamente dito. Após a instalação do servidor, instalaremos a ferramenta cliente, que é o **MySQL Workbench**. No quadro a seguir, são mostradas as telas de instalação.

“ Após a instalação do servidor, instalaremos a ferramenta cliente, que é o MySQL Workbench.”

Quadro 01: Instalação do MySQL.

The screenshot shows the 'MySQL Installer' interface. On the left, a sidebar lists steps: 'Adding Community', 'Choosing a Setup Type' (which is selected), 'Installation', 'Product Configuration', and 'Installation Complete'. The main panel title is 'TELA DO PROCESSO DE INSTALAÇÃO' and the sub-section is 'Choosing a Setup Type'. It displays five options:

- Developer Default**: Installs all products needed for MySQL development purposes.
- Server only**: Installs only the MySQL Server product.
- Client only**: Installs only the MySQL Client products, without a server.
- Full**: Installs all included MySQL products and features.
- Custom**: Manually select the products that should be installed on the system.

A detailed description for 'Developer Default' states: "Installs the MySQL Server and the tools required for MySQL application development. This is useful if you intend to develop applications for an existing server." A scrollable list below includes: \* MySQL Server, \* MySQL Shell (The new MySQL client application to manage MySQL Servers and InnoDB cluster instances), and \* MySQL Router (High availability router daemon for InnoDB cluster setups to be installed on application). At the bottom right are 'Next >' and 'Cancel' buttons.

**OBSERVAÇÃO:**  
Nesta tela, são apresentados os tipos de instalação possíveis do MySQL.  
Para as nossas práticas com a linguagem SQL, o tipo de instalação **Developer Default**, que instala os produtos necessários para um servidor de desenvolvimento, será a mais recomendável.



**MySQL® Installer**  
Adding Community

Choosing a Setup Type  
Installation  
Product Configuration  
Installation Complete

**Installation**

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.21	Ready to download		
MySQL Workbench 8.0.22	Ready to download		
MySQL for Visual Studio 1.2.9	Ready to download		
MySQL Shell 8.0.22	Ready to download		
MySQL Router 8.0.22	Ready to download		
Connector/ODBC 8.0.22	Ready to download		
Connector/C++ 8.0.22	Ready to download		
Connector/J 8.0.22	Ready to download		
Connector/.NET 8.0.22	Ready to download		
Connector/Python 8.0.22	Ready to download		
MySQL Documentation 8.0.21	Ready to download		
Samples and Examples 8.0.21	Ready to download		

Click [Execute] to install the following packages.

#### OBSERVAÇÃO:

A tela apresenta os produtos que serão instalados. Se verificarmos a coluna “*Status*”, veremos que todos os produtos estão como “*Ready to download*”. Isto significa que quando selecionarmos o botão [**Execute**], o programa instalador baixará todos eles para o nosso computador para serem devidamente instalados.

**MySQL® Installer**  
Adding Community

Choosing a Setup Type  
Installation  
Product Configuration  
Installation Complete

**Installation**

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.21	Downloaded		
MySQL Workbench 8.0.22	Downloaded		
MySQL for Visual Studio 1.2.9	Complete		
MySQL Shell 8.0.22	Downloaded		
MySQL Router 8.0.22	Downloaded		
Connector/ODBC 8.0.22	Downloaded		
Connector/C++ 8.0.22	Downloaded		
Connector/J 8.0.22	Complete		
Connector/.NET 8.0.22	Installing	47%	
Connector/Python 8.0.22	Downloaded		
MySQL Documentation 8.0.21	Downloaded		
Samples and Examples 8.0.21	Downloaded		

Show Details >

#### OBSERVAÇÃO:

Nesta fase, os produtos serão baixados para o nosso computador e instalados. Na tela ao lado, alguns produtos estão com o *Status* “**Downloaded**”, ou seja, já foram baixados, existe um que está com o *Status* “**Installing**” significando que está sendo instalado e na coluna **Progress** mostra o percentual desta instalação. Outros produtos já foram instalados, portanto, estão com o *Status* “**Complete**”. Após todos os produtos estarem com o *Status* “**Complete**”, selecionaremos o botão [**Next**].

**MySQL® Installer**  
Adding Community

Choosing a Setup Type  
Installation  
**Product Configuration**  
Installation Complete

**Product Configuration**

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.21	Ready to configure
MySQL Router 8.0.22	Ready to configure
Samples and Examples 8.0.21	Ready to configure

**OBSERVAÇÃO:**  
Esta tela mostra que todos os produtos foram instalados com sucesso e estão prontos para a configuração.  
No nosso caso, estamos interessados no produto **MySQL Server 8.0.21** que é o servidor MySQL.  
Outros produtos também foram instalados como o MySQL Router 8.0.22, que não estudaremos aqui, além de um pacote de exemplos.

Fonte: Elaborado pelo autor.

## 1.2. Configuração

**Quadro 02:** Configuração do MySQL.

**TELA DO PROCESSO DE CONFIGURAÇÃO**

**MySQL® Installer**  
MySQL Server 8.0.21

Type and Networking

**Server Configuration Type**  
Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

**Config Type:** Development Computer

**Connectivity**  
Use the following controls to select how you would like to connect to this server.

<input checked="" type="checkbox"/> TCP/IP	Port: 3306	X Protocol Port: 33060
<input checked="" type="checkbox"/> Open Windows Firewall ports for network access		
<input type="checkbox"/> Named Pipe	Pipe Name: MYSQL	
<input type="checkbox"/> Shared Memory	Memory Name: MYSQL	

**Advanced Configuration**  
Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

Show Advanced and Logging Options

**OBSERVAÇÃO:**  
Aqui, poderemos configurar o tipo de servidor (*Type*) que queremos e definir as configurações de rede (*Networking*). As configurações de rede propostas pelo instalador não devem ser modificadas. No caso do tipo de servidor, para as nossas tarefas, o tipo "**Development Computer**" é o ideal. Existem mais duas opções: "**Server Computer**" e "**Dedicated Computer**". Selecionar o botão **[NEXT]**.

**MySQL® Installer**  
MySQL Server 8.0.21

Type and Networking  
Authentication Method  
Accounts and Roles  
Windows Service  
Apply Configuration

**Authentication Method**

**Use Strong Password Encryption for Authentication (RECOMMENDED)**  
MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.

 Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching\_sha2\_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

**Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)**  
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.

#### OBSERVAÇÃO:

Nesta tela, poderemos selecionar o método de autenticação, observemos que o método “**Use Strong Password Encryption for Authentication**” já vem marcado e recomendado (**RECOMMENDED**). Esta parte da configuração está ligada à segurança dos acessos de usuários ao banco de dados, definindo a forma de encriptação da senha. Seleccionaremos o botão **[NEXT]**.

**MySQL® Installer**  
MySQL Server 8.0.21

Type and Networking  
Authentication Method  
Accounts and Roles  
Windows Service  
Apply Configuration

**Accounts and Roles**

**Root Account Password**  
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:  

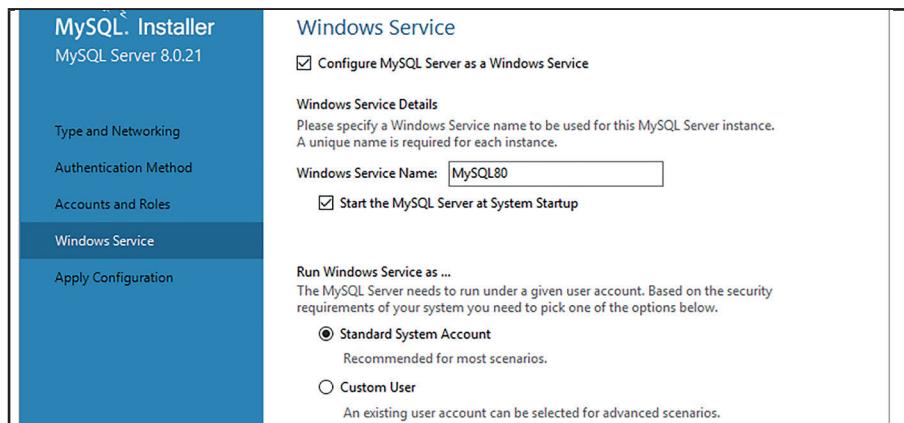
Repeat Password:

**MySQL User Accounts**  
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role	Add User
			
			

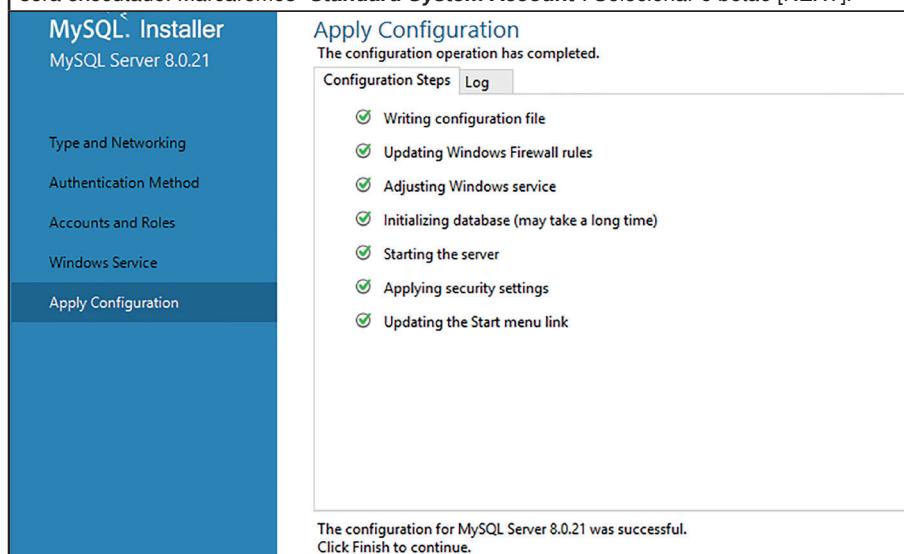
#### OBSERVAÇÃO:

Agora, teremos que definir uma senha para o usuário “root”, que é um usuário que tem todos os privilégios no MySQL, ele é o administrador do SGBD. Geralmente, criamos usuários com menores privilégios para o uso normal dos bancos de dados. Iremos utilizar o usuário “root”, então, digitaremos a senha “**P@ssw0rd**”, que é a que utilizaremos para os acessos ao MySQL. Selecionar **[NEXT]**.



