Transformando o futuro das pessoas
e as pessoas para o futuro.

#Senacfaz75

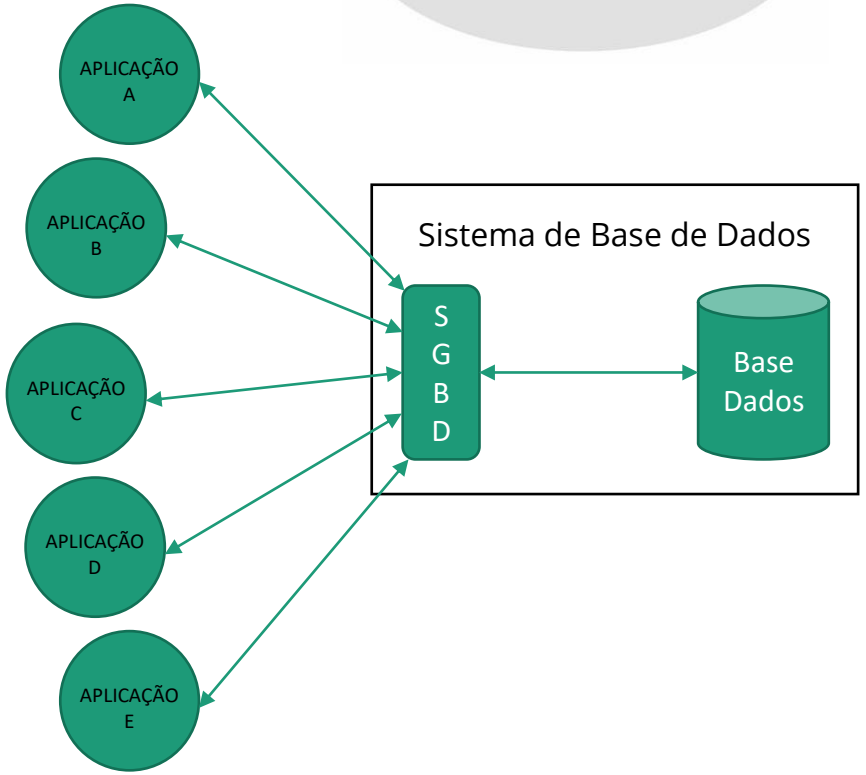
Senac

Fundamentos de
Banco de Dados

O que é um banco de dados?

Um banco de dados ou base de dados nada mais é do que uma coleção de dados inter-relacionados, com algum significado inerente, isto é, informações de interesse de uma ou mais organizações. Ele é projetado, construído e preenchido com dados para um propósito específico de uma ou mais organizações e destinado à utilização por grupo de usuários, diretamente ou por meio de aplicações pré-concebidas.

Podemos também ver um banco de dados como um conjunto de arquivos estruturados de forma a facilitar o acesso aos conjuntos de dados armazenados. Esses arquivos encontram-se, de alguma forma, relacionados. Por exemplo, em um banco de dados escolar podemos encontrar alguns arquivos, tais como: dados dos alunos (nome, endereço, CPF, sexo), dos cursos propriamente ditos (título, ano de lançamento, formato, etc.) e dados sobre as disciplinas (nome, ementa, código, etc.). Para obter informações sobre uma dada disciplina e os alunos que participam dela, como nome da disciplina e nome dos alunos, será necessário consultar os dois primeiros arquivos, que devem estar relacionados. Desta forma, o relacionamento entre os arquivos é uma das condições para que tenhamos um banco de dados, pois somente através destes relacionamentos, teremos a informação propriamente dita.



Technology vector created by user14579558 - www.freepik.com

Entretanto, apenas com a utilização de arquivos relacionados ainda é muito difícil de se acessar estas informações. Dado o acréscimo do volume e dos tipos de dados nas empresas, há uma necessidade de se utilizar softwares especiais para gerenciar as informações, e esses softwares são denominados SGBDs (Sistemas Gerenciadores de Banco de Dados).

Os sistemas gerenciadores de bancos de dados são softwares que manipulam, com eficiência grandes coleções de informações estruturadas e armazenadas de uma forma consistente e integrada. Estes sistemas possuem módulos para consulta, atualização e as interfaces entre o sistema e o usuário. Podemos afirmar, então, que um SGBD e constituído por um conjunto de dados associados a um conjunto de programas para acesso a eles [SILBERSCHATZ, 2006].

Utilidade:

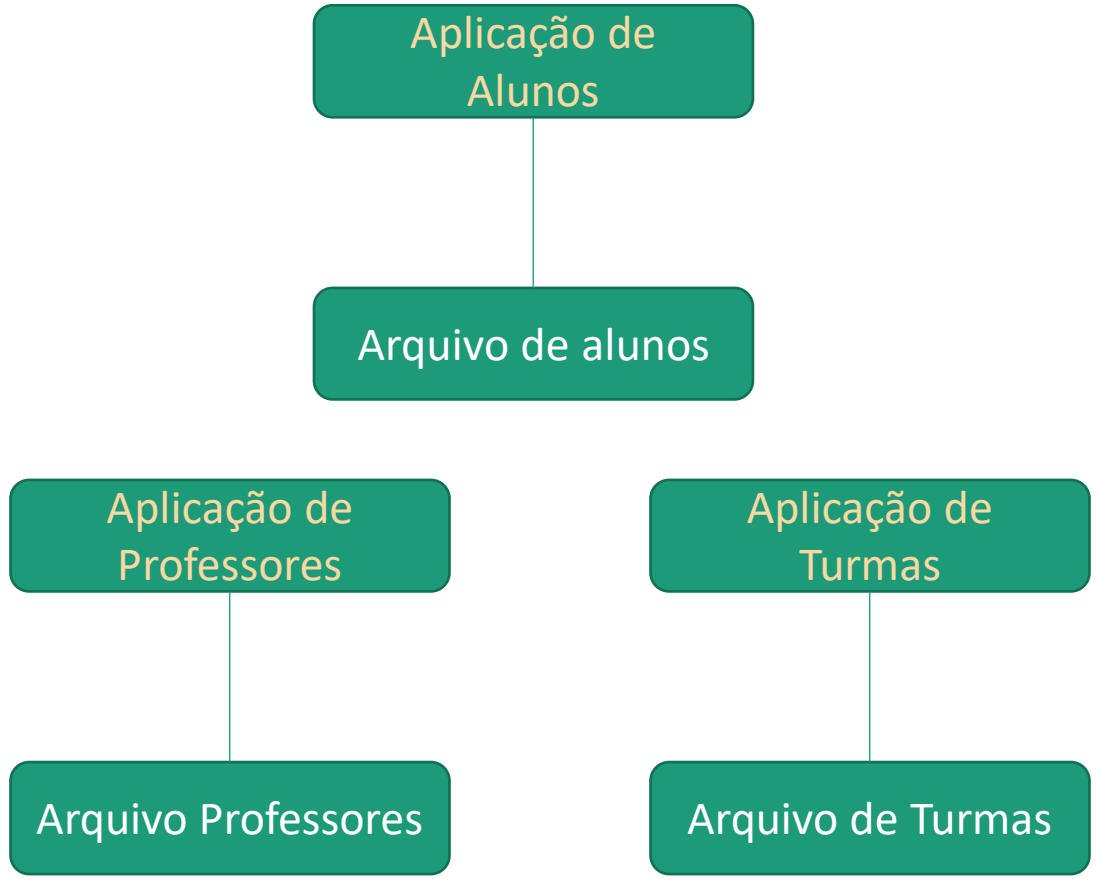
- Sistema acadêmico
- Agências Bancárias
- Rede Hoteleira
- Bibliotecas digitais
- Redes de supermercado etc

Exemplos de Sistemas Gerenciadores de Banco de Dados

Para saber mais sobre cada um deles, faça uma pesquisa para conhecê-los mais a fundo.

Problemas com a não utilização de um Sistema de Gerenciamento de Banco de Dados

Se não houvesse tais sistemas, nossas informações poderiam ficar armazenadas em um conjunto muito grande de arquivos e sem ligação entre si. Assim os dados de uma aplicação não estariam integrados com os dados de outra aplicação, poderia haver redundância, dados inconsistentes, ou seja, tem um valor em um arquivo e pode ter valor diferente em outro. Não teríamos uma integralidade de dados, veja a figura abaixo:



Problema 1: Redundância

Informações sendo repetidas em vários arquivos. Sendo assim poderíamos ter uma informação dentro de aluno com um valor e a mesma informação dentro de turma com um valor diferente, ou seja, gerando inconsistência em toda a estrutura.

Problema 2: Dificuldade de Acesso

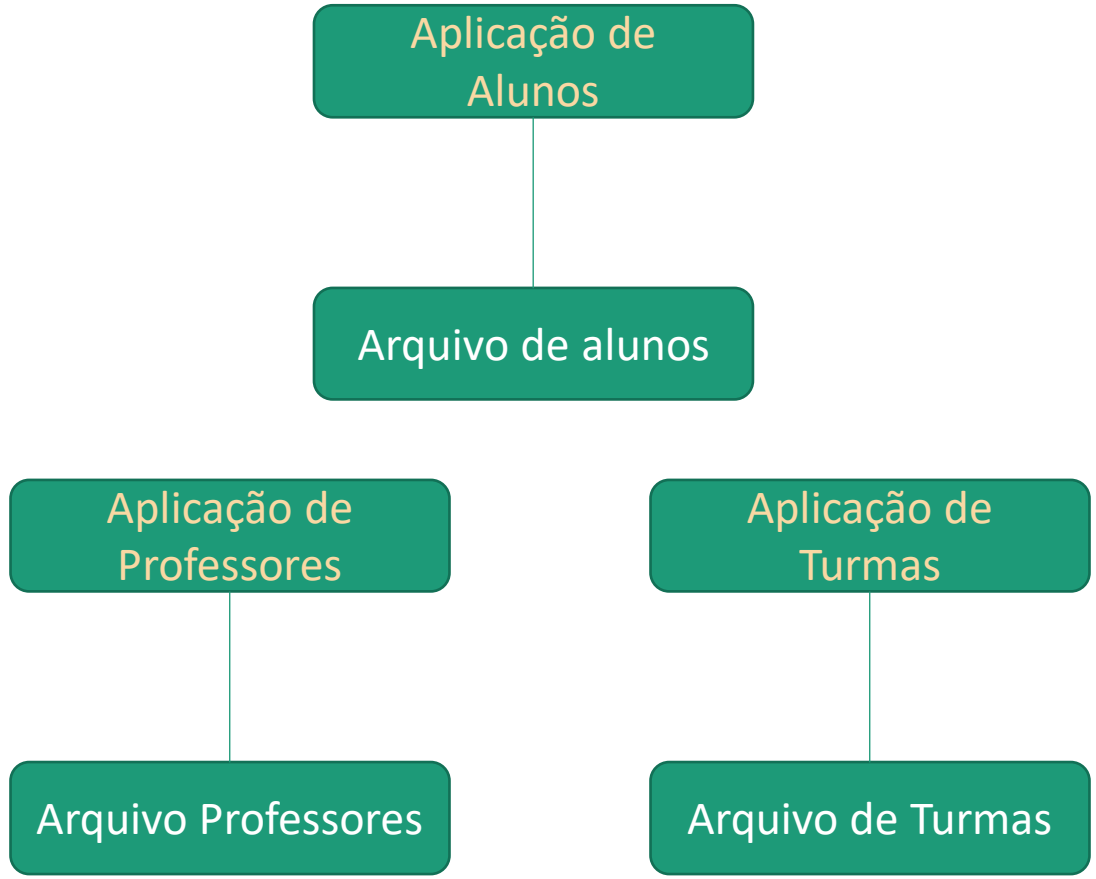
Sistemas de arquivos não possuem ambiente de recuperação de dados armazenados.

Problema 3: Isolamento de dados

Como em cada um dos arquivos que armazenam dados de formas diferentes, fica difícil a construção de uma aplicação que converse com tantas informações guardas sem uma regra de armazenamento.

Problema 4: Atomicidade

Em SGBDs essa é uma regra de ouro, uma coisa é ou não é! Não há espaço para o meio termo. Eu faço tudo quando estiver tudo certo ou não faço nada. Isso não existe em sistemas de arquivos, pois em seu armazenamento não há algum tipo de controle. Exemplo: transferência bancária.



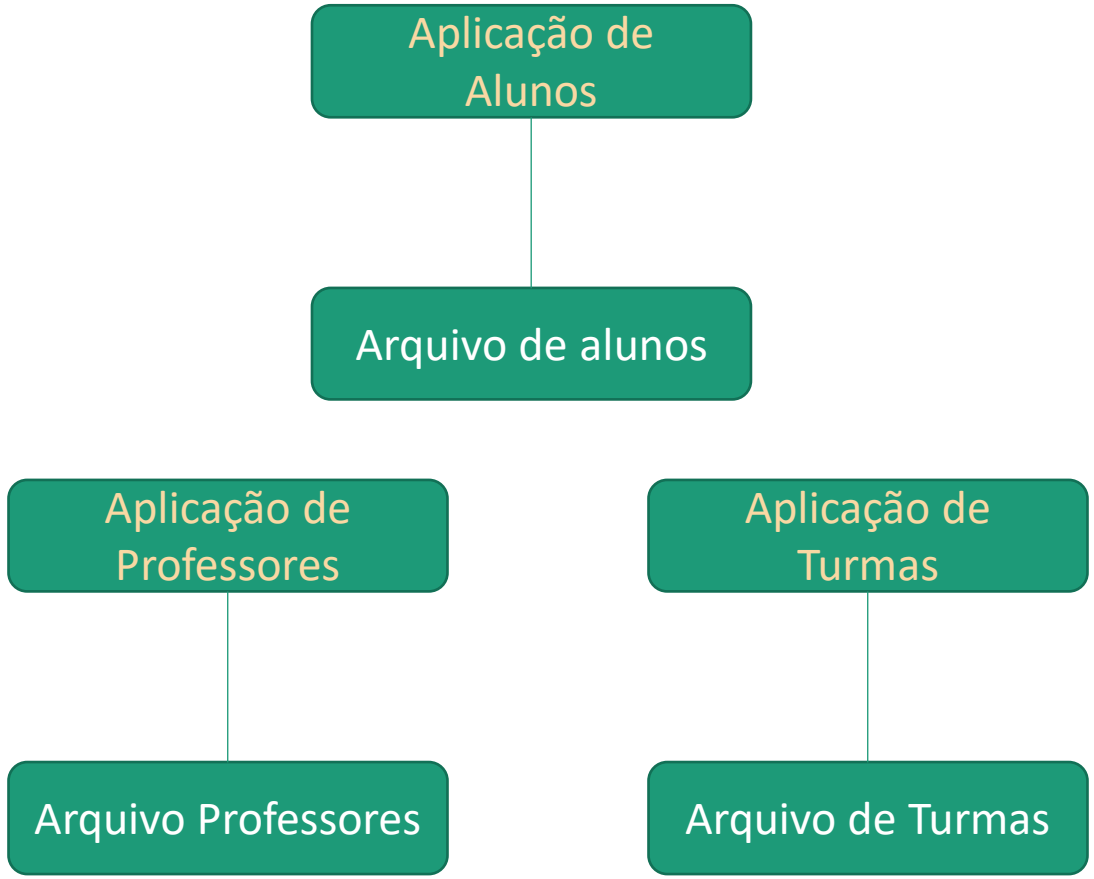
Problema 5: Concorrência de acesso

Em sistemas de arquivos não temos controle de quem está acessando e o quê simultaneamente. veja exemplo:

> Tomemos como exemplo que uma conta conjunta A de fundo de formatura- com saldo igual a R$ 1000,00 - foi acessada de forma simultânea pelos alunos Carolina e Pedro. Carolina sacou R$100,00 para pagar os convites e Pedro, R$200,00 para comprar refrigerantes. Pergunta-se: qual o saldo da conta após os saques? Se ambos leram o valor do saldo igual a R$1000,00, podemos ter como possíveis valores : R$900,00, R$800,00, levando-se em conta qual valor foi escrito por ultimo. Nesse caso, nenhum dos dois valores seria valido. O correto seria ter um saldo igual a R$700,00 e o banco de dados garante isso, mas arquivos isolados podem não garantir. (BRAGA, 2013. P.11.

Problema 6: Segurança

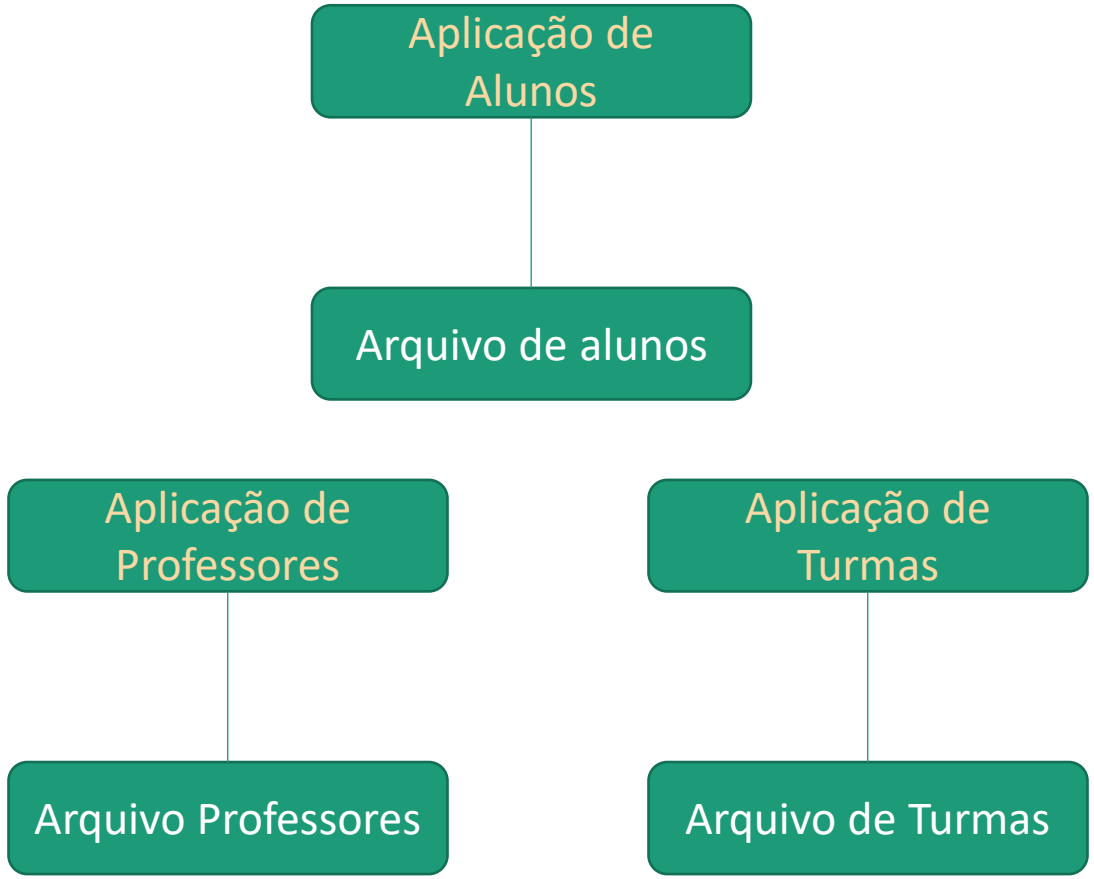
Será que todos os usuários tem perfil para acessar a base de dados em sistemas de arquivos? Imaginemos um usuários tendo total acesso aos dados de um professor em um sistema acadêmico, com acesso a cpf, salários etc. Não seria no mínimo politicamente correto.



Problema 7: Integridade

Vejamos um caso peculiar:

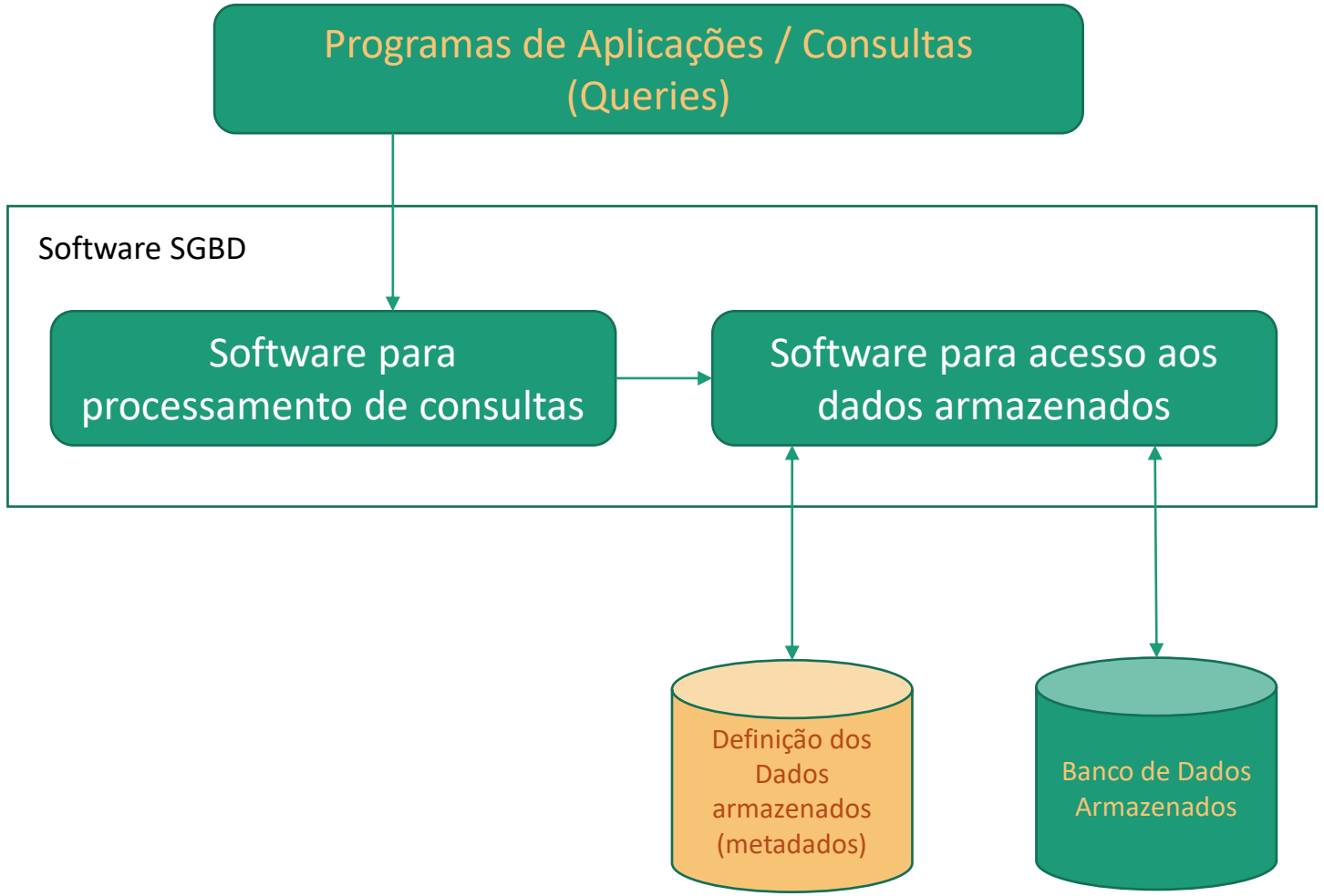
> Imaginem o seguinte problema, estamos cadastrando todos os alunos de uma dada turma e, alguns alunos não tem CPF próprio e usam o CPF da mãe. Imaginem agora que temos dois alunos gêmeos e eles vão ser cadastrados usando o mesmo CPF e estamos utilizando o CPF para identificação dos alunos. O que teríamos neste caso: INCONSISTÊNCIA, uma vez que dois alunos tem o mesmo CPF. Em um sistema de arquivos, isso poderia passar desapercebido mas em um banco de dados, existe o controle de **integridade de chave primária** que não deixa isso acontecer. (BRAGA, 2013. p. 12)



Sistema de Gerenciamento de Banco de Dados (Estrutura)

Sistema de Gerenciamento de Banco de Dados (SGBD) em sua composição possui um conjunto de software para gerenciar um banco de dados.

Veja na figura a seguir.

