Centro Paula Souza

ETEC Dr Geraldo José Rodrigues Alckmin

Apostila de Banco de Dados

Conceitos Básicos

Componentes Curriculares:
Banco de Dados I e II
Programação Web I e II

Séries atendidas: 1ª, 2ª e 3ª

Autores: **Gilberto Abud Junior**
Reginaldo Luiz Gonçalves

HOME EDITORA

2023

Diagramação

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

Design da capa e projeto gráfico

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

Imagem da capa

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

Revisão de texto

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

Home Editora
CNPJ: 39.242.488/0002-80
www.homeeditora.com
contato@homeeditora.com
9198473-5110
Av. Augusto Montenegro, 4120 - Parque Verde, Belém - PA, 66635-110

Editor-Chefe
Prof. Dr. Ednilson Souza
Bibliotecária
Janaina Karina Alves Trigo Ramos
Produtor editorial
Nazareno Da Luz

Catalogação na publicação
Elaborada por Bibliotecária Janaina Ramos – CRB-8/9166

A165a

Abud Junior, Gilberto

Apostila de banco de dados - conceitos básicos / Gilberto Abud Junior, Reginaldo Luiz Gonçalves. – Belém: Home, 2023.

56 p.; 14,8 X 21 cm

ISBN 978-65-5889-514-5

1. Banco de dados - Desenvolvimento. I. Abud Junior, Gilberto. II. Gonçalves, Reginaldo Luiz. III. Título.

CDD 005.75

Índice para catálogo sistemático

I. Banco de dados - Desenvolvimento

Ficha de Identificação de Material Didático e Autor

1. Título: **Apostila de Banco de Dados – Conceitos Básicos**.
2. Assunto: **Banco de Dados e Linguagem de Programação para Internet.**
3. Resumo: **Além da explicação teórica rápida, utilizando Modelagem e Programação em SQL, a apostila apresenta vários exemplos práticos desenvolvidos em sala de aula com os alunos.**
4. Área / habilitação a que se destina:
   Área/Eixo – Informática e Comunicação
   Habilitações: Desenvolvimento de Sistemas e Informática para Internet.
5. Componentes Curriculares que atinge:
   Banco de Dados I e II; Programação Web I e II
6. Séries atendidas: **1ª, 2ª e 3ª Série**
7. Nome do (s) autor (es): **Gilberto Abud Junior**
   Reginaldo Luiz Gonçalves
8. Unidade Escolar dos autores:
   Etec Dr. Geraldo José Rodrigues Alckmin – Etec Taubaté.
9. Telefones de contato dos autores: **(12) 99104-9082**
   (12) 99108-3878
10. E-mail dos autores:
    gilberto.junior101@etec.sp.gov.br
    reginaldo.goncalves@etec.sp.gov.br
11. Número de páginas: 56, caracteres com espaço: 32.100, laudas: 23.

Sumário

Os autores

Gilberto Abud Junior

Brasileiro, nascido em Taubaté – SP, é graduado Tecnólogo em Processamento de Dados, Pedagogia e Matemática. Possui Formação Pedagógica para Educação Profissional em Nível Médio. É Especialista em Informática em Educação e Administração Escolar. Atua como professor no curso Técnico em Informática desde 1998 no Colégio UNITAU da Universidade de Taubaté e desde 2000 nas unidades das Etecs do Centro Paula Souza, atualmente na unidade de Taubaté – SP.

Reginaldo Luiz Gonçalves

Brasileiro, nascido em Taubaté – SP, é graduado em Computação Científica e Pedagogia. Possui Formação Pedagógica para Educação Profissional em Nível Médio. É Especialista em Informática em Educação, Administração Escolar, Educação a Distância e em Currículo, Didática e Metodologias Ativas. Atua como professor no curso Técnico em Informática desde 1994 no Colégio UNITAU da Universidade de Taubaté e desde 2003 nas unidades das Etecs do Centro Paula Souza, atualmente na unidade de Taubaté – SP.

Agradecimentos

Aos nossos familiares, pela paciência, pelo incentivo e pelo apoio incondicional nos momentos difíceis, principalmente nos momentos de nossa ausência para a dedicação a esse trabalho.

Apresentação

Olá,

Caros estudantes, a programação é uma ciência que utiliza a lógica de nosso pensamento para resolvermos problemas dos mais variados níveis. Com ela conseguimos transmitir e "ensinar" ao computador o que ele deve fazer para a solução do problema proposto. Aprendendo a lógica com certeza você estará habilitado a programar em qualquer linguagem que se interessar.

Esta apostila tem o objetivo de atender uma demanda com mais exemplos práticos no que diz respeito a desenvolvimento de Banco de Dados que precisam ser bem entendidos. Além da explicação teórica rápida, acompanha alguns exemplos práticos desenvolvidos em sala de aula com os alunos dos Cursos Técnicos em Informática, Informática para Internet e Desenvolvimento de Sistemas. A metodologia de resolução de problemas é aplicada e cada atividade aqui resolvida foi estudada, explicada e construída pelo professor e pelos alunos dos respectivos cursos.

Bons estudos aos leitores!

Introdução

Analisar nossa maneira de pensar nos leva a melhorar o pensamento lógico. As linguagens de programação servem para isso. Ao determinarmos o que desejamos que o computador faça, estamos "conversando" com ele. Os computadores, só fazem aquilo que mandamos, e não necessariamente o que desejamos que eles façam. Não deve haver nenhuma ambiguidade nas instruções dos programas que fornecemos ao computador, nem a possibilidade de interpretações alternativas.

O computador sempre tomará algum caminho em suas ações; muito cuidado é necessário para assegurar que o computador siga pelo único caminho correto possível que leve aos resultados desejados. Quando o aluno está interagindo com o computador ele está manipulando conceitos e isso contribui para seu desenvolvimento mental. Ele está adquirindo conceitos da mesma maneira que ele adquire conceitos quando interage com objetos do mundo.

Os alunos aprendem, porque essa interação com o computador propicia um ambiente riquíssimo e bastante efetivo do ponto de vista de construção do conhecimento.

Desejamos aos leitores um bom estudo e sucesso na programação e na construção do conhecimento.

Capítulo 01

O que é um Banco de Dados?

Podemos definir um banco de dados como sendo uma coleção de dados devidamente armazenados e relacionados de forma lógica, representando aspectos da realidade e projetado para atender um determinado grupo de usuários. Ele pode ser criado, gerenciado e acessado por um conjunto de ferramentas que chamamos de **Sistema Gerenciador de Banco de Dados (SGBD)**. O modelo aqui estudado será o **Relacional**, desenvolvido por **Edgar Frank Codd** que descreve como ele deve funcionar. Podemos ver a importância dos Bancos de Dados hoje que são utilizados em todos os sistemas, em grandes corporações, em sites de e-commerce e até nos aplicativos mobile.

Veremos então de forma direta e bem objetiva os passos que devemos seguir na criação de um projeto de banco de dados, passando pela modelagem conceitual até sua implementação física.

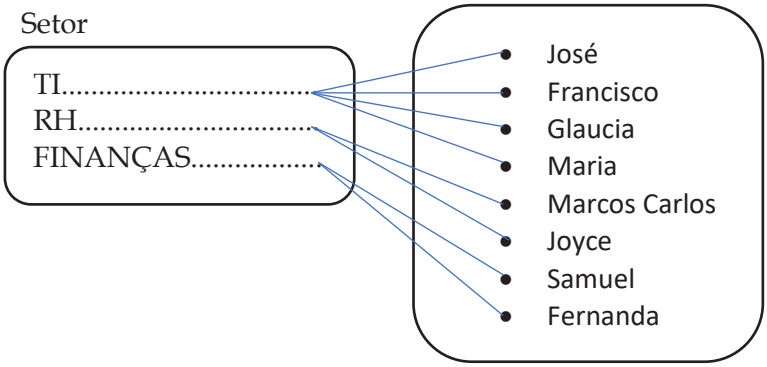
Primeiro Passo

Para criarmos nosso modelo conceitual, usaremos uma ferramenta chamada **Diagrama Entidade-Relacionamento (DER)**, ferramenta esta desenvolvida em meados dos anos 70 por **Peter Chen**. A primeira pergunta é a seguinte: De quais objetos queremos armazenar dados? Quando definirmos isso já

teremos dado início ao nosso projeto de banco de dados. A esses objetos damos o nome de Entidades. Elas são representadas em nosso DER por meio de retângulos. As entidades não ficam soltas ou desligadas em nosso projeto. Elas se conectam por meio de relacionamentos que são representados por um losango. Definido as entidades, os relacionamentos, podemos determinar qual a relação entre as ocorrências de cada entidade determinando qual a sua cardinalidade. As possibilidades são cardinalidade 1:1 (um pra um), 1:N (um pra muitos) e N:N (muitos pra muitos). A cardinalidade nos mostra como uma ocorrência de uma entidade está associada a(s) ocorrências da outra entidade. Veja um exemplo.

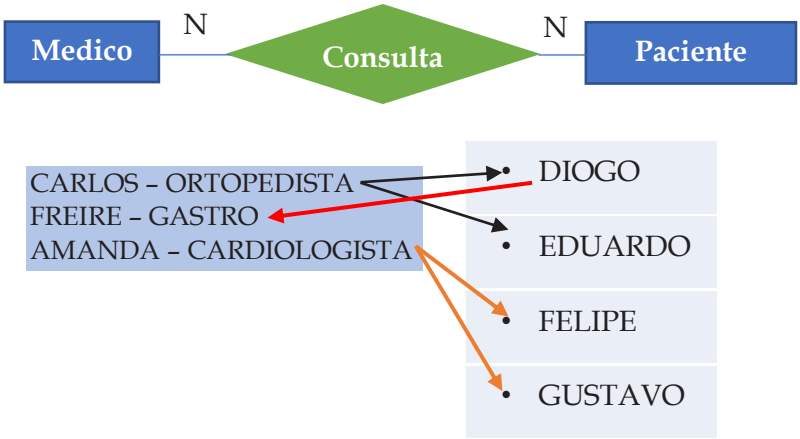
Vamos supor que em uma empresa tenha alguns setores como TI, RH e Finanças por exemplo e cada um desses setores tenham em média 7 funcionários. O Diagrama ficará como na figura a seguir.

Nesse caso a cardinalidade será 1:N, pois apesar da empresa ter muitos setores e muitos funcionários, não será isso que definirá a cardinalidade. Vamos então entender isso. O setor de TI terá um grupo de funcionários que trabalham de forma exclusiva para ele. Isso também ocorrerá com os demais setores que terão seus funcionários de forma exclusiva. Então entendemos que um setor tem muitos funcionários e cada funcionários poderá trabalhar em apenas um setor. Veja as associações a seguir.