#### OBSERVAÇÃO:

O nosso servidor MySQL que estamos configurando deverá ser executado como um serviço do Windows, e é nesta tela que definiremos isto, marcaremos a opção “**Configure MySQL Server as a Windows Service**”. Poderemos ainda dar um nome para este serviço na caixa de texto “**Windows Service Name**”. Poderemos modificar sob qual conta de usuário o serviço será executado. Marcaremos “**Standard System Account**”. Selecionar o botão [NEXT].



#### OBSERVAÇÃO:

Esta é a tela de aplicação das configurações no seu estado final. Aqui, todas as configurações foram aplicadas com sucesso, sendo isto sinalizado pelos ícones em verde. Inicialmente, os passos de configuração (*Configurations Steps*) estão desmarcados e um botão [**Execute**] é mostrado, basta selecionar este botão e as configurações são aplicadas. Este é o passo final da configuração do nosso servidor, selecionar o botão [**Finish**].

 MySQL. Installer

Adding Community

Choosing a Setup Type

Installation

**Product Configuration**

Installation Complete

## Product Configuration

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.21	Configuration complete.
MySQL Router 8.0.22	Ready to configure
Samples and Examples 8.0.21	Ready to configure

### OBSERVAÇÃO:

Esta tela mostra que o nosso servidor MySQL Server 8.0.21 foi configurado com sucesso, observemos a coluna Status que sinaliza que a Configuração está completa (*Configuration complete*). Selecionando o botão [**NEXT**], inicia-se a configuração dos demais produtos.

 MySQL. Installer

MySQL Router 8.0.22

MySQL Router Configuration

## MySQL Router Configuration

Bootstrap MySQL Router for use with InnoDB cluster

This wizard can bootstrap MySQL Router to direct traffic between MySQL applications and a MySQL InnoDB cluster. Applications that connect to the router will be automatically directed to an available read/write or read-only member of the cluster.

The bootstrapping process requires a connection to the InnoDB cluster. In order to register the MySQL Router for monitoring, use the current Read/Write instance of the cluster.

Hostname:

Port:

Management User:

Password:

MySQL Router requires specification of a base port (between 80 and 65532). The first port is used for classic read/write connections. The other ports are computed sequentially after the first port. If any port is indicated to be in use, please change the base port.

Classic MySQL protocol connections to InnoDB cluster:

Read/Write:

Read Only:

MySQL X protocol connections to InnoDB cluster:

Read/Write:

Read Only:

### OBSERVAÇÃO:

Este produto não utilizaremos, então, simplesmente selecionaremos o botão [**Finish**].

**MySQL® Installer**

Adding Community

Choosing a Setup Type

Installation

**Product Configuration**

Installation Complete

**Product Configuration**

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.21	Configuration complete.
MySQL Router 8.0.22	Configuration not needed.
Samples and Examples 8.0.21	Ready to configure

OBSERVAÇÃO:  
Esta é a próxima tela mostrada, selecionaremos [NEXT].

**MySQL® Installer**

Samples and Examples

Connect To Server

Select the MySQL server instances from the list to receive sample schemas and data.

Server	Port	Arch...	Type	Status
<input checked="" type="checkbox"/> MySQL Server 8.0.21	3306	X64	Stand-alone Server	Connection succeeded.

Provide the credentials that should be used (requires root privileges).  
Click "Check" to ensure they work.

User name:  Credentials provided in Server configuration

Password:

OBSERVAÇÃO:  
Aqui, iremos fazer um teste de conexão com o nosso servidor MySQL. Como só definimos um único usuário, que é o usuário “root”, para o qual configuramos a senha “P@ssw0rd”, iremos digitá-la e selecionar o botão [Check]. Na coluna Status surgirá a mensagem “**Connection succeeded**”. Selecionar o botão [NEXT].

**MySQL® Installer**  
Samples and Examples

Connect To Server  
Apply Configuration

### Apply Configuration

The configuration operation has completed.

Configuration Steps Log

- Checking if there are any features installed that need configuration.
- Running Scripts

#### OBSERVAÇÃO:

Esta é uma tela de verificação se o processo de configuração ocorreu com sucesso ou se algum produto ainda necessita de configuração. Antes de sua execução é mostrado um botão [Execute], selecionamos este botão e a verificação é feita. Selecionar o botão [Finish].

**MySQL® Installer**  
Adding Community

Choosing a Setup Type  
Installation  
Product Configuration  
Installation Complete

### Product Configuration

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.21	Configuration complete.
MySQL Router 8.0.22	Configuration not needed.
Samples and Examples 8.0.21	Configuration complete.

#### OBSERVAÇÃO:

Aqui é mostrado os produtos instalados e o *Status* da sua configuração, no caso, todos foram configurados corretamente. Selecionar o botão [Next].

**MySQL® Installer**  
Adding Community

Choosing a Setup Type  
Installation  
Product Configuration  
Installation Complete

### Installation Complete

The installation procedure has been completed.

Start MySQL Workbench after setup  
 Start MySQL Shell after setup

The MySQL Shell is an advanced MySQL client application that can be used to work with single MySQL Server instances. Further, it can be used to create and manage an InnoDB cluster, an integrated solution for high availability and scalability of MySQL databases, without requiring advanced MySQL expertise.

Refer to the following links for documentation, tutorials and examples on MySQL Shell:  
[MySQL Shell Documentation](#)    [Setting up a Real World Cluster Blog](#)  
[The All New MySQL InnoDB ReplicaSet Blog](#)    [Changing Cluster Options Live Blog](#)

Esta é a tela final do processo de instalação e configuração do MySQL. Selecionar o botão [Finish].

Finalizada a instalação, teremos instalado tanto o servidor MySQL quanto a ferramenta cliente MySQL Workbench.

## 2. A LINGUAGEM SQL (**STRUCTURE QUERY LANGUAGE**)

A linguagem estruturada de pesquisa, mais conhecida como **SQL**, é a **linguagem de programação** que irá permitir que enviamos comandos para o SGBD MySQL, atuando como um instrumento de ligação entre as aplicações e o SGBD. Ela foi desenvolvida no início dos anos 70, pela **IBM**, com a finalidade de interagir com os **bancos de dados relacionais**. A partir de seu desenvolvimento, diversos fabricantes de sistemas gerenciadores de banco de dados começaram a desenvolver suas versões próprias da linguagem, que passaram a se chamar de dialetos ou extensões da linguagem SQL. Atualmente, é uma das linguagens mais populares do mundo na área de desenvolvimento de software que utilizam bancos de dados e existem inúmeras definições para ela, a seguir estão listadas algumas:

- Uma linguagem de dados relacional que disponibiliza um conjunto consistente de palavras da língua inglesa para consultas, definição de dados, manipulação de dados e controle de dados (Gartner);
- Uma linguagem padrão para sistemas gerenciadores de dados relacionais nos últimos 40 anos (Beth Narrish e Dan Hilton);
- Uma interface padrão para sistemas gerenciadores de dados relacionais (Andrew Pavlo e Matthew Aslett);
- Linguagem utilizada para gerenciar e administrar servidores de banco de dados (Database Journal);
- Linguagem mais comum para consultas e manipulação de dados (Angela Zhang, Forbes);
- Uma linguagem de consulta projetada para organizar, gerenciar, desenvolver e consultar os grandes bancos de dados relacionais através das redes de computadores (IBM);
- Uma linguagem especializada para atualização, exclusão e requisição de informações dos bancos de dados (Indiana University).

Fonte: <https://bit.ly/2JwP9hL>.

Em função do aparecimento dos **dialetos** ou **extensões** da linguagem SQL original, surgiu a necessidade de uma **padronização**. Então, organizações como o Instituto Nacional Americano de Padrões (*American National Standards Institute – ANSI*) e a Organização Internacional de Padronização (*International Standards Organization – ISO*) resolveram padronizar a linguagem, surgindo, em 1986, o **padrão ANSI** e, em 1987, o **padrão ISO**. A partir destes padrões iniciais, várias atualizações foram sendo acrescidas, sempre que necessário, sendo incluídos nos padrões novas funcionalidades e comandos.