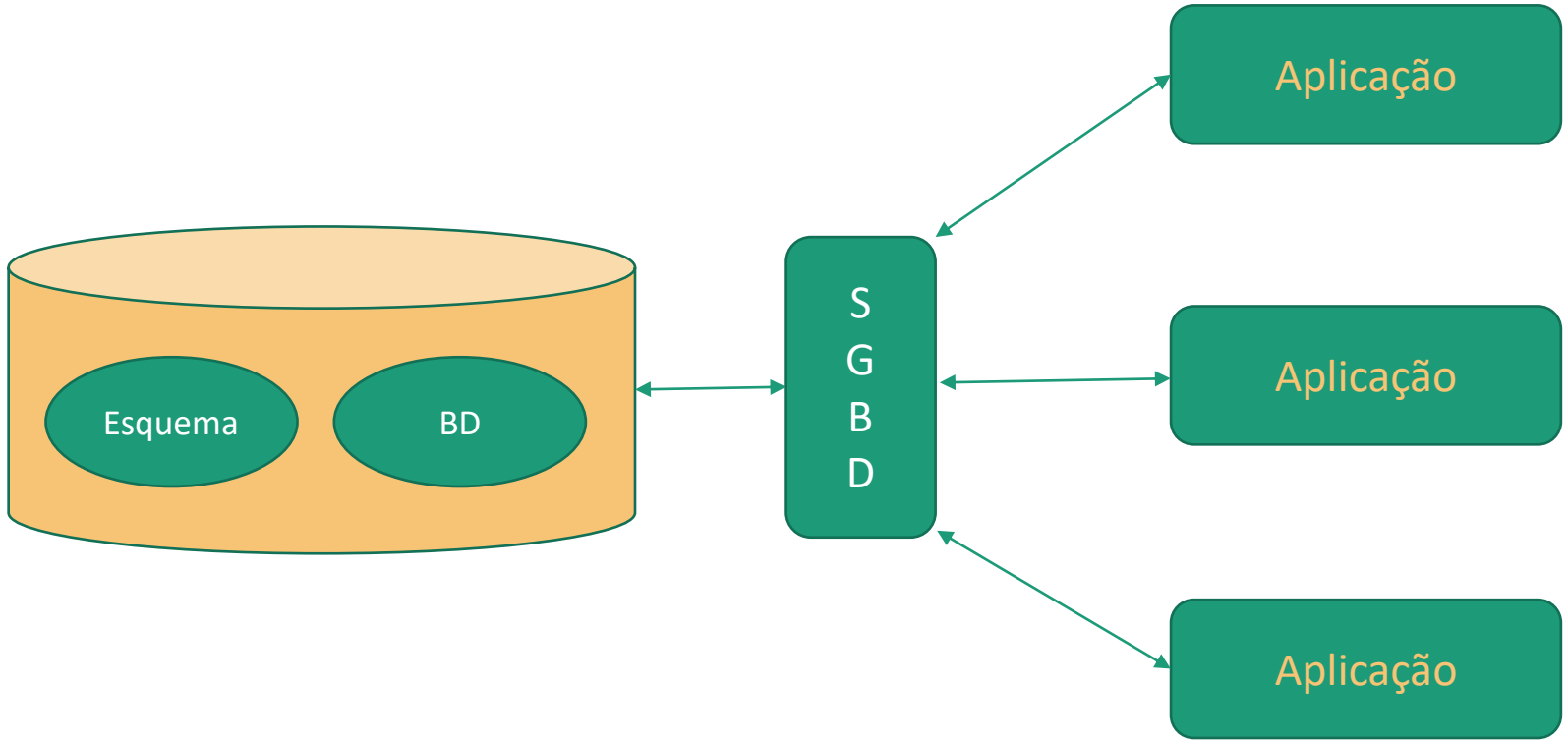


Sistema de Gerenciamento de Banco de Dados (Estrutura)

O Banco de Dados é em si um repositório de informações organizadas que se relacionam. Isso quer dizer que os dados serão armazenados nos arquivos relacionados ao SGBD e também ao esquema que será montado pelo SGBD.

O SGBD permite armazenamento e acesso multiusuário eficiente a uma grande quantidade de dados armazenados, além de garantir a integridade e segurança dos dados.

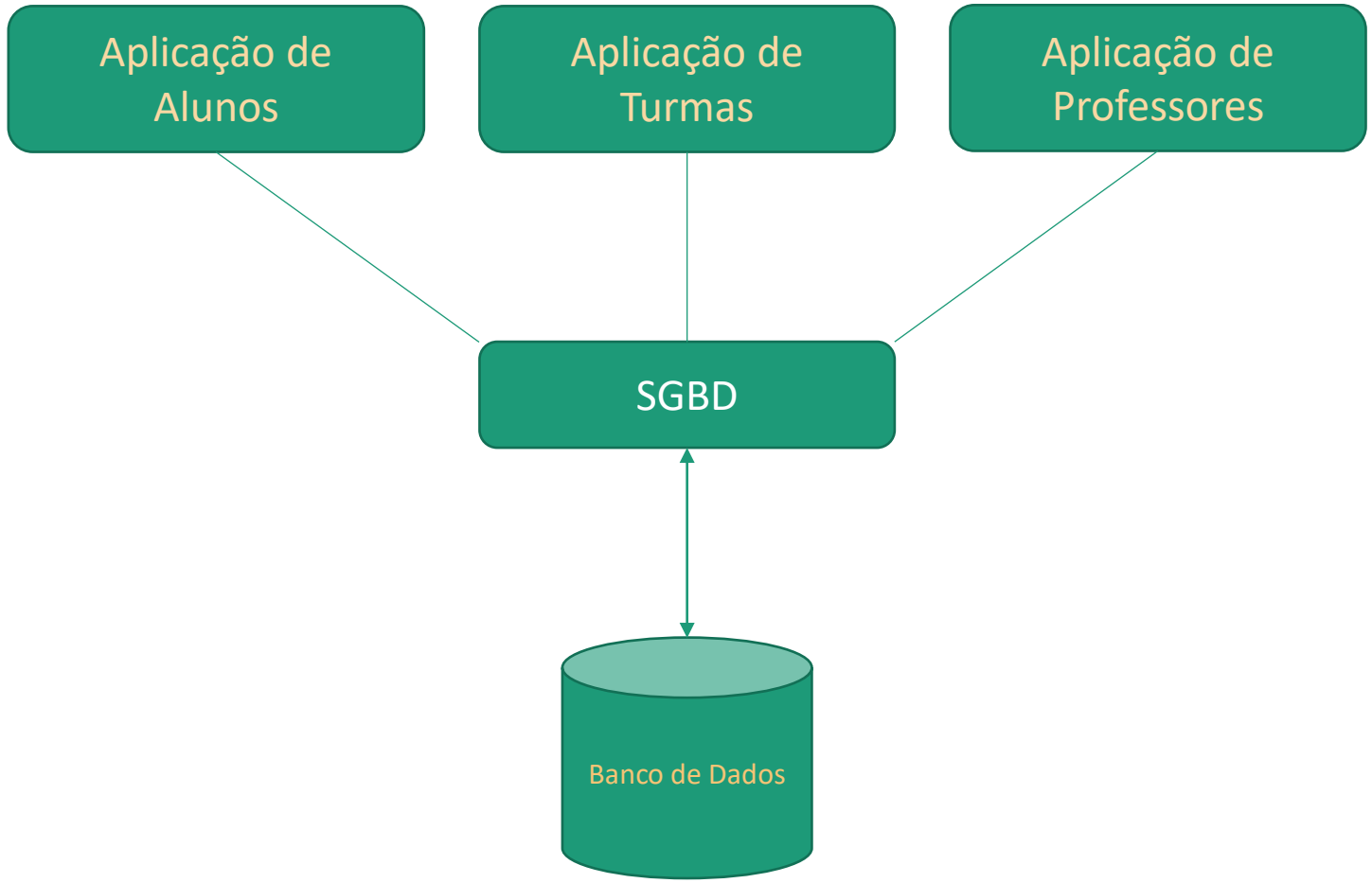
• Ao lado temos uma visão minimizada do que realmente um SGBD é em usa totalidade. Veremos mais adiante que ele é bem mais complexo do que isso.



Sistema de Gerenciamento de Banco de Dados (Estrutura)

Usando com ponto de partida uma aplicação, o manuseio de dados com o uso dos SGBDs se tornam simplificados, ou seja, as informações são integradas no Banco e o SGBD faz todo o controle de inconsistência e redundância.

Vejamos ao lado, como ficaria o exemplo dado no slide 5, quando começamos a falar sobre os problemas com base de dados em arquivos.



Instâncias e Esquemas

Estes termos, embora relacionados, não significam a mesma coisa. Um esquema de banco de dados é um esboço de um banco de dados planejado. Na verdade, ele não contém dados.

Uma instância de banco de dados, por outro lado, é um retrato de um banco de dados da forma como existia em um determinado momento. Sendo assim, instâncias de banco de dados podem mudar ao longo do tempo, enquanto um esquema de banco de dados é geralmente estático, já que é difícil mudar a estrutura de um banco de dados a partir do momento que estiver operacional.

Esquemas e instâncias de banco de dados podem se afetar mutuamente por meio de um sistema de gerenciamento de banco de dados (SGBD). O SGBD assegura que cada instância de banco de dados esteja em conformidade com as restrições impostas pelos designers do banco de dados no esquema de banco de dados.

Esquema

id	nome	numero	turma
1	Laura Dern	10	A
2	Amy Adams	5	B
3	Ana Kendrick	18	C
4	Antony Hopkins	15	D

Instância

Instâncias e Esquemas

Estes termos, embora relacionados, não significam a mesma coisa. Um esquema de banco de dados é um esboço de um banco de dados planejado. Na verdade, ele não contém dados.

Uma instância de banco de dados, por outro lado, é um retrato de um banco de dados da forma como existia em um determinado momento. Sendo assim, instâncias de banco de dados podem mudar ao longo do tempo, enquanto um esquema de banco de dados é geralmente estático, já que é difícil mudar a estrutura de um banco de dados a partir do momento que estiver operacional.

Esquemas e instâncias de banco de dados podem se afetar mutuamente por meio de um sistema de gerenciamento de banco de dados (SGBD). O SGBD assegura que cada instância de banco de dados esteja em conformidade com as restrições impostas pelos designers do banco de dados no esquema de banco de dados.

Esquema

id	nome	numero	turma
1	Laura Dern	10	A
2	Amy Adams	5	B
3	Ana Kendrick	18	C
4	Antony Hopkins	15	D

Instância

Independência de Dados

É a capacidade de fazer alterações na definição dos esquemas em determinado nível, sem afetar o esquema do nível superior.

Independência de dados física:

Faz uma modificação no esquema do BD sem que, com isso, qualquer programa aplicativo precise ser reescrito. Exemplificando essa teoria, nós poderíamos ter modificado a estrutura de dados utilizada para o armazenamento dos dados sem afetar os demais níveis. O que vocês acham disso? É claro que não vai afetar em nada os níveis acima pois a estrutura lógica ficará a mesma!

Independência de dados lógica:

Faz uma modificação no esquema do BD sem que, com isso, qualquer programa aplicativo precise ser reescrito. Um exemplo concreto seria: Vamos considerar os alunos do curso. Imaginemos que que além dos dados como nome, data de nascimento, turma e disciplina, teríamos também que agora armazenar os endereços dos alunos, ou seja, teríamos que modificar a tabela alunos para incluir este novo campo! Qual o impacto disso? Com certeza um pouco maior que a mudança no nível físico, pois pelo menos na aplicação onde é feito o cadastramento dos alunos, teríamos que fazer mudanças!!!

Modelo de Dados

Descreve a estrutura conceitual dos dados armazenados no banco de dados, detalhando a organização dos dados internamente no banco.

Como exemplo, se criarmos um banco de dados para um sistema acadêmico. O modelo de dados a ser criado deverá descrever o que este banco de dados deve armazenar:

- Para **alunos**: Matrícula, nome, endereço, cidade, estado...
- Para as **notas**: Matrícula, turma, aluno...

O modo como as informações são organizadas no banco também pode mudar. Assim, um BD pode ser classificado em:

- **Relacional (nosso estudo)**
- Redes
- Hierárquico
- Orientado a objetos
- Objeto-relacional

Modelo Relacional

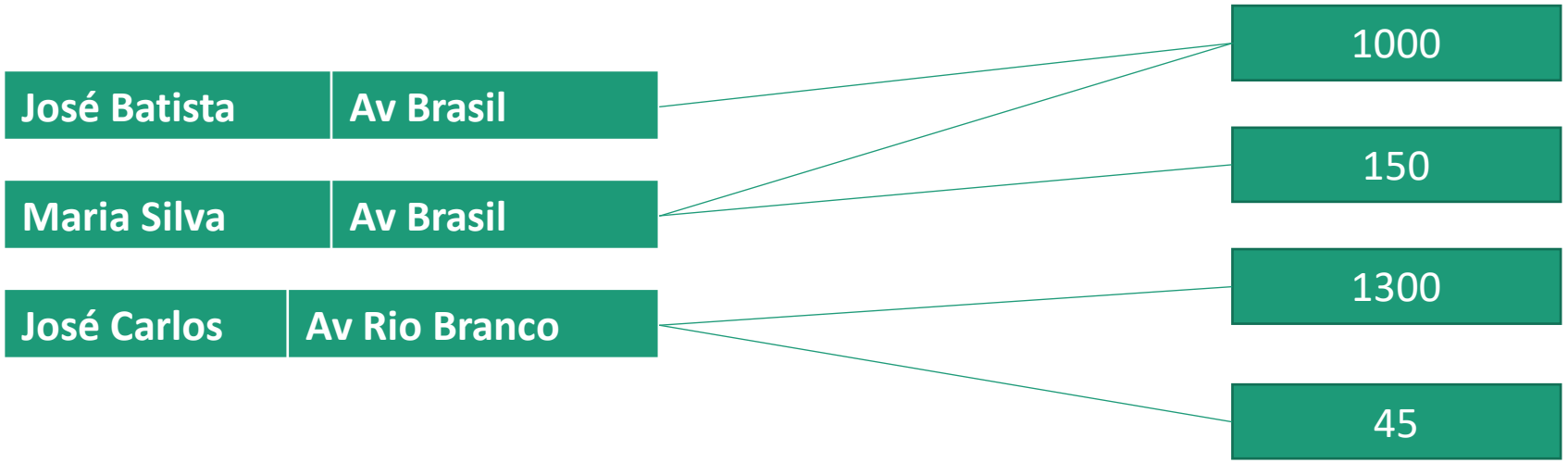
Este modelo é amplamente utilizado atualmente pelos Sistemas Gerenciadores de Banco de Dados. A **simplicidade** e a uniformidade são suas características mais relevantes em conjunto com sua linguagem padronizada para consulta (SQL - Structured Query Language).

Um conjunto de tabelas é usado para representar tanto os dados quanto a relação entre eles. As ligações entre as tabelas e feita por meio dos valores dos atributos ou colunas, conforme descrito posteriormente. Veja exemplo a seguir:

id	nome	Endereço	Cidade	Estado	Cep
1	John Doe	Rua Roque Picorelli, 15 - Grajaú	Juiz de Fora	MG	36052-330
2	Jane Silva	Rua Adofo Kirchmaier, 166 – São Pedro	Juiz de Fora	MG	36058-120
3	João Ferreira	Rua São Mateus, 1350 – São Mateus	Barbacena	MG	37055-330
4	Maria Pereira	Rua Halfeld, 72 - Centro	Juiz de Fora	MG	35100-220

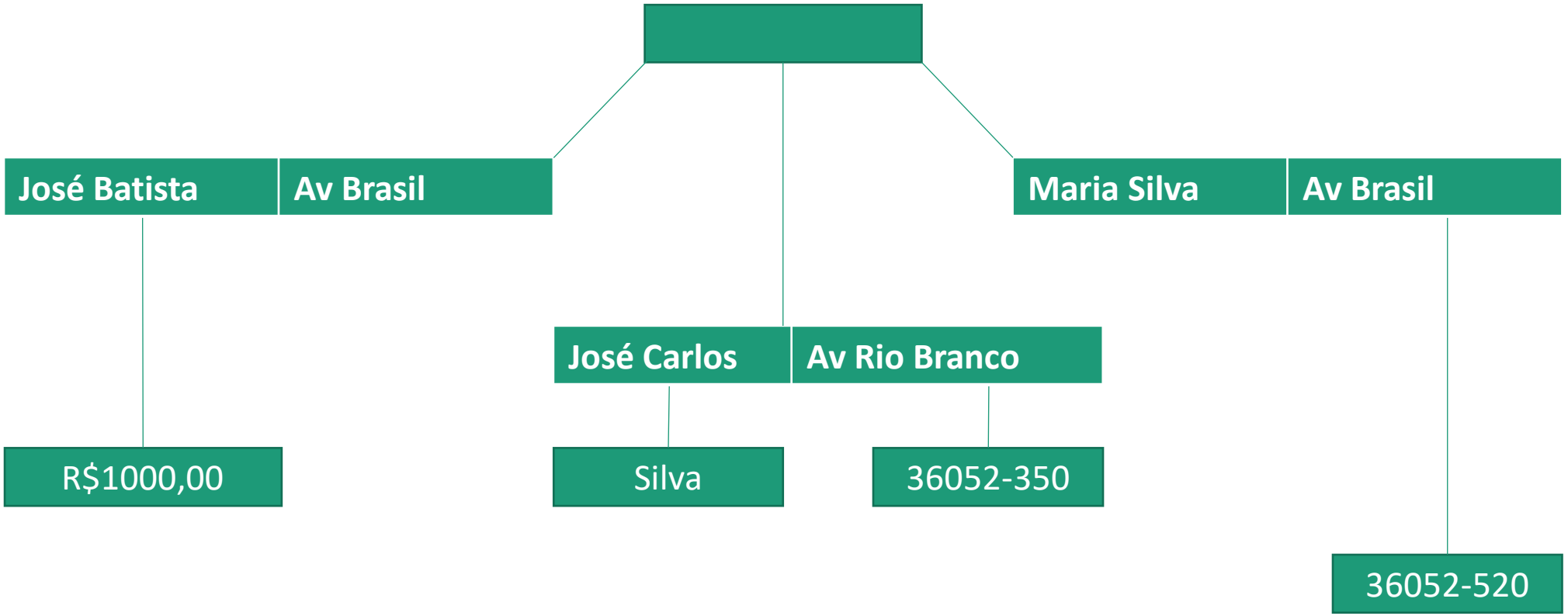
Modelo de Redes

Uma coleção de registro que são concatenados uns aos outros por meio de ligações é considerada um banco de dados em rede. Não é um modelo de Banco de Dados mais utilizado pelos SGBSs, sendo mais ligado a forma de se acessar o dado fisicamente, ao contrário do modelo relacional, que permite uma visão mais lógica do banco de dados. Os registros dos Banco de Dados são organizados por um conjunto arbitrário de grafos. Veja exemplo:



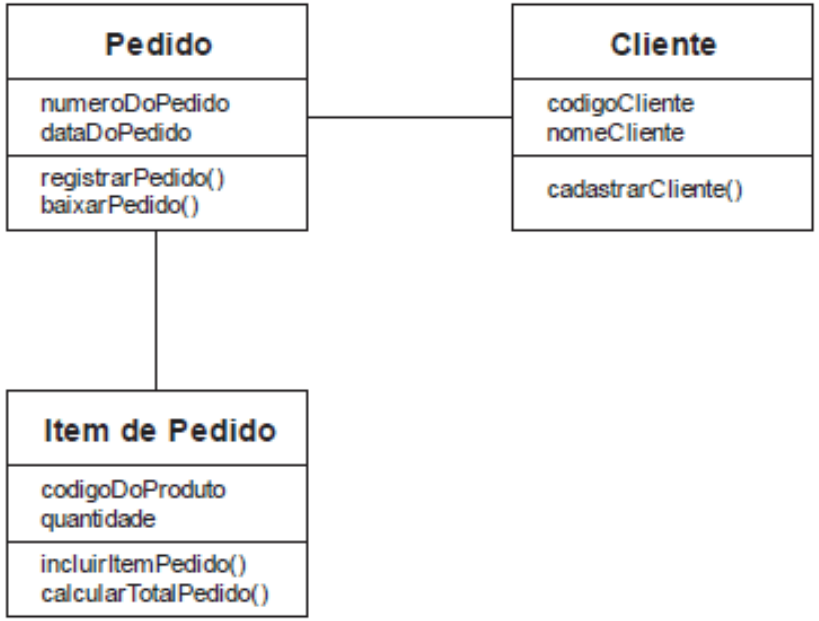
Modelo Hierárquico

Este modelo de BD se assemelha bastante com o modelo de Rede pois os dados e suas relações são representados por registros e ligações. A diferença é que no modelo hierárquico os registros estão organizados em árvores ao invés de grafos aleatórios como modelo de redes. O modelo hierárquico também não é mais utilizado como modelo de BDs atuais mesmo tendo como grande vantagem o desempenho das consultas por utilizar uma estrutura em árvore.



Modelo Orientado a Objetos

Este modelo apareceu para suprir uma lacuna no armazenamento de dados providos a partir das linguagens de programação que utilizam a Orientação a Objetos. Se levarmos em consideração as aplicações onde trabalhamos a partir de objetos (como em Java), por que armazenar os dados dos objetos em tabelas e não em objetos? É nessa premissa que se baseou a construção de BDs orientados a objetos, pois conceitos como classes, herança, métodos, associações entre classes foram incorporados nos bancos de dados. Entretanto, mesmo de toda a flexibilidade do modelo, este tipo de banco de dados não foi bem aceito no mercado.



Conceitos de Manipulação de Banco de Dados

Linguagem de Definição de Dados (DDL - Data-Definition Language **)**

Para manipularmos a criação de uma Base de Dados precisamos esquematizar um Banco de Dados, ou seja, criação de tabelas, índices, etc., para isso utilizamos a Linguagem de Definição de Dados DDL com os seguintes comandos:

CREATE TABLE: para criar uma tabela
ALTER TABLE: alteração de tabelas
DROP TABLE: apagar uma tabela
CREATE INDEX: criação de índices
ALTER INDEX: alteração de índices
DROP INDEX: apagar um índice

Linguagem de Manipulação de Dados (DML - Data Manipulation Language**)**

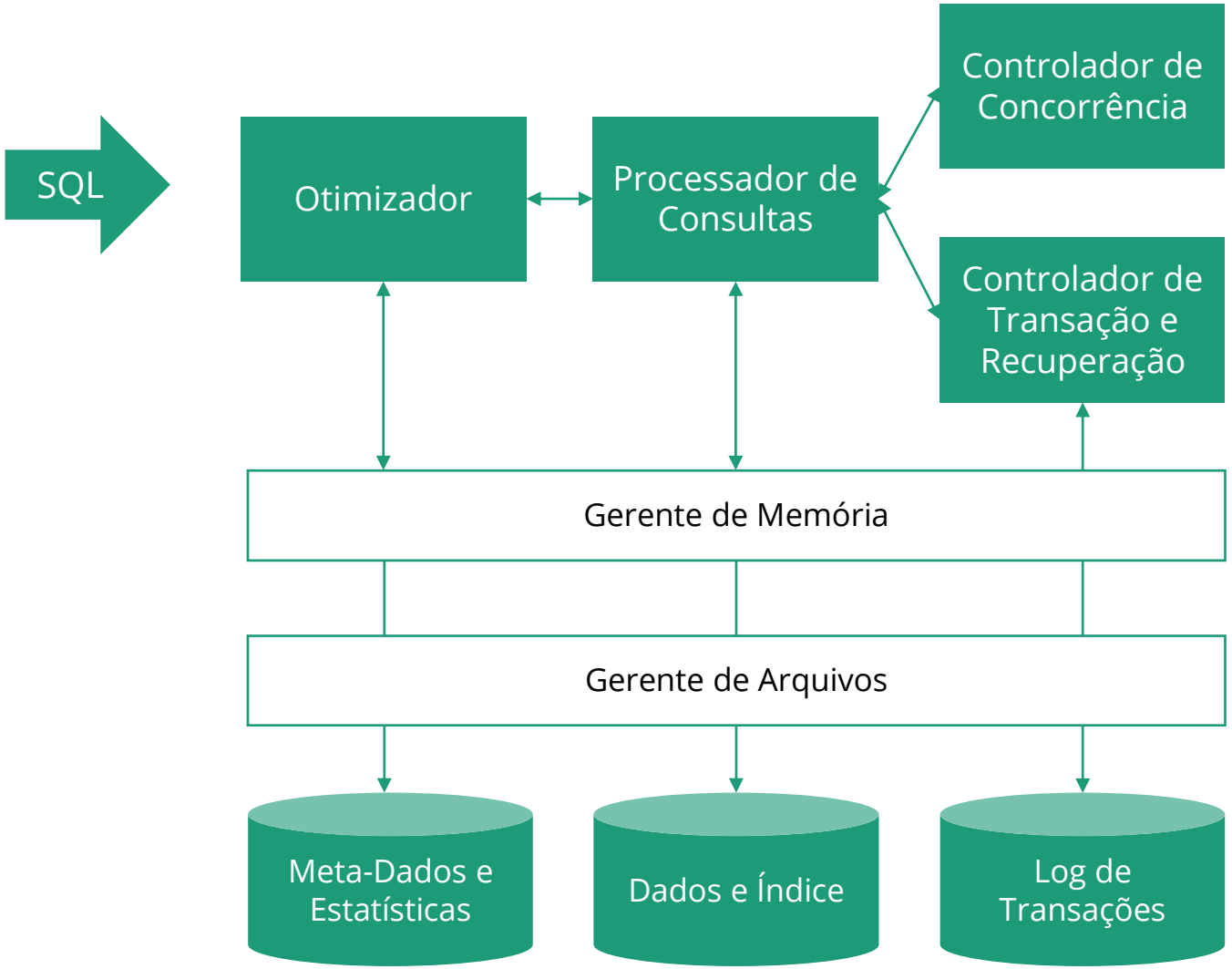
A DML permite a manipulação de dados. Podemos realizar as seguintes operações a partir da LMD:

- SELECT - recuperar dados de um banco de dados
- INSERT - inserir dados em uma tabela
- UPDATE - atualiza os dados existentes em uma tabela
- DELETE - Exclui todos os registros de uma tabela de banco de dados

A DML viabiliza o acesso (manipulação) dos dados de forma compatível ao modelo de dados apropriado. Tanto a DDL quanto a LMD fazem parte da linguagem SQL (Structured Query Language), que veremos mais adiante.