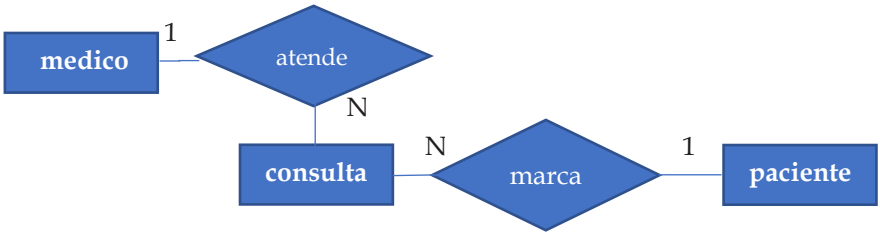


Percebam que o Setor de TI possui muitos funcionários e cada um deles está somente associado ao Setor de TI. Isso definirá a cardinalidade desse modelo. Vejamos abaixo, outros exemplos de DER.

Nesse caso temos um consultório onde há apenas um médico e evidentemente atende muitos pacientes e os pacientes só podem ser consultados por ele, construindo assim um relacionamento 1:N, sem nenhuma dúvida. Mas, e se o DER for de uma clínica com vários médicos (especialidades)? É possível então que todos os médicos atendam vários pacientes em dias e horários diferentes e, que os pacientes possam ser atendidos por vários médicos de acordo com a necessidade de cada um. Veja a seguir o modelo inicial que garantirá um relacionamento N:N, gerando um tratamento especial para a solução desse problema.



Nesse caso o Dr. Carlos atende muitos Pacientes (passou de 1 devemos considerar muitos), assim como os demais irão fazer. Mas o paciente Diogo por exemplo além de se consultar com o ortopedista poderá marcar também uma consulta com qualquer outro, no caso do exemplo no Dr. Freire, determinando assim um relacionamento N:N. Para que nossa tabela futuramente fique bem elaborada daremos um tratamento especial aos relacionamentos N:N. Vamos desmembrá-lo e criar dois relacionamentos 1:N e assim futuramente nossa tabela ficará bem construída. O modelo ficará como a seguir.



Ficou definido então que um médico pode atender várias (ou muitas) consultas e os pacientes podem marcar muitas consultas também. O que aconteceu então foi que colocamos uma entidade nova entre as duas existentes e depois nessa entidade colocamos a cardinalidade "N" nos dois relacionamentos, e nas pontas, ou seja, nas entidades que já estavam indicadas desde o início colocaremos a cardinalidade "1".

Vamos aprender nas próximas páginas a colocar em prática a modelagem e logo em seguida utilizarmos a linguagem SQL para a implementação de nosso banco de dados.

O que é o SQL?

SQL é uma linguagem de grande utilização que significa **Structured Query Language**, ou seja, **Linguagem Estruturada de Pesquisa**. Teve seus fundamentos no modelo relacional de **Edgard Frank Codd** em 1970.

Essa linguagem fez tanto sucesso desde então pelo modelo matemático formal que serviu para o seu desenvolvimento, que foi crescendo dentro do ambiente de banco de dados pela forma que se manipulava os dados, que se tornou uma linguagem padrão nesse ambiente. E, em 1982 a **American National Standarts Institute (ANSI)** tornou a **SQL** uma linguagem padrão no ambiente relacional.

Antes de prosseguirmos vamos definir o que é um banco de dados. Podemos dizer que um banco de dados é um

conjunto de dados armazenados de forma lógica, organizada e bem estruturada. Ele é mais comum em nossa rotina diária do que imaginamos. Se escrevermos em nossa agenda os compromissos da semana, ou uma agenda de contatos de nossos amigos, estamos trabalhando com um banco de dados. Quando o desenvolvemos em um ambiente computacional, temos em mente alguns objetivos específicos como por exemplo uma futura consulta desses dados sem que haja redundância ou inconsistência desses dados e que toda a manipulação no que diz respeito a inclusão, deleção, consulta e atualização desses dados seja rápida com a possibilidade de vários usuários (o banco é feito para atendê-los) utilizarem os seus recursos.

Todo esse trabalho de criarmos um Banco de Bados, suas tabelas relacionadas e sua manipulação será possível por meio dos **Sistemas Gerenciadores de Banco de Dados (SGBD)**, que por sua vez proporcionam um ambiente capaz de gerenciar todos os recursos com segurança.

O que podemos declarar como sendo vantagens da linguagem **SQL** para o nosso banco de dados?

Primeiro sua independência em relação a qualquer Sistema Gerenciador de Banco de Dados. Praticamente todos a utilizam;

Possui também portabilidade entre plataformas de hardware e software;

É formado por comandos em inglês e de fácil entendimento;

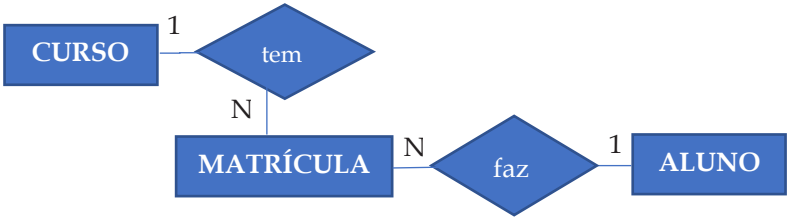
Enfim, veremos como é fácil e rápido criarmos nosso banco de dados, iniciando com a modelagem e depois com o

desenvolvimento de suas respectivas tabelas. Vamos em frente....

Comando CREATE

Vamos então utilizar nosso primeiro comando: CREATE. Inicialmente iremos criar nosso banco de dados exemplo. Nosso banco de dados se chamará "colégio" e possuirá três tabelas relacionadas para que possamos implementar os exemplos necessários para o aprendizado.

O nosso banco de dados "colégio" terá a tabela "curso" para armazenar os dados básicos dos cursos existentes no colégio. Para isso, nossa tabela "curso" será composta pelos campos código, nome, valor. A tabela "aluno" será composta por ra, nome, nota e falta. A tabela "matrícula" será composta por um código, data da matrícula e as chaves estrangeiras correspondentes. Abaixo está o modelo conceitual representado pelo Diagrama Entidade-Relacionamento (DER) do colégio que iremos implementar.



Vamos criar e implementar o nosso banco de dados "colégio", de forma simples e rápida. Olhe só como é fácil....

```
CREATE DATABASE colégio
```

Simples não? Esse comando escrito em qualquer SGBD criará o banco de dados "colégio".

Vamos criar agora as tabelas que fazem parte de nosso banco de dados e a cada linha será explicada cada sentença.

```
CREATE TABLE curso
(
Cur_codigo integer not null primary key,
Cur_nome varchar (60),
Cur_preco decimal (6,2)
);
```

Pareceu um pouco confuso? Mas é muito fácil de compreender. Como falamos inicialmente vamos ter 3 campos em nossa tabela com dados referentes ao curso.

A linha CREATE TABLE curso indica que a tabela será criada com os campos desejados que ficam entre parênteses. Vamos a eles:

A linha "Cur_codigo integer not null primary key," indica que eu preciso de um campo chamado Cur_codigo

(nome que você pode escolher), do tipo inteiro (integer), não nulo (not null), ou seja esse campo é de digitação obrigatória pelo usuário no momento do cadastramento. Esse campo também é o que chamamos de chave primária (primary key), que se define como um identificador único para o registro de uma tabela. Esse valor a ser digitado deverá ser diferente para cada registro digitado.

A linha "Cur_nome varchar (60)," define um campo alfanumérico onde colocaremos o nome do curso e perceba que há uma indicação dentro dos parênteses, o número 60. Isso significa que reservamos 60 caracteres para isso.

A linha "Cur_preco decimal (6,2)", define um campo para um número com casas decimais. Para isso foi indicado dentro dos parênteses 6,2. O primeiro dígito indica a quantidade de dígitos do número que no caso aqui é 6. O segundo dígito indica a quantidade de casas decimais. O número então ficará com 6 dígitos e duas casas decimais. Veja abaixo os exemplos que contemplam essa formatação:

XXXX,XX

3245,00

6453,57

45,37

Pronto após a digitação dessas linhas nossa primeira tabela estará pronta. Vamos seguir os mesmos procedimentos com as demais tabelas e veremos como ficou a estrutura do nosso banco de dados completo.

```
CREATE TABLE  aluno
(
Alu_ra integer not null primary key,
Alu_nome varchar (80),
Alu_nota decimal (4,2),
Alu_faltas integer
);
```

Explicando as linhas...

A linha "Alu_ra integer not null primary key," declara um campo do tipo inteiro não nulo que é também a chave primária como explicado na tabela anterior.

A linha "Alu_nome varchar (80)," declara um campo que armazenará o nome do aluno e esse campo foi declarado como um campo alfanumérico com 80 caracteres.

A linha "Alu_nota decimal (4,2)," declara um campo para armazenar a nota do aluno que tem o formato de até 4 dígitos com duas casas decimais. Exemplo de notas que podem ser digitadas nesse campo: 10,00, 9,75, 5,90 etc.

A linha "Alu_faltas integer" declara um campo para se armazenar as faltas do aluno que será um número inteiro.

Vamos agora terminar o nosso banco de dados com a tabela "matrícula" que segue abaixo

```
CREATE TABLE  matricula
(
Mat_codigo integer not null primary key,
Mat_data varchar (10),
Alu_ra integer not null ,
Foreign key (Alu_ra) references aluno (Alu_ra),
Cur_codigo integer not null,
Foreign key (Cur_codigo) references curso (Cur_codigo)
);
```

As duas primeiras linhas dessa tabela são semelhantes as das outras tabelas anteriores. O que aparece de novo aqui é um novo componente que chamamos de chave estrangeira (Foreign Key). Uma chave estrangeira é campo que criamos que estabelece o relacionamento entre as tabelas envolvidas. A coluna ou campo da tabela que possui chave estrangeira se corresponde a mesma coluna da chave primária da outra tabela. outra tabela, mantendo a integridade referencial. Vamos ver isso quando estivermos fazendo consultas em nossas tabelas. Qual será a resposta da consulta com e sem a chave estrangeira.

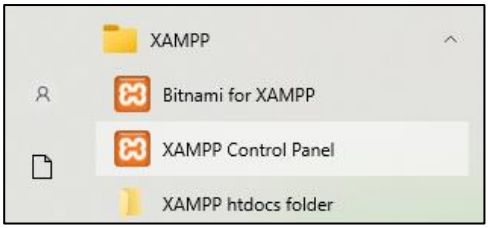
Onde construir o nosso banco?