A linguagem SQL é rica em **comandos**, permitindo um eficiente desenvolvimento para bancos de dados, seus comandos podem ser agrupados de acordo com o papel de cada um dentro da linguagem. Para uma explicação mais concisa, vamos observar o quadro a seguir:

**Quadro 03:** Organização dos comandos da linguagem SQL.

LINGUAGEM DE DEFINIÇÃO DE DADOS ( <i>Data Definition Language – DDL</i> )	Este subconjunto de comandos permite a criação dos objetos do banco de dados. Os principais comandos são: <i>CREATE</i> – função de criação dos objetos de banco de dados; <i>ALTER</i> – função de alteração dos objetos de banco de dados; <i>DROP</i> – função de exclusão de objetos de banco de dados.
LINGUAGEM DE MANIPULAÇÃO DE DADOS ( <i>Data Manipulation Language – DML</i> )	Este subconjunto de comandos tem a função de manipular os dados dentro de um banco de dados. Os comandos deste subconjunto são: <i>INSERT</i> – função de inserção de dados; <i>SELECT</i> - função de busca de dados; <i>UPDATE</i> – função de atualização de dados; <i>DELETE</i> – função de exclusão de dados.
LINGUAGEM DE CONTROLE DE DADOS ( <i>Data Control Language – DCL</i> )	Este subconjunto de comandos está ligado a conceder ou retirar privilégios de usuários nos objetos de banco de dados, os principais comandos são: <i>GRANT</i> – função de dar privilégios aos usuários nos objetos do banco de dados; <i>REVOKE</i> – função de retirar privilégios dados aos usuários nos objetos de banco de dados.
LINGUAGEM DE CONTROLE DE TRANSAÇÃO ( <i>Transaction Control Language - TCL</i> )	Este subconjunto de comandos é responsável por confirmar ou reverter transações. São eles: <i>COMMIT</i> – tem a função de confirmar os comandos dentro de uma transação; <i>ROLLBACK</i> – tem a função de reverter os comandos dentro de uma transação.

Fonte: Elaborado pelo autor.



## FIQUE ATENTO

Poderemos encontrar em alguma literatura de banco de dados referência à **Linguagem de consulta de dados** (*Data Query Language - DQL*), situação na qual o comando “SELECT” faria parte sozinho deste subconjunto de comandos, portanto, saindo do subconjunto de comandos da linguagem de manipulação de dados. Outra situação é encontrarmos os comandos “COMMIT” e “ROLLBACK” fazendo parte do subconjunto de comandos da linguagem de controle de dados.

O conceito de TRANSAÇÃO será discutido posteriormente, mas de uma forma resumida poderemos dizer que uma transação é um mecanismo que permite selecionar um **grupo de operações** e executá-las de forma a garantir que todas serão executadas ou nenhuma delas. No caso, o comando *COMMIT* é responsável por confirmar que todos do grupo de comandos foram executados, e o comando *ROLLBACK* é responsável por reverter a situação dos comandos, garantindo que nenhum deles seja executado.

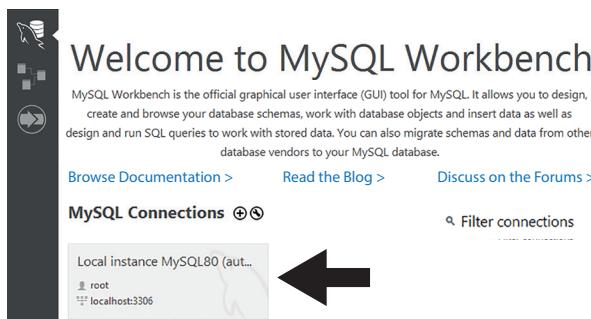
Os **objetos** de banco de dados são as tabelas, as visões, os índices, as funções, os procedimentos armazenados, os triggers etc.

## 2.1. Utilizando o MySQL para a criação e manipulação de banco de dados

Para a criação do banco de dados de exemplo, manipulação de dados e criação de mecanismos para que o banco de dados tenha um alto desempenho, necessitaremos utilizar o **MySQL Workbench**, que é a ferramenta cliente que irá permitir que enviamos comandos SQL a serem executados.

Ao iniciarmos o MySQL Workbench, a tela inicial que nos será apresentada é a tela a seguir:

**Figura 01:** Tela inicial do MySQL Workbench.

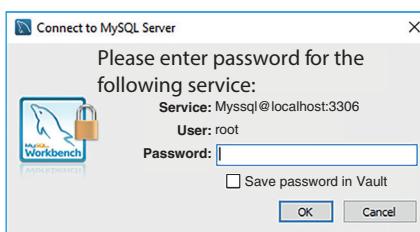


Fonte: MySQL Workbench.

Quando da instalação do MySQL, ele cria um usuário chamado “**root**”, que é o usuário administrador do banco de dados, e para este usuário criamos a senha “P@ssw0rd” que será solicitada ao pressionarmos o botão indicado pela seta na figura anterior. Ou seja, iremos nos conectar com a instância do MySQL que instalamos. Uma **instância** é basicamente uma instalação do MySQL, poderemos instalar mais de uma instância em um mesmo servidor, bastando para isto nomeá-las. No nosso caso, durante a instalação, aceitamos o nome padrão dado pelo MySQL, que foi “MySQL80”. Poderíamos também ter criados outros usuários para utilizar, mas no nosso caso atual, iremos utilizar o usuário “root”. Resumindo:

- Iremos nos conectar à instância local chamada MySQL80;
- Como o MySQL está instalado no computador local, ele é identificado como “localhost”, e a porta TCP-IP utilizada é a padrão 3306;
- Iremos utilizar o usuário “root” que tem a senha “P@ssw0rd”.

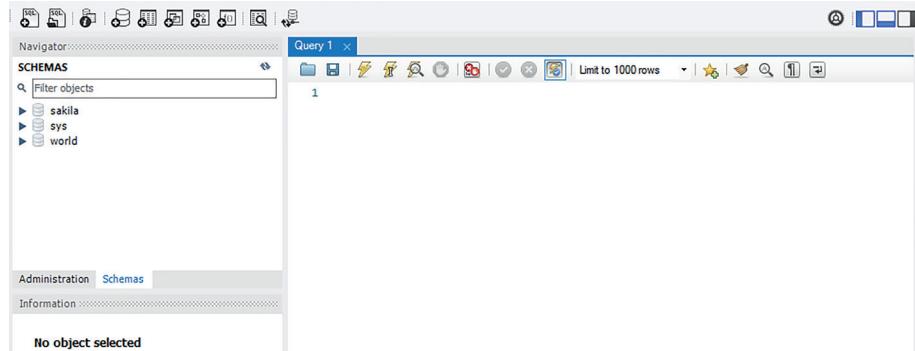
**Figura 02:** Tela de *Login* do MySQL Workbench.



Fonte: MySQL Workbench.

No campo **Password** da tela de autenticação, iremos digitar a senha: P@ssw0rd. Tendo sido digitada corretamente a senha, será mostrada a tela a seguir, que é a nossa tela de trabalho. Nela, construiremos o nosso banco de dados.

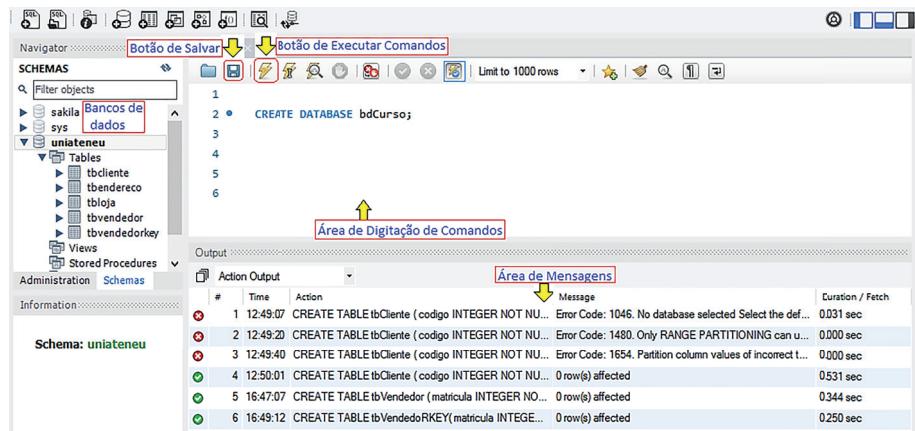
**Figura 03:** Área de trabalho do MySQL Workbench.



Fonte: MySQL Workbench.

Como já explicado anteriormente, o **MySQL Workbench** será a nossa ferramenta de trabalho, e todos os comandos comentados a partir de agora devem ser executados nesta ferramenta. A figura a seguir mostra as **funcionalidades** principais que serão necessárias para trabalho nas atividades práticas que se seguirão.

**Figura 04:** Área de comando SQL do MySQL Workbench.



Fonte: MySQL Workbench.

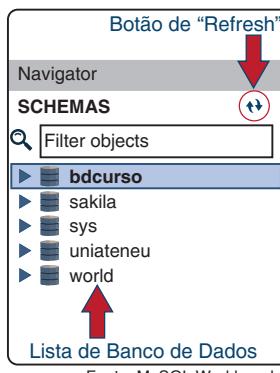
### 2.1.1. Criação do banco de dados

Para a criação do banco de dados, inicialmente, deveremos digitar o comando **CREATE DATABASE** bdcurso (se desejarmos excluir este banco de dados, utilizaremos o comando **DROP DATABASE** bdcurso) na “Área de Digitação de Comandos” e pressionar o “Botão de Executar Comandos”, como mostrado na figura anterior. Observe na “Área de Mensagens” que surgirá a mensagem:



Esta mensagem está marcada com um **ícone verde**, sinalizando que o comando foi **executado** com sucesso. Verifique na parte esquerda da tela, onde está a palavra “SCHEMAS”, que o novo banco de dados será colocado na lista de Bancos de Dados. Caso isto não ocorra, pressione o “Botão de Refresh” e a lista de bancos de dados será atualizada, mostrando o banco de dados recentemente criado.