Principais componentes de um SGBD

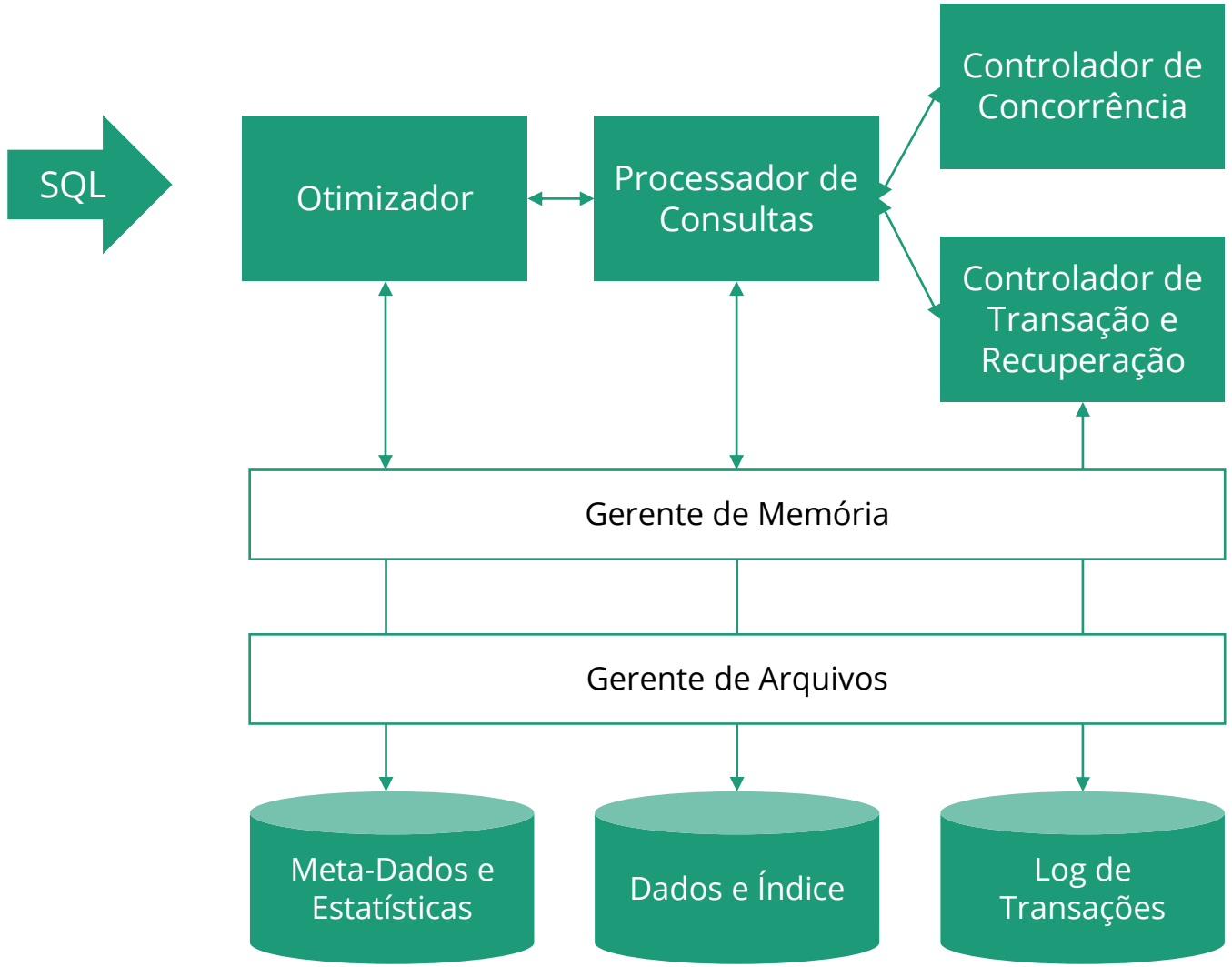
Um SGBD é dividido em módulos específicos, de modo a compreender a todas as suas funções, algumas delas advindas d sistema operacional. Os módulos podem ser organizados em dois grupos: o de processamentos de consultas e o de administração do armazenamento de dados.



Principais componentes de um SGBD

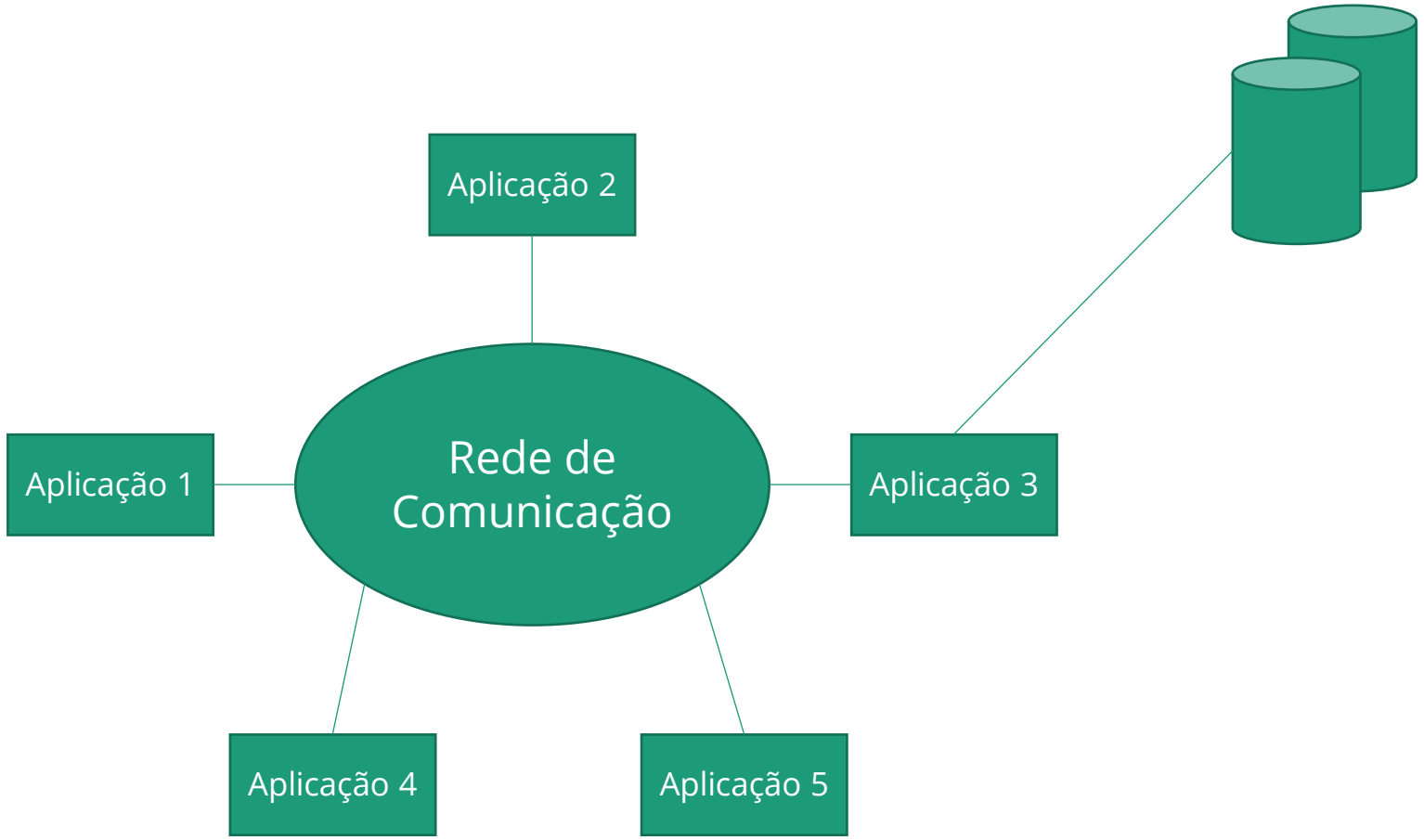
Processamento de Consultas: Constituído pelo **otimizador** e **processador de consultas**. Seu principal objetivo é receber uma requisição do usuário especificada em SQL e melhorar esta requisição em termos de desempenho (fazer com que a consulta seja executada o menor tempo possível) e acessar o e disponibilizar os dados requeridos pelo usuário;

Administração do armazenamento de dados: Garante o estado consistente do BD, não permitindo acesso não autorizado aos dados e nem o acesso simultâneo aos dados. Gerencia a alocação de espaço em disco e as estruturas de dados utilizadas para armazenar as informações. Outro componente importante é o gerenciamento das trocas de dados entre o disco e a memória, o que é feito pelo **gerente de memória**.



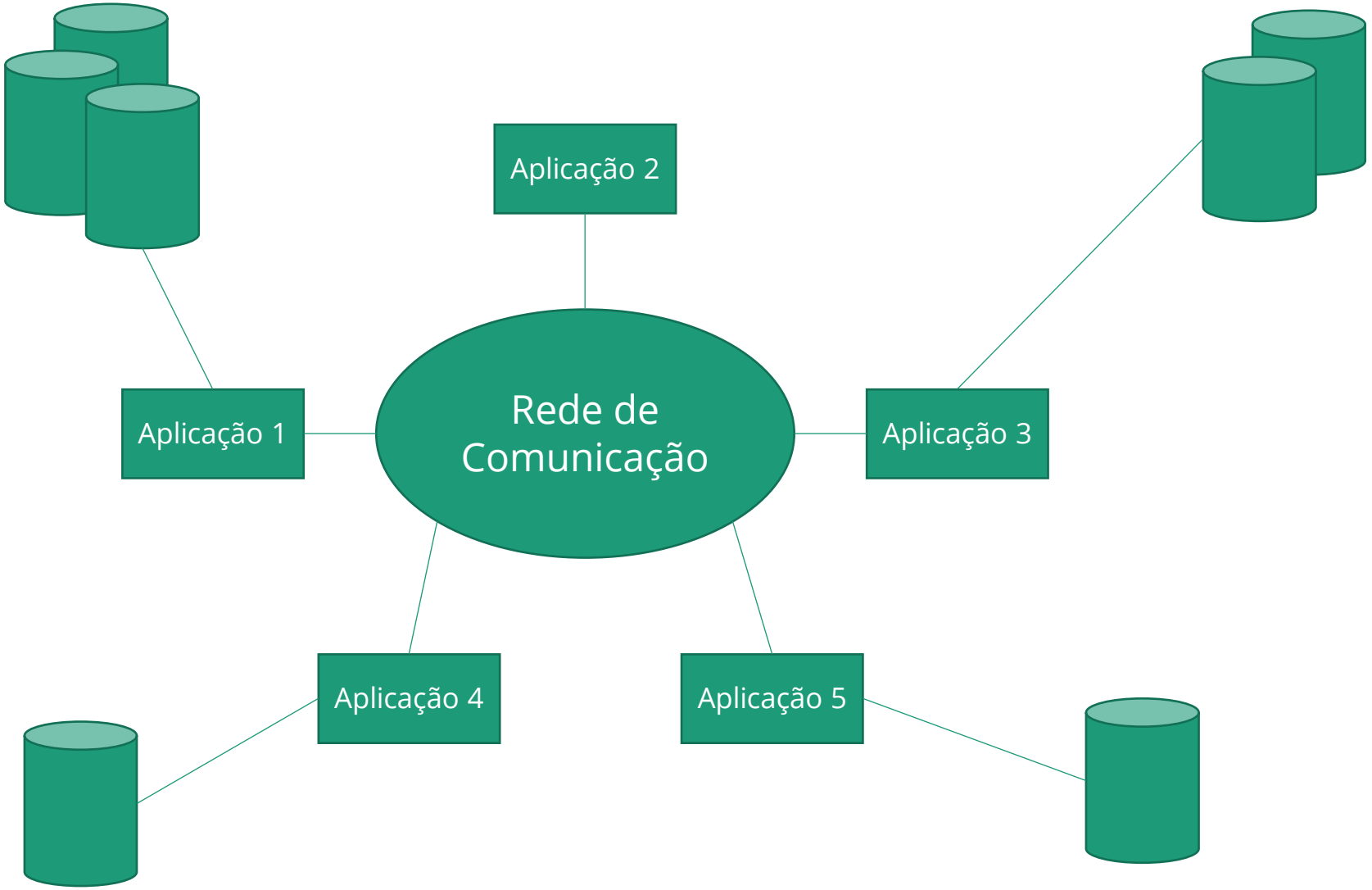
Arquitetura de um SGBD

Arquitetura Centralizada: O Sistema Gerenciador de Banco de Dados é executado e armazenado em uma máquina, podendo ser consultado através de outros computadores com acesso a este computador denominado servidor de SGBD. O acesso pode ser multiusuário, mas o SGBD está centralizado é uma máquina (o servidor).



Arquitetura de um SGBD

Arquitetura Distribuída: O Sistema Gerenciador de Banco de Dados é distribuído em várias máquinas e/ou aplicações/sites, sendo que o processamento de qualquer requisição é gerenciada de forma a acessar as "partes" onde estão armazenadas as informações relevantes.



Usuários de um banco de dados

Usuários leigos: Fazem acesso de alto nível ao BD por meio das telas de aplicativos que acessam o base de dados;

Usuários avançados: usuários que trabalham com os BDs por intermédio de telas específicas disponíveis no SGBD. Desenvolvem consultas SQL e executam sem a necessidade de escrever uma aplicação para esse fim;

Programadores aplicações: usuários com formação em computação que constroem aplicações, por meio de ferramentas (compiladores) destinadas para esse fim. Com essas ferramentas, desenvolvem interfaces para as aplicações, adicionando formulários e relatórios que acessam o bancos de dados;

Administrador de Banco de Dados (DBA – DataBase Administrator): tipo de usuário especializado. Cabe a ele a administração dos BDs, definir a melhor estrutura de armazenamento desses dados, definir aspectos de segurança, programação de copias de segurança, dentre outros.

Atividade 1 BD

1. Cite quatro sistemas que você tenha usado/visualizado recentemente e que provavelmente tenha um SGBD para realizar a persistência dos dados.
2. Detalhe quais são as principais funções de um administrador de um banco de dados
3. Descreva pelos menos 3 tabelas que você acredita que precisariam ser usadas para armazenar informações em um sistema de redes sociais como o Facebook.
4. Liste os primeiros passos que você seguiria para a criação de um banco de dados para uma empresa e justifique.
5. Defina as principais categorias de modelos de dados.
6. Dê uma vantagem e uma desvantagem de se usar um SGBD em um sistema, justifique sua resposta.

Fundamentos de
Banco de Dados e linguagem SQL

Introdução

Os bancos de dados relacionais foram beneficiados com a implementação da linguagem de consulta SQL que por sua vez fez com que ele se estabelecesse no mercado. Desde sua definição como padrão, em 1986, passou por diversas revisões, gerando publicações de novas versões. A linguagem SQL não é somente uma linguagem de consulta. Ela possui diversas outras funcionalidades, veja a lista a seguir :

Linguagem de Definição de Dados (DDL)

Agrega comandos para definição de esquemas de relações, exclusão de relações, criação de índices e modificações do esquema de relações. Alguns exemplos de comandos:

CREATE TABLE - Cria uma nova tabela com seus campos e define as restrições de campo.
ALTER TABLE - Altera as definições de campos e de restrições.
CREATE INDEX - Cria um novo índice em uma tabela existente.
DROP - Exclui uma tabela existente de um banco de dados ou exclui um índice existente de uma tabela.

Linguagem de manipulação de dados (DML) - agrega comandos para inserção, exclusão e modificação de registros no banco de dados.

INSERT : inserção de registros.
DELETE : deleção de registros.
UPDATE : atualização de registros.
SELECT : seleção de registros.

Linguagem de Controle de Dados - DCL (Data Control Language) - agrega comandos para concessão/autorização de acessos as informações no banco de dados.

GRANT : concessão de privilégios a tabelas e visões
o REVOKE : revogação de privilégios a tabelas e visões

Linguagem para Controle de Transação - possibilita o controle do processamento das transações (ações executadas no banco de dados)

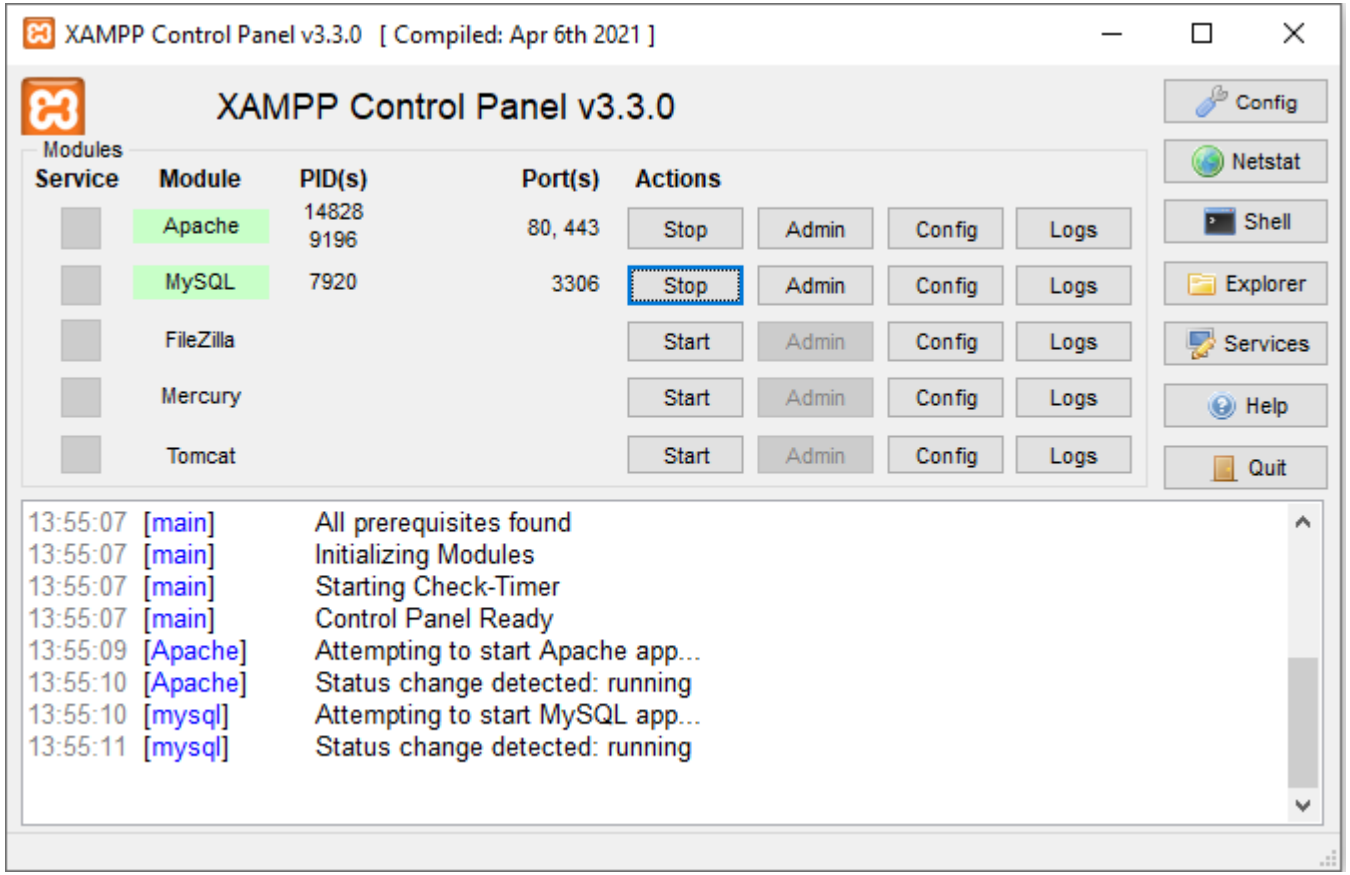
COMMIT : efetiva uma alteração no banco de dados
ROLLBACK : desfaz uma alteração antes de ser efetivada no banco
SAVEPOINT : permite uma subdivisão lógica de uma transação longa

Restrições de integridade: são usados para garantir a exatidão e a consistência dos dados em uma Banco de dados relacional. Ou seja, garantir que dados representem assertivamente a realidade modelada. A integridade dos dados é tratada nas bases de dados através do conceito de integridade relacional e é garantida pelo próprio SGBD.

STORED PROCEDURES (procedimentos armazenados no banco)
TRIGGERS (gatilhos)

Criando um banco de Dados

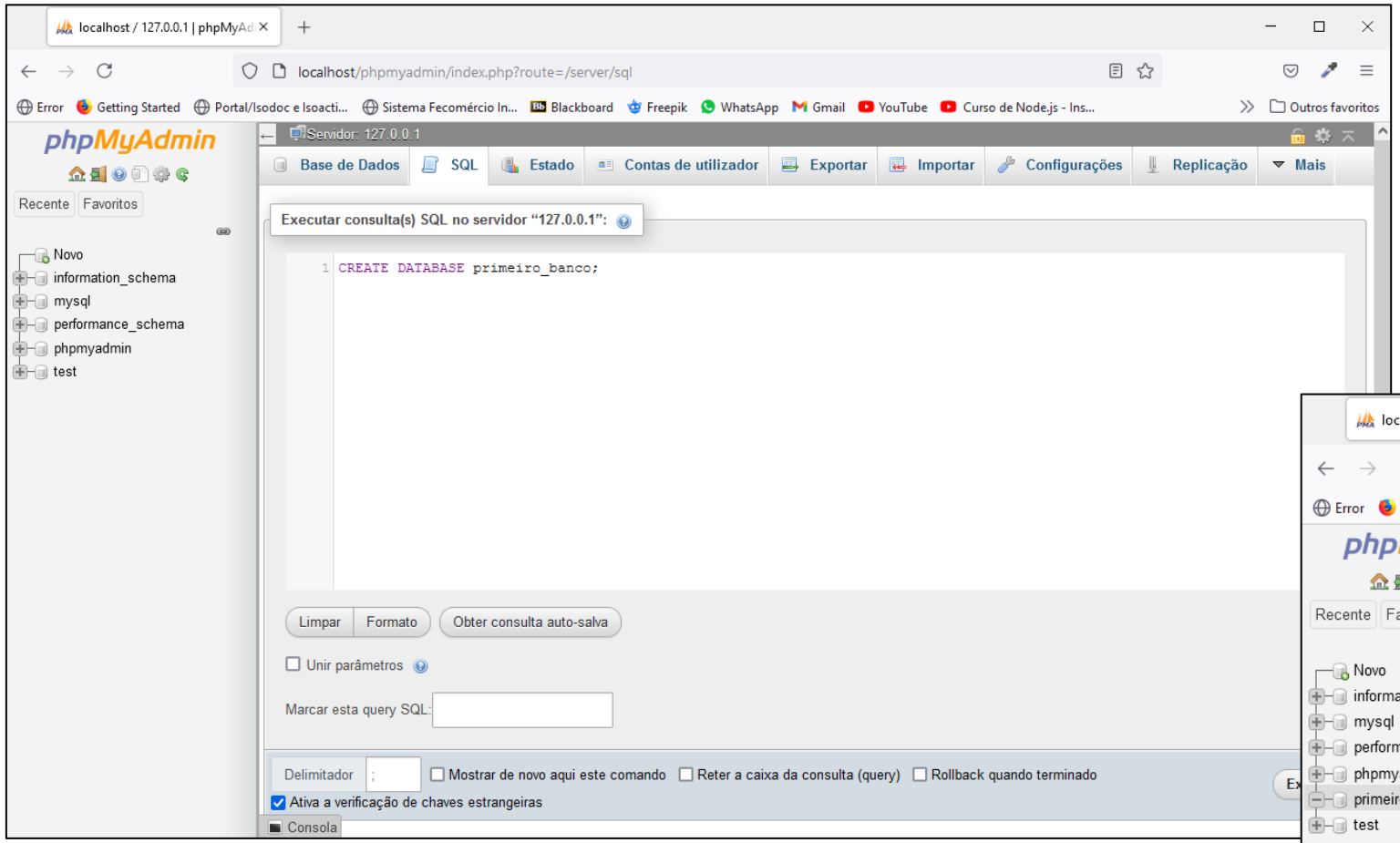
Com XAMPP devidamente instalado siga os passo:



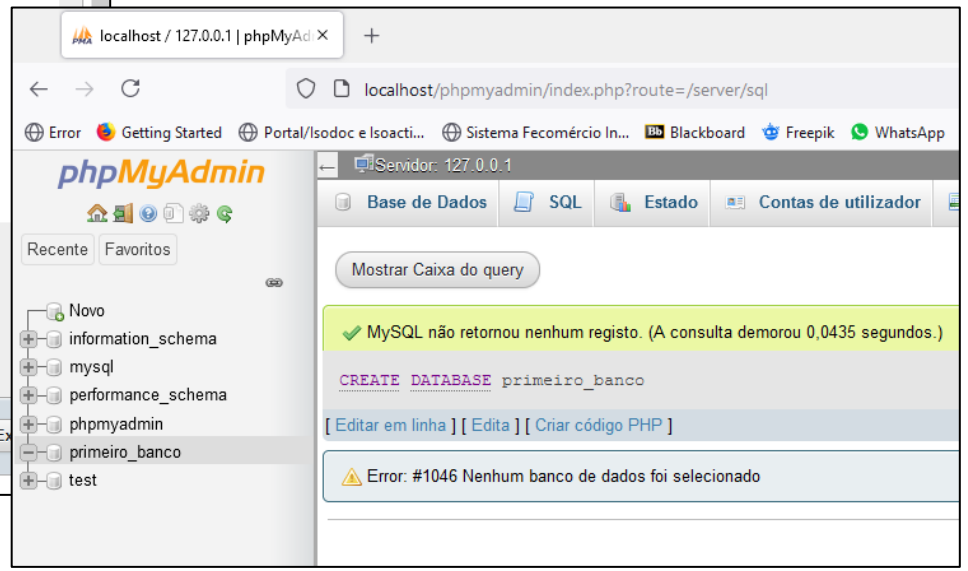
- No painel de controle do XAMPP dê start nos módulos Apache e MySQL.
- No módulo SQL clique em Admin, você será levado a página do PHPMyAdmin, veja a seguir

Criando um banco de Dados

Com a janela do browser aberta vá até a aba SQL



- Clique na aba SQL
- Digite: CREATE DATABASE primeiro_banco;
- clique em executar

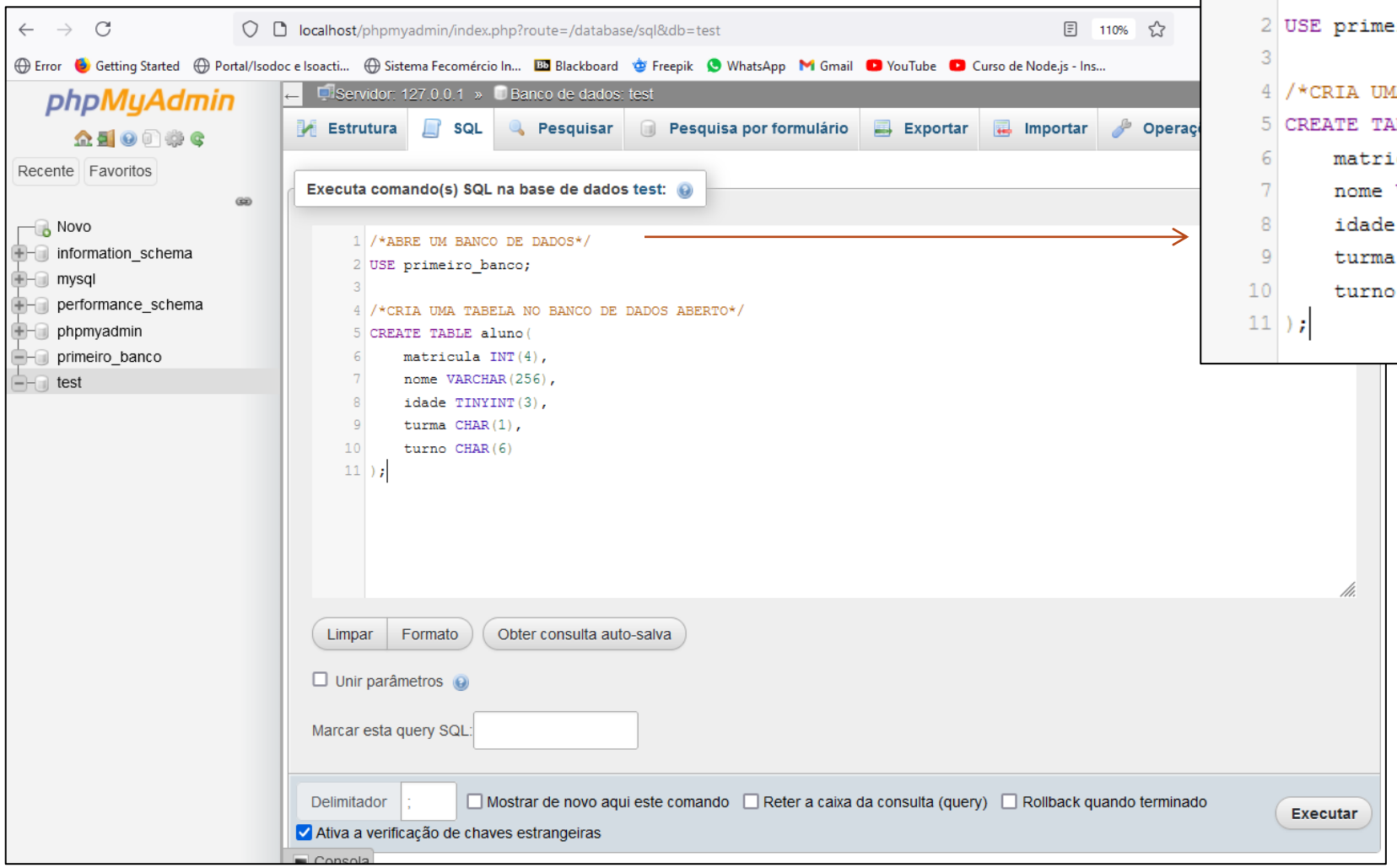


Tipos primitivos de um banco de dados

Numérico	Inteiro	TinyInt, SmallInt, Int, MediumInt, BigInt
	Real	Decimal, Float, Double, Real
	Lógico	Bit, Boolean
Data/Tempo		Date, DateTime, TimeStamp, Time, Year
Literal	Caractere	Char, VarChar
	Texto	TinyText, Text, MediumText, LongText
	Binário	TinyBlob, Blob, MediumBlob, LongBlob
	Coleção	Enum, Set
Espacial		Geometry, Point, Polygon, MultiPolygon

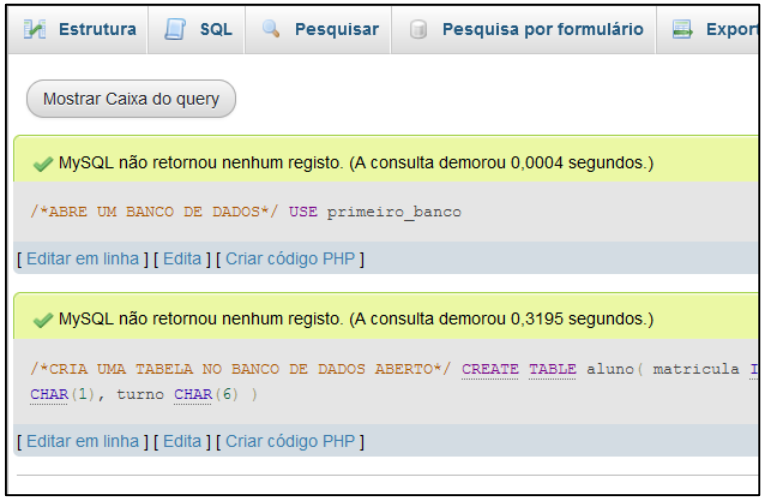
Criando uma tabela no banco de dados

Com a janela do browser aberta vá até a aba SQL



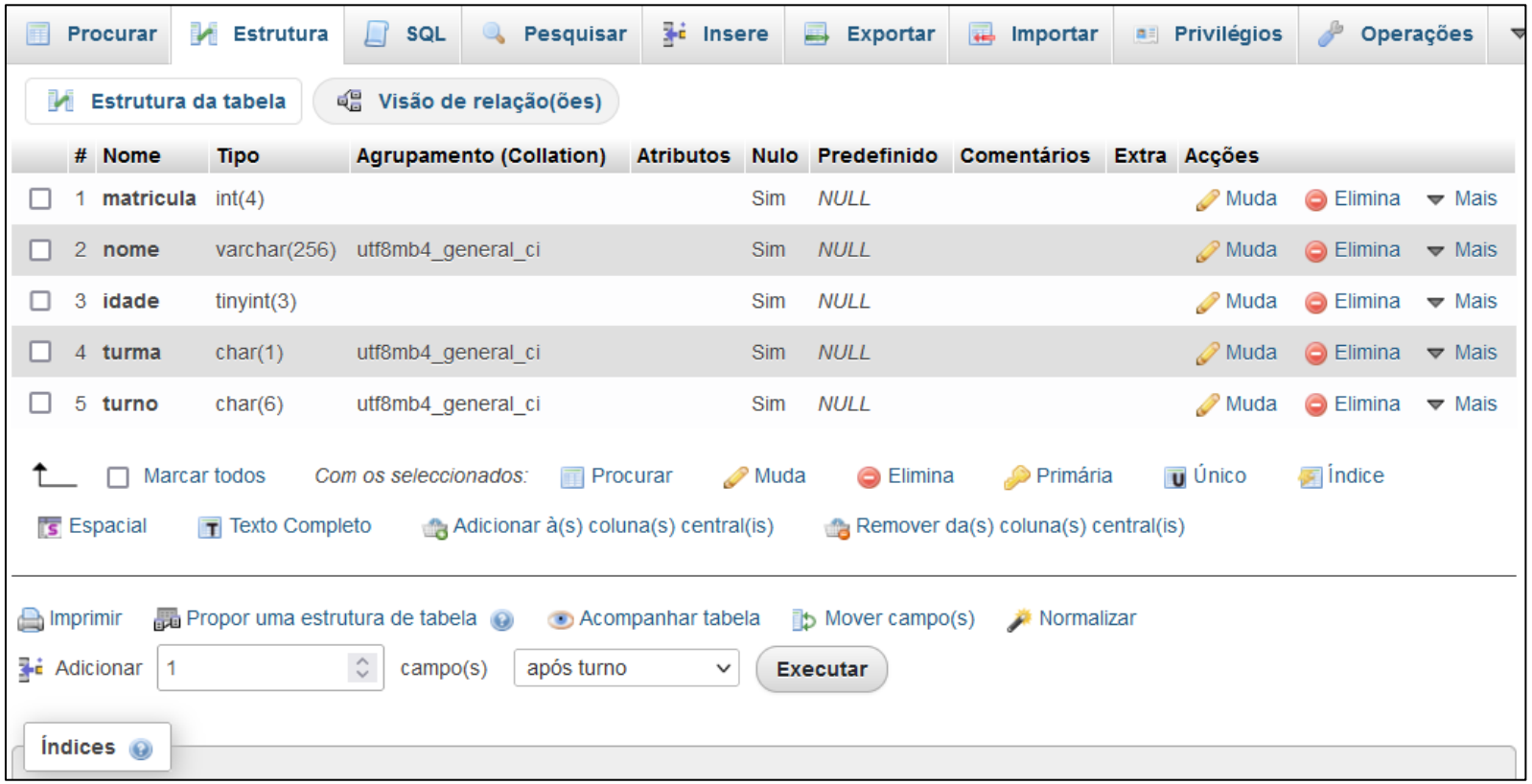

```
/*ABRE UM BANCO DE DADOS*/
USE primeiro_banco;

/*CRIA UMA TABELA NO BANCO DE DADOS ABERTO*/
CREATE TABLE aluno(
    matricula INT(4),
    nome VARCHAR(256),
    idade TINYINT(3),
    turma CHAR(1),
    turno CHAR(6)
);
```



Criando uma tabela no banco de dados

No PHPMyAdmin, selecione a tabela criada e clique em estrutura.



Criando uma tabela no banco de dados

Também podemos ver a estrutura da tabela utilizando o comando DESCRIBE seguido do nome da tabela

Mostrar Caixa do query

A seleção atual não contém uma coluna exclusiva. Os recursos de edição de grade, caixa de seleção, Editar, Copiar e Apagar não estão disponíveis.

A sua consulta SQL foi executada com êxito.

```
DESCRIBE aluno
```

[Editar em linha] [Edita] [Criar código PHP]

+ Opções

Field	Type	Null	Key	Default	Extra
matricula	int(4)	YES		*NULL*	
nome	varchar(256)	YES		*NULL*	
idade	tinyint(3)	YES		*NULL*	
turma	char(1)	YES		*NULL*	
turno	char(6)	YES		*NULL*	

Operações resultantes das consultas

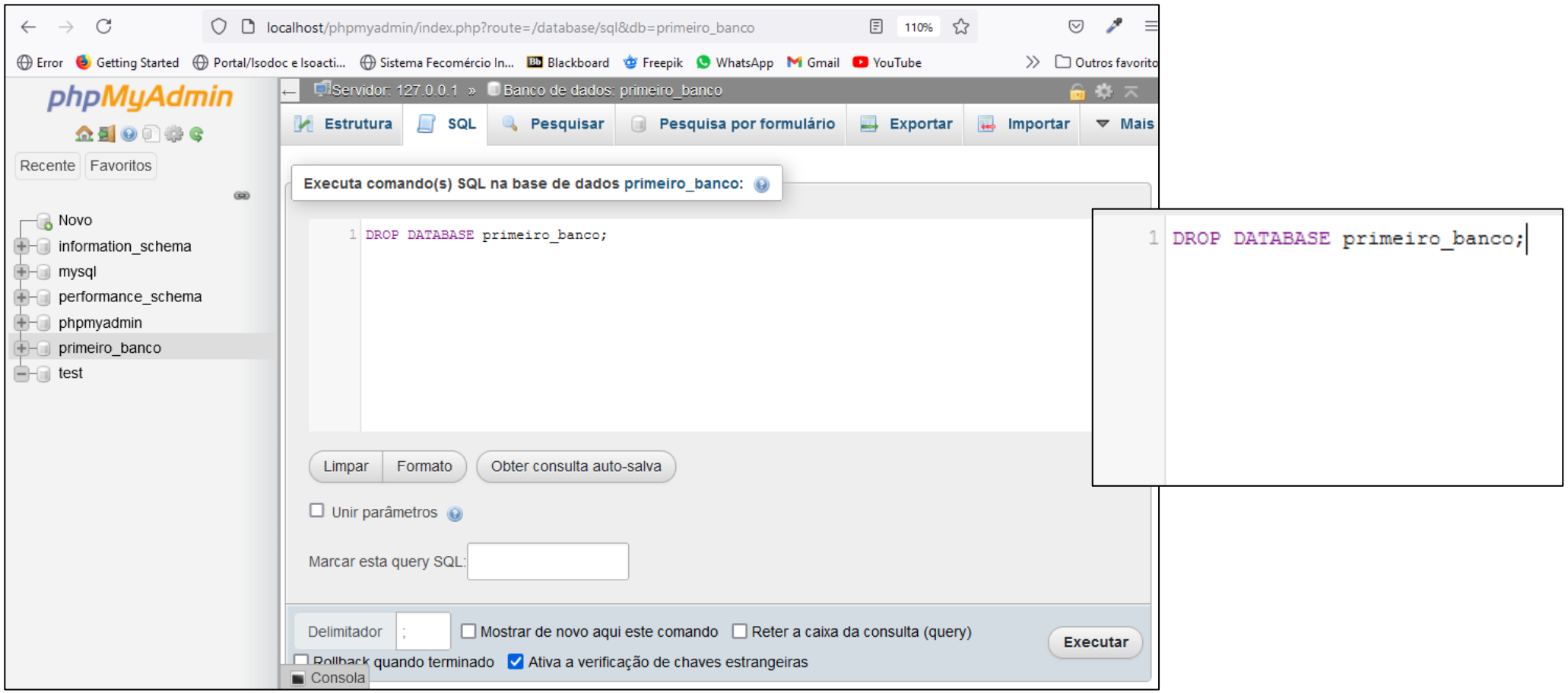
Imprimir Copiar para área de transferência Criar visualização

```
1 DESCRIBE aluno;
```

Apagando um banco de Dados

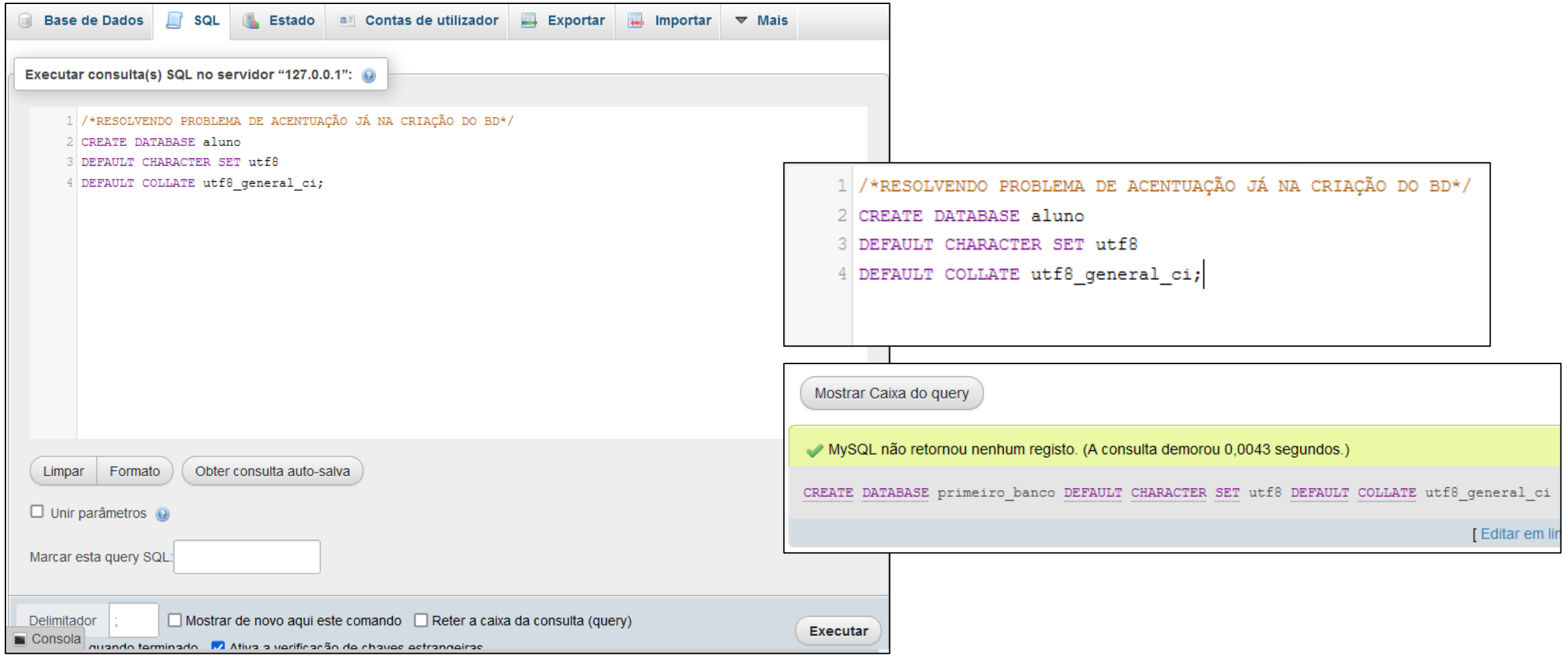
Para apagar um banco de dados basta utilizar o comando Drop Database.

Tenha **cuidado** antes de eliminar um banco de dados. A exclusão de um banco de dados resultará na perda de informações completas armazenadas no banco de dados!



Utilizando Constraints para resolver problemas de acentuação

Quando criamos nosso banco de dados é possível logo de início configurar a codificação do país para termos nossos caracteres apresentados da forma correta. O ponto e vírgula na última linha indica o fim do comando.



Melhorando a criação de tabelas

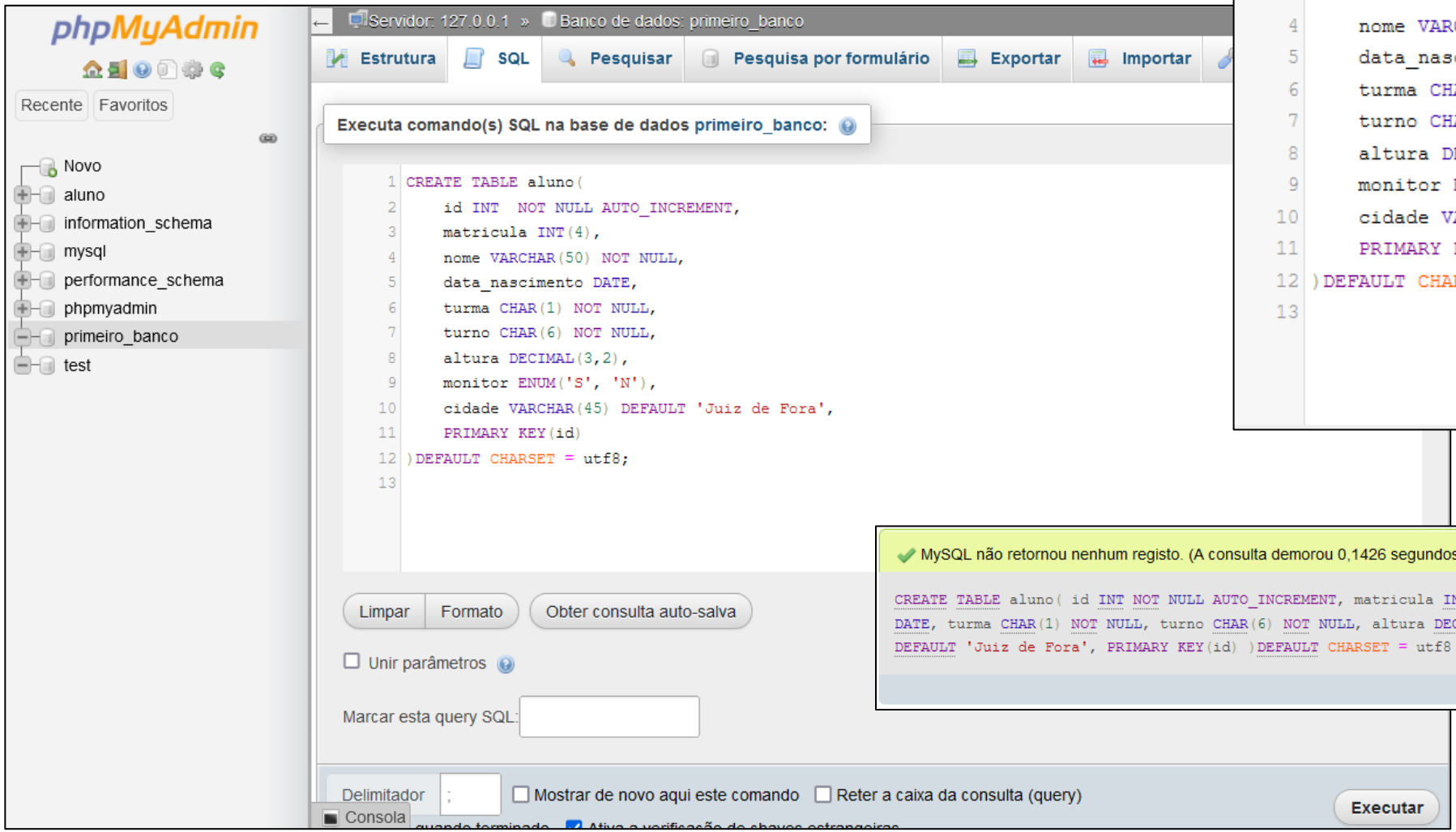
Vamos melhorar a nossa estrutura de tabela incluindo tipos primitivos mais consistentes como date, enum e decimal.

Chaves primárias (em inglês, Primary keys ou "PK"): sob o ponto de vista de um banco de dados relacional, referem-se aos conjuntos de um ou mais campos, cujos valores, considerando a combinação de valores em caso de mais de uma chave primária, nunca se repetem na mesma tabela e, desta forma, podem ser usadas como um índice de referência para criar relacionamentos com as demais tabela do banco de dados (daí vem o nome banco de dados relacional). Portanto, uma chave primária nunca pode ter valor nulo, nem repetição.