Vamos utilizar o **XAMPP** que é um pacote com os principais servidores de código aberto existentes no mercado de informática. O **XAMPP** pode ser baixado no endereço abaixo:

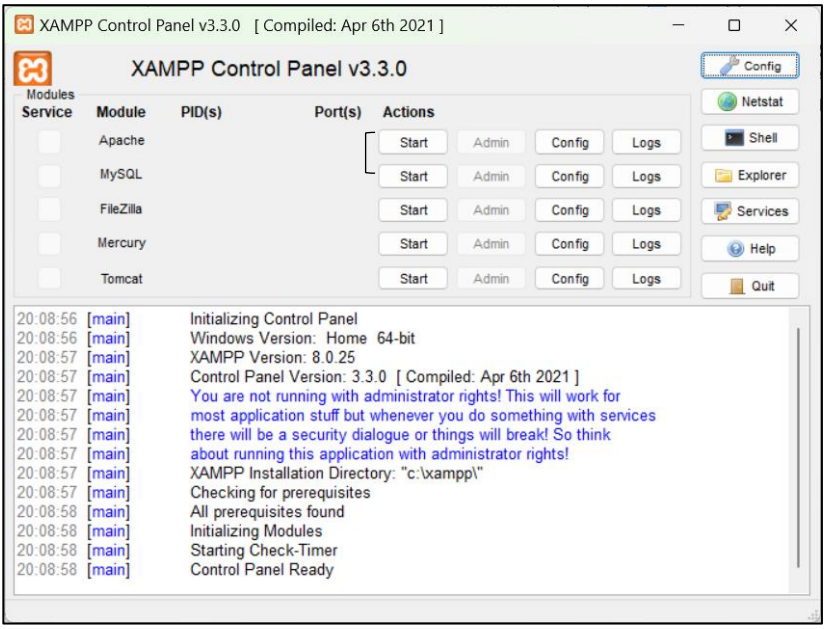
https://www.apachefriends.org/pt_br/download.html

Ao instalar o **XAMPP** conseguiremos utilizar o **MySQL** e todos os seus recursos por meio do **phpMyAdmin**, que permite a administração do **MySQL**. Com ele podemos criar e apagar bancos de dados, inserir, atualizar, deletar tabelas de maneira fácil e rápida.

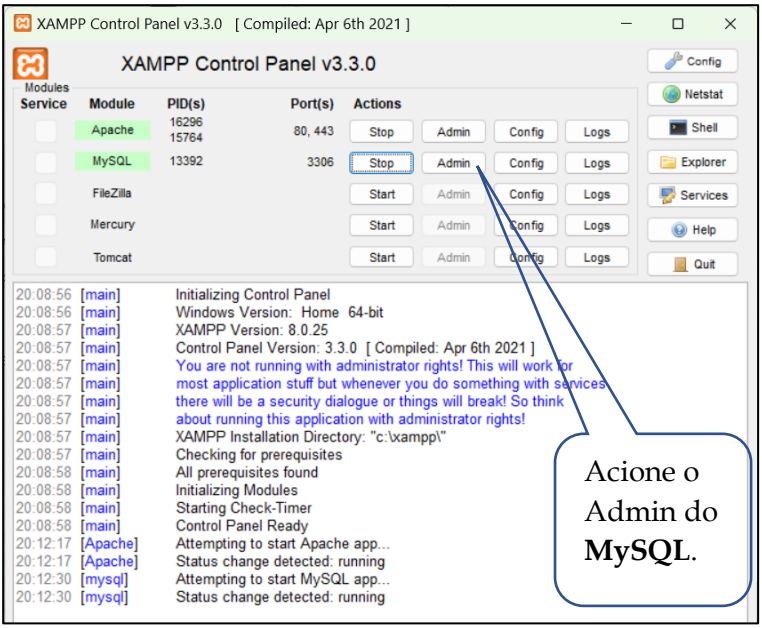
Após a instalação do **XAMPP** devemos abri-lo na opção **XAMPP**, Control Panel.



Uma janela será aberta e devemos iniciar (Start) nas opções **Apache** e **MySQL**:



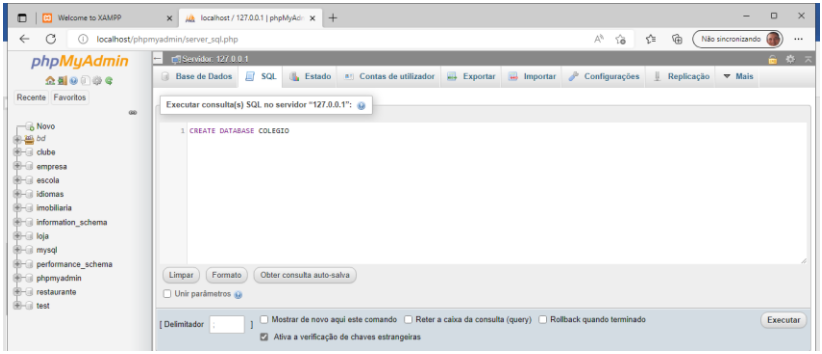
As duas opções ficarão assinaladas, permitindo o acesso por meio da opção Admin. Para criarmos os bancos de dados utilizados nos exemplos nos próximos capítulos acionaremos o Admin do **MySQL**.



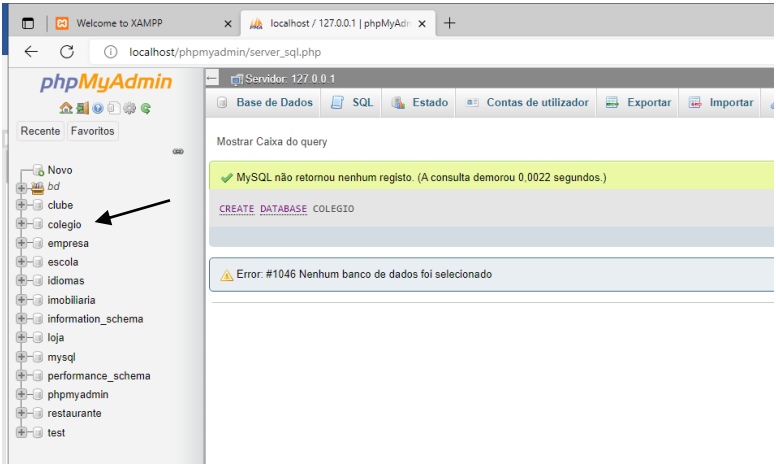
Pelo Admin do **MySQL** entraremos direto no **phpMyAdmin** para criarmos o banco de dados *colegio* e as três tabelas chamadas *curso, aluno e matrícula.* Na imagem abaixo inserimos na caixa de texto abaixo de Criar base de dados o nome do banco de dados que queremos criar que é *colegio.*

Tela do **phpMyAdmin**

Na guia do **phpMyAdmin** escrevemos o comando de criação do banco de dados como visto acima. Em seguida clique no botão executar para que a ação seja realizada.

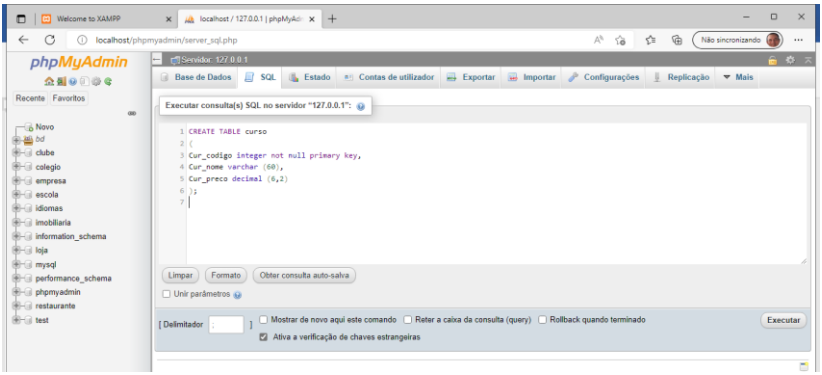


Tela dos comandos de criação do Banco de Dados colegio



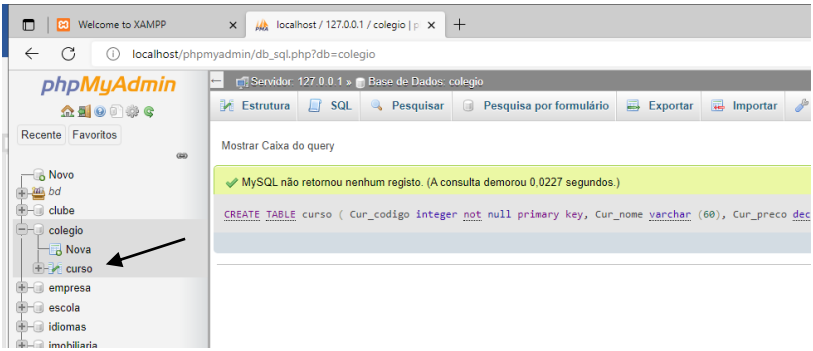
Tela do banco de dados colégio já criado.

Agora vamos criar a nossa primeira tabela que é a tabela curso com os comandos já explicados anteriormente. Após digitar é só clicar no botão executar. Veja a figura abaixo:



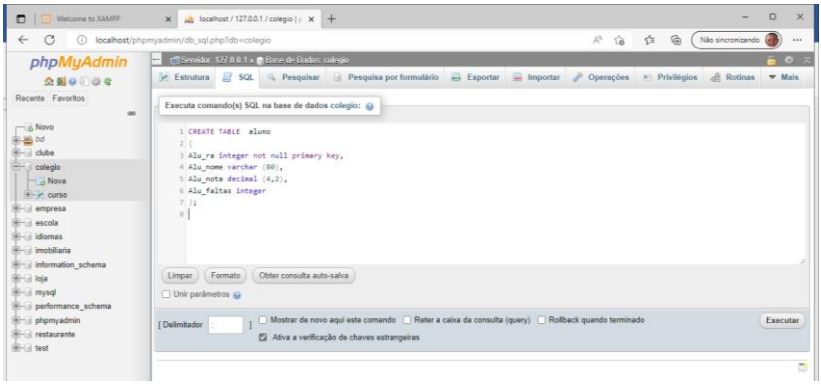
Tela dos comandos de criação da tabela *curso.*

Perceba que após a execução dos comandos de criação da tabela "curso" ela já aparece na coluna ao lado a esquerda.