**Figura 05:** Lista de bancos de dados.



Fonte: MySQL Workbench.

#### Observação:

- I. No sistema gerenciador de banco de dados MySQL, o termo “SCHEMA” é sinônimo de “DATABASE”. Portanto, os comandos abaixo têm a mesma função:

**CREATE DATABASE** bdcurso;

**CREATE SCHEMA** bdcurso.

- II. Na área de mensagens do **MySQL Workbench**, veremos dois tipos de mensagens, aquelas que são marcadas com um **ícone vermelho**, indicando que houve um erro de execução do comando, e outra marcada com um **ícone verde**, indicando que houve sucesso, ou seja, o comando foi executado corretamente. Na ocorrência de erro, a coluna MESSAGE indica o motivo do erro, devendo este ser corrigido e o comando executado novamente até ter sucesso. A figura a seguir mostra a parte da tela onde os erros são mostrados.

**Figura 06:** Mensagens de erro.

Action Output	#	Time	Action	Message	Duration / Fetch
	✖ 2	12:49:20	CREATE TABLE tbCliente ( codigo INTEGER NOT ...	Error Code: 1480. Only RANGE PARTITIONING can ...	0.000 sec
	✖ 3	12:49:40	CREATE TABLE tbCliente ( codigo INTEGER NOT ...	Error Code: 1654. Partition column values of incorrect...	0.000 sec
	✔ 4	12:50:01	CREATE TABLE tbCliente ( codigo INTEGER NOT ...	0 row(s) affected	0.531 sec
	✔ 5	16:47:07	CREATE TABLE tbVendedor ( matricula INTEGER N...	0 row(s) affected	0.344 sec

Fonte: MySQL Workbench.

## 2.2. Criação das tabelas do banco de dados

Uma vez criado o nosso banco de dados (**bdcurso**), o próximo passo será a criação das tabelas, para isto, deveremos consultar o nosso **dicionário de dados** e executar os comandos de criação das **tabelas** nele definidas.

Deveremos observar que na lista de banco de dados no **MySQL Workbench** além do nosso banco de dados existem outros bancos. Para que os comandos de criação de tabelas sejam executados de forma que as tabelas sejam geradas dentro do banco de dados **bdcurso**, deveremos informar isto ao SGBD MySQL. Para que isto aconteça, deveremos executar o seguinte comando, antes dos comandos de criação das tabelas: **USE bdcurso**.

Para nos certificarmos que o comando foi executado com sucesso, ou seja, que o banco de dados onde serão criadas as tabelas é o “**bdcurso**”, basta verificar na **área de mensagens** se o comando dado (**USE bdcurso**) está marcado com um **ícone verde**. Outra forma de termos a certeza que o banco de dados que está em uso é o “**bdcurso**” é na lista de banco de dados (Figura 05) e verificarmos se o nome do banco de dados está em **negrito**.

Com o banco de dados em uso, deveremos agora criar as **tabelas**, lembrando, como já visto, que o comando CREATE DATABASE e CREATE TABLE são comandos de definição de dados (*Data Definition Language - DDL*). Para facilitar o entendimento, iremos trazer uma parte das informações do quadro do dicionário de dados que criamos anteriormente e acrescentar uma coluna que chamaremos de COMANDOS SQL, em que constam os comandos de criação das tabelas que serão executados no MySQL WorkBench.

**Quadro 04:** Dicionário de dados e comandos de criação das tabelas.

TABELA	NOME DO CAMPO	TIPO DE DADOS	TAMANHO	COMANDOS SQL
tbcliente	Cliente_Codigo	Integer	Pré-definido	CREATE TABLE `tbcliente` ( `Cliente_Codigo` INT NOT NULL, `Cliente_Nome` VARCHAR(100) NOT NULL)
	Cliente_Nome	Varchar	100	
tbproduto	Produto_Codigo	Integer	Pré-definido	CREATE TABLE `tbproduto` ( `Produto_Codigo` INT NOT NULL, `Produto_Nome` VARCHAR(100) NOT NULL)
	Produto_Nome	Varchar	100	
tblaja	Loja_Codigo	Integer	Pré-definido	CREATE TABLE `tblaja` ( `Loja_Codigo` INT NOT NULL, `Loja_Nome` VARCHAR(100) NOT NULL)
	Loja_Nome	Varchar	100	
tbvendedor	Vendedor_Matricula	Inteiro	Pré-definido	CREATE TABLE `tbvendedor` ( `Vendedor_Matricula` INT NOT NULL, `Vendedor_Nome` VAR- CHAR(100) NOT NULL)
	Vendedor_Nome	Varchar	100	
tbdepartamento	Departamento_Codigo	Integer	Pré-definido	CREATE TABLE `tbdepartamen- to` ( `Departamento_Codigo` INT NOT NULL, `Departamento_Nome` VAR- CHAR(100) NOT NULL)
	Departamento_Nome	Varchar	100	

tbvenda	Venda_Codigo	Integer	Pré-definido	
	Venda_Data	Datetime	Pré-definido	
	Venda_Quantidade	Integer	Pré-definido	
	Venda_Cliente	Integer	Pré-definido	
	Venda_Produto	Integer	Pré-definido	
	Venda_Loja	Integer	Pré-definido	
	Venda_Departamento	Integer	Pré-definido	
	Venda_Vendedor	Integer	Pré-definido	
tbestado	Estado_Codigo	Integer	Pré-definido	CREATE TABLE `tbestado` ( `Estado_Codigo` INT NOT NULL, `Estado_Nome` VARCHAR(100) NOT NULL, `Estado_Sigla` VARCHAR(2) NOT NULL)
	Estado_Nome	VARCHAR	100	
	Estado_Sigla	VARCHAR	2	
tbcidade	Cidade_Codigo	Integer	Pré-definido	CREATE TABLE `tbcidade` ( `Cidade_Codigo` INT NOT NULL, `Cidade_Nome` VARCHAR(100) NOT NULL)
	Cidade_Nome	VARCHAR	100	
tbbairro	Bairro_Codigo	Integer	Pré-definido	CREATE TABLE `tbbairro` ( `Bairro_Codigo` INT NOT NULL, `Bairro_Nome` VARCHAR(100) NOT NULL)
	Bairro_Nome	VARCHAR	100	
tblogradouro	Logradouro_Codigo	Integer	Pré-definido	CREATE TABLE `tblogradouro` ( `Logradouro_Codigo` INT NOT NULL, `Logradouro_Nome` VARCHAR(100) NOT NULL)
	Logradouro_Nome	VARCHAR	100	

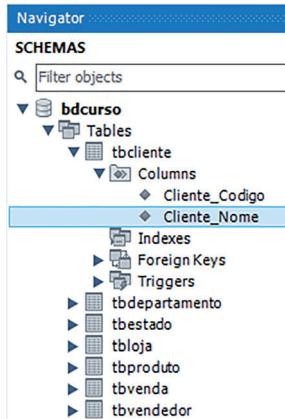
Fonte: Elaborado pelo autor.

No caso de necessitarmos excluir alguma das tabelas criadas, utilizaremos o comando **DROP TABLE** seguido do nome da tabela. Como exemplo, para excluir a tabela **tbbairro**, executaríamos o comando abaixo:

**DROP TABLE tbbairro.**

Após a finalização da execução dos comandos de criação das tabelas, todas elas deverão estar dentro do banco de dados “bdcurso”, como mostrado na figura a seguir. Aqui, deveremos observar a cláusula “**NOT NULL**” nos comandos de criação das tabelas, sua utilização significa que informações devem ser obrigatoriamente enviadas para a coluna, ou seja, a coluna não pode conter valor NULO.

**Figura 07:** Tabelas.



Fonte: MySQL Workbench.

## 2.3. Criação das chaves primárias

Com todas as tabelas criadas, agora, o próximo passo será a criação das **chaves primárias**. No quadro a seguir, estão colocadas as informações sobre as tabelas, campos, nomes das chaves primárias e os comandos SQL que devem ser executados para a criação destas chaves no banco de dados.