Simplificando, quando a chave primária é simples, ou seja, é formada por um único campo da tabela, esse campo não pode ter dois ou mais registros de mesmo valor e também não pode conter nenhum registro nulo. Se a chave primária é composta, ou seja, formada por mais de um campo, os valores de cada campo podem se repetir, mas nunca a combinação desses valores. Exemplo: a tabela 'Livros_Autores' tem como chave primária (cod_livro, cod_autor). Podem existir nessa tabela os registros:

Tipos melhorados e chave primária

```sql
CREATE TABLE aluno(
    id INT  NOT NULL AUTO_INCREMENT,
    matricula INT(4),
    nome VARCHAR(50) NOT NULL,
    data_nascimento DATE,
    turma CHAR(1) NOT NULL,
    turno CHAR(6) NOT NULL,
    altura DECIMAL(3,2),
    monitor ENUM('S', 'N'),
    cidade VARCHAR(45) DEFAULT 'Juiz de Fora',
    PRIMARY KEY(id)
)DEFAULT CHARSET = utf8;
```



Usando Describe para ver a estrutura

Estrutura | SQL | Pesquisar | Pesquisa por formulário | Exportar | Importar | Operações | Mais

[Editar em linha] [Edita] [Criar código PHP]

+ Opções

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
matricula	int(4)	YES		*NULL*	
nome	varchar(50)	NO		*NULL*	
data_nascimento	date	YES		*NULL*	
turma	char(1)	NO		*NULL*	
turno	char(6)	NO		*NULL*	
altura	decimal(3,2)	YES		*NULL*	
monitor	enum('S','N')	YES		*NULL*	
cidade	varchar(45)	YES		Juiz de Fora	

Operações resultantes das consultas

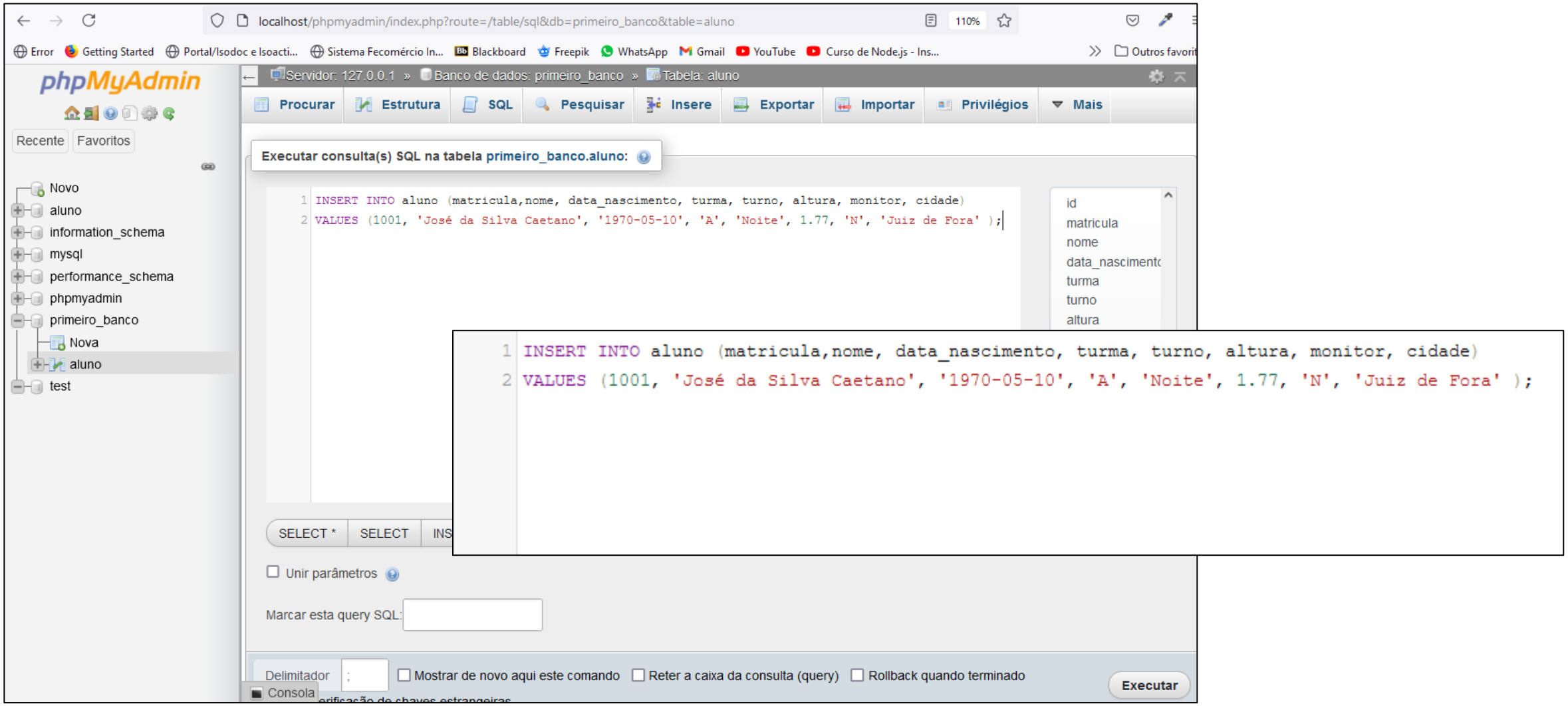
Imprimir | Copiar para área de transferência | Criar visualização

Marcar este comando SQL

Rótulo: ☐ Deixar todos os utilizadores acederem a este marcador

Marcar este comando SQL

Inserindo dados na tabela do banco de dados, siga os passos:



1 INSERT INTO aluno (matricula,nome, data_nascimento, turma, turno, altura, monitor, cidade)
2 VALUES (1001, 'José da Silva Caetano', '1970-05-10', 'A', 'Noite', 1.77, 'N', 'Juiz de Fora');

Visualizando os dados: SELECT * FROM tabela;

Mostrar Caixa do query

A mostrar registos de 0 - 0 (1 total, A consulta demorou 0,0011 segundos.)

```
SELECT * FROM aluno
```

Perfil [Editar em linha] [Edita] [Explicar SQL] [Criar código PHP] [Actualizar]

Mostrar tudo | Número de registos: 25 Filtrar registos: Pesquisar esta tabela

+ Opções

	id	matricula	nome	data_nascimento	turma	turno	altura	monitor	cidade
Edita Copiar Apagar	1	1001	José da Silva Caetano	1970-05-10	A	Noite	1.77	N	Juiz de Fora

Marcar todos *Com os seleccionados:* Edita Copiar Apagar Exportar

Mostrar tudo | Número de registos: 25 Filtrar registos: Pesquisar esta tabela

Operações resultantes das consultas

Imprimir Copiar para área de transferência Exportar Mostrar gráfico Criar visualização

Atividade 2 Banco de Dados

Crie uma banco de dados para uma escola utilizando todos os dados da tabela abaixo, fiquem a vontade para escolher os nomes:

Num	S	Data Enturmação	Código	Nome	Sexo	Cond. Matrícula	Data Nascimento	Etnia	Bolsa Família	Transporte Escolar	Tipo de Transporte Escolar
01		28/04/2021	140013		M	NOVATO	05/03/2009	Branca	Não	Não	-
02		05/05/2021	210014		M	NOVATO	25/09/2009	Parda	Não	Não	-
03		28/04/2021	140005		M	NOVATO	26/08/2009	Branca	Não	Não	-
04		28/04/2021	120006		M	NOVATO	06/06/2007	Branca	Não	Não	-
05		28/04/2021	140014		M	NOVATO	29/06/2009	Não declarada	Sim	Sim (Rural)	Ônibus
06		28/04/2021	140011		M	NOVATO	17/07/2009	Parda	Não	Não	-
07		28/04/2021	140008		M	NOVATO	20/08/2009	Branca	Não	Não	-
08		05/05/2021	210016		M	NOVATO	22/06/2010	Não declarada	Não	Não	-
09		28/04/2021	140019		M	NOVATO	07/02/2010	Não declarada	Não	Não	-
10		05/05/2021	210018		M	NOVATO	05/10/2009	Não declarada	Sim	Sim (Rural)	Vans/Kombis
11		28/04/2021	140012		F	NOVATO	07/08/2009	Não declarada	Não	Não	-
12		04/05/2021	210013		F	NOVATO	02/10/2009	Não declarada	Não	Sim (Rural)	Micro-ônibus
13		28/04/2021	140007		F	NOVATO	28/05/2010	Parda	Não	Não	-
14		10/05/2021	210020		F	NOVATO	30/10/2009	Parda	Sim	Sim (Rural)	Vans/Kombis
15		05/05/2021	210017		F	NOVATO	01/12/2009	Branca	Sim	Não	-
16		28/04/2021	130006		F	NOVATO	03/09/2008	Branca	Sim	Não	-
17		28/04/2021	130028		F	NOVATO	09/11/2007	Parda	Não	Não	-

Modificando o nome da Tabela

ALTER TABLE(DDL): permite alterar a estrutura de uma tabela existente. Por exemplo, você pode adicionar ou excluir colunas, criar ou destruir índices, alterar o tipo de colunas existentes, ou renomear colunas ou a mesma tabela.

DROP TABLE (DDL): O comando remove tabelas do banco de dados.

Novas Constraints

UNIQUE: Podemos usar as restrições UNIQUE para garantir que não há valores duplicados inseridos em colunas específicas que não participam de uma chave primária. Embora a restrição UNIQUE e a restrição PRIMARY KEY impõem exclusividade, use a restrição UNIQUE em vez da restrição PRIMARY KEY quando for impor a exclusividade de uma coluna, ou uma combinação de colunas, que não seja uma chave primária.

Alterando a estrutura de um banco de dados.

Para alterar a estrutura de um BD vamos utilizar dois comando:

ALTER TABLE(DDL): permite alterar a estrutura de uma tabela existente. Por exemplo, você pode adicionar ou excluir colunas, criar ou destruir índices, alterar o tipo de colunas existentes, ou renomear colunas ou a mesma tabela.

DROP TABLE (DDL): O comando remove tabelas do banco de dados.

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
altura	decimal(3,2)	YES		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora

Adicionar o campo CEP a nossa estrutura

```
1 /*Adicionando um campo*/
2 ALTER TABLE aluno ADD COLUMN CEP VARCHAR(20);
```

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
altura	decimal(3,2)	YES		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

Alterando a estrutura de um banco de dados.

Para alterar a estrutura de um BD vamos utilizar dois comando:

ALTER TABLE(DDL): permite alterar a estrutura de uma tabela existente. Por exemplo, você pode adicionar ou excluir colunas, criar ou destruir índices, alterar o tipo de colunas existentes, ou renomear colunas ou a mesma tabela.

DROP TABLE (DDL): O comando remove tabelas do banco de dados.

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
altura	decimal(3,2)	YES		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

```
1 /*Apagando uma campo*/
2 ALTER TABLE aluno DROP COLUMN altura;
```

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

Alterando a estrutura de um banco de dados.

Parâmetros do ADD TABLE

AFTER

Insere um campo após o campo indicado pelo comando AFTER.

FIRST

Insere um campo antes do primeiro campo

Observações: Não existe BEFORE e LAST. Se não colocar nada o campo será criado na última posição

Prática:

- Insira um campo teste antes do campo nome
- Visualize a estrutura
- Apague o campo teste

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

```
1 /*Adicionando um campo após o campo nome*/
2 ALTER TABLE aluno ADD COLUMN especial enum('S', 'N')AFTER nome;
```

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

Alterando a estrutura de um banco de dados.

Parâmetros do ADD TABLE

AFTER

Insere um campo após o campo indicado pelo comando AFTER.

FIRST

Insere um campo antes do primeiro campo

Observações: Não existe BEFORE e LAST. Se não colocar nada o campo será criado na última posição

Prática:

- Insira um campo teste antes do campo matrícula
- Visualize a estrutura
- Apague o campo teste

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

```
1 /*Adicionando um campo na primeira posição*/
2 ALTER TABLE aluno
3 ADD COLUMN teste VARCHAR(30) FIRST;
```

+ Opções

Field	Type	Null	Key	Default
teste	varchar(30)	YES		*NULL*
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

Alterando a estrutura de um banco de dados.

Resultado final

```
1 DESCRIBE aluno;
```

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

```
1 SELECT * FROM aluno;
```

+ Opções

				id	matricula	nome	especial	data_nascimento	turma	turno	monitor	cidade	CEP
☐	Edita	Copiar	Apagar	1	1001	José da Silva Caetano	*NULL*	1970-05-10	A	Noite	N	Juiz de Fora	*NULL*
☐	Edita	Copiar	Apagar	2	1001	João Pedro Ferreira	*NULL*	2000-11-28	A	Manhã	N	Juiz de Fora	*NULL*
☐	Edita	Copiar	Apagar	3	550	Juca da Silva	*NULL*	2010-11-28	A	Noite	N	Juiz de Fora	*NULL*
☐	Edita	Copiar	Apagar	4	1001	Kleber Pedro Nogueira	*NULL*	2015-11-28	A	Manhã	N	Juiz de Fora	*NULL*
☐	Edita	Copiar	Apagar	5	1001	Maria Beatriz Oliveira	*NULL*	1984-11-28	B	Noite	N	Juiz de Fora	*NULL*
☐	Edita	Copiar	Apagar	6	1001	Ana Beatriz Nogueira	*NULL*	1960-11-28	B	Noite	N	Juiz de Fora	*NULL*

Modificando a estrutura de um banco de dados.

MODIFY COLUMN

Este comando modifica a estrutura do tabela do BD, ou seja, trabalha diretamente com as propriedades de um campo. **Não podemos renomear o campo**.

Exemplo: Alterando o tamanho do campo CEP para 8 caracteres.

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(20)	YES		*NULL*

```
1 /*Aterando a Estrutura da tabela*/
2 ALTER TABLE aluno MODIFY COLUMN CEP VARCHAR(8);
```

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(8)	YES		*NULL*

Modificando a estrutura de um banco de dados.

Modificando o campo CEP para tamanho 10 com um campo default.

Pratica:

Modificando um campo NOT NULL com parâmetro DEFAULT.

- Modifique o tamanho do campo cidade para 50 e valor default para Barbacena;
- Volte com o padrão Juiz de Fora.

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(8)	YES		*NULL*

```
1 /*Modificando Campos e constraints*/
2 ALTER TABLE aluno
3 MODIFY COLUMN CEP VARCHAR(10) NOT NULL DEFAULT '00.000-000';
```

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(10)	NO		00.000-000

Modificando o nome da Coluna/Campo

CHANGE

Altera o nome da coluna da seguinte maneira:

nome velho → nome novo

É preciso manter as Constraints para não perder a composição da sua estrutura.

Exemplo:

Mudar o nome da coluna data_nascimento para nascimento.

+ Opções

Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
data_nascimento	date	YES		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	YES		Juiz de Fora
CEP	varchar(10)	NO		00.000-000

```
1 /*Mudando o nome da coluna data_nascimento*/
2 ALTER TABLE aluno
3 CHANGE COLUMN data_nascimento nascimento DATE NOT NULL;
```

+ Opções

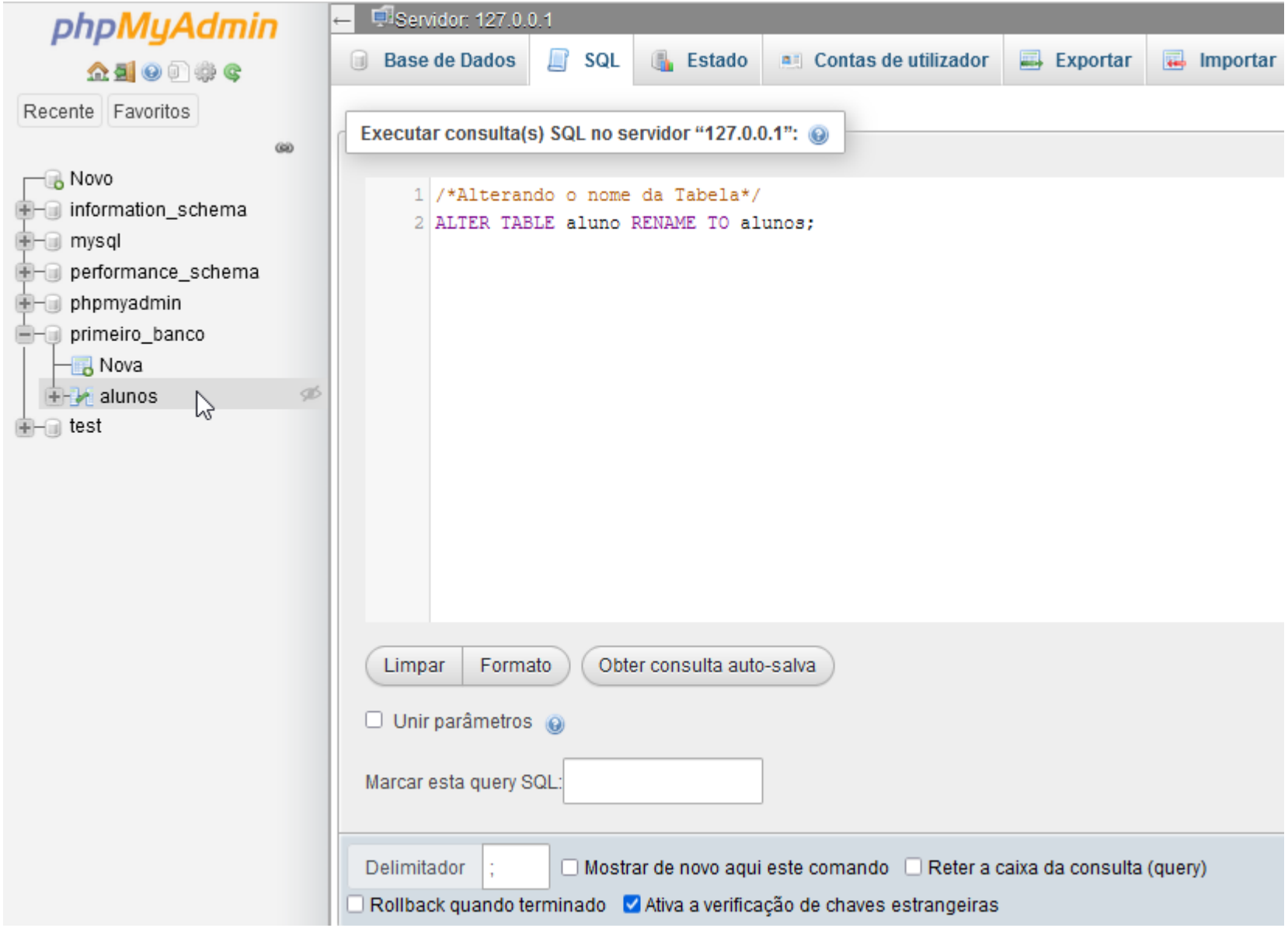
Field	Type	Null	Key	Default
id	int(11)	NO	PRI	*NULL*
matricula	int(4)	YES		*NULL*
nome	varchar(50)	NO		*NULL*
especial	enum('S','N')	YES		*NULL*
nascimento	date	NO		*NULL*
turma	char(1)	NO		*NULL*
turno	char(6)	NO		*NULL*
monitor	enum('S','N')	YES		*NULL*
cidade	varchar(45)	NO		Juiz de Fora
CEP	varchar(10)	NO		00.000-000

Modificando o nome da Tabela

RENAME TO

Altera o nome da Tabela.

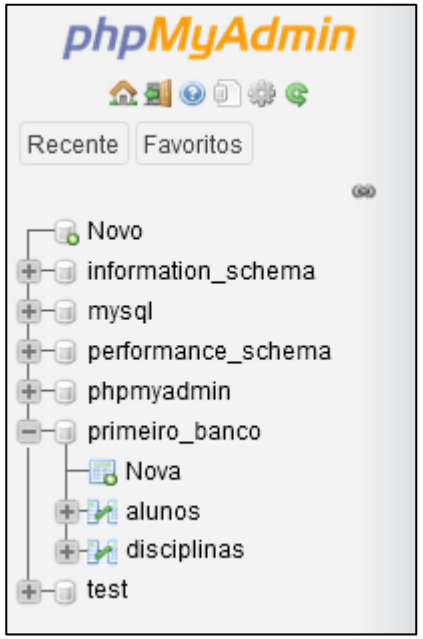
```
1 /*Alterando o nome da Tabela*/
2 ALTER TABLE aluno RENAME TO alunos;
```



Modificando...

Criando uma nova tabela (disciplinas).

```sql
/*Constrains novas
IF NOT EXISTS: Só cria se a tabela não existir
UNIQUE: Não é chave primária, mas não deixa duplicar o dado
UNSIGNED: Sem sinal
*/
CREATE TABLE IF NOT EXISTS disciplinas (
    nome VARCHAR(25) NOT NULL UNIQUE,
    professor VARCHAR(25) NOT NULL,
    turno VARCHAR(6) NOT NULL DEFAULT 'manhã'
) DEFAULT charset=utf8;
```



+ Opções

Field	Type	Null	Key	Default
nome	varchar(25)	NO	PRI	*NULL*
professor	varchar(25)	NO		*NULL*
turno	varchar(6)	NO		manhã

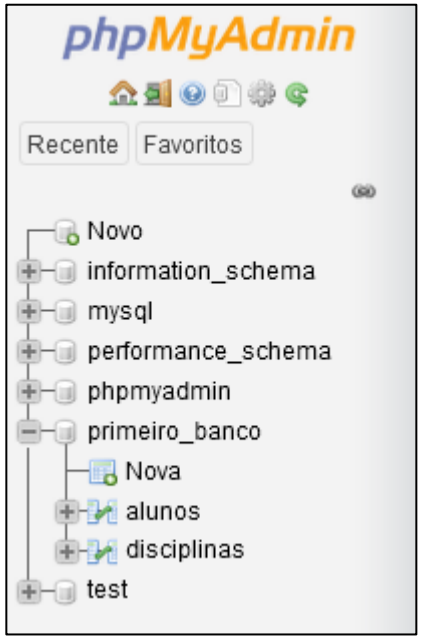
Novas Constraints

Criando um campo para a primária na tabela disciplina

```sql
/*Criando um campo para a chave primária*/
ALTER TABLE disciplinas ADD COLUMN id_disciplina int FIRST;
```

Configurando uma chave primária para a nova tabela (disciplinas).

```sql
/*Configurando a chave primária*/
ALTER TABLE disciplinas ADD PRIMARY KEY (id_disciplina);
```



+ Opções

Field	Type	Null	Key	Default
id_disciplina	int(11)	NO	PRI	*NULL*
nome	varchar(25)	NO	UNI	*NULL*
professor	varchar(25)	NO		*NULL*
turno	varchar(6)	NO		manhã

Apagando uma tabela

Usar a tabela disciplina com o nome disciplina_drop

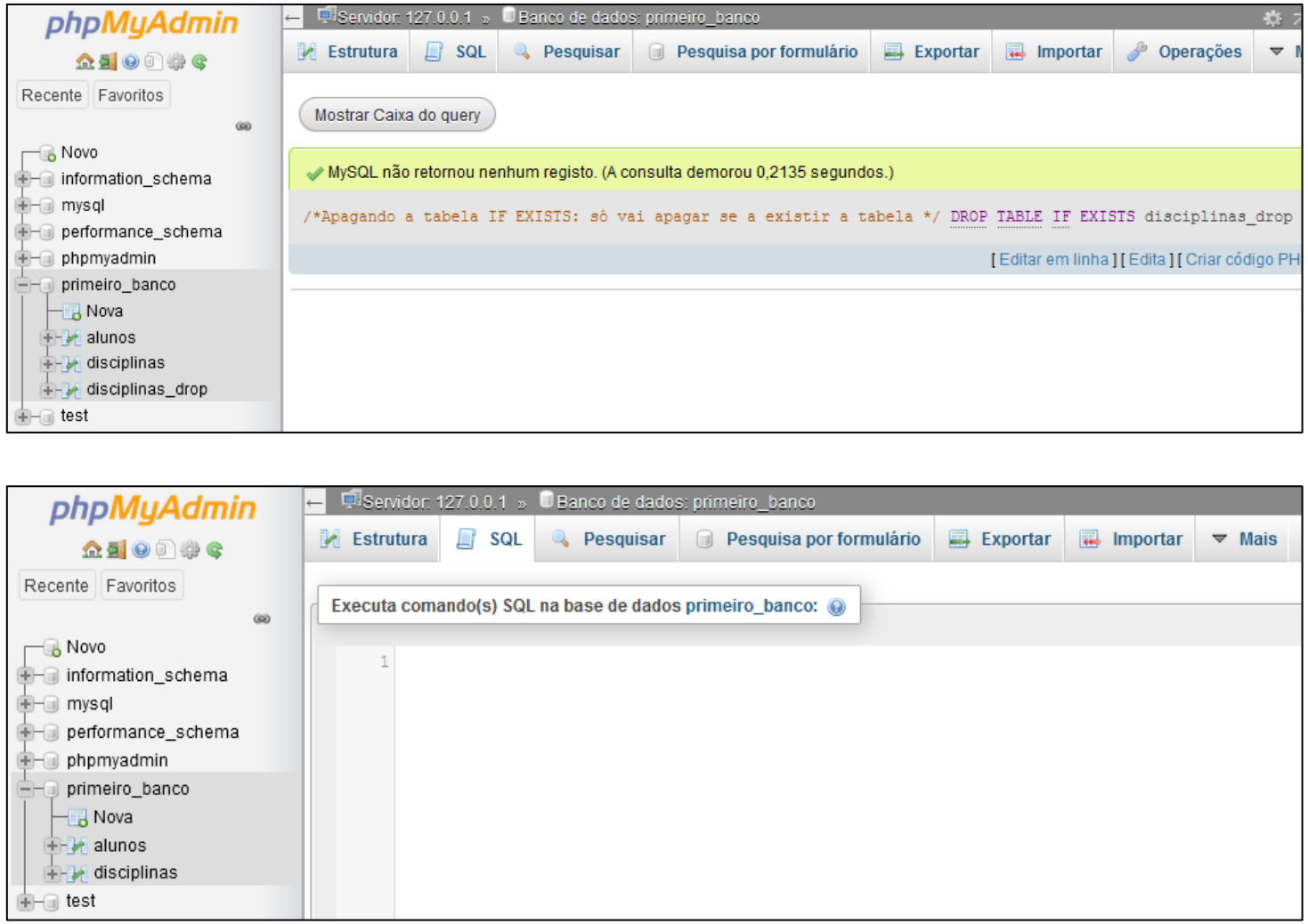
DROP TABLE

```sql
/*Tabela exemplo para utilização do comendo DROP*/
CREATE TABLE IF NOT EXISTS disciplinas_drop (
    nome VARCHAR(25) NOT NULL UNIQUE,
    professor VARCHAR(25) NOT NULL,
    turno VARCHAR(6) NOT NULL DEFAULT 'manhã'
) DEFAULT charset=utf8;
```

```sql
/*Apagando a tabela
IF EXISTS: só vai apagar se a existir a tabela
*/
DROP TABLE IF EXISTS disciplinas_drop;
```