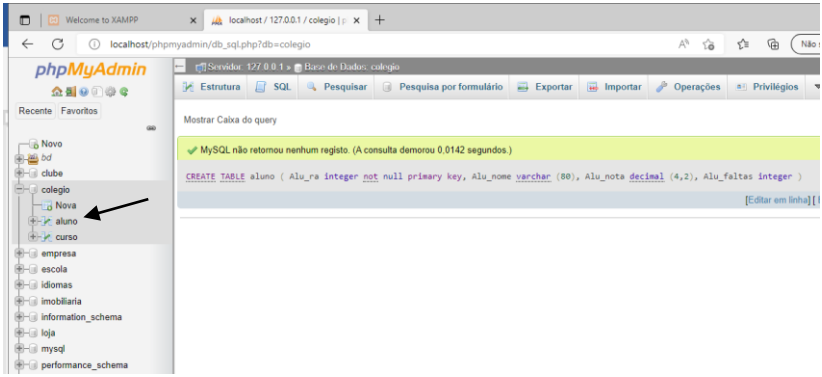


Tela exibindo a tabela *curso* criada

Continuando com nossa tabela "aluno" agora utilizando também os comandos já estudados anteriormente e executando-os. Veja a figura a seguir:

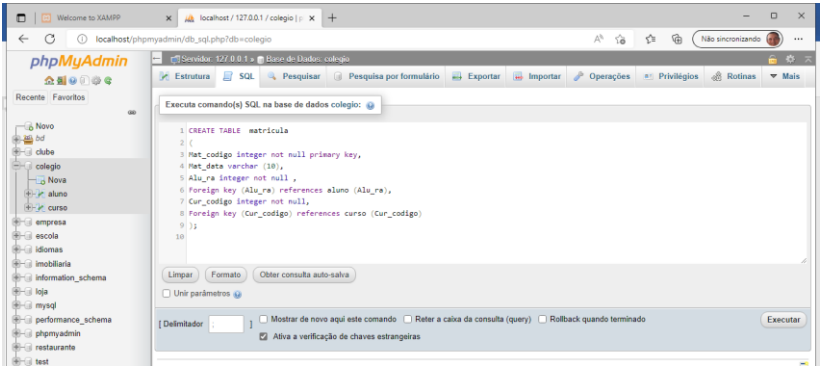


Tela dos comandos de criação da tabela *aluno*

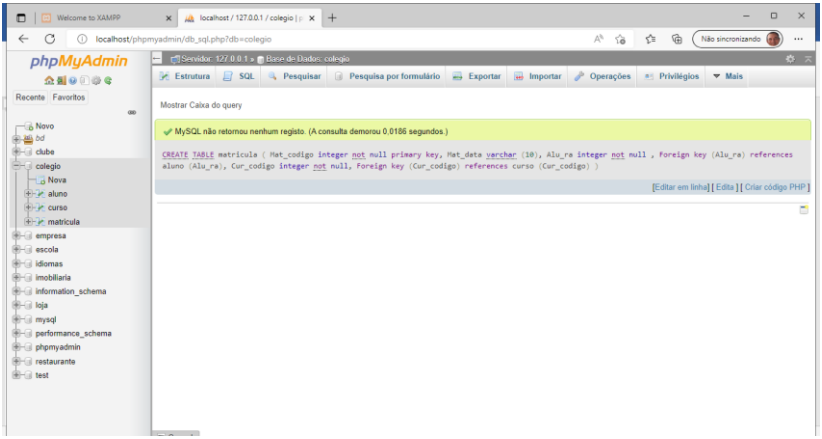


Tela exibindo a tabela *aluno* criada

A última tabela a ser criada será a matrícula que se relaciona com as outras duas, finalizando nosso banco de dados.



Tela dos comandos de criação da tabela *matrícula*



Tela exibindo a tabela *matrícula* criada

Capítulo 02

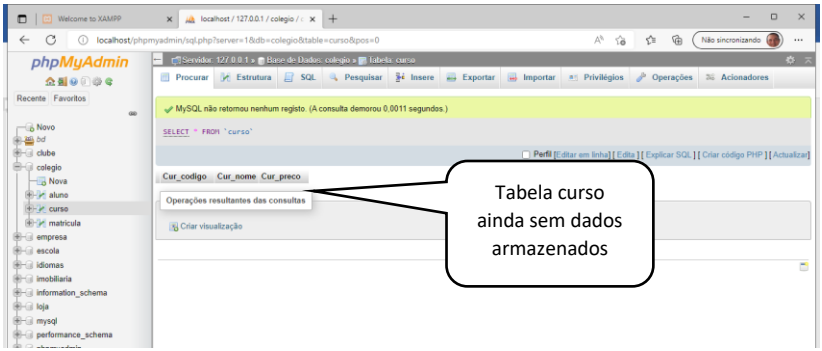
Inserindo dados nas tabelas

A linha de comando para inserir dados na tabela curso é a que segue, considerando as colunas existentes. Veja o que vai acontecer:

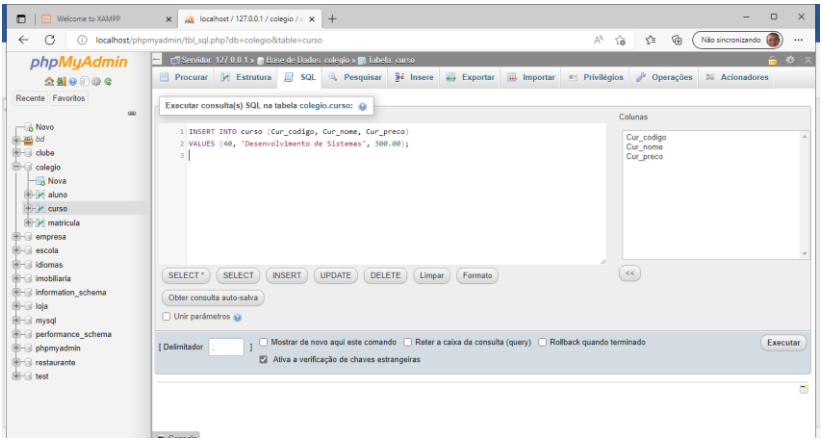
INSERT INTO curso (Cur_codigo, Cur_nome, Cur_preco)

VALUES (40, 'Desenvolvimento de Sistemas', 300.00);

Podemos traduzir da seguinte maneira as linhas de comandos acima: Inserir dentro da tabela curso, nos campos cur_codigo, cur_nome e cur_preco os valores 40 (número inteiro sem apóstrofo), Desenvolvimento de Sistemas (com apóstrofo por ser caractere alfanumérico) e 300.00 (também sem apóstrofo por ser valor numérico decimal).

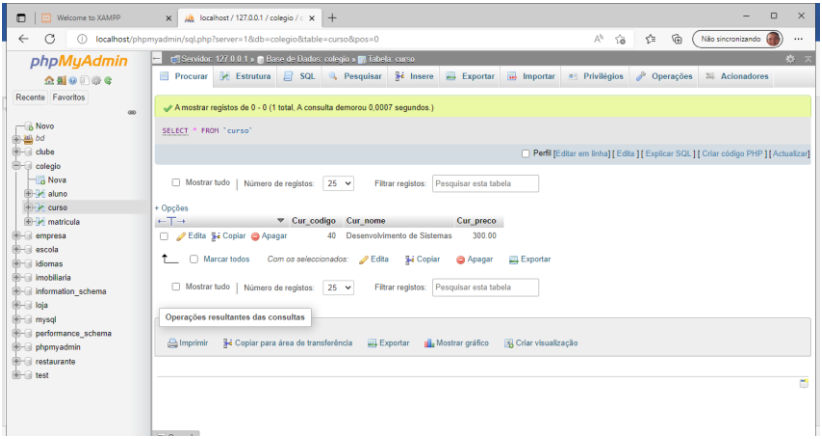


Tela exibindo a conteúdo da tabela *curso*



Tela exibindo a inclusão de dados na tabela *curso*

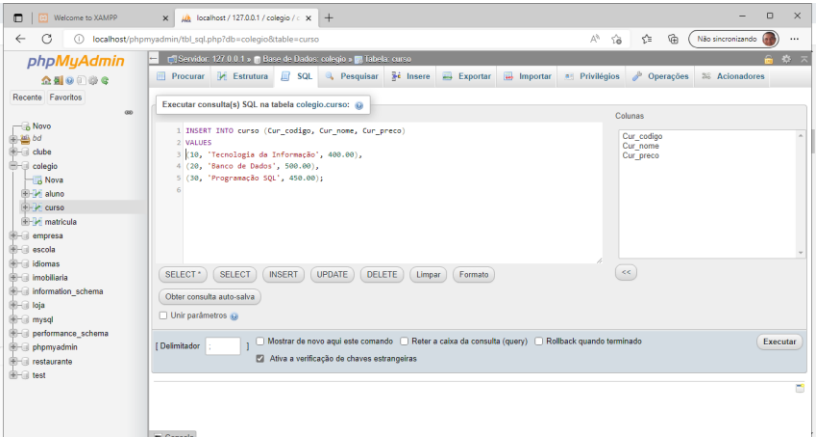
Perceba na tela abaixo onde há indicação da seta que foi inserido o curso desejado na tabela curso.



Tela exibindo o curso cadastrado

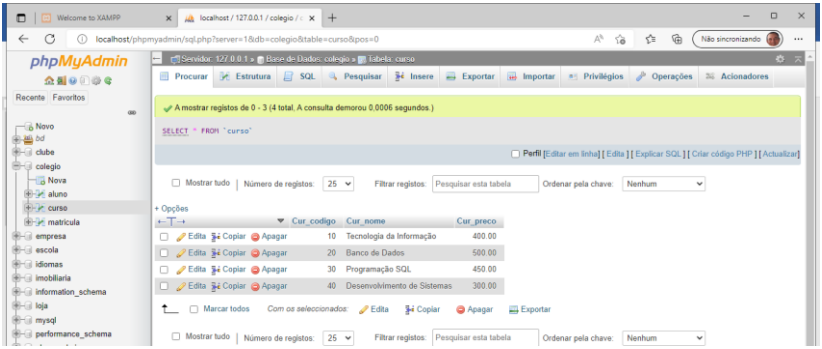
Podemos inserir vários registros num único comando também. Veremos abaixo um exemplo inserindo mais três cursos do nosso colégio. Digitamos o comando

```
INSERT INTO CURSO (Cur_codigo, Cur_nome, Cur_preco)
VALUES
(10, 'Tecnologia da Informação', 400.00),
(20, 'Banco de Dados', 500.00),
(30, 'Programação SQL', 450.00);
```



Tela dos comandos de inclusão de cursos

Vamos ver como ficou a nossa tabela depois dessa inclusão de mais de um curso de uma só vez. Além de inserir os dados, tudo foi ordenado de forma crescente pelo valor da chave primária.