**Quadro 05:** Dicionário de dados e comandos de criação das chaves primárias.

TABELA	NOME DO CAMPO	NOME DA CHAVE PRIMÁRIA	COMANDOS SQL
tbcliente	Cliente_Codigo	pk_Cliente_Codigo	<code>ALTER TABLE tbcliente ADD PRIMARY KEY pk_Cliente_Codigo (Cliente_Codigo);</code>
tbproduto	Produto_Codigo	pk_Produto_Codigo	<code>ALTER TABLE tbproduto ADD PRIMARY KEY pk_Produto_Codigo (Produto_Codigo);</code>

tbloja	Loja_Codigo	pk_Loja_codigo	<b>ALTER TABLE</b> tbloja <b>ADD PRIMARY KEY</b> pk_Loja_Codigo (Loja_Codigo);
tbvendedor	Vendedor_Matricula	pk_Vendedor_Codigo	<b>ALTER TABLE</b> tbvendedor <b>ADD PRIMARY KEY</b> pk_Vendedor_Codigo (Vendedor_Codigo);
tbdepartamento	Departamento_Codigo	pk_Departamento_codigo	<b>ALTER TABLE</b> tbdepartamento <b>ADD PRIMARY KEY</b> pk_Departamento_Codigo (Departamento_Codigo);
tbvenda	Venda_Codigo	pk_Venda_Codigo	<b>ALTER TABLE</b> tbvenda <b>ADD PRIMARY KEY</b> pk_Venda_Codigo (Venda_Codigo);
tbestado	Estado_Codigo	pk_Estado_Codigo	<b>ALTER TABLE</b> tbestado <b>ADD PRIMARY KEY</b> pk_Estado_Codigo (Estado_Codigo);
tbcidade	Cidade_Codigo	pk_Cidade_Codigo	<b>ALTER TABLE</b> tbcidade <b>ADD PRIMARY KEY</b> pk_Cidade_Codigo (Cidade_Codigo);
tbairro	Bairro_Codigo	pk_Bairro_Codigo	<b>ALTER TABLE</b> tbbairro <b>ADD PRIMARY KEY</b> pk_Bairro_Codigo (Bairro_Codigo);
tblogradouro	Logradouro_Codigo	pk_Logradouro_codigo	<b>ALTER TABLE</b> tblogradouro <b>ADD PRIMARY KEY</b> pk_Logradouro_Codigo (Logradouro_Codigo);

Fonte: Elaborado pelo autor.

Todas as chaves primárias são criadas tendo como referência **colunas** cujo tipo de dado é INTEIRO. Existe uma propriedade das colunas das tabelas que é o AUTO INCREMENTO, os sistemas gerenciadores de banco de dados possuem um mecanismo automático que controla a **numeração** das colunas que tiverem esta propriedade ativada, ou seja, o usuário não precisa se preocupar em enviar os dados para estas colunas, o SGBD irá incluir automaticamente e controlar o seu incremento.

## 2.4. Implementando a propriedade de AUTO INCREMENTO

Para a implementação da propriedade de AUTO INCREMENTO nas colunas que são chave primária em todas as tabelas, necessitaremos executar os comandos a seguir:

```
ALTER TABLE tbcliente MODIFY COLUMN Cliente_Codigo INT  
AUTO_INCREMENT  
  
ALTER TABLE tbproduto MODIFY COLUMN Produto_Codigo INT  
AUTO_INCREMENT  
  
ALTER TABLE tbloja MODIFY COLUMN Loja_Codigo INT AUTO_  
INCREMENT  
  
ALTER TABLE tbvendedor MODIFY COLUMN Vendedor_Matricula  
INT AUTO_INCREMENT  
  
ALTER TABLE tbdepartamento MODIFY COLUMN Departamento_  
Codigo INT AUTO_INCREMENT  
  
ALTER TABLE tbvenda MODIFY COLUMN Venda_Codigo INT AUTO_  
INCREMENT  
  
ALTER TABLE tbestado MODIFY COLUMN Estado_Codigo INT  
AUTO_INCREMENT  
  
ALTER TABLE tbcidade MODIFY COLUMN Cidade_Codigo INT  
AUTO_INCREMENT  
  
ALTER TABLE tbbairro MODIFY COLUMN Bairro_Codigo INT AUTO_  
INCREMENT  
  
ALTER TABLE tblogradouro MODIFY COLUMN Logradouro_Codigo  
INT AUTO_INCREMENT
```

### **Observação:**

Por questões didáticas, para discutir os assuntos separadamente, criamos as tabelas, depois criamos as chaves primárias para todas elas e, por fim, implementamos a propriedade de auto incremento nas colunas correspondentes às chaves primárias de todas as tabelas. No entanto, estes passos podem ser feitos de uma única vez no momento da criação da tabela. Abaixo, o exemplo da criação de uma tabela, já definindo a chave primária e o auto incremento no comando CREATE TABLE:

```
CREATE TABLE `tbcidade` (
    `Cidade_Codigo` INT NOT NULL AUTO_INCREMENT,
    `Cidade_Nome` VARCHAR(45) NOT NULL,
    PRIMARY KEY pk_tbcidade_codigo (`Cidade_Codigo`));
```

#### **Observação:**

Nos comandos SQL de criação das tabelas, utilizamos, propositalmente, os **nomes** das tabelas e dos campos entre **aspas simples**. No entanto, isto não é uma sintaxe obrigatória, poderemos perfeitamente retirar as aspas e os comandos funcionarão sem problemas, como mostrado abaixo:

```
CREATE TABLE tbcidade (
    Cidade_Codigo INT NOT NULL AUTO_INCREMENT,
    Cidade_Nome VARCHAR(45) NOT NULL,
    PRIMARY KEY pk_tbcidade_codigo (Cidade_Codigo));
```

## **2.5. Implementando os relacionamentos entre as tabelas**

---

Como foi discutido na Unidade 01, fizemos a normalização das entidades e criamos novas, entre elas, a entidade venda que se relaciona com as demais, e a entidade cliente, que além de se relacionar com a entidade venda, relaciona-se também com as entidades estado, cidade, bairro e logradouro. Para cada entidade, criamos uma tabela que a representa dentro do banco de dados.

Para que possamos construir os relacionamentos entre a tabela **tbcliente** que se relaciona com as tabelas tbestado, tbcidade, tbairro, tblogradouro e a tabela **tbvenda**, que se relaciona com as tabelas tbcliente, tbvendedor, tbproduto, tbloja e tbdepartamento, será necessário a criação de novas colunas em tbcliente e tbvenda. Basta que observemos a Figura 10: Entidades e relacionamentos (Unidade 01) para que se entenda a lógica do relacionamento. Iremos criar as colunas que se apresentam na cor azul nesta figura. Da mesma forma, no diagrama de entidades e relacionamentos que temos na Figura 11: Diagrama de entidades e relacionamentos no MySQL (Unidade 01), estas mesmas colunas estão identificadas por pequenos quadriláteros na cor vermelha. Lembramos que todas as colunas que serão criadas são do tipo INTEIRO. Vejamos o quadro a seguir:

**Quadro 06:** Criação das colunas para os relacionamentos.

TABELA	COLUNA	COMANDO SQL
tbcliente	Cliente_Estado	<code>ALTER TABLE tbcliente ADD COLUMN Cliente_Estado INTEGER;</code>
	Cliente_Cidade	<code>ALTER TABLE tbcliente ADD COLUMN Cliente_Cidade INTEGER;</code>
	Cliente_Bairro	<code>ALTER TABLE tbcliente ADD COLUMN Cliente_Bairro INTEGER;</code>
	Cliente_Logradouro	<code>ALTER TABLE tbcliente ADD COLUMN Cliente_Logradouro INTEGER;</code>
tqvenda	Cliente_codigo	<code>ALTER TABLE tbvenda ADD COLUMN Cliente_Codigo INTEGER;</code>
	Vendedor_Matricula	<code>ALTER TABLE tbvenda ADD COLUMN Vendedor_Matricula INTEGER;</code>
	Produto_Codigo	<code>ALTER TABLE tbvenda ADD COLUMN Produto_Codigo INTEGER;</code>
	Loja_Codigo	<code>ALTER TABLE tbvenda ADD COLUMN Loja_Codigo INTEGER;</code>
	Departamento_Codigo	<code>ALTER TABLE tbvenda ADD COLUMN Departamento_Codigo INTEGER;</code>

Fonte: Elaborado pelo autor.

Agora, iremos implementar este **relacionamento** que está ilustrado na Figura 11: Diagrama de entidades e relacionamentos no MySQL (Unidade 01), utilizando as chaves estrangeiras. Para isto, iremos consultar o nosso dicionário de dados (Quadro 09: Dicionário de dados – CHAVES ESTRANGEIRAS (Unidade 01)), na parte que se refere às definições das chaves estrangeiras e, baseado nele, iremos escrever os **comandos SQL** para implementar os **objetos correspondentes** no nosso banco de dados, conforme mostrado no quadro abaixo:

**Quadro 07:** Geração das chaves estrangeiras.

		CHAVE ESTRANGEIRA	Lado N do relacionamento	COLUNA	Lado 1 do relacionamento	CHAVE PRIMÁRIA	COMANDOS SQL
fk_cliente_logradouro	fk_cliente_bairro	fk_cliente_cidade	tbcliente	Estado_Codigo	tbestado	Estado_Codigo	<pre>ALTER TABLE tbcliente ADD CONSTRAINT fk_cliente_estado FOREIGN KEY ( Estado_Codigo ) REFERENCES tbestado (Estado_Codigo)</pre>
Logradouro_Codigo	Bairro_Codigo	Cidade_Codigo		tbcidade		Cidade_Codigo	<pre>ALTER TABLE tbcliente ADD CONSTRAINT fk_cliente_cidade FOREIGN KEY ( Cliente_Cidade ) REFERENCES tbccidade (Cidade_Codigo)</pre>
tblogradouro						Bairro_Codigo	<pre>ALTER TABLE tbcliente ADD CONSTRAINT fk_cliente_bairro FOREIGN KEY ( Bairro_Codigo ) REFERENCES tbairro (Bairro_Codigo)</pre>
Logradouro_Codigo							<pre>ALTER TABLE tbcliente ADD CONSTRAINT fk_cliente_logradouro FOREIGN KEY ( Logradouro_Codigo ) REFERENCES tblogradouro (Logradouro_Codigo)</pre>

fk_venda_departamento	fk_venda_produto	fk_venda_loja	fk_venda_cliente	
tbvenda				
Departamento_codigo	Produto_Codigo	Vendedor_Matricula	Loja_Codigo	ALTER TABLE tbvenda ADD CONSTRAINT fk_venda_cliente FOREIGN KEY ( Cliente_codigo ) REFERENCES tbcliente (Cliente_Codigo) ;
tbdepartamento	tbproduto	tvendedor	tbluja	ALTER TABLE tbvenda ADD CONSTRAINT fk_venda_loja FOREIGN KEY ( Loja_codigo ) REFERENCES tbloja (Loja_Codigo) ;
Departamento_codigo	Produto_Codigo	Vendedor_Matricula	Loja_Codigo	ALTER TABLE tbvenda ADD CONSTRAINT fk_venda_vendedor FOREIGN KEY (Vendedor_Matriculada) REFERENCES tbvendedor (Vendedor_Matricula) ;
				ALTER TABLE tbvenda ADD CONSTRAINT fk_venda_produto FOREIGN KEY ( Produto_Codigo ) REFERENCES tbproduto (Produto_Codigo) ;
				ALTER TABLE tbvenda ADD CONSTRAINT fk_venda_cliente FOREIGN KEY ( Departamento_Codigo ) REFERENCES tbdepartamento (Departamento_Codigo) ;

Fonte: Elaborado pelo autor.

Observação:

Até agora, temos criado **objetos** (tabelas, colunas, chaves primárias, chaves estrangeiras) no banco de dados, portanto, relembrando a organização dos comandos do SQL (**Quadro 03**: Organização dos comandos da linguagem SQL.), nós temos utilizado os comandos SQL da **Linguagem de Definição de Dados**.

## 2.6. Manipulação dos dados

O nosso banco de dados está construído, mas ainda não possui **dados** dentro das **tabelas**, se utilizarmos o comando SELECT para verificar a existência dos dados, veremos que todas as tabelas estão vazias. A título de ilustração, colocamos abaixo dois comandos SELECT. Vamos observar que o primeiro comando utiliza um “\*”, significando que todos os campos da tabela serão retornados quando da execução do comando SELECT. No segundo comando, definimos que colunas desejamos que sejam retornadas.

```
select * from tbcliente;
```

```
select Venda_Data,Venda_Cliente,Venda_Produto from tbvenda;
```

O próximo passo será fazer a inclusão dos dados que se encontram na **Figura 10: Entidades e relacionamentos** (Unidade 01). Observemos aqui que existe uma restrição para que a inclusão de dados seja feita, no que se refere às tabelas tbvenda e tbcliente, pois ambas se relacionam com outras tabelas. No caso da tabela tbcliente, as tabelas tbestado, tbcidade, tbbairro e tblogradouro devem ter, inicialmente, seus dados incluídos. Já no caso da tabela tbvenda, todas as outras tabelas devem ter seus dados incluídos antes dela receber os dados. Para que as inclusões dos dados sejam feitas, utilizaremos o comando INSERT, conforme mostrado no quadro a seguir:

**Quadro 08:** Comandos para a inserção dos dados.

TABELA	COMANDOS SQL
<b>tbestado</b>	<pre>Insert into tbestado(Estado_Nome,Estado_Sigla) Values ('CEARÁ','CE'),('SÃO PAULO','SP'),('RIO DE JANEIRO','RJ'), ('PARÁ','PA'),('BAHIA','BA');</pre>
<b>tbcidade</b>	<pre>Insert into tbcidade(Cidade_Nome) Values ('CEARÁ','CE'),('SÃO PAULO','SP'),('RIO DE JANEIRO','RJ'), ('PARÁ','PA'),('BAHIA','BA');</pre>

<b>tbbairro</b>	<code>Insert into tbbairro(Bairro_Nome) Values ('MARAPONGA'),('CENTRO'),('ÁGUAS BELAS'), ),('AÇAÍ'), ('MONTESE');('PARÁ','PA'),('BAHIA','BA');</code>
<b>tblogradouro</b>	<code>Insert into tblogradouro(Logradouro_Nome) Values ('RUA'),('AVENIDA'),('TRAVESSA'),('VILA');</code>
<b>tbloja</b>	<code>Insert into tbloja(Loja_Nome) Values ('MARAPONGA'),('CENTRO'),('NITERÓI'),('SANTARÉM'), ('PELOURINHO'),('MONTESE');</code>
<b>tbdepartamento</b>	<code>Insert into tbdepartamento(Departamento_Nome) Values ('INFORMÁTICA'),('AUTOMOTIVO'),('ELETRODOMÉSTICO'), ('TELECOMUNICAÇÕES'),('ROUPAS E CALÇADOS'), ('INSTRUMENTOS MUSICais');</code>
<b>tbproduto</b>	<code>Insert into tbproduto(Produto_Nome) Values ('IMPRESSORA LASER'),('PNEUS'),('GELADEIRA'),('CPU DELL'), ('SAPATO MASCULINO'),('VIOLÃO');</code>
<b>tbvendedor</b>	<code>Insert into tbvendedor(Vendedor_Nome) Values ('MANUEL ORLANDO'),('ELIANE FARIAS'),('ANTÔNIO MATOS'), ('ANDREA FILGUEIRAS'),('PEDRO ANDRÉ'), ('GILBERTO FREIRE');</code>
<b>tbcliente</b>	<code>Insert into tbcliente(Cliente_Nome,Cliente_Endereco,Cliente_Cpf, Cliente_Logradouro,Cliente_Bairro,Cliente_Cidade,Cliente_Estado) Values ('JOSÉ DA SILVA','JÚLIO ALCIDES,323','12556947849',1,1,1,1), ('MARTA PEREIRA','JÚLIO ALCIDES,323','15923485256',2,2,2,2), ('FÁBIA OLIVEIRA','JPEREIRA FILGUEIRAS,567','16942588869',3,3,3,3), ('CARLOS SOBREIRA','SÃO JOSÉ,234','17852396289',4,4,4,4), ('JOSUÉ ANDERSON','FLORIANO PEIXOTO,876','12865896378',1,2,5,5), ('MÁRCIA NUNES','GOMES DE MATOS,6789','13796522552',1,5,1,1);</code>
<b>tbvenda</b>	<code>Insert into tbvenda (Venda_Data,Venda_Quantidade,Venda_Cliente, Venda_Vendedor,Venda_Produto,Venda_Loja,Venda_Departamento) Values ('11/11/2015',1,1,1,1,1,1),('2020-05-20',4,2,2,2,2,2), ('2014-05-14',3,3,3,3,3,3),('2013-02-25',1,4,4,4,4,1), ('2003-08-15',10,5,5,5,5,5),('2020-01-01',2,6,6,6,6,6);</code>

Fonte: Elaborado pelo autor.

Além da INSERÇÃO dos dados também será importante podermos modificar os dados, alterar ou excluir registros. No caso da alteração, utilizaremos o comando **UPDATE**. Como exemplo, vamos alterar o nome do vendedor “Pedro André” para “Pedro Luis”, poderemos fazer de duas formas, a primeira e a mais recomendada utilizando na cláusula **WHERE** do comando, a matrícula do vendedor, esta é a maneira recomendada, pois a coluna matrícula da tabela tbvendedor é uma chave primária, portanto, não há possibilidade de matrículas repetidas.

A segunda forma de atualização seria utilizar na cláusula WHERE o nome do vendedor, neste caso, falhas poderiam ocorrer, pois existem pessoas com o mesmo nome, então, se utilizarmos esta forma, todos os vendedores que tenham o nome “Pedro André” terão seus nomes trocados para “Pedro Luis”. Os comandos são mostrados a seguir:

- I. Modificação utilizando a chave primária, sabemos que a matrícula do vendedor é a de número “5”, então, o comando será:

```
UPDATE tbvendedor SET nome='PEDRO LUIS' WHERE matricula=5
```

- II. Modificação utilizando a coluna “nome”:

```
UPDATE tbvendedor SET nome='PEDRO LUIS' WHERE nome='PEDRO  
ANDRÉ'
```

No caso da necessidade de **exclusão** de um **registro** nesta mesma tabela, poderemos usar as duas formas, lembrando que no caso de utilização do nome, havendo homônimos, todos serão excluídos. Vejamos o exemplo abaixo:

- I. Exclusão utilizando a chave primária:

```
DELETE FROM tbvendedor WHERE matricula=5
```

- II. Exclusão utilizando a coluna “nome”:

```
DELETE FROM tbvendedor WHERE nome='PEDRO ANDRÉ'
```

Com todas as tabelas populadas, vamos executar um comando SELECT na tabela **tbvenda**, que foi criada para gravar as informações constantes na nossa planilha original que foi mostrada anteriormente no **Quadro 03: Planilha de controle de vendas**, e que foi a base para o planejamento do nosso banco de dados. O comando SELECT está mostrado abaixo:

```
select Venda_Codigo,Venda_Data,Venda_Quantidade,Venda_Cliente,  
Venda_Vendedor,Venda_Produto,Venda_Loja,Venda_Departamento  
from tbvenda;
```

Observando as **informações retornadas** pelo comando, poderemos concluir que elas não são muito úteis, pois trazem somente os números referentes aos **códigos de identificação** do cliente, da loja, do departamento e do produto, para compor um relatório gerencial é totalmente inviável.