Observação:

DROP não tem como voltar



Manipulando Tuplas (Registros)

Uso dos comando DELETE, UPDATE e TRUNCATE

Vamos apagar a tabela disciplinas e recriá-la da seguinte forma

```sql
/*
Passo 1: Apagar a tabela disciplina com Drop
Passo 2: Criar a tabela disciplinas:
*/

CREATE TABLE IF NOT EXISTS disciplinas (
    id_disciplina INT NOT NULL AUTO_INCREMENT,
    nome VARCHAR(25) NOT NULL UNIQUE,
    professor VARCHAR(25) NOT NULL,
    turno VARCHAR(6) NOT NULL DEFAULT 'manhã',
    PRIMARY KEY (id_disciplina)
) DEFAULT charset=utf8;
```

Manipulando Tuplas (Registros)

Uso dos comando DELETE, UPDATE e TRUNCATE

Inserindo dados na nova tabela disciplinas

```sql
/*
Passo 3: Inserção de Dados
*/

INSERT INTO disciplinas VALUES
(DEFAULT,'Matemática I-II-II', 'Cláudio Rodrigues', 'Noite' ),
(DEFAULT, 'Análise de Sistemas', 'Carla Maria', 'Tarde' ),
(DEFAULT, 'Engenharia de Software', 'João Pedro', 'Manhã' ),
(DEFAULT, 'Banco de Dados', 'Cláudio Rodrigues', 'Noite' ),
(DEFAULT, 'Linguagem de Programação I-II', 'Érika Torres', 'Noite' ),
(DEFAULT, 'Pré-Cálculo', 'Fernanda Souza', 'Tarde' ),
(DEFAULT, 'Cálculo I-II', 'Louis Scyfer da Silva', 'Noite' ),
(DEFAULT, 'Física', 'Louis Scyfer da Silva', 'Noite' );
```

Manipulando Tuplas (Registros): Uso dos comando DELETE, UPDATE e TRUNCATE

+ Opções

Field	Type	Null	Key	Default	Extra
id_disciplina	int(11)	NO	PRI	*NULL*	auto_increment
nome	varchar(25)	NO	UNI	*NULL*	
professor	varchar(25)	NO		*NULL*	
turno	varchar(6)	NO		manhã	

```
1 DESCRIBE disciplinas;
```

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas	Carla Maria	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I-II	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

```
1 SELECT * FROM disciplinas;
```

Alterando uma instância/linha/registro do Banco de Dados:

UPDATE

A instrução é usada para modificar os registros existentes em uma tabela.

Sintaxe:

UPDATE nome_tabela
SET Coluna1 = Valor1, Valor2 = valor3, ...
WHERE Condição;

Tenha cuidado ao atualizar os registros em uma tabela! Observe a cláusula **WHERE** na declaração **UPDATE**. A cláusula WHERE especifica quais registros devem ser atualizados. Se você omitir a cláusula **WHERE**, todos os registros na tabela serão atualizados!

Alterando uma instância/linha/registro do Banco de Dados:

Mudar na **linha 7** o nome da disciplina para Cálculo I, II e III.

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas	Carla Maria	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I-II	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

```
1 UPDATE disciplinas SET nome = 'Cálculo I-II-III'
2 WHERE id_disciplina = '7';
```

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas	Carla Maria	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I-II-III	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

Alterando uma instância/linha/registro do Banco de Dados:

Na **linha 2**, mudar o nome do professor(a) para **Maria Fernanda** e nome da disciplina para **Análise de Sistemas I**.

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas	Carla Maria	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I-II-III	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

```
1 UPDATE disciplinas
2 SET nome = 'Análise de Sistemas I', professor = 'Maria Fernanda'
3 WHERE id_disciplina = '2';
```

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

Alterando uma instância/linha/registro do Banco de Dados:

Na **linha 4**, mudar o **turno** para **Tarde** limitando a alteração somente nessa linha.

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

```
1 UPDATE disciplinas
2 /*Limite não deixa que outras linhas sejam afetadas*/
3 SET turno = 'Tarde' WHERE id_disciplina = '4' LIMIT 1;
```

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Tarde
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

Observação:

UPDATE disciplinas

SET turno = 'Tarde'

WHERE turno = 'Manhã';

Todos os turno inseridos como **Manhã** se tornarão **Tarde**. Se usar o **LIMIT**, somente a primeira ocorrência receberá o update.

O Workbench tem um safe update!

Apagando linhas do Banco de Dados:

Uso dos comando DELETE, UPDATE e TRUNCATE

DELETE

Esta instrução é usada para excluir registros existentes em uma tabela.

Sintaxe: **DELETE FROM** nome_tabela **WHERE** Condição;

Tenha cuidado ao excluir registros em uma tabela! Observe a cláusula **WHERE** na declaração **DELETE**. A cláusula **WHERE** especifica quais registros devem ser excluídos. **Se você omitir a cláusula WHERE, todos os registros da tabela serão excluídos**!

Prática...

Crie mais 4 registros na tabela disciplinas

```
1 INSERT INTO disciplinas VALUES
2 (DEFAULT,'História', 'Helena de Souza', 'Manhã' ),
3 (DEFAULT, 'Geografia', 'Fabiana Rodrigues', 'Tarde' ),
4 (DEFAULT, 'Inglês', 'Humberto Chaves', 'Manhã' ),
5 (DEFAULT, 'Português', 'Danilo de Assis', 'Noite' );
```

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Tarde
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	9	História	Helena de Souza	Manhã
☐	Edita	Copiar	Apagar	10	Geografia	Fabiana Rodrigues	Tarde
☐	Edita	Copiar	Apagar	11	Inglês	Humberto Chaves	Manhã
☐	Edita	Copiar	Apagar	12	Português	Danilo de Assis	Noite

Apagando linhas do Banco de Dados:

```
1 /*Apagando um registro
2 da tabela disciplinas
3 */
4 DELETE FROM disciplinas
5 WHERE id_disciplina = '12';
```

Observação:

DELETE FROM disciplinas

WHERE turno = 'Tarde' ;

Todos os turnos inseridos como **Tarde** serão apagados. Se usar o **LIMIT**, somente a primeira ocorrência receberá a ação do comando.

O Workbench tem um safe update que funciona para a Delete múltiplo!

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Tarde
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	9	História	Helena de Souza	Manhã
☐	Edita	Copiar	Apagar	10	Geografia	Fabiana Rodrigues	Tarde
☐	Edita	Copiar	Apagar	11	Inglês	Humberto Chaves	Manhã
☐	Edita	Copiar	Apagar	12	Português	Danilo de Assis	Noite

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Tarde
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	9	História	Helena de Souza	Manhã
☐	Edita	Copiar	Apagar	10	Geografia	Fabiana Rodrigues	Tarde
☐	Edita	Copiar	Apagar	11	Inglês	Humberto Chaves	Manhã

Apagando linhas do Banco de Dados:

Praticando:

- Remova os registro 9, 10 e 11
- Execute uma linha por vez

```
1  DELETE FROM disciplinas
2  WHERE id_disciplina = '9';
3
4  DELETE FROM disciplinas
5  WHERE nome = 'Geografia';
6
7  DELETE FROM disciplinas
8  WHERE professor = 'Humberto Chaves';
```

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Tarde
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite
☐	Edita	Copiar	Apagar	9	História	Helena de Souza	Manhã
☐	Edita	Copiar	Apagar	10	Geografia	Fabiana Rodrigues	Tarde
☐	Edita	Copiar	Apagar	11	Inglês	Humberto Chaves	Manhã

+ Opções

				id_disciplina	nome	professor	turno
☐	Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
☐	Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
☐	Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
☐	Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Tarde
☐	Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
☐	Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
☐	Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
☐	Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

Limpando uma tabela do Banco de Dados

TRUNCATE

O comando TRUNCATE TABLE exclui os dados de uma tabela, mas não a própria tabela.

Sintaxe:

TRUNCATE TABLE nome_tabela;

```
1 /*Limpando os registros da
2 tabela disciplinas */
3 TRUNCATE TABLE disciplinas;
```

+ Opções

			id_disciplina	nome	professor	turno
Edita	Copiar	Apagar	1	Matemática I-II-II	Cláudio Rodrigues	Noite
Edita	Copiar	Apagar	2	Análise de Sistemas I	Maria Fernanda	Tarde
Edita	Copiar	Apagar	3	Engenharia de Software	João Pedro	Manhã
Edita	Copiar	Apagar	4	Banco de Dados	Cláudio Rodrigues	Tarde
Edita	Copiar	Apagar	5	Linguagem de Programação	Érika Torres	Noite
Edita	Copiar	Apagar	6	Pré-Cálculo	Fernanda Souza	Tarde
Edita	Copiar	Apagar	7	Cálculo I, II, III	Louis Scyfer	Noite
Edita	Copiar	Apagar	8	Física	Louis Scyfer da Silva	Noite

id_disciplina	nome	professor	turno

Operações resultantes das consultas

Criar visualização

Observação:

Esse comando também não tem volta!

Backup de um Banco de Dados

Passos:

Entrar no PHPMyAdmin

Escolher o banco de dados

Aperte na aba Export (Exportar), que está localizada no topo da página.
Escolha Export Method (Método de Exportação). O phpMyAdmin oferece 2 métodos de exportação:

1) Rápido – Exibe o Mínimo de Opções
2) Personalizado – Exibe Todas as Opções Possíveis.

Para importar basta clicar no botão importar, escolher o arquivo .sql e executar.

Observação: É preciso ter um banco criado para ocorrer a importação das tabelas

Backup de um Banco de Dados

Passos:

Entrar no PHPMyAdmin

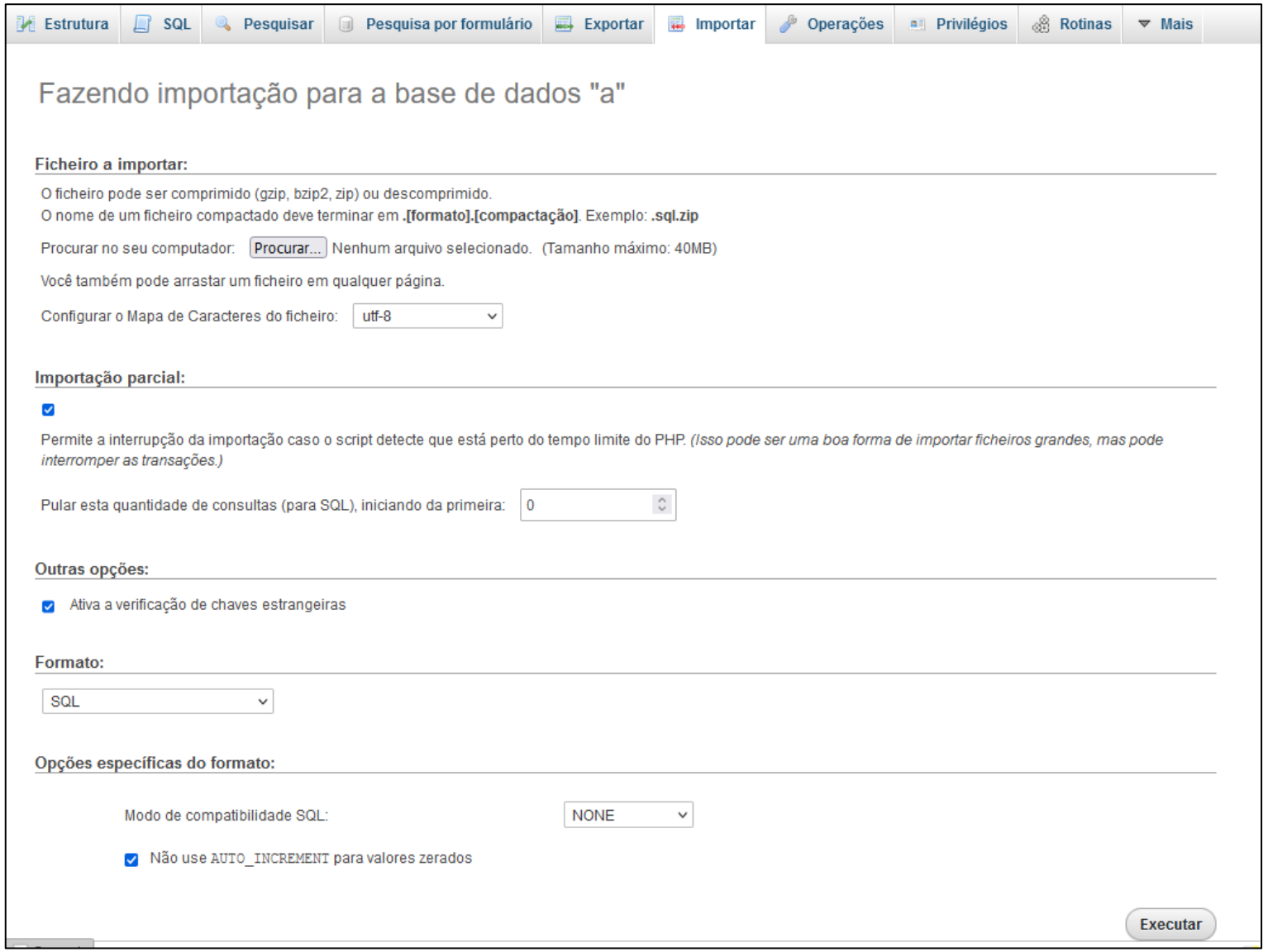
Escolher o banco de dados

Aperte na aba Export (Exportar), que está localizada no topo da página.
Escolha Export Method (Método de Exportação). O phpMyAdmin oferece 2 métodos de exportação:

1) Rápido – Exibe o Mínimo de Opções
2) Personalizado – Exibe Todas as Opções Possíveis.

Para importar basta clicar no botão importar escolha o arquivo .sql e execute.

Observação: É preciso ter um banco criado para ocorrer a importação das tabelas



Fundamentos de
Banco de Dados
e linguagem SQL
SELECT

SELECT

A instrução **SQL SELECT** retorna um conjunto de registros de resultados de uma ou mais tabelas.
Uma instrução **SELECT** recupera zero ou mais linhas de uma ou mais tabelas de banco de dados ou visualizações de banco de dados . Na maioria dos aplicativos, **SELECT** é o comando de linguagem de manipulação de dados (**DML**) mais comumente usado . Como SQL é uma linguagem de programação declarativa , as consultas **SELECT** especificam um conjunto de resultados, mas não especificam como calculá-lo. O banco de dados traduz a consulta em um " plano de consulta" que pode variar entre execuções, versões de banco de dados e software de banco de dados. Essa funcionalidade é chamada de " otimizador de consulta ", pois é responsável por encontrar o melhor plano de execução possível para a consulta, dentro das restrições aplicáveis.

A instrução SELECT tem muitas cláusulas opcionais:

SELECT: é a lista de colunas ou expressões SQL que devem ser retornadas pela consulta. Esta é aproximadamente a operação de projeção da álgebra relacional.
FROM: especifica de qual tabela obter os dados.
WHERE: especifica quais linhas recuperar. Esta é aproximadamente a operação de seleção de álgebra relacional .
ORDER BY: especifica como ordenar as linhas retornadas.

Banco de Dados e Tabela para utilização do SELECT

```sql
CREATE DATABASE escola
DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci;
```

```sql
CREATE TABLE IF NOT EXISTS alunos(
    num INT NOT NULL AUTO_INCREMENT,
    data_enumeracao DATE NOT NULL,
    codigo INT(6) NOT NULL,
    nome CHAR(45) NOT NULL,
    sexo ENUM('M','F') NOT NULL,
    cond_matricula CHAR(15) NOT NULL,
    data_nascimento DATE NOT NULL,
    Etnia CHAR(15) NOT NULL DEFAULT 'Não declarada',
    bolsa_familia ENUM('Sim','Não') NOT NULL,
    transporte_escolar ENUM('Sim(Rural)','Não') NOT NULL,
    tipo_trans_escolar CHAR(15) NOT NULL DEFAULT '-',
    PRIMARY KEY(num)
)DEFAULT CHARSET = utf8;
```

Tabela para utilização do Select

```
INSERT INTO alunos VALUES
    (DEFAULT,'2021-04-28',140013,'Alex André','M','NOVATO','2009-03-05','Branca','Não','Não',DEFAULT),
    (DEFAULT,'2021-05-05',210014,'Felipe Marcos','M','NOVATO','2009-09-25','Parda','Não','Não',DEFAULT),
    (DEFAULT,'2021-04-28',140005,'Bryan Cranston','M','NOVATO','2009-08-26','Branca','Não','Não',DEFAULT),
    (DEFAULT,'2021-04-28',120006,'Aaron Paul','M','NOVATO','2007-06-06','Branca','Não','Não',DEFAULT),
    (DEFAULT,'2021-04-28',140014,'Bob Odenkirk','M','NOVATO','2009-06-29','Não Declarada','Não','Sim(Rural)','Ônibus'),
    (DEFAULT,'2021-04-28',140011,'Steven Michael Quezada','M','NOVATO','2009-07-17','Parda','Não','Não',DEFAULT),
    (DEFAULT,'2021-04-28',140008,'Jonathan Banks','M','NOVATO','2009-08-20','Branca','Não','Não',DEFAULT),
    (DEFAULT,'2021-05-05',210016,'Giancarlo Esposito','M','NOVATO','2010-06-22','Não Declarada','Sim','Não',DEFAULT),
    (DEFAULT,'2021-04-28',140019,'Anna Gunn','F','NOVATO','2010-02-07','Não Declarada','Não','Não',DEFAULT),
    (DEFAULT,'2021-05-05',210018,'Betsy Brandt','F','NOVATO','2009-10-05','Não Declarada','Não','Sim(Rural)','Vans/Kombis'),
    (DEFAULT,'2021-04-28',140012,'Laura Fraser','F','NOVATO','2009-08-07','Não Declarada','Não','Não',DEFAULT),
    (DEFAULT,'2021-05-04',210013,'Krysten Ritter','F','NOVATO','2009-10-02','Não Declarada','Sim','Sim(Rural)','Micro-ônibus'),
    (DEFAULT,'2021-04-28',140007,'Carmen Serano','F','NOVATO','2010-05-28','Parda','Sim','Não',DEFAULT),
    (DEFAULT,'2021-05-10',210020,'Emily Rios','F','NOVATO','2009-10-30','Parda','Sim','Sim(Rural)','Vans/Kombis'),
    (DEFAULT,'2021-05-05',210017,'Tina Parker','F','NOVATO','2009-12-01','Branca','Não','Não',DEFAULT),
    (DEFAULT,'2021-04-28',130006,'RJ Mitte','M','NOVATO','2008-09-03','Branca','Não','Sim (Rural)',DEFAULT),
    (DEFAULT,'2021-04-28',130028,'Dean Norris','M','NOVATO','2007-11-09','Parda','Não','Não',DEFAULT);
```

SELECT

Seleção de todos os campos

```sql
/*
* Significa todos
*/
SELECT * FROM alunos;
```

Result Set

num	data_enumeracao	codigo	nome	sexo	cond_matricula	data_nascimento	Etnia	bolsa_familia	transporte_escolar	tipo_trans_escolar
1	2021-04-28	140013	Alex André	M	NOVATO	2009-03-05	Branca	Não	Não	-
2	2021-05-05	210014	Felipe Marcos	M	NOVATO	2009-09-25	Parda	Não	Não	-
3	2021-04-28	140005	Bryan Cranston	M	NOVATO	2009-08-26	Branca	Não	Não	-
4	2021-04-28	120006	Aaron Paul	M	NOVATO	2007-06-06	Branca	Não	Não	-
5	2021-04-28	140014	Bob Odenkirk	M	NOVATO	2009-06-29	Não Declarada	Não	Sim(Rural)	Ônibus
6	2021-04-28	140011	Steven Michael Quezada	M	NOVATO	2009-07-17	Parda	Não	Não	-
7	2021-04-28	140008	Jonathan Banks	M	NOVATO	2009-08-20	Branca	Não	Não	-
8	2021-05-05	210016	Giancarlo Esposito	M	NOVATO	2010-06-22	Não Declarada	Sim	Não	-
9	2021-04-28	140019	Anna Gunn	F	NOVATO	2010-02-07	Não Declarada	Não	Não	-
10	2021-05-05	210018	Betsy Brandt	F	NOVATO	2009-10-05	Não Declarada	Não	Sim(Rural)	Vans/Kombis
11	2021-04-28	140012	Laura Fraser	F	NOVATO	2009-08-07	Não Declarada	Não	Não	-
12	2021-05-04	210013	Krysten Ritter	F	NOVATO	2009-10-02	Não Declarada	Sim	Sim(Rural)	Micro-ônibus
13	2021-04-28	140007	Carmen Serano	F	NOVATO	2010-05-28	Parda	Sim	Não	-
14	2021-05-10	210020	Emily Rios	F	NOVATO	2009-10-30	Parda	Sim	Sim(Rural)	Vans/Kombis
15	2021-05-05	210017	Tina Parker	F	NOVATO	2009-12-01	Branca	Não	Não	-
16	2021-04-28	130006	RJ Mitte	M	NOVATO	2008-09-03	Branca	Não	Não	-
17	2021-04-28	130028	Dean Norris	M	NOVATO	2007-11-09	Parda	Não	Não	-

SELECT

Seleção de todos os campos com ordenação feita pelo nome do aluno

```sql
SELECT * FROM alunos
ORDER BY nome;
```

num	data_enumeracao	codigo	nome ▲ 1	sexo	cond_matricula	data_nascimento	Etnia	bolsa_familia	transporte_escolar	tipo_trans_escolar
4	2021-04-28	120006	Aaron Paul	M	NOVATO	2007-06-06	Branca	Não	Não	-
1	2021-04-28	140013	Alex André	M	NOVATO	2009-03-05	Branca	Não	Não	-
9	2021-04-28	140019	Anna Gunn	F	NOVATO	2010-02-07	Não Declarada	Não	Não	-
10	2021-05-05	210018	Betsy Brandt	F	NOVATO	2009-10-05	Não Declarada	Não	Sim(Rural)	Vans/Kombis
5	2021-04-28	140014	Bob Odenkirk	M	NOVATO	2009-06-29	Não Declarada	Não	Sim(Rural)	Ônibus
3	2021-04-28	140005	Bryan Cranston	M	NOVATO	2009-08-26	Branca	Não	Não	-
13	2021-04-28	140007	Carmen Serano	F	NOVATO	2010-05-28	Parda	Sim	Não	-
17	2021-04-28	130028	Dean Norris	M	NOVATO	2007-11-09	Parda	Não	Não	-
14	2021-05-10	210020	Emily Rios	F	NOVATO	2009-10-30	Parda	Sim	Sim(Rural)	Vans/Kombis
2	2021-05-05	210014	Felipe Marcos	M	NOVATO	2009-09-25	Parda	Não	Não	-
8	2021-05-05	210016	Giancarlo Esposito	M	NOVATO	2010-06-22	Não Declarada	Sim	Não	-
7	2021-04-28	140008	Jonathan Banks	M	NOVATO	2009-08-20	Branca	Não	Não	-
12	2021-05-04	210013	Krysten Ritter	F	NOVATO	2009-10-02	Não Declarada	Sim	Sim(Rural)	Micro-ônibus
11	2021-04-28	140012	Laura Fraser	F	NOVATO	2009-08-07	Não Declarada	Não	Não	-
16	2021-04-28	130006	RJ Mitte	M	NOVATO	2008-09-03	Branca	Não	Não	-
6	2021-04-28	140011	Steven Michael Quezada	M	NOVATO	2009-07-17	Parda	Não	Não	-
15	2021-05-05	210017	Tina Parker	F	NOVATO	2009-12-01	Branca	Não	Não	-