Tela de apresentação dos cursos já cadastrados

Vamos utilizar os comandos estudados acima para também preencher de uma só vez a tabela “aluno” com pelo menos 5 alunos para que possamos futuramente ainda nessa apostila fazermos as consultas que desejarmos. Os comandos digitados e depois executados, mais o resultado do armazenamento dos dados será apresentado depois das linhas de comandos abaixo respectivamente.

INSERT INTO ALUNO (Alu_ra, Alu_nome, Alu_nota, Alu_falta)

VALUES

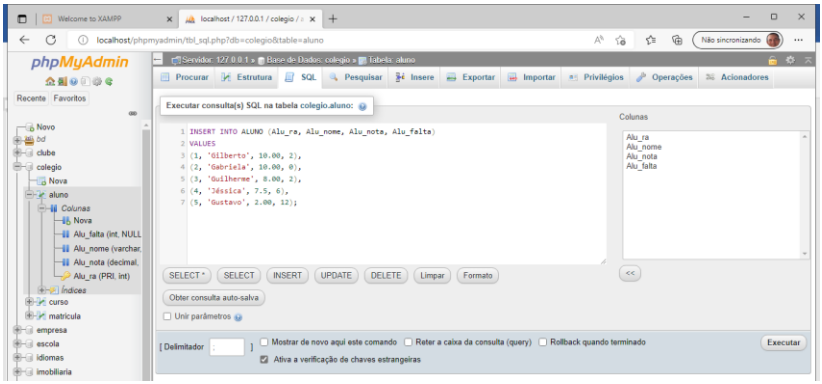
(1, 'Gilberto’, 10.00, 2),

(2, 'Gabriela’, 10.00, 0),

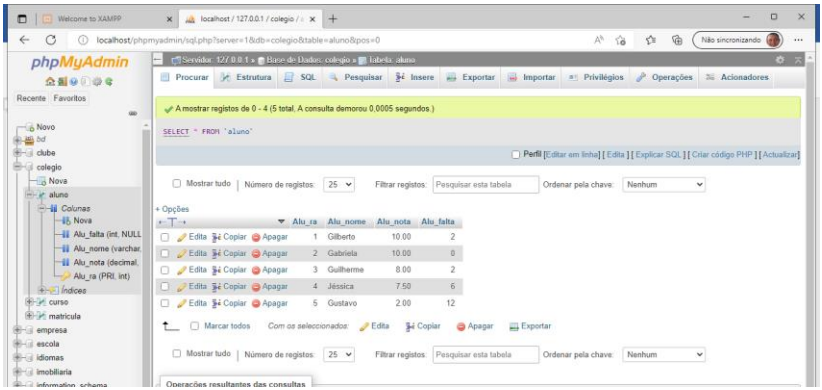
(3, 'Guilherme’, 8.00, 2),

(4, 'Jéssica’, 7.5, 6),

(5, 'Gustavo’, 2.00, 12);



Tela dos comandos de inclusão de dados



Tela dos dados Armazenados na tabela "aluno"

O próximo passo é realizar as matrículas dos alunos com os cursos existentes em nosso colégio. Veremos então o relacionamento que protegerá a integridade dos dados por meio das chaves estrangeiras existentes na tabela "matrícula". Segue os comandos necessários para essa tarefa:

INSERT INTO MATRICULA (Mat_codigo, Mat_data, Alu_ra, cur_codigo)

VALUES

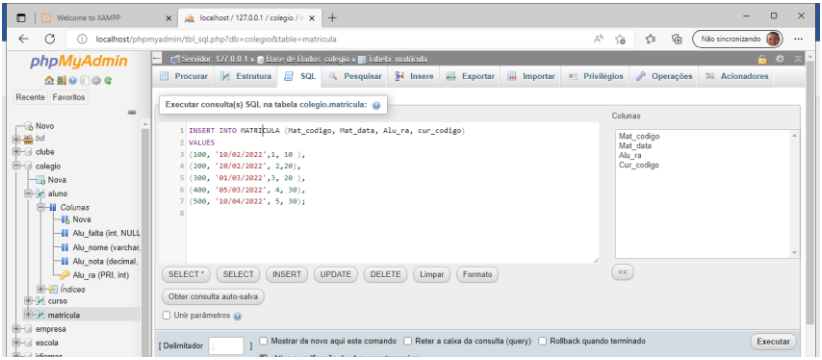
(100, '10/02/2022',1, 10),

(200, '20/02/2022', 2,20),

(300, '01/03/2022',3, 20),

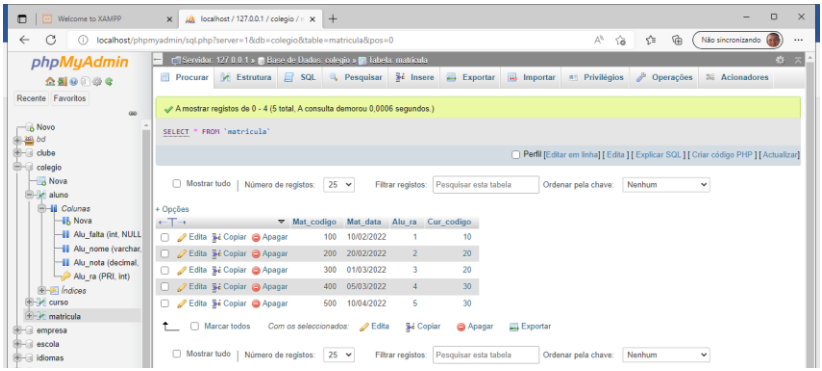
(400, '05/03/2022', 4, 30),

(500, '10/04/2022', 5, 30);



Tela de comandos de inclusão de dados

A seguir veja a tela de dados já cadastrados na tabela matrícula. Os 5 alunos que temos cadastrados forma matriculados em três dos quatro cursos existentes em nosso banco de dados "colégio". Veja o resultado a seguir.



Tela dos dados cadastrados na tabela matricula

Capítulo 3

Consultas básicas em SQL

Quando criamos um banco de dados, seja ele qual for, pensamos em atender uma comunidade de pessoas ou usuários que farão uso dos dados armazenados para diversos fins. E a linguagem SQL é capaz de fazer essas consultas de forma rápida e fácil. A seguir veremos algumas consultas básicas que faremos diretamente em nosso banco de dados "colégio".

A sintaxe da estrutura de consulta com Select é a seguinte:

SELECT <campo1>, <campo2>

FROM <tabela1>, <tabela2>

WHERE <critérios>

Primeiras consultas

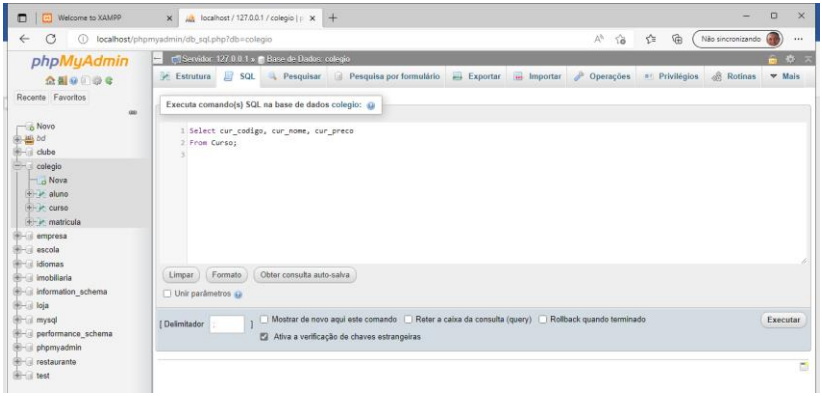
Quero exibir todas as colunas da minha tabela curso.

Select cur_codigo, cur_nome, cur_preco

From Curso;

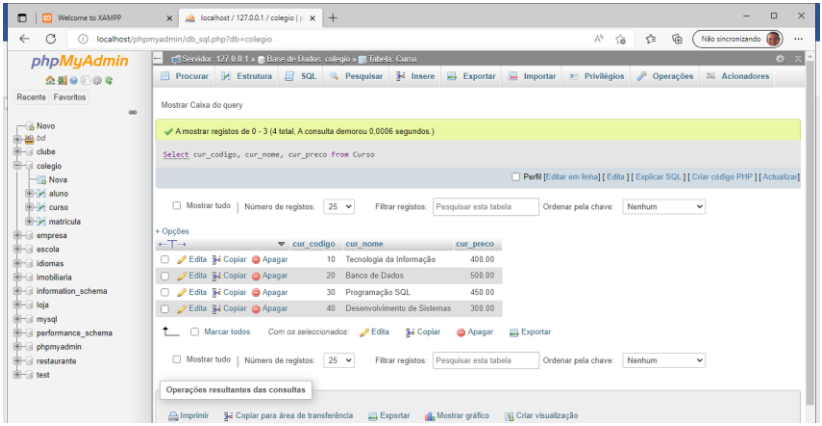
Podemos também usar o caractere * que significa todas as colunas também. Veja a seguir

Select * From Curso;



Tela da linha de comando select

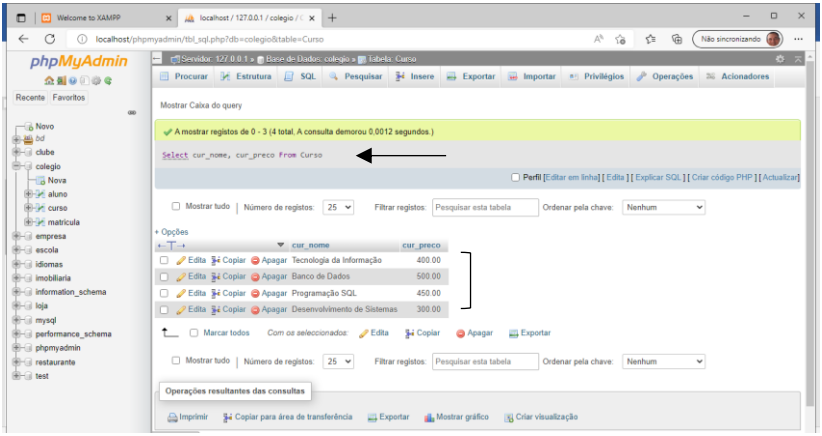
Segue abaixo o resultado dos dados solicitados pelo comando acima.