Para um melhor entendimento de como funciona a **junção** entre as **tabelas**, vamos montar um exemplo simples, como mostrado na figura a seguir. Neste caso, temos duas tabelas: a tabela `tbcliente_compra`, que contém o código do cliente, seu nome e código do produto que o cliente comprou, e a outra tabela, a `tbproduto`, que contém os códigos dos produtos e seus respectivos nomes. A relação entre estas tabelas pode ser definida como “Um CLIENTE compra um PRODUTO, ou seja, como o exemplo é bem simples, a **cardinalidade** da relação é de **“1 para 1”**, conceito já discutido anteriormente. Agora, iremos utilizar o comando `SELECT` e as cláusulas de junção de tabelas: `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN` e `FULL JOIN`.

**Figura 08:** Ilustração da junção entre tabelas.

TABELA: tbcliente_compra			TABELA: tbproduto	
Código	Nome	Compra	Código	Produto
1	José	1101	1101	Computador
2	Maria		1201	Frigobar
3	Andreza		1301	Microondas
4	Zélia	1401	1401	Televisor
5	Roberto	1501	1501	Refrigerador
6	Raul		1601	Sapato
7	Yuri	1701	1701	Telefone
8	Murilo		1801	Carro
9	Felipe		1901	Pneu
10	Marta	2001	2001	Furadeira

Fonte: Elaborada pelo autor.

- **INNER JOIN**

Podemos notar que somente alguns clientes compraram os produtos. Utilizamos a cláusula `INNER JOIN` quando queremos selecionar somente os **clientes** que fizeram **compras**, a sintaxe do comando é a seguinte:

```
select tbcliente_compra.nome,tbproduto.produto  
from tbcliente_compra INNER JOIN tbproduto  
on tbcliente_compra.compra=tbproduto.codigo;
```

Com este comando SELECT utilizando a cláusula INNER JOIN, enviamos ao SGBD as seguintes informações:

- Mostrar a coluna “nome” da tabela tbcliente\_compra que contém o nome do cliente;
- Mostrar a coluna “produto” da tabela tbproduto que contém o nome do produto;
- A junção entre as duas tabelas será do tipo INNER JOIN, ou seja, serão mostrados somente os clientes que fizeram compras;
- As colunas que irão relacionar as tabelas serão: na tabela tbcliente\_compra a coluna “compra” que contém o código do produto comprado, deverá ser igual à coluna “código” da tabela tbproduto, que identifica o produto.

**Quadro 09:** Resultado do comando SELECT com INNER JOIN.

NOME	PRODUTO
JOSÉ	COMPUTADOR
ZÉLIA	TELEVISOR
ROBERTO	REFRIGERADOR
YURI	TELEFONE
MARTA	FURADEIRA

Fonte: Elaborado pelo autor.

- **LEFT JOIN ou LEFT OUTER JOIN**

Ao utilizarmos o LEFT JOIN, estaremos informando ao SGBD que queremos mostrar todos os **registros** da tabela que se encontram à **esquerda** do relacionamento. No nosso exemplo, será a tabela tbcliente\_compra e todos os seus registros serão mostrados, não importando se os clientes fizeram ou não aquisição de produtos. Para os clientes que adquirirem produtos, estes serão mostrados, caso contrário, será mostrado o valor “NULL”. O comando SELECT é o descrito abaixo:

```
select tbcliente_compra.nome,tbproduto.produto  
from tbcliente_compra LEFT JOIN tbproduto  
on tbcliente_compra.compra=tbproduto.codigo;
```

Com este comando SELECT utilizando a cláusula LEFT JOIN, enviamos ao SGBD as seguintes informações:

- Mostrar a coluna “nome” da tabela tbcliente\_compra que contém o nome do cliente;
- Mostrar a coluna “produto” da tabela tbproduto que contém o nome do produto;
- A junção entre as duas tabelas será do tipo LEFT JOIN, ou seja, serão mostrados todos os clientes, independentemente que tenham comprado produtos ou não, aqueles que fizeram compra de produto, o nome do produto adquirido será mostrado. Aqueles que não o fizeram, será mostrado “NULL” na coluna do produto;
- As colunas que irão relacionar as tabelas serão: na tabela tbcliente\_compra a coluna “compra”, que contém o código do produto comprado, deverá ser igual à coluna “código” da tabela tbproduto, que identifica o produto.

**Quadro 10:** Resultado do comando SELECT com LEFT JOIN.

NOME	PRODUTO
JOSÉ	COMPUTADOR
MARIA	NULL
ANDREZA	NULL
ZÉLIA	TELEVISOR
ROBERTO	REFRIGERADOR
RAUL	NULL
YURI	TELEFONE
MURILO	NULL
FELIPE	NULL
MARTA	FURADEIRA

Fonte: Elaborado pelo autor.

- **RIGHT JOIN ou RIGHT OUTER JOIN**

A cláusula RIGHT JOIN determina que todos os **registros** da tabela que está **à direita** do relacionamento devem ser mostrados. Neste caso, estamos nos referindo à tabela tbproduto. Todos os registros da tabela

tbproduto devem ser mostrados e se existirem correspondentes na tabela tbcliente\_compra, ou seja, o produto foi adquirido por algum cliente, o nome do cliente deve ser mostrado, caso contrário, será mostrado o valor NULL. O comando SELECT é o descrito abaixo:

```
select tbcliente_compra.nome,tbproduto.produto  
from tbcliente_compra  RIGHT JOIN  tbproduto  
on tbcliente_compra.compra=tbproduto.codigo;
```

**Quadro 11:** Resultado do comando SELECT com RIGHT JOIN.

NOME	PRODUTO
JOSÉ	COMPUTADOR
NULL	FRIGOBAR
NULL	MICROONDAS
ZÉLIA	TELEVISOR
ROBERTO	REFRIGERADOR
NULL	SAPATO
YURI	TELEFONE
NULL	CARRO
NULL	PNEU
MARTA	FURADEIRA

Fonte: Elaborado pelo autor.

- **FULL JOIN ou FULL OUTER JOIN**

No caso da utilização da cláusula FULL OUTER JOIN, todos os registros da **tabela da direita** e todos os registros da **tabela da esquerda** do relacionamento serão mostrados. A tabela da direita do relacionamento que tenha registro correspondente na tabela da esquerda será mostrado, assim como o contrário também acontecerá. Para os registros que não têm correspondentes serão mostrados o valor NULL. O comando SELECT é o descrito abaixo:

```
select tbcliente_compra.nome,tbproduto.produto  
from tbcliente_compra  FULL OUTER JOIN  tbproduto  
on tbcliente_compra.compra=tbproduto.codigo;
```

**Quadro 12:** Resultado do comando SELECT com FULL JOIN.

NOME	PRODUTO
JOSÉ	COMPUTADOR
MARIA	NULL
ANDREZA	NULL
ZÉLIA	TELEVISOR
ROBERTO	REFRIGERADOR
RAUL	NULL
YURI	TELEFONE
MURILO	NULL
FELIPE	NULL
MARTA	FURADEIRA
NULL	FRIGOBAR
NULL	MICROONDAS
NULL	SAPATO
NULL	CARRO
NULL	PNEU

Fonte: Elaborado pelo autor.

Agora que já entendemos o funcionamento das junções de tabelas, poderemos voltar ao nosso comando SELECT mostrado abaixo, para implementá-lo de forma que possa **retornar informações** úteis. Já vimos que ele está retornando somente informações de códigos de cliente, vendedor, produto, loja e departamento. Vamos modificá-lo utilizando a técnica de **junção de tabela** para que ele possa gerar um **relatório** com informações completas sobre cliente, vendedor, produto, loja e departamento.

```
select Venda_Codigo,Venda_Data,Venda_Quantidade,Venda_Cliente,  
Venda_Vendedor,Venda_Produto,Venda_Loja,Venda_Departamento from  
tbvenda;
```

A nossa tabela que contém todas as informações que utilizaremos como ponto de partida é a tabela tbvenda. Dela iremos querer todos os **registros**, uma vez que iremos fazer um relatório de todas as vendas, portanto, utilizaremos a cláusula LEFT JOIN. O comando SELECT ficará da seguinte forma:

```
select  
Venda_Codigo,Venda_Data,Venda_Quantidade,tbcliente.Cliente_  
Nome,  
tbvendedor.Vendedor_Nome,tbproduto.Produto_Nome,tbloja.Loja_  
Nome,tbdepartamento.Departamento_Nome  
from tbvenda  
LEFT JOIN tbcliente  
on tbvenda.Venda_cliente=tbcliente.Cliente_Codigo  
LEFT JOIN tbvendedor  
on tbvenda.Venda_Vendedor=tbvendedor.Vendedor_Matricula  
LEFT JOIN tbproduto  
n tbvenda.Venda_produto=tbproduto.Produto_Codigo  
LEFT JOIN tbloja  
on tbvenda.Venda_Loja=tbloja.Lojas_Codigo  
LEFT JOIN tbdepartamento  
on tbvenda.Venda_Departamento=tbdepartamento.Departamento_  
Codigo;
```

Fizemos neste comando SELECT a **junção** de todas as **tabelas** que possuem as informações importantes para o nosso relatório. Partimos da tabela tbvenda e juntamos com as tabelas tbcliente, tbvendedor, tbproduto, tbloja e tbdepartamento. Isto nos permite o acesso a qualquer coluna destas tabelas. Observemos que utilizamos como forma de referenciar as tabelas e suas colunas a sintaxe: “**nome da tabela**” seguido de um “**ponto**” e depois o “**nome da coluna**”. Esta sintaxe não é obrigatória, poderíamos ter colocado simplesmente o nome da coluna, no entanto, esta forma de escrever torna o comando mais claro. Ela somente se torna obrigatória se existirem tabelas que possuam colunas com nomes iguais. Como exemplo, a tabela **tbproduto** poderia ter uma coluna chamada nome que guardaria o **nome** do produto, assim como a **tbcliente** poderia ter uma coluna **nome** que guardaria o nome do cliente, neste caso, para evitar ambiguidades seria obrigatório referenciar estas colunas colocando antes os nomes das tabelas seguido de “ponto” e depois os nomes das colunas.