SELECT

Seleção de todos os campos com ordenação feita pelo nome do aluno:

Ascendente (ASC ou nada) e Descendente (DESC)

```sql
SELECT * FROM alunos
ORDER BY nome DESC;
```

num	data_enumeracao	codigo	nome 1	sexo	cond_matricula	data_nascimento	Etnia	bolsa_familia	transporte_escolar	tipo_trans_escolar
15	2021-05-05	210017	Tina Parker	F	NOVATO	2009-12-01	Branca	Não	Não	-
6	2021-04-28	140011	Steven Michael Quezada	M	NOVATO	2009-07-17	Parda	Não	Não	-
16	2021-04-28	130006	RJ Mitte	M	NOVATO	2008-09-03	Branca	Não	Não	-
11	2021-04-28	140012	Laura Fraser	F	NOVATO	2009-08-07	Não Declarada	Não	Não	-
12	2021-05-04	210013	Krysten Ritter	F	NOVATO	2009-10-02	Não Declarada	Sim	Sim(Rural)	Micro-ônibus
7	2021-04-28	140008	Jonathan Banks	M	NOVATO	2009-08-20	Branca	Não	Não	-
8	2021-05-05	210016	Giancarlo Esposito	M	NOVATO	2010-06-22	Não Declarada	Sim	Não	-
2	2021-05-05	210014	Felipe Marcos	M	NOVATO	2009-09-25	Parda	Não	Não	-
14	2021-05-10	210020	Emily Rios	F	NOVATO	2009-10-30	Parda	Sim	Sim(Rural)	Vans/Kombis
17	2021-04-28	130028	Dean Norris	M	NOVATO	2007-11-09	Parda	Não	Não	-
13	2021-04-28	140007	Carmen Serano	F	NOVATO	2010-05-28	Parda	Sim	Não	-
3	2021-04-28	140005	Bryan Cranston	M	NOVATO	2009-08-26	Branca	Não	Não	-
5	2021-04-28	140014	Bob Odenkirk	M	NOVATO	2009-06-29	Não Declarada	Não	Sim(Rural)	Ônibus
10	2021-05-05	210018	Betsy Brandt	F	NOVATO	2009-10-05	Não Declarada	Não	Sim(Rural)	Vans/Kombis
9	2021-04-28	140019	Anna Gunn	F	NOVATO	2010-02-07	Não Declarada	Não	Não	-
1	2021-04-28	140013	Alex André	M	NOVATO	2009-03-05	Branca	Não	Não	-
4	2021-04-28	120006	Aaron Paul	M	NOVATO	2007-06-06	Branca	Não	Não	-

SELECT

Seleção por meio dos campos ou filtragem. A ordem das colunas pode ser alterada e o **ORDER BY pode também ter mais de um campo**.

```sql
/*
Selecionando pelo nome, codigo e data de enumeração em ordem alfabética pelo nome
*/
SELECT nome, codigo, data_enumeracao from alunos
ORDER BY nome;
```

nome ▲ 1	codigo	data_enumeracao
Aaron Paul	120006	2021-04-28
Alex André	140013	2021-04-28
Anna Gunn	140019	2021-04-28
Betsy Brandt	210018	2021-05-05
Bob Odenkirk	140014	2021-04-28
Bryan Cranston	140005	2021-04-28
Carmen Serano	140007	2021-04-28
Dean Norris	130028	2021-04-28
Emily Rios	210020	2021-05-10
Felipe Marcos	210014	2021-05-05
Giancarlo Esposito	210016	2021-05-05
Jonathan Banks	140008	2021-04-28
Krysten Ritter	210013	2021-05-04
Laura Fraser	140012	2021-04-28
RJ Mitte	130006	2021-04-28
Steven Michael Quezada	140011	2021-04-28
Tina Parker	210017	2021-05-05

SELECT

Seleção por meio linhas utilizando a cláusula WHERE, ou seja, Selecione todos os campos da tabela alunos onde a data de enumeração seja igual a 28/04/2021.

```
1 SELECT * FROM alunos
2 WHERE data_enumeracao = '2021-04-28'
3 ORDER BY nome;
```

num	data_enumeracao	codigo	nome ▲ 1	sexo	cond_matricula	data_nascimento	Etnia	bolsa_familia	transporte_escolar	tipo_trans_escolar
4	2021-04-28	120006	Aaron Paul	M	NOVATO	2007-06-06	Branca	Não	Não	-
1	2021-04-28	140013	Alex André	M	NOVATO	2009-03-05	Branca	Não	Não	-
9	2021-04-28	140019	Anna Gunn	F	NOVATO	2010-02-07	Não Declarada	Não	Não	-
5	2021-04-28	140014	Bob Odenkirk	M	NOVATO	2009-06-29	Não Declarada	Não	Sim(Rural)	Ônibus
3	2021-04-28	140005	Bryan Cranston	M	NOVATO	2009-08-26	Branca	Não	Não	-
13	2021-04-28	140007	Carmen Serano	F	NOVATO	2010-05-28	Parda	Sim	Não	-
17	2021-04-28	130028	Dean Norris	M	NOVATO	2007-11-09	Parda	Não	Não	-
7	2021-04-28	140008	Jonathan Banks	M	NOVATO	2009-08-20	Branca	Não	Não	-
11	2021-04-28	140012	Laura Fraser	F	NOVATO	2009-08-07	Não Declarada	Não	Não	-
16	2021-04-28	130006	RJ Mitte	M	NOVATO	2008-09-03	Branca	Não	Não	-
6	2021-04-28	140011	Steven Michael Quezada	M	NOVATO	2009-07-17	Parda	Não	Não	-

SELECT

Selecione o código, o nome, a data de nascimento e o bolsa família onde, o bolsa família seja igual a Sim e ordene a busca pela data de nascimento do aluno.

```
1 SELECT codigo, nome, data_nascimento, bolsa_familia  FROM alunos
2 WHERE bolsa_familia = 'Sim'
3 ORDER BY data_nascimento;
```

codigo	nome	data_nascimento ▲ 1	bolsa_familia
210013	Krysten Ritter	2009-10-02	Sim
210020	Emily Rios	2009-10-30	Sim
140007	Carmen Serano	2010-05-28	Sim
210016	Giancarlo Esposito	2010-06-22	Sim

Observação: o campo para o meu ORDER BY não precisa fazer parte do meu **result set**.

SELECT

Selecione o nome, a data de nascimento da tabela alunos onde, a data de nascimento do aluno seja maior ou igual a 20/08/2009 e ordene a busca pelo nome.

Os operadores lógicos são basicamente os mesmo da linguagem de programação.

```
1 SELECT nome, data_nascimento FROM alunos
2 WHERE data_nascimento >= '2009-08-20'
3 ORDER BY nome;
```

nome ▲ 1	data_nascimento
Anna Gunn	2010-02-07
Betsy Brandt	2009-10-05
Bryan Cranston	2009-08-26
Carmen Serano	2010-05-28
Emily Rios	2009-10-30
Felipe Marcos	2009-09-25
Giancarlo Esposito	2010-06-22
Jonathan Banks	2009-08-20
Krysten Ritter	2009-10-02
Tina Parker	2009-12-01

SELECT: busca com BETWEEN (dentro de uma intervalo de valores)

Selecione o código, o nome, a data de enumeração da tabela alunos onde, a data de enumeração esteja **entre** 28/04/2021 e 4/05/2021 e ordene a busca pelo código.

```sql
SELECT codigo, nome, data_enumeracao FROM alunos
WHERE data_enumeracao BETWEEN '21-04-28' AND '2021-05-04'
ORDER BY codigo;
```

codigo ▲ 1	nome	data_enumeracao
120006	Aaron Paul	2021-04-28
130006	RJ Mitte	2021-04-28
130028	Dean Norris	2021-04-28
140005	Bryan Cranston	2021-04-28
140007	Carmen Serano	2021-04-28
140008	Jonathan Banks	2021-04-28
140011	Steven Michael Quezada	2021-04-28
140012	Laura Fraser	2021-04-28
140013	Alex André	2021-04-28
140014	Bob Odenkirk	2021-04-28
140019	Anna Gunn	2021-04-28
210013	Krysten Ritter	2021-05-04

SELECT: busca com IN (especifico os valores)

Faz a busca pelos campos contendo somente as etnias pardas e brancas.

```sql
SELECT codigo, nome, data_nascimento, Etnia FROM alunos
WHERE Etnia IN ('parda', 'branca')
ORDER BY codigo;
```

codigo ▲ 1	nome	data_nascimento	Etnia
120006	Aaron Paul	2007-06-06	Branca
130006	RJ Mitte	2008-09-03	Branca
130028	Dean Norris	2007-11-09	Parda
140005	Bryan Cranston	2009-08-26	Branca
140007	Carmen Serano	2010-05-28	Parda
140008	Jonathan Banks	2009-08-20	Branca
140011	Steven Michael Quezada	2009-07-17	Parda
140013	Alex André	2009-03-05	Branca
210014	Felipe Marcos	2009-09-25	Parda
210017	Tina Parker	2009-12-01	Branca
210020	Emily Rios	2009-10-30	Parda

SELECT:

Podemos usar expressões lógicas juntamente com operadores relacionais para melhorar a filtragem de nossa busca.

AND e OU com =, >, <, >=, >=...

```sql
SELECT codigo, nome, sexo FROM alunos
WHERE sexo = 'F' AND bolsa_familia = 'Sim'
ORDER BY nome;
```

codigo	nome ▲ 1	sexo
140007	Carmen Serano	F
210020	Emily Rios	F
210013	Krysten Ritter	F

SELECT: LIKE

O operador **LIKE** é usado em uma cláusula **WHERE** para pesquisar um padrão especificado em uma coluna.

```
1 SELECT codigo, nome, data_nascimento FROM alunos
2 WHERE nome LIKE 'A%';
```

Existem dois curingas geralmente usados em conjunto com o esse operador:

% (porcentagem): representa zero, um ou vários caracteres

_ (Sublinhado): representa um único caractere

codigo	nome	data_nascimento
140013	Alex André	2009-03-05
120006	Aaron Paul	2007-06-06
140019	Anna Gunn	2010-02-07

O exemplo ao lado nos retorna um **result set** com uma lista dos alunos cujo nome começa com o caractere 'a' (o operador **LIKE** é case sensitive).

Sintaxe:

SELECT campo1, campo2, ...
FROM nome_tabela
WHERE campo_N LIKE ' ';

Observação: Podemos negar o **LIKE** com **NOT LIKE**

WHERE nome LIKE 'a%'	Encontra os valores que começam com a letra "a"
WHERE nome LIKE '%a'	Encontra os valores que terminam com a letra"a"
WHERE nome LIKE '%or%'	Encontra os valores que tenham "or" em alguma posição
WHERE nome LIKE '_r%'	Encontra os valores que tem "r" na segunda posição
WHERE nome LIKE 'a_%'	Encontra os valores que começam com"a" e têm pelo menos 2 caracteres de comprimento
WHERE nome LIKE 'a__%'	Encontra os valores que começam com "a" e têm pelo menos 3 caracteres de comprimento
WHERE nome LIKE 'a%o'	Encontra todos os valores que começam com "a" e terminam com "o"

SELECT: LIKE

```sql
SELECT codigo, nome, Etnia, bolsa_familia
FROM alunos
WHERE Etnia LIKE 'Não%';
```

codigo	nome	Etnia	bolsa_familia
140014	Bob Odenkirk	Não Declarada	Não
210016	Giancarlo Esposito	Não Declarada	Sim
140019	Anna Gunn	Não Declarada	Não
210018	Betsy Brandt	Não Declarada	Não
140012	Laura Fraser	Não Declarada	Não
210013	Krysten Ritter	Não Declarada	Sim

```sql
SELECT codigo, nome, data_nascimento
FROM alunos
WHERE nome NOT LIKE '%a%';
```

codigo	nome	data_nascimento
140014	Bob Odenkirk	2009-06-29
210013	Krysten Ritter	2009-10-02
210020	Emily Rios	2009-10-30
130006	RJ Mitte	2008-09-03

```sql
SELECT nome, bolsa_familia, transporte_escolar
FROM alunos
WHERE sexo LIKE 'F%';
```

nome	bolsa_familia	transporte_escolar
Anna Gunn	Não	Não
Betsy Brandt	Não	Sim(Rural)
Laura Fraser	Não	Não
Krysten Ritter	Sim	Sim(Rural)
Carmen Serano	Sim	Não
Emily Rios	Sim	Sim(Rural)
Tina Parker	Não	Não

SELECT: DISTICT

A instrução **SELECT DISTINCT** é usada para retornar apenas valores distintos (diferentes).

Dentro de uma tabela, uma coluna geralmente contém muitos valores duplicados; e às vezes você só deseja listar os valores diferentes (distintos).

Sintaxe:

SELECT DISTINCT campo1, campo2, ...
FROM nome_tabela;

A = 20
A = 10
A = 10
B = 20
C = 30
C = 30
D = 30

DISTICT

B = 20
D = 30

```
1 SELECT DISTINCT data_enumeracao, Etnia
2 FROM alunos;
```

data_enumeracao	Etnia
2021-04-28	Branca
2021-05-05	Parda
2021-04-28	Não Declarada
2021-04-28	Parda
2021-05-05	Não Declarada
2021-05-04	Não Declarada
2021-05-10	Parda
2021-05-05	Branca

Observação para a Distinção
A etnia branca com data 28/04/2021 só existe 1 na tabela do banco de dados. Assim como a parda e a não declarada com suas respectivas datas.

Novo Banco de Dados (Mercado)

```sql
CREATE DATABASE IF NOT EXISTS mercado
DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci;
```

```sql
CREATE TABLE IF NOT EXISTS produtos (
  id_produto INT AUTO_INCREMENT,
  nome VARCHAR(50) NOT NULL,
  unidade CHAR(3) NOT NULL,
  quantidade INT NOT NULL,
  marca VARCHAR(20),
  preço DECIMAL(7,2) NOT NULL,
  estoque INT NOT NULL,
  validade DATE NOT NULL,
  PRIMARY KEY (id_produto)
)DEFAULT CHARSET = utf8;

```

+ Opções

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
nome	varchar(50)	NO		*NULL*	
unidade	char(3)	NO		*NULL*	
quantidade	int(11)	NO		*NULL*	
marca	varchar(20)	YES		*NULL*	
preço	decimal(7,2)	NO		*NULL*	
estoque	int(11)	NO		*NULL*	
validade	date	NO		*NULL*	

```sql
INSERT INTO produtos VALUES
(DEFAULT, 'Arroz', 'pct', 1, 'Sepé', 4.79, 100, '2022-10-15'),
(DEFAULT, 'Feijão', 'pct', 1, 'Fartura', 6.49, 70, '2022-10-06'),
(DEFAULT, 'Açúcar', 'pct', 1, 'União', 3.46, 120, '2022-11-12'),
(DEFAULT, 'Fubá', 'pct', 1, 'Yoky', 3.19, 40, '2023-10-10'),
(DEFAULT, 'Arroz', 'pct', 1, 'Prato Fino', 6.99, 100, '2022-07-12'),
(DEFAULT, 'Farinha de Trigo', 'pct', 1, 'Dona benta', 3.89, 70, '2022-10-04'),
(DEFAULT, 'Café', 'pct', 1, 'Toko', 9.98, 60, '2023-07-10'),
(DEFAULT, 'Macarrão', 'pct', 1, 'Santa Amália', 3.78, 200, '2022-05-05'),
(DEFAULT, 'café', 'pct', 1, 'Melita', 8.30, 120, '2023-05-23'),
(DEFAULT, 'Feijão', 'pct', 1, 'Canil', 7.25, 150, '2022-06-23'),
(DEFAULT, 'Leite em pó', 'pct', 1, 'Ninho', 13.66, 80, '2023-08-20');
```

id_produto	nome	unidade	quantidade	marca	preço	estoque	validade
1	Arroz	pct	1	Sepé	4.65	100	2022-10-10
2	Arroz	pct	1	Sepé	4.79	100	2022-10-15
3	Feijão	pct	1	Fartura	6.49	70	2022-10-06
4	Açúcar	pct	1	União	3.46	120	2022-11-12
5	Fubá	pct	1	Yoky	3.19	40	2023-10-10
6	Arroz	pct	1	Prato Fino	6.99	100	2022-07-12
7	Farinha de Trigo	pct	1	Dona benta	3.89	70	2022-10-04
8	Café	pct	1	Toko	9.98	60	2023-07-10
9	Macarrão	pct	1	Santa Amália	3.78	200	2022-05-05
10	café	pct	1	Melita	8.30	120	2023-05-23
11	Feijão	pct	1	Canil	7.25	150	2022-06-23
12	Leite em pó	pct	1	Ninho	13.66	80	2023-08-20

SELECT: COUNT()

A função **COUNT()** retorna o número de linhas que correspondem a um critério especificado.

Sintaxe:

SELECT COUNT(campo/coluna)

FROM nome_tabela

WHERE condição;

Veremos a seguir outra formas de utilizar a função **COUNT()**

id_produto	nome	unidade	quantidade	marca	preço	estoque	validade
1	Arroz	pct	1	Sepé	4.65	100	2022-10-10
2	Arroz	pct	1	Sepé	4.79	100	2022-10-15
3	Feijão	pct	1	Fartura	6.49	70	2022-10-06
4	Açúcar	pct	1	União	3.46	120	2022-11-12
5	Fubá	pct	1	Yoky	3.19	40	2023-10-10
6	Arroz	pct	1	Prato Fino	6.99	100	2022-07-12
7	Farinha de Trigo	pct	1	Dona benta	3.89	70	2022-10-04
8	Café	pct	1	Toko	9.98	60	2023-07-10
9	Macarrão	pct	1	Santa Amália	3.78	200	2022-05-05
10	café	pct	1	Melita	8.30	120	2023-05-23
11	Feijão	pct	1	Canil	7.25	150	2022-06-23
12	Leite em pó	pct	1	Ninho	13.66	80	2023-08-20

```
6 /*Retorna o cálculo da quantidade de todos os Registros*/
7 SELECT COUNT(*) from produtos;
```

+ Opções

COUNT(*)
12

SELECT: COUNT()

Usando o COUNT() com DISTINCT

```
1 SELECT DISTINCT nome FROM produtos;
```

nome
Arroz
Feijão
Açúcar
Fubá
Farinha de Trigo
Café
Macarrão
Leite em pó

```
1 SELECT COUNT(DISTINCT nome) FROM produtos;
```

COUNT(DISTINCT nome)
8

SELECT: COUNT() utilizando uma condição com o parâmetro WHERE

```
1 SELECT * from produtos
2 WHERE preco < 5.00;
```

id_produto	nome	unidade	quantidade	marca	preco	estoque	validade
1	Arroz	pct	1	Sepé	4.65	100	2022-10-10
2	Arroz	pct	1	Sepé	4.79	100	2022-10-15
4	Açúcar	pct	1	União	3.46	120	2022-11-12
5	Fubá	pct	1	Yoky	3.19	40	2023-10-10
7	Farinha de Trigo	pct	1	Dona benta	3.89	70	2022-10-04
9	Macarrão	pct	1	Santa Amália	3.78	200	2022-05-05

```
1 SELECT COUNT(*) from produtos
2 WHERE preco < 5.00;
```

COUNT(*)
6

SELECT: MIN() e MAX() utilizando também o Where para estabelecer uma condição

MIN ()

Esta função retorna o menor valor da coluna selecionada.

MAX()

Esta função retorna o maior valor da coluna selecionada.

Sintaxe MIN ()

SELECT MIN(coluna)
FROM nome_tabela
WHERE Condição;

Sintaxe MAX ()

SELECT MAX(coluna)
FROM nome_tabela
WHERE Condição;

```sql
/*Mostrar os campos*/
SELECT id_produto, nome, preco FROM produtos;
```

id_produto	nome	preco
1	Arroz	4.65
2	Arroz	4.79
3	Feijão	6.49
4	Açúcar	3.46
5	Fubá	3.19
6	Arroz	6.99
7	Farinha de Trigo	3.89
8	Café	9.98
9	Macarrão	3.78
10	café	8.30
11	Feijão	7.25
12	Leite em pó	13.66

```sql
/*Menor preço*/
SELECT MIN(preco) FROM produtos
WHERE nome = 'Fubá';
```

MIN(preco)
3.19

```sql
/*Maior preço*/
SELECT MAX(preco) FROM produtos
WHERE nome = 'Arroz';
```

MAX(preco)
6.99

SELECT: SUM() e AVG()

A função **SUM()** calcula a soma de um conjunto de valores.

Nota: os valores **NULL** são ignorados.

Sintaxe

SELECT SUM(coluna)

FROM nome_tabela

WHERE condição;

A função **AVG()** retorna o valor médio de uma coluna numérica.

Sintaxe

SELECT AVG(coluna)

FROM nome_tabela

WHERE condição;

```sql
/*Selecionando os dados*/
SELECT id_produto, nome, marca, preco, estoque FROM produtos
WHERE estoque < 80;
```

id_produto	nome	marca	preco	estoque
3	Feijão	Fartura	6.49	70
5	Fubá	Yoky	3.19	40
7	Farinha de Trigo	Dona benta	3.89	70
8	Café	Toko	9.98	60

```sql
/*Selecionando os dados*/
SELECT SUM(estoque) FROM produtos
WHERE estoque < 80;
```

SUM(estoque)
240

```sql
/*Media*/
SELECT AVG(preco) FROM produtos
WHERE estoque < 80;
```

AVG(preco)
5.887500

GROUP BY

Enquanto o comando **SELECT DISTINCT**, distingue nos dados, ou seja, entre um grupo de dados com o mesmo valor, separa-se 1 e descarta os outros valores, o **GROUP BY** vai realizar o agrupamento das informações.

Qual usar?

- Se você muita duplicidade: Utilize GROUP BY
- Se você tem pouca duplicidade: Utilize DISTINCT

Exemplo:

Selecionando da tabela produtos a quantidade de cada produto no estoque.

```
1 SELECT * FROM produtos;
```

id_produto	nome	unidade	quantidade	marca	preco	estoque	validade
1	Arroz	pct	1	Sepé	4.65	100	2022-10-10
2	Arroz	pct	1	Sepé	4.79	100	2022-10-15
3	Feijão	pct	1	Fartura	6.49	70	2022-10-06
4	Açúcar	pct	1	União	3.46	120	2022-11-12
5	Fubá	pct	1	Yoky	3.19	40	2023-10-10
6	Arroz	pct	1	Prato Fino	6.99	100	2022-07-12
7	Farinha de Trigo	pct	1	Dona benta	3.89	70	2022-10-04
8	Café	pct	1	Toko	9.98	60	2023-07-10
9	Macarrão	pct	1	Santa Amália	3.78	200	2022-05-05
10	café	pct	1	Melita	8.30	120	2023-05-23
11	Feijão	pct	1	Canil	7.25	150	2022-06-23
12	Leite em pó	pct	1	Ninho	13.66	80	2023-08-20

```
1 SELECT DISTINCT nome from produtos;
```

nome
Arroz
Feijão
Açúcar
Fubá
Farinha de Trigo
Café
Macarrão
Leite em pó

```
1 SELECT nome, estoque, COUNT(*) FROM produtos
2 GROUP BY estoque ORDER BY nome;
```

nome ▲ 1	estoque	COUNT(*)
Açúcar	120	2
Arroz	100	3
Café	60	1
Feijão	150	1
Feijão	70	2
Fubá	40	1
Leite em pó	80	1
Macarrão	200	1

GROUP BY

Exemplo:

Vamos pegar da tabela produtos o nome, a marca e o preço onde, o preço seja menor que 7.0.