Tela dos dados exibidos da tabela curso

No momento não quero exibir todas as colunas, mas sim apenas duas das três existentes na tabela curso. Preciso saber apenas o preço e o nome do curso. Veja como fica:

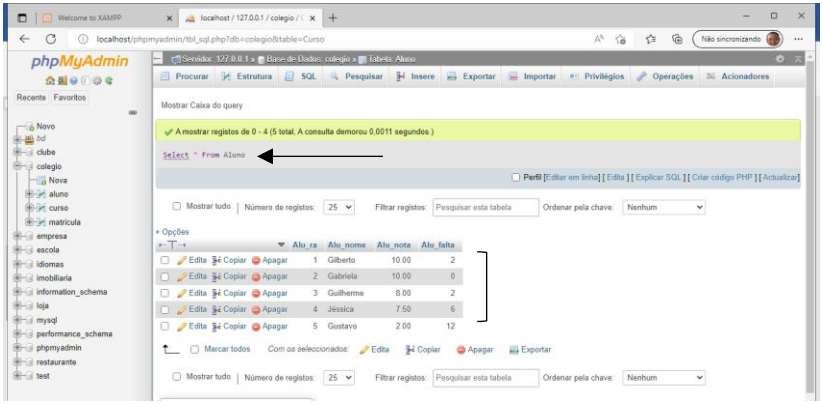
```
Select  cur_nome, cur_preco

From Curso;
```



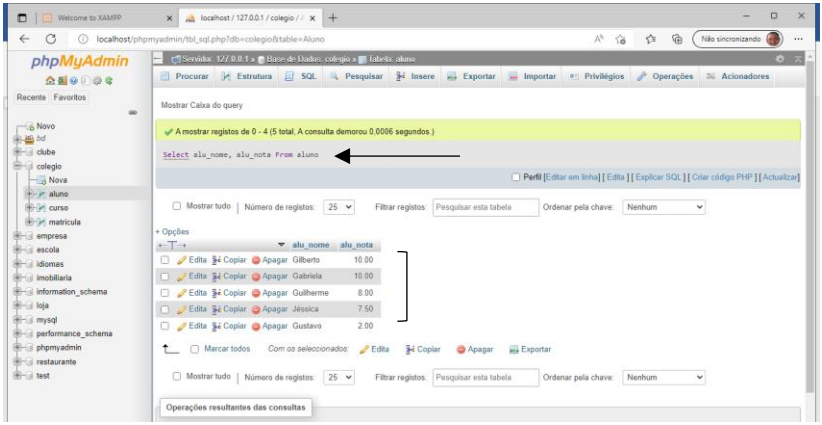
Tela dos dados resultantes da consulta

Faremos agora algumas consultas na tabela aluno na seguinte sequência:

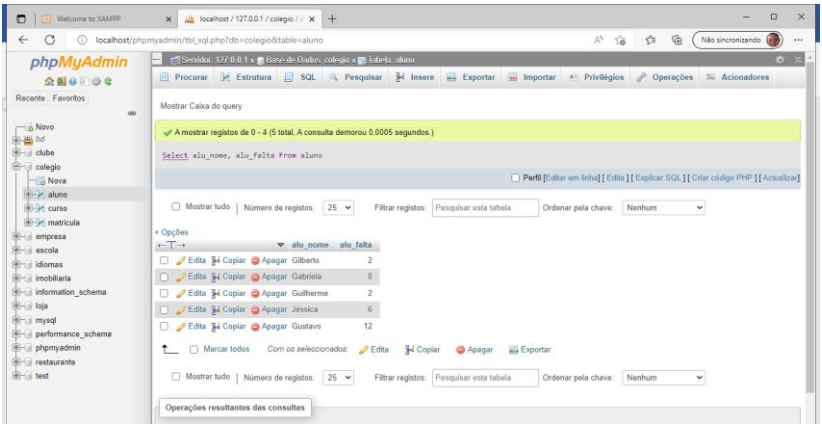
1. Consulta de todos os campos da tabela aluno;
2. Consulta do nome e da nota do aluno;
3. Consulta do nome e das faltas do aluno;



Tela de resultado da consulta 1



Tela de resultado da consulta 2



Tela de resultado da consulta 3

A partir de agora faremos consultas estabelecendo critérios para a resposta desejada. Por exemplo, listar o nome dos alunos que possuem nota maior que 7.5. Nesse momento iremos utilizar um operador relacional, muito usado em programação. Abaixo segue uma tabela com os operadores mais utilizados.

Operador	Comparação
=	Igual
<>	diferente
<	menor
>	maior
< =	menor igual a
> =	maior igual a

Vamos aos exemplos práticos que serão exibidos na sequência...

Consultas com critérios utilizando WHERE.

1. Consulta dos nomes dos alunos que possuem nota maior que 7.5;
2. Consulta dos nomes dos alunos que possuem falta abaixo de 5;
3. Consulta dos nomes e notas dos alunos que possuem faltas acima de 5;

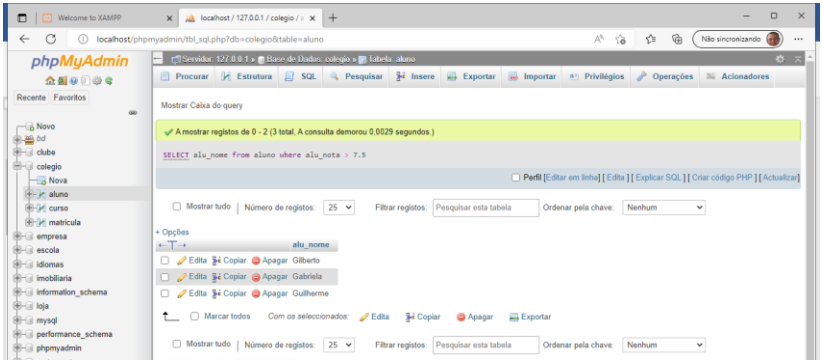
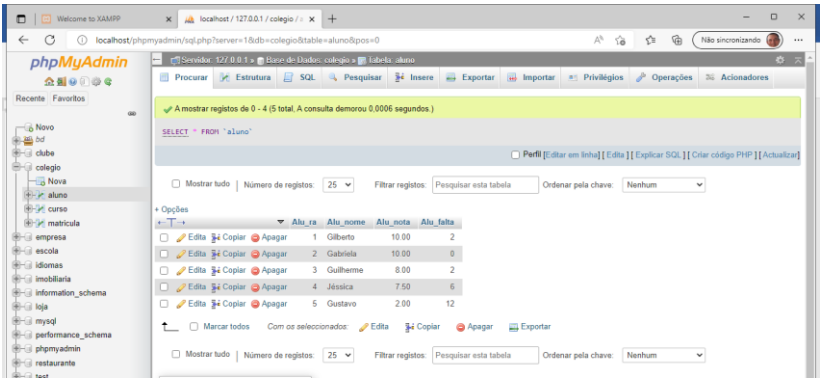


Tabela de resultados da tabela 1.

Veja que a consulta funcionou corretamente escolhendo apenas os alunos com nota maior que 7.5. A aluna Jéssica ficou de fora, pois, sua nota é exatamente 7.5. Sendo assim não entra no critério que pede apenas a nota maior que 7.5.

+ Opções

	Alu_ra	Alu_nome	Alu_nota	Alu_falta
Edita Copiar Apagar	1	Gilberto	10.00	2
Edita Copiar Apagar	2	Gabriela	10.00	0
Edita Copiar Apagar	3	Guilherme	8.00	2
Edita Copiar Apagar	4	Jéssica	7.50	6
Edita Copiar Apagar	5	Gustavo	2.00	12



Tela do resultado da consulta 2

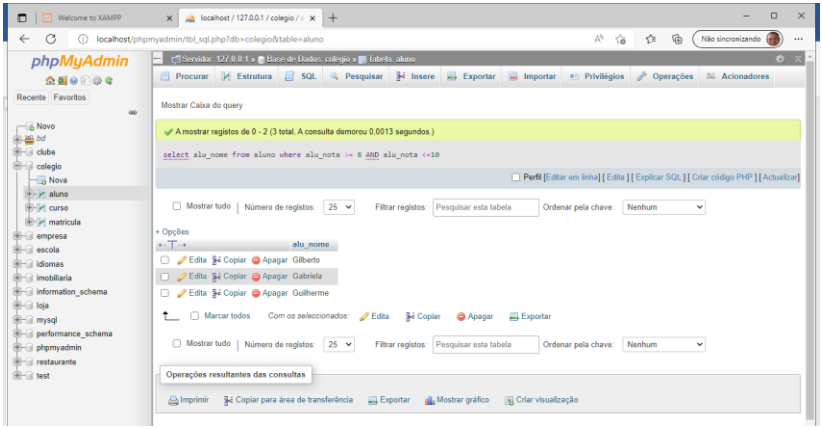
Tela dos resultados da consulta 3

Corretamente foram listados apenas o nome e a nota dos alunos Jéssica e Gustavo que possuem faltas acima de 5.

Quando precisamos fazer uma consulta que envolva uma faixa de valores vamos precisar utilizar os operadores lógicos. Existem vários operadores, porém, nessa apostila iremos utilizar apenas o AND (Se as expressões forem todas verdadeiras) e o OR (Se uma das expressões for verdadeira).

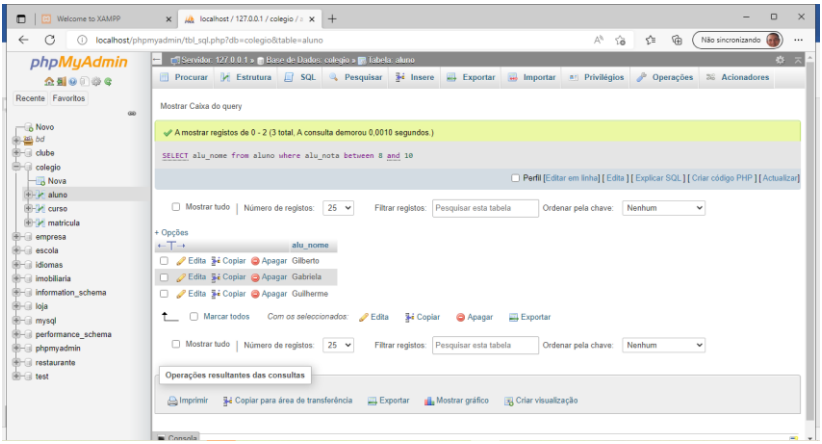
Vamos ao exemplo:

1. Consulta do nome dos alunos com notas entre 8 e 10;
2. Consulta do nome e nota dos alunos com faltas entre 0 e 5.



Tela do resultado da consulta 1

Podemos utilizar também o BETWEEN para consultar uma faixa de valores. Veja o mesmo exemplo acima utilizando o between.