Assim como poderemos filtrar registros de uma única tabela, por exemplo, montar um comando SELECT que retorne o nome de um estado, utilizando na cláusula WHERE a sigla, desta forma:

```
SELECT * FROM tbestado WHERE Estado_Sigla='CE'
```

Poderemos utilizar um comando SELECT complexo e colocar filtros, como mostrado no exemplo a seguir, onde filtramos os registros da loja “3”:

```
select  
Venda_Codigo,Venda_Data,Venda_Quantidade,tbcliente.Cliente_  
Nome,  
tbvendedor.Vendedor_Nome,tbproduto.Produto_Nome,tbloja.Loja_  
Nome,tbdepartamento.Departamento_Nome  
from tbvenda  
LEFT JOIN tbcliente  
on tbvenda.Venda_cliente=tbcliente.Cliente_Codigo  
LEFT JOIN tbvendedor  
on tbvenda.Venda_Vendedor=tbvendedor.Vendedor_Matricula  
LEFT JOIN tbproduto  
n tbvenda.Venda_produto=tbproduto.Produto_Codigo  
LEFT JOIN tbloja  
on tbvenda.Venda_Loja=tbloja.Lojas_Codigo  
LEFT JOIN tbdepartamento  
on tbvenda.Venda_Departamento=tbdepartamento.Departamento_  
Codigo  
WHERE tbloja.Lojas_Codigo=3;
```

### 3. TABELAS TEMPORÁRIAS

As tabelas temporárias, diferentemente das tabelas que criamos até o momento, têm um **tempo de vida transitório**. Ao nos conectarmos ao SGBD MySQL utilizando MySQL Workbench, criamos uma sessão de usuário e se criarmos uma tabela temporária ela só existirá enquanto a sessão existir, ou seja, se nos desconectarmos do MySQL a tabela deixa de existir.

### 3.1. Vantagens das tabelas temporárias

---

Apesar da sua existência temporária, este tipo de tabela pode ser muito útil em várias situações, como as citadas a seguir:

- Em muitas situações, poderemos ter que desenvolver consultas complexas, com código muito grande, que acaba tornando o uso mais trabalhoso, então, poderemos colocar os dados em tabelas temporárias simplificando o trabalho;
- Consultas complexas são mais difíceis de serem manipuladas e, muitas vezes, precisamos gerar os dados que elas retornam inúmeras vezes. Colocar estes resultados em tabelas temporárias é uma forma eficiente de se ter os resultados em mãos sempre que forem necessários;
- O uso de tabelas temporárias para projetar os dados que serão necessários em uma determinada situação, na qual, posteriormente, será criada uma tabela permanente no ambiente de produção para guardar estes dados, é uma forma eficiente de economizar o trabalho de modificação de estruturas de tabelas permanentes, pois as tabelas temporárias são mais fáceis de manusear, e quando se definir a versão da tabela permanente final, cria-se a tabela e transfere-se os dados da tabela temporária correspondente.

### 3.2. Criação de tabelas temporárias

---

O comando SQL para a criação de tabelas temporárias é bem parecido com o comando para a criação das tabelas permanentes, o detalhe está na cláusula TEMPORARY que vem logo após a cláusula CREATE. Vamos supor que necessitamos criar uma tabela temporária com a mesma estrutura da tabela permanente mostrada no comando a seguir:

```
CREATE TABLE tbcidade (
    Cidade_Codigo INT NOT NULL,
    Cidade_Nome VARCHAR(45) NOT NULL);
```

Para a criação da tabela temporária correspondente basta acrescentar a cláusula TEMPORARY, como mostrado abaixo:

```
CREATE TEMPORARY TABLE tmp_tbcidade (
    Cidade_Codigo INT NOT NULL,
    Cidade_Nome VARCHAR(45) NOT NULL);
```

Todos os comandos de manipulação de dados válidos para uma tabela permanente também serão válidos para uma tabela temporária.

### 3.3. Criação de tabelas temporárias com informações vindas de um comando SELECT

Anteriormente, relatamos que uma das vantagens das **tabelas temporárias** é a facilidade de sua **criação**, nas atividades de desenvolvimento de banco de dados, muitas vezes, nos deparamos com comandos SELECTs complexos. Estes comandos podem ter seus resultados facilmente enviados para uma tabela temporária, que pode ser trabalhada e havendo a necessidade de torná-la permanente, pois sabemos que ao término da sessão do usuário a tabela temporária deixa de existir, poderemos criar a estrutura permanente e enviar os dados vindos da tabela temporária.

Para criar uma tabela temporária com informações vindas de um comando SELECT é bem simples. Como exemplo, no comando a seguir, criaremos a tabela temporária **tmp\_tbestado** com informações vindas da tabela permanente **tbestado**:

```
create temporary table tmp_tbestado
select Estado_Codigo,Estado_nome,Estado_Sigla from tbestado;
```

### 3.4. Exclusão das tabelas temporárias

Como já foi explicado, as tabelas temporárias deixam de existir quando o usuário que as criou desconecta-se do SGDB, mas havendo necessidade de exclusão de alguma tabela, utilizamos o comando abaixo:

```
DROP TABLE tmp_tbestado;
```



## PRATIQUE

1) Escreva duas definições da linguagem SQL.

---

---

---

---

2) Qual a importância da linguagem SQL?

---

---

---

---

Para todas as questões a seguir, escrever o comando SQL.

3) Criar um banco de dados chamado **bdunidade02**.

---

---

---

---

4) Usar o banco de dados criado na questão anterior e criar uma tabela chamada **tbund02cliente**, utilizando a estrutura abaixo:

COLUNA	TIPO DE DADO
codigo	inteiro
nome	Cadeia de caracteres com tamanho 100.

---

---

---

---

5) Na tabela **tbund02cliente**, faça a inserção dos dados mostrados na estrutura abaixo:

CODIGO	NOME
1	José
2	Maria
3	Pedro
4	Ribeiro

6) Altere o nome do cliente cujo código é 2, para o nome **PEDRO LUIS**.

---

---

---

7) Altere tabela **tbund02cliente** colocando a propriedade de **AUTO INCREMENTO** na coluna código.

---

---

---

8) Crie na tabela **tbund02cliente**, utilizando a coluna código, uma **CHAVE PRIMÁRIA**.

---

---

---

9) Crie uma tabela temporária chamada **tbclientetmp**, com os dados vindos da tabela **tbund02cliente**.

---

---

---

---

---

- 10) Crie as tabelas, mostradas abaixo, e escreva um comando *SELECT* que mostre o nome do fornecedor e o nome do produto.

tb fornecedor	
codigo	nome
1	Enel
2	Tim
3	Aço cearense
4	M. Dias Branco

tb produto	
codigo	nome
1	Telefonia
2	Aço
3	Biscoitos
4	Energia



## RELEMBRE

Nesta unidade, aprendemos a instalar e configurar o MySQL. Entendemos que para podermos utilizar os comandos SQL é necessário termos um software servidor, no caso, o SGBD MySQL é uma ferramenta cliente, o MySQL Workbench que nos permite escrever comandos SQL e enviar para o processamento pelo servidor.

Praticamos a linguagem SQL, através da criação de banco de dados, de tabelas e de chaves primárias. Praticamos os conceitos aprendidos na Unidade 01, implementando os relacionamentos entre tabelas através dos comandos SQL. Estudamos ainda as tabelas temporárias, que são objetos importantes para o desenvolvimento de consultas complexas.



## REFERÊNCIAS

BURLESON, Donald K.; CELKO, Joe; GULUTZAN, Peter. **Advanced SQL database programmer handbook**. Rampant Techpress, 2003.

MYSQL. Disponível em: <https://www.mysql.com/>. Acesso em: 03 dez. 2020.

SILBERSCHATZ, Abraham. **Sistema de banco de dados**. Rio de Janeiro, 2006.



## ANOTAÇÕES

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## **ANOTAÇÕES**



## **ANOTAÇÕES**

[www.UniATENEU.edu.br](http://www.UniATENEU.edu.br)



Núcleo de Educação a Distância

Rua Coletor Antônio Gadelha, Nº 621  
Messejana, Fortaleza – CE  
CEP: 60871-170, Brasil  
Telefone (85) 3033.5199