O agrupamento será feito pela marca do produto e a ordem de exibição dos produtos, pelo nome com um contador para as ocorrências dos valores do campo nome.

```sql
SELECT COUNT(nome), nome, marca, preco FROM produtos
WHERE preco < 7.0
GROUP BY marca ORDER BY nome;
```

```sql
SELECT * FROM produtos;
```

id_produto	nome	unidade	quantidade	marca	preco	estoque	validade
1	Arroz	pct	1	Sepé	4.65	100	2022-10-10
2	Arroz	pct	1	Sepé	4.79	100	2022-10-15
3	Feijão	pct	1	Fartura	6.49	70	2022-10-06
4	Açúcar	pct	1	União	3.46	120	2022-11-12
5	Fubá	pct	1	Yoky	3.19	40	2023-10-10
6	Arroz	pct	1	Prato Fino	6.99	100	2022-07-12
7	Farinha de Trigo	pct	1	Dona benta	3.89	70	2022-10-04
8	Café	pct	1	Toko	9.98	60	2023-07-10
9	Macarrão	pct	1	Santa Amália	3.78	200	2022-05-05
10	café	pct	1	Melita	8.30	120	2023-05-23
11	Feijão	pct	1	Canil	7.25	150	2022-06-23
12	Leite em pó	pct	1	Ninho	13.66	80	2023-08-20

COUNT(nome)	nome ▲ 1	marca	preco
1	Açúcar	União	3.46
2	Arroz	Sepé	4.65
1	Arroz	Prato Fino	6.99
1	Farinha de Trigo	Dona benta	3.89
1	Feijão	Fartura	6.49
1	Fubá	Yoky	3.19
1	Macarrão	Santa Amália	3.78

GROUP BY HAVING

A cláusula **HAVING** foi adicionada ao **SQL** porque não podemos usar o **WHERE** com funções de agregação (exemplo Where Count()). O **HAVING** só funciona com o campo do **GROUP BY**.

```
1 SELECT * FROM produtos;
```

id_produto	nome	unidade	quantidade	marca	preco	estoque	validade
1	Arroz	pct	1	Sepé	4.65	100	2022-10-10
2	Arroz	pct	1	Sepé	4.79	100	2022-10-15
3	Feijão	pct	1	Fartura	6.49	70	2022-10-06
4	Açúcar	pct	1	União	3.46	120	2022-11-12
5	Fubá	pct	1	Yoky	3.19	40	2023-10-10
6	Arroz	pct	1	Prato Fino	6.99	100	2022-07-12
7	Farinha de Trigo	pct	1	Dona benta	3.89	70	2022-10-04
8	Café	pct	1	Toko	9.98	60	2023-07-10
9	Macarrão	pct	1	Santa Amália	3.78	200	2022-05-05
10	café	pct	1	Melita	8.30	120	2023-05-23
11	Feijão	pct	1	Canil	7.25	150	2022-06-23
12	Leite em pó	pct	1	Ninho	13.66	80	2023-08-20

```
1 SELECT COUNT(*), nome, validade FROM produtos
2 GROUP by nome HAVING validade > 2019
3 ORDER BY nome;
```

COUNT(*)	nome ▲ 1	validade
1	Açúcar	2022-11-12
3	Arroz	2022-10-10
2	Café	2023-07-10
1	Farinha de Trigo	2022-10-04
2	Feijão	2022-10-06
1	Fubá	2023-10-10
1	Leite em pó	2023-08-20
1	Macarrão	2022-05-05

GROUP BY HAVING

Aninhando um SELECT

```
1 SELECT AVG(preco) FROM produtos;
```

+ Opções

AVG(preco)
6.369167

```
1 SELECT * FROM produtos;
```

id_produto	nome	unidade	quantidade	marca	preco	estoque	validade
1	Arroz	pct	1	Sepé	4.65	100	2022-10-10
2	Arroz	pct	1	Sepé	4.79	100	2022-10-15
3	Feijão	pct	1	Fartura	6.49	70	2022-10-06
4	Açúcar	pct	1	União	3.46	120	2022-11-12
5	Fubá	pct	1	Yoky	3.19	40	2023-10-10
6	Arroz	pct	1	Prato Fino	6.99	100	2022-07-12
7	Farinha de Trigo	pct	1	Dona benta	3.89	70	2022-10-04
8	Café	pct	1	Toko	9.98	60	2023-07-10
9	Macarrão	pct	1	Santa Amália	3.78	200	2022-05-05
10	café	pct	1	Melita	8.30	120	2023-05-23
11	Feijão	pct	1	Canil	7.25	150	2022-06-23
12	Leite em pó	pct	1	Ninho	13.66	80	2023-08-20

```
1 SELECT id_produto, nome, preco, COUNT(*) FROM produtos
2 WHERE preco > 6.0 GROUP BY preco
3 HAVING preco > (SELECT AVG(preco) FROM produtos);
```

id_produto	nome	preco	COUNT(*)
3	Feijão	6.49	1
6	Arroz	6.99	1
11	Feijão	7.25	1
10	café	8.30	1
8	Café	9.98	1
12	Leite em pó	13.66	1

Relacionamentos

Para criar tabelas relacionadas, defina um relacionamento entre duas tabelas. Um relacionamento é estabelecido quando o valor em um campo, chamado de campo de correspondência (às vezes chamado de campo chave) em um dos lados do relacionamento for comparado com um valor no campo de correspondência do outro lado do relacionamento, de acordo com os critérios estabelecidos no relacionamento.

Por exemplo, você pode criar um relacionamento de modo que, quando o valor no campo ID do aluno em uma tabela Alunos for igual ao valor do campo ID do aluno em uma tabela Classes, os registros nas duas tabelas estão relacionados.

Campos usados em relacionamentos
Um campo de correspondência pode ser um campo ou uma combinação de campos que identifica um registro em uma tabela. Por exemplo, um campo de correspondência pode conter datas que, quando comparadas com os critérios do relacionamento, determinam se o relacionamento é bem-sucedido.
Uma chave é um tipo de campo de correspondência. As chaves normalmente contêm valores que são usados com IDs (como uma ID de produto). Há dois tipos de chaves:

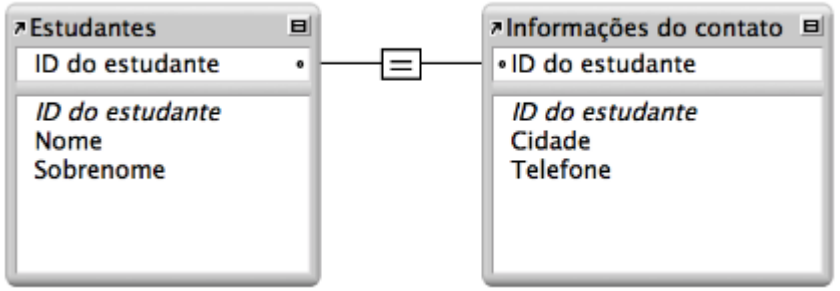
• **chave primária** – um campo que está na mesma tabela que o registro que identifica. Um valor de chave primária deve ser exclusivo e não deve estar vazio (não nulo). Existe somente uma chave primária em uma tabela, mas a chave pode consistir em mais de um campo. Por padrão, quando você cria um arquivo ou uma tabela na caixa de diálogo Gerenciar banco de dados, a nova tabela contém o campo de chave primária.

• **chave externa (Estrangeira)** – um campo em uma tabela que identifica um registro em outra tabela. Os valores das chaves externas não precisam ser exclusivos na tabela e podem estar vazios (ser nulos). Pode haver várias chaves externas em uma tabela.

Fonte: FileMaker. Disponível em: https://fmhelp.filemaker.com/help/18/fmp/pt/index.html#page/FMP_Help%2Frelationships.html%23

Relacionamentos

Relacionamentos um para um
Em um relacionamento um para um, um registro em uma tabela está associado a um e somente um registro em outra tabela. Por exemplo, em um banco de dados escolar, cada aluno tem somente um ID do aluno e cada ID do aluno é atribuído a somente uma pessoa.

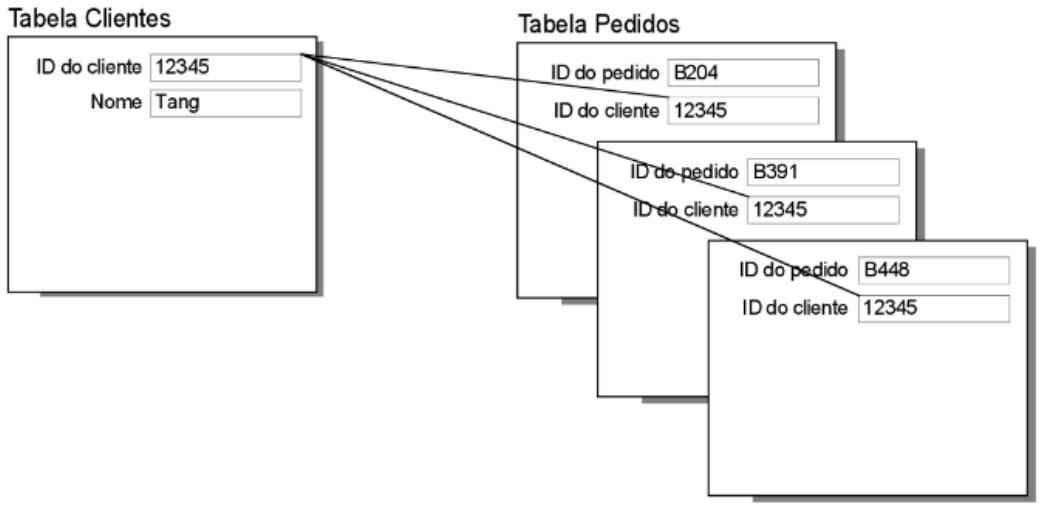


Nesse exemplo, o campo de chave em cada tabela, ID do aluno, foi concebido para conter valores exclusivos. Na tabela Alunos, o campo ID do aluno é a chave primária; na tabela Informações de contato, o campo ID do aluno é uma chave externa.

Esse relacionamento retorna registros relacionados quando o valor no campo ID do aluno na tabela Informações de contato é o mesmo ID do aluno na tabela Alunos.

Relacionamento um para muitos
Em um relacionamento um para muitos, um registro em uma tabela está associado a um ou mais registros em outra tabela. Por exemplo, cada cliente pode ter muitos pedidos de vendas.

Um relacionamento um para muitos tem a seguinte aparência no gráfico de relacionamentos:
Nesse exemplo, o campo de chave primária na tabela Clientes, ID do cliente, foi concebido para conter valores exclusivos. O campo chave externa na tabela Pedidos, o ID do cliente foi concebido para permitir várias instâncias de mesmo valor.



Esse relacionamento retorna registros relacionados quando o valor no campo ID do cliente na tabela Pedidos é o mesmo ID do cliente na tabela Clientes.

Fonte: FileMaker. Disponível em: https://fmhelp.filemaker.com/help/18/fmp/pt/index.html#page/FMP_Help%2Frelationships.html%23

Relacionamentos

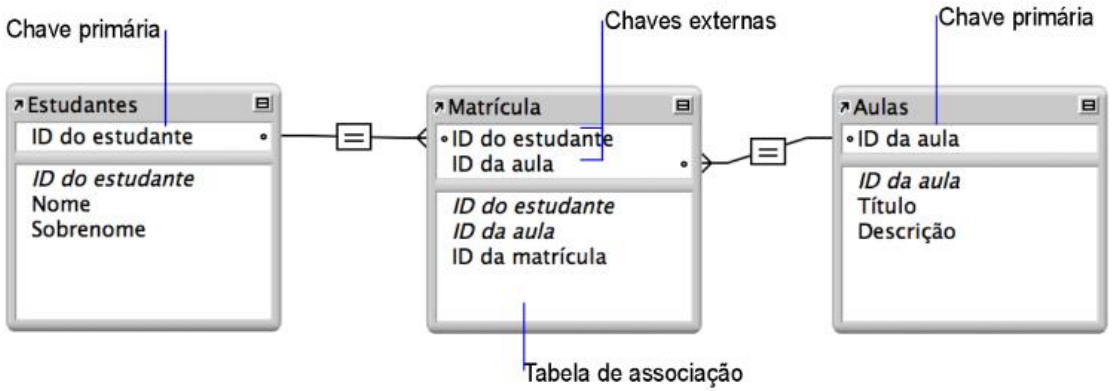
Relacionamento muitos para muitos

Um relacionamento muitos para muitos ocorre quando vários registros em uma tabela são associados a vários registros em outra tabela. Por exemplo, um relacionamento muitos para muitos existe entre clientes e produtos: clientes podem comprar vários produtos e produtos podem ser comprados por muitos clientes.

Sistemas de bancos de dados relacionais normalmente **não permitem implementar um relacionamento muitos para muitos direto entre duas tabelas**. Considere o exemplo de acompanhamento de faturas. Se houver muitas faturas no mesmo número de fatura e um dos seus clientes fez consulta sobre esse número de fatura, você não saberia a qual número ele estava se referindo. Esse é um motivo para atribuir um valor exclusivo a cada fatura.

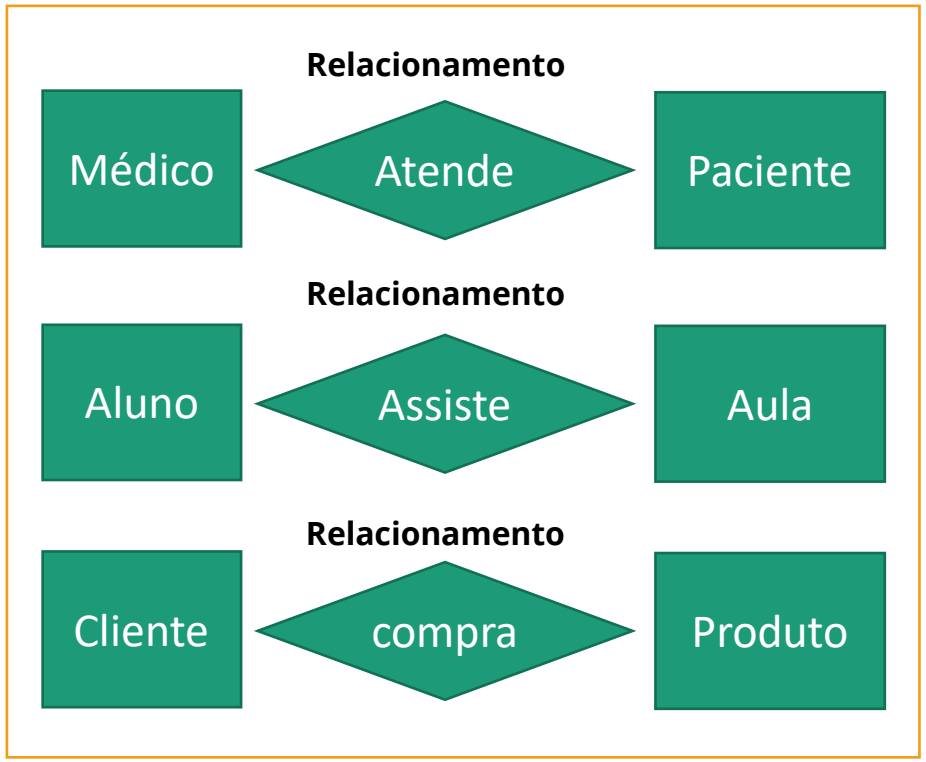
Para evitar esse problema você pode dividir o relacionamento muitos para muitos em dois relacionamentos um para muitos usando uma terceira tabela, chamada de **tabela de associação**. Cada registro em uma tabela de associação inclui um campo de correspondência que contém o valor das chaves primárias das duas tabelas que ela associa. (Na tabela de associação, esses campos de correspondência são chaves externas.) Esses campos de chaves externas são preenchidos com dados, pois os registros da tabela de associação são criados de ambas as tabelas que ela associa.

Um exemplo típico de um relacionamento muitos para muitos é aquele entre alunos e classes. Um aluno pode inscrever-se em várias classes e uma classe pode incluir muitos alunos.

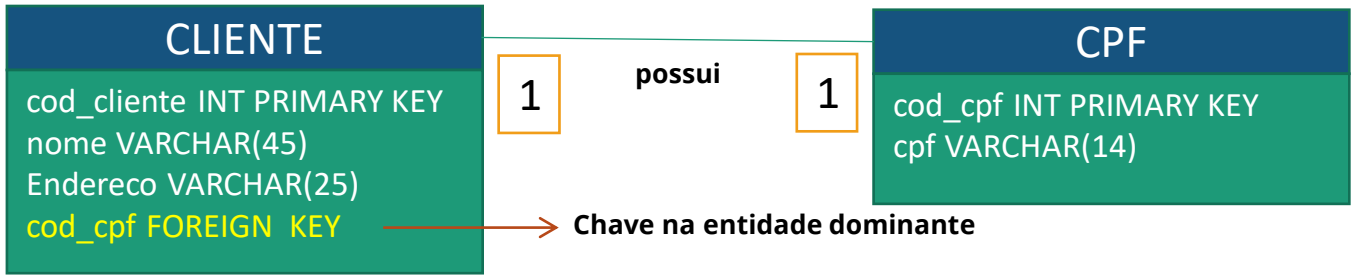


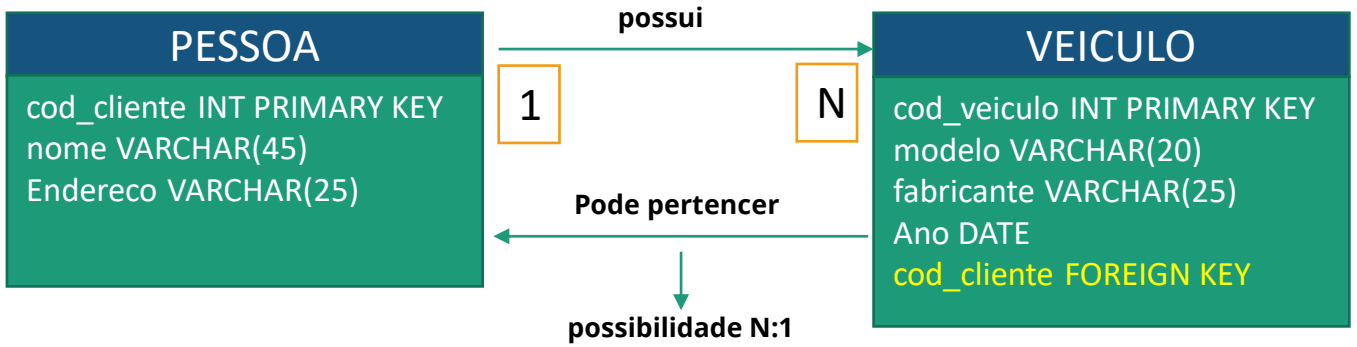
Fonte: FileMaker. Disponível em: https://fmhelp.filemaker.com/help/18/fmp/pt/index.html#page/FMP_Help%2Frelationships.html%23

Diagrama Entidade-Relacionamento (DER) – Simplório!



Cardinalidade





Relacionamentos

InnoDB é um mecanismo de armazenamento para o sistema de gerenciamento de banco de dados MySQL e **MariaDB** . Desde o lançamento do MySQL 5.5.5 em 2010, ele substituiu o MyISAM como o tipo de tabela padrão do MySQL. Ele fornece os recursos de transação compatíveis com ACID padrão , junto com o suporte de chave estrangeira (Integridade Referencial Declarativa). Ele é incluído como padrão na maioria dos binários distribuídos pela MySQL, com exceção de algumas versões OEM.

A saber: Na ciência da computação , **ACID** (atomicidade , consistência , isolamento , durabilidade) é um conjunto de propriedades de transações de banco de dados destinadas a garantir a validade dos dados apesar de erros, falhas de energia e outros contratempos. No contexto de bancos de dados , uma sequência de operações de banco de dados que satisfaz as propriedades ACID (que podem ser percebidas como uma única operação lógica nos dados) é chamada de transação . Por exemplo, uma transferência de fundos de uma conta bancária para outra, mesmo envolvendo várias alterações, como débito em uma conta e crédito em outra, é uma única transação.

A saber: **MariaDB** é um fork desenvolvido pela comunidade e com suporte comercial do sistema de gerenciamento de banco de dados relacional MySQL (RDBMS), destinado a permanecer um software livre e de código aberto sob a GNU General Public License . O desenvolvimento é liderado por alguns dos desenvolvedores originais do MySQL, que o bifurcaram devido a preocupações sobre sua aquisição pela Oracle Corporation em 2009.

Relacionamentos

Crie um banco de dados com as seguintes colunas:

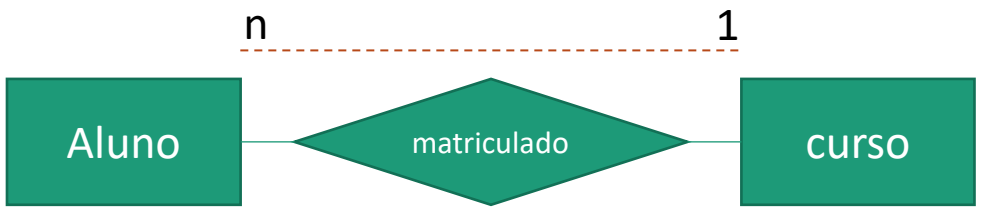
Criando uma tabela Aluno

```sql
CREATE TABLE IF NOT EXISTS aluno(
    id_aluno INT  NOT NULL AUTO_INCREMENT,
    matricula INT(4),
    nome VARCHAR(50) NOT NULL,
    PRIMARY KEY(id_aluno)
)DEFAULT CHARSET = utf8;
```

Criando uma tabela curso

```sql
CREATE TABLE IF NOT EXISTS curso(
    id_curso INT  NOT NULL AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL,
    carga_horaria INT(4),
    PRIMARY KEY(id_curso)
)DEFAULT CHARSET = utf8;
```

Nesta Escola o aluno só pode escolher 1 curso



Relacionamentos

Visualizando as tabelas: DESCRIBE aluno;

Field	Type	Null	Key	Default	Extra
id_aluno	int(11)	NO	PRI	*NULL*	auto_increment
matricula	int(4)	YES		*NULL*	
nome	varchar(50)	NO		*NULL*	

Visualizando as tabelas: DESCRIBE curso;

Field	Type	Null	Key	Default	Extra
id_curso	int(11)	NO	PRI	*NULL*	auto_increment
nome	varchar(50)	NO		*NULL*	
carga_horaria	int(4)	YES		*NULL*	

Relacionamentos

Criando o campo para chave estrangeira

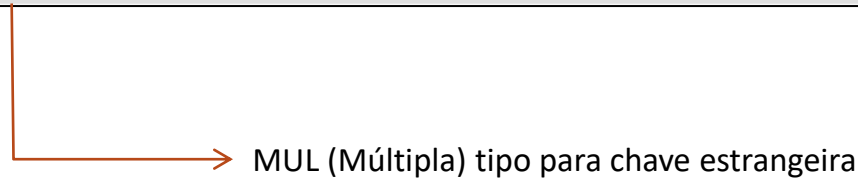
```
1 ALTER TABLE aluno ADD COLUMN id_curso INT;
```

Adicionando a chave estrangeira

```
1 ALTER TABLE aluno ADD FOREIGN KEY (id_curso)
2 REFERENCES curso(id_curso);
```

Visualizando as tabelas: DESCRIBE aluno;

Field	Type	Null	Key	Default	Extra
id_aluno	int(11)	NO	PRI	*NULL*	auto_increment
matricula	int(4)	YES		*NULL*	
nome	varchar(50)	NO		*NULL*	
id_curso	int(11)	YES	MUL	*NULL*	



MUL (Múltipla) tipo para chave estrangeira

Relacionamentos

Inserindo dados na tabela curso

```
1  INSERT INTO curso VALUES
2  (DEFAULT, 'Sistema de Bancos de Dados', 460),
3  (DEFAULT, 'Robótica', 460),
4  (DEFAULT, 'Fundamentos de Redes', 600),
5  (DEFAULT, 'Microsof office 365 avançado', 460),
6  (DEFAULT, 'Linguagem de programação Java', 1200),
7  (DEFAULT, 'Linguagem de programação Python', 600),
8  (DEFAULT, 'Wordpress Básico ao Avançado', 360),
9  (DEFAULT, 'Lógica de Programação', 260),
10 (DEFAULT, 'Inteligência artificial', 460);
```

Exibindo os dados: SELECT * FROM curso ;

id_curso	nome	carga_horaria
1	Sistema de Bancos de Dados	460
2	Robótica	460
3	Fundamentos de Redes	600
4	Microsof office 365 avançado	460
5	Linguagem de programação Java	1200
6	Linguagem de programação Python	600
7	Wordpress Básico ao Avançado	360
8	Lógica de Programação	260
9	Inteligência artificial	460

Relacionamentos

Inserindo dados na tabela aluno

```sql
INSERT INTO aluno VALUES
(DEFAULT, '1234', 'Keanu Reeves', '1'),
(DEFAULT, '1234', 'Christina Ricci', '2'),
(DEFAULT, '1234', 'Jessica Henwick', '1'),
(DEFAULT, '1234', 'Carrie-Anne Moss', '3'),
(DEFAULT, '1234', 'Yahya Abdul-Mateen II', '9'),
(DEFAULT, '1234', 'Priyanka Chopra Jonas', '1'),
(DEFAULT, '1234', 'Jada Pinkett Smith', '3'),
(DEFAULT, '1234', 'Brian J. Smith', '1'),
(DEFAULT, '1234', 'Neil Patrick Harris', '4'),
(DEFAULT, '1234', 'Eréndira Ibarra', '9'),
(DEFAULT, '1234', 'Ellen Hollman', '1');
```

Exibindo os dados: SELECT * FROM aluno ;

id_aluno	matricula	nome	id_curso
1	1234	Keanu Reeves	9
2	1234	Christina Ricci	2
3	1234	Jessica Henwick	1
4	1234	Carrie-Anne Moss	3
5	1234	Yahya Abdul-Mateen II	9
6	1234	Priyanka Chopra Jonas	1
7	1234	Jada Pinkett Smith	3
8	1234	Brian J. Smith	1
9	1234	Neil Patrick Harris	4
10	1234	Eréndira Ibarra	9
11	1234	Ellen Hollman	1