Tela de resultado utilizando o between

Consultas ordenadas

Podemos fazer todas as consultas que quisermos inclusive reaproveitar as que foram feitas anteriormente e escolher um critério para ordená-las em ordem crescente ou decrescente com ORDER BY. Vamos ver alguns exemplos práticos.

Select alu_nome, alu_nota

From Aluno

Order By alu_nome ASC;

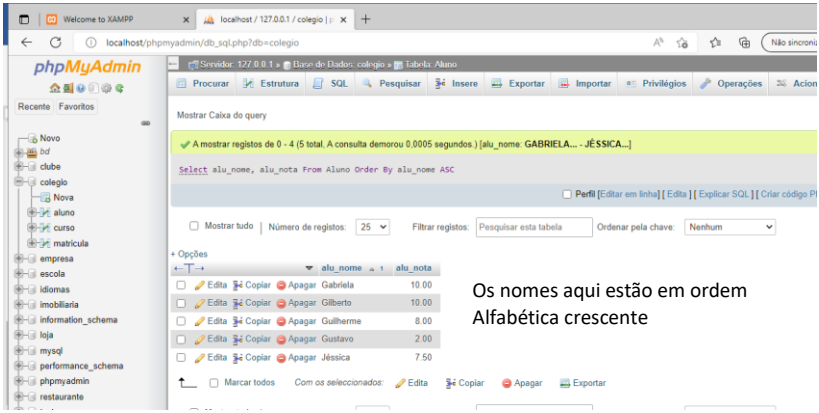
O que foi pedido aqui é que seja listada o nome e a nota do aluno e que seja considerado o nome para ser ordenado em ordem crescente com o ASC.

Perceba que ao cadastrar os alunos, eles foram ordenados pela chave primária e com o comando acima será nos dado como resposta os nomes dos alunos e suas respectivas notas em ordem alfabética crescente.

+ Opções

	Alu_ra	Alu_nome	Alu_nota	Alu_falta
Edita Copiar Apagar	1	Gilberto	10.00	2
Edita Copiar Apagar	2	Gabriela	10.00	0
Edita Copiar Apagar	3	Guilherme	8.00	2
Edita Copiar Apagar	4	Jéssica	7.50	6
Edita Copiar Apagar	5	Gustavo	2.00	12

Tela de cadastro dos alunos



Tela do resultado da consulta ordenada

Se eu quiser inverter basta ao final das linhas de comando não colocar ASC e sim DESC. Fazendo isso os dados serão apresentados em ordem decrescente. Faça isso como atividade e você vai ver os resultados desejados.

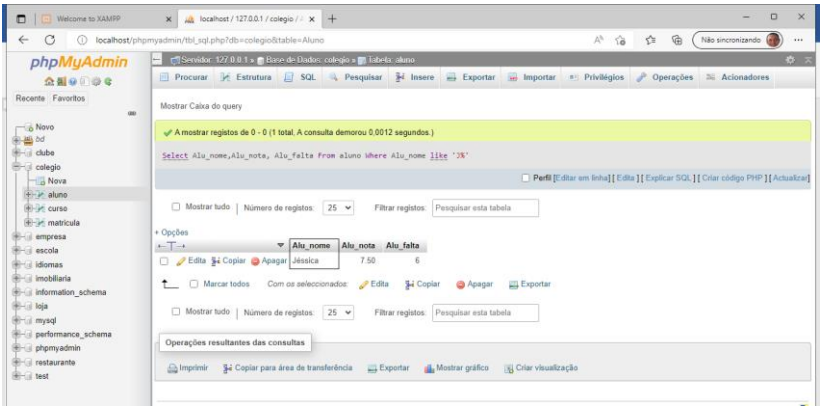
Filtro de Texto

Evidentemente podemos fazer também um filtro com dados do tipo caractere, como por exemplo procurar nomes, endereços desejados em nosso banco de dados. Para isso utilizamos o LIKE. Por exemplo quero listar os alunos que tenham a inicial do nome J, não importando o que virá depois, serão válidos nomes como João, José, etc. Vamos aos exemplos práticos.

Select Alu_nome,Alu_nota, Alu_falta

From aluno

Where Alu_nome like 'J%';

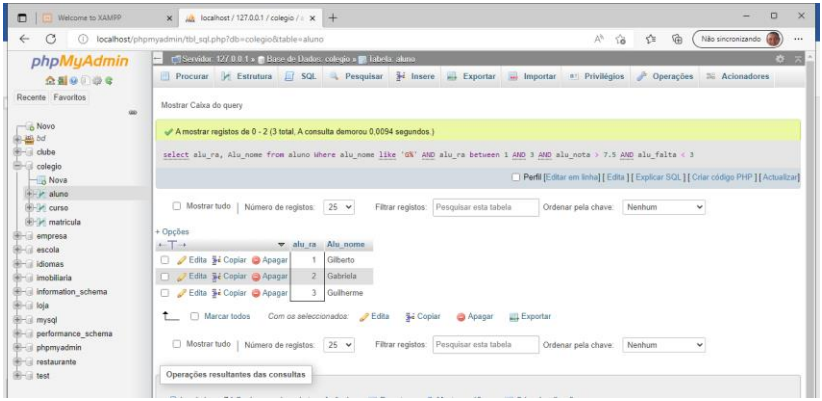


Tela do resultado da consulta com filtro de texto

Nós temos vários alunos cadastrados, porém somente a aluno Jéssica foi dada como resultado pois o LIKE determinou que fosse alguém com nome iniciado com J e o símbolo % deixa

livre os caracteres que vem após o J, não importando qual nome será, desde que se inicie com J.

Podemos fazer consultas em nosso banco de dados mesclando os critérios desejados. Vamos ver um exemplo em que envolveremos mais de um critério e qual será a resposta da consulta.



Tela de resposta da consulta de múltiplos critérios

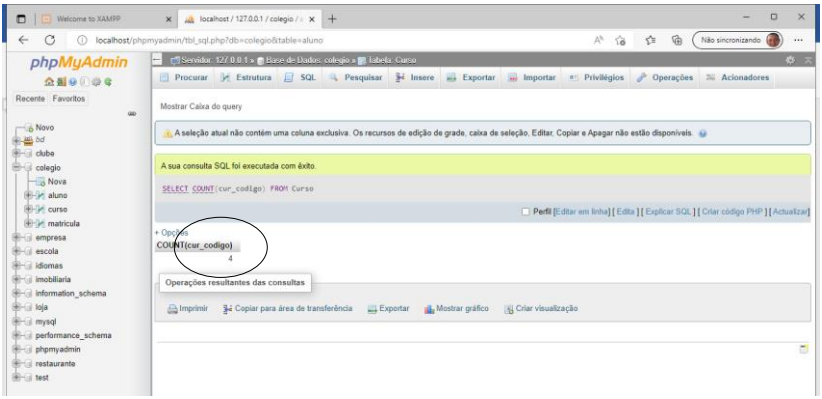
A resposta foi dada corretamente. Nomes da tabela que se iniciam com G, com ra variando de 1 a 3, notas maiores que 7.5 e com faltas menores que 3. Vários critérios podem ser utilizados em todas as nossas consultas desde que sejam necessários.

O **SQL** também possibilita o uso de funções predefinidas que iremos estudar a seguir.

Funções

Função COUNT

Essa função retorna a quantidades de itens da seleção desejada. Conforme a figura abaixo usamos o COUNT para que a resposta de quantos cursos temos no colégio seja exibida.

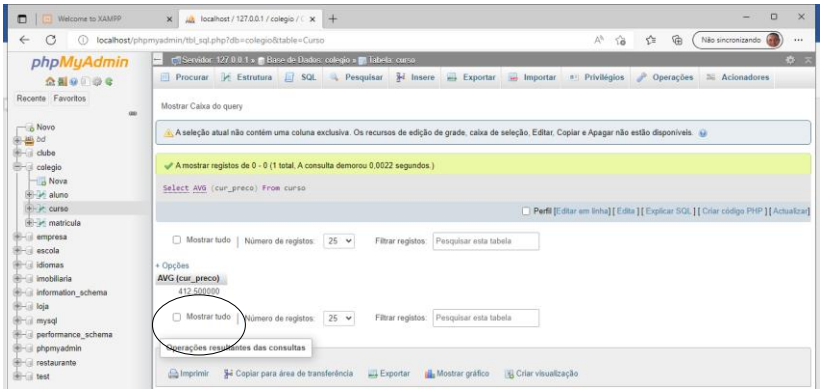


Tela do resultado da função count

Função AVG

Essa função retorna a média dos valores do campo especificado. Podemos escrever o comando para a média do preço dos cursos. Vejamos a linha de comando a ser escrita no **phpMyAdmin** que fará a soma dos valores dos quatro cursos existentes e evidentemente dividirá esse valor por 4.

```
Select AVG (cur_preco)
From curso;
```



Tela de resultado da média calculada pela função avg

Funções MIN...MAX...SUM

Essas funções determinam ou como costumamos dizer retornam o menor valor, o maior valor e a soma de um determinado grupo de registros respectivamente. Segue as linhas de comando das três funções e os resultados de cada uma delas.

1. Select MIN (alu_nota)

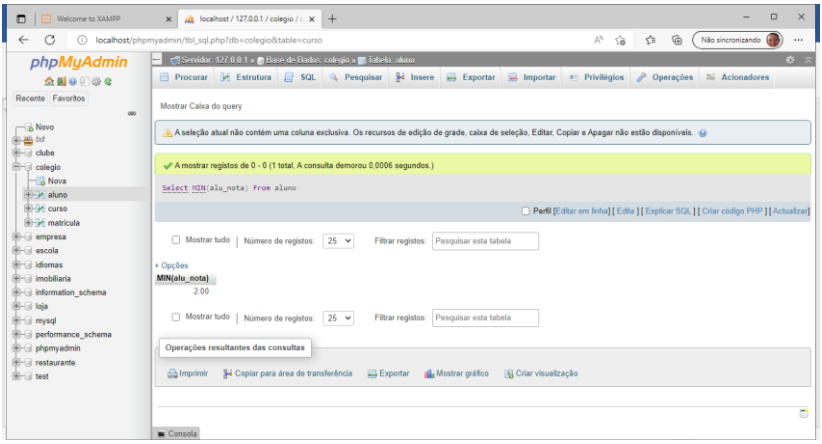
   From aluno;

2. Select MAX (alu_nota)

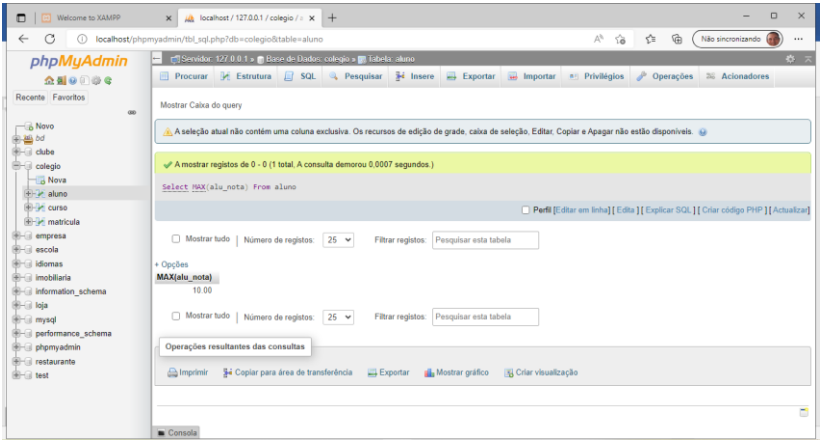
   From aluno;

3. Select SUM (cur_preco)

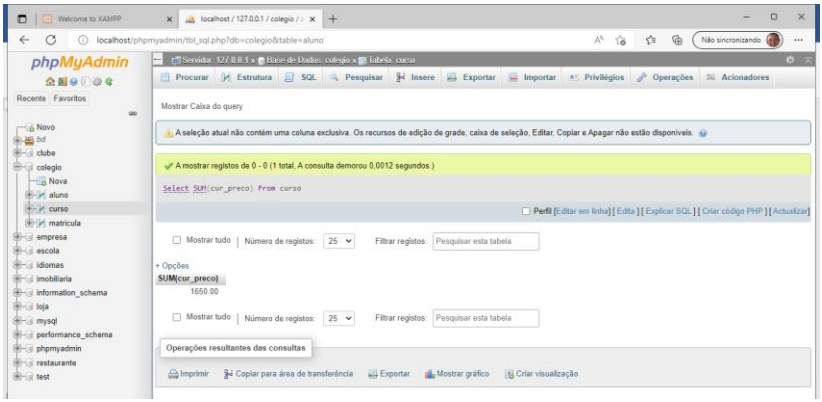
   From curso;



Tela do resultado da função MIN



Tela do resultado da função MAX



Tela do resultado da função SUM

Consultas simultâneas em mais de uma tabela

O SELECT também permite fazer consultas em duas ou mais tabelas simultaneamente exibindo um único resultado. Nesse caso as tabelas envolvidas são unificadas por meio das chaves primária e estrangeira que garantirão a integridade das respostas, evitando inclusive duplicidade de dados como no exemplo a seguir. As tabelas estão relacionadas conforme abaixo:

Mat_codigo	Mat_data	Alu_ra	Cur_codigo
100	10/02/2022	1	10
200	20/02/2022	2	20
300	01/03/2022	3	20
400	05/03/2022	4	30
500	10/04/2022	5	30

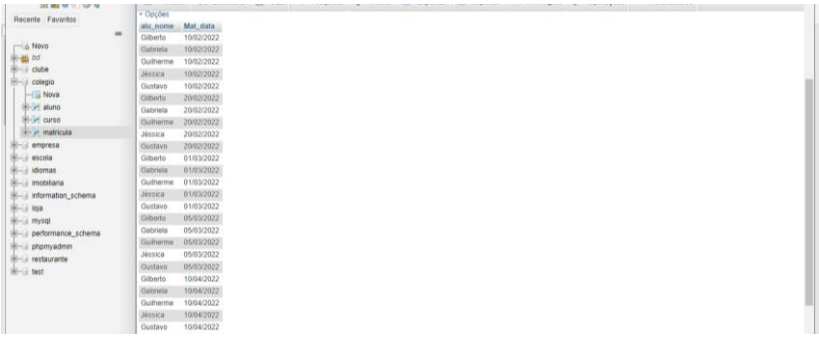
Alu_ra	Alu_nome	Alu_nota	Alu_falta
1	Gilberto	10.00	2
2	Gabriela	10.00	0
3	Guilherme	8.00	2
4	Jéssica	7.50	6
5	Gustavo	2.00	12

Cur_codigo	Cur_nome	Cur_preco
10	Tecnologia da Informação	400.00
20	Banco de Dados	500.00
30	Programação SQL	450.00
40	Desenvolvimento de Sistemas	300.00

Devemos seguir a orientação para se fazer uma consulta, porém se não utilizarmos as chaves envolvidas não teremos uma reposta correta como apresentado abaixo:

select alu_nome, Mat_data (colocamos os campos desejados)

from aluno, matricula; (as tabelas envolvidas)



Tela de resultado incorreto sem a utilização das chaves

A resposta exibida faz uma conexão sem sentido de que todos os alunos fizeram matriculas em todas as datas existentes. Apesar de não aparecer nenhum erro a ser corrigido, a resposta está totalmente equivocada.

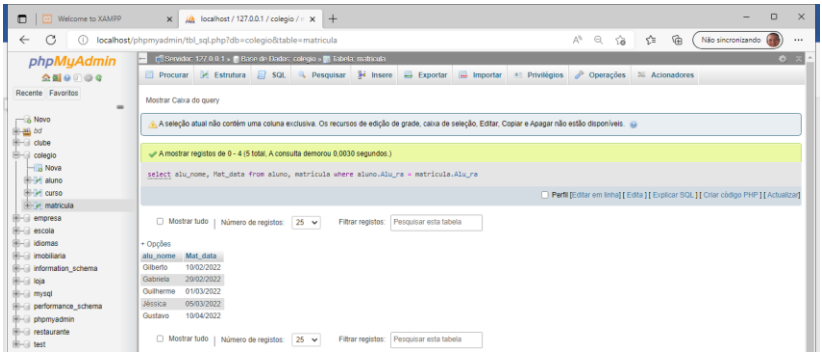
Para que isso não aconteça vamos ver o primeiro exemplo envolvendo as chaves.

select alu_nome, Mat_data (colocamos os campos desejados)

from aluno, matricula (as tabelas envolvidas)

where aluno.Alu_ra = matricula.Alu_ra;

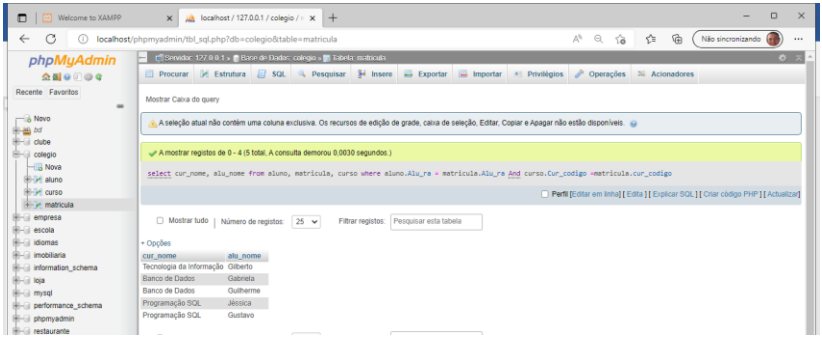
Chave primária	=	Chave estrangeira



Tela do resultado correto com uso das chaves

Vamos agora envolver as três tabelas do nosso banco de dados simultaneamente. O resultado a ser exibido deve ser o nome do curso e o nome dos alunos matriculados.

```
select cur_nome, alu_nome

from aluno, matricula, curso

where aluno.Alu_ra = matricula.Alu_ra

And    curso.Cur_codigo =matricula.cur_codigo;
```



Tela de resultados com 3 tabelas

Referências

ABUD Junior, Gilberto; GONÇALVES, Reginaldo Luiz. **Apostila de Estruturas de Repetição em C# - Versão 1.0**. Home Editora. Belém-Pará, 2023.

ALVES, Willian Pereira: **Construindo uma Aplicação Completa com PHP e MySQL**. Editora Novatec. 2017. São Paulo.

DATE, C.J.: **Introdução a Sistemas de Banco de Dados**. Editora GEN LTC. 2004. São Paulo.

GONÇALVES, Reginaldo Luiz; Abud, Gilberto J. **Algoritmos e Programação:** Exemplos Práticos. 1ª Edição. Volume 1. Worges Editoração. Belém-Pará. 2022.

GONÇALVES, Reginaldo Luiz; Abud, Gilberto J**. Programação em Linguagem C:** Exemplos Práticos. 1ª Edição. Volume 2. Worges Editoração. Belém-Pará. 2022.

GONÇALVES, Reginaldo Luiz; Abud, Gilberto J. **Programando em JavaScript:** Exemplos Práticos. 1ª Edição. Volume 3. Worges Editoração. Belém-Pará. 2022.

GONÇALVES, Reginaldo Luiz; Abud, Gilberto J. **Programando em PHP:** Exemplos Práticos. 1ª Edição. Volume 4. Worges Editoração. Belém-Pará. 2022.

LACERDA, Ivan Marx Freire; OLIVEIRA, Ana Liz Souto: **Programador WEB: Um guia para programação e manipulação de banco de dados**. Editora SENAC. São Paulo

MACHADO, Felipe; ABREU, Mauricio: **Projeto de Banco de Dados: Uma visão Prática. Editora Érica Ltda**. 2018. São Paulo.

MILANI, André: **PostgreSQL: Guia do Programador**. Novatec Editora Ltda. 2008. São Paulo.

TEOREY, Toby; LIGHTSTONE, Sam; NADEAU, Tom: **Projeto e Modelagem de Bancos de Dados**. Elsevier. 2007.Rio de Janeiro

Sites

Acesso em 22/04/2023:
https://www.apachefriends.org/pt_br/download.html.

Comentários finais

Convido a todos a lerem e estudarem também os nossos livros da **Coleção Programando: Aprenda Rápido**, Algoritmos e Programação: Exemplos Práticos, Programação em Linguagem C: Exemplos Práticos, Programando em JavaScript: Exemplos Práticos e Programando em PHP: Exemplos Práticos. Os recursos do PHP unido ao JavaScript, HTML e CSS são inúmeros e merecem ser estudados.

Leiam também as Apostilas de C#: Conceitos Básicos – Versão 1.0, Apostila de SQL: Conceitos Básicos – Versão 1.0 e a Apostila de Estruturas de Repetição em C# - Versão 1.0.

Bons estudos aos leitores!

Home Editora
CNPJ: 39.242.488/0002-80
www.homeeditora.com
contato@homeeditora.com
9198473-5110
Av. Augusto Montenegro, 4120 - Parque Verde, Belém - PA, 66635-110