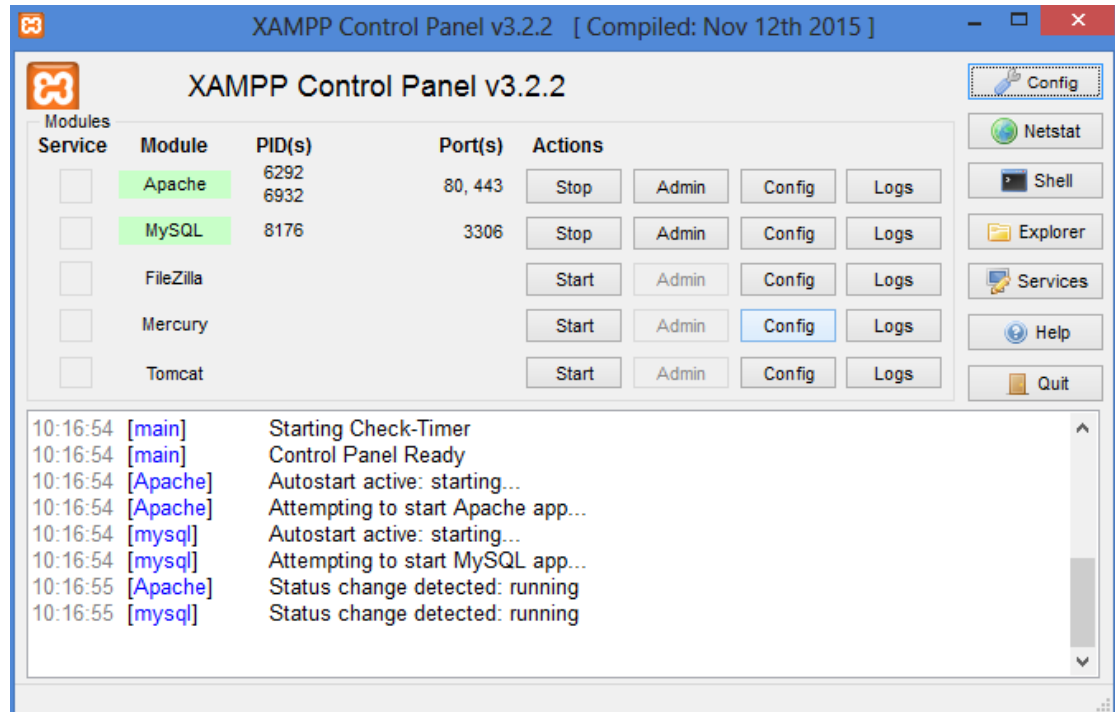
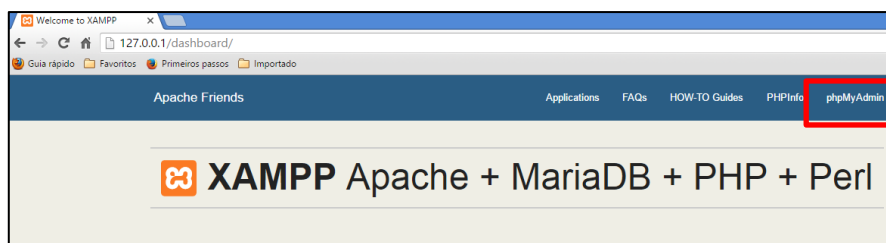


## Criando Banco de Dados com phpMyAdmin

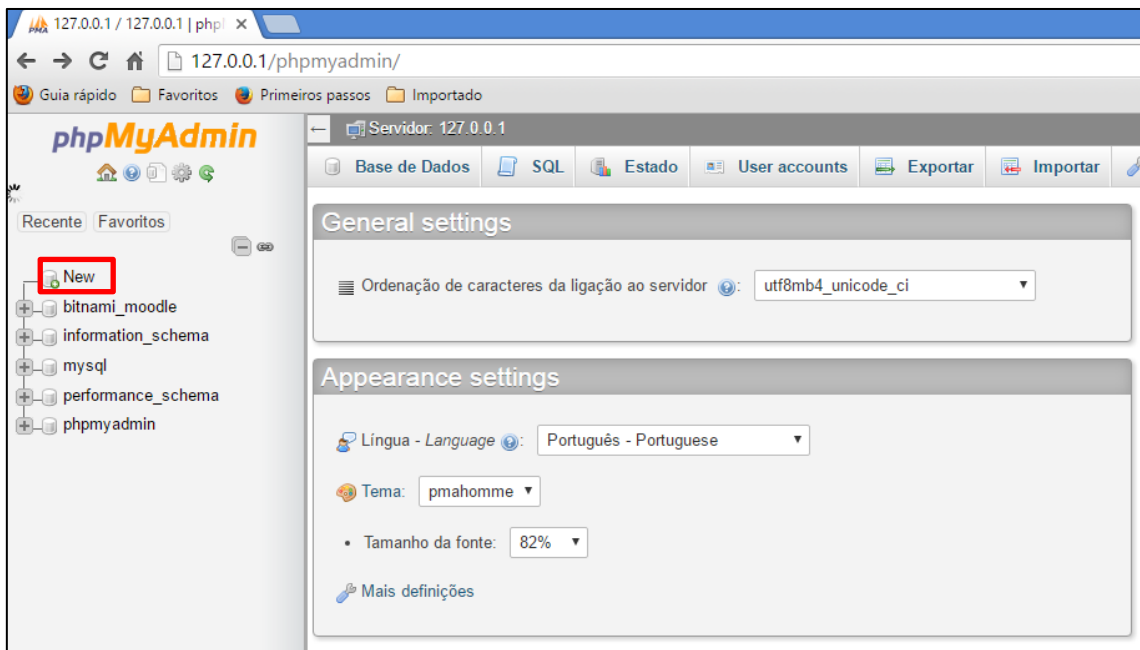
O XAMPP fornece um aplicativo para criar e gerenciar banco de dados MySQL, chamado phpMyAdmin. Para acionar esta funcionalidade, é necessário acionar o Apache e o MySQL, através do painel de controle do Servidor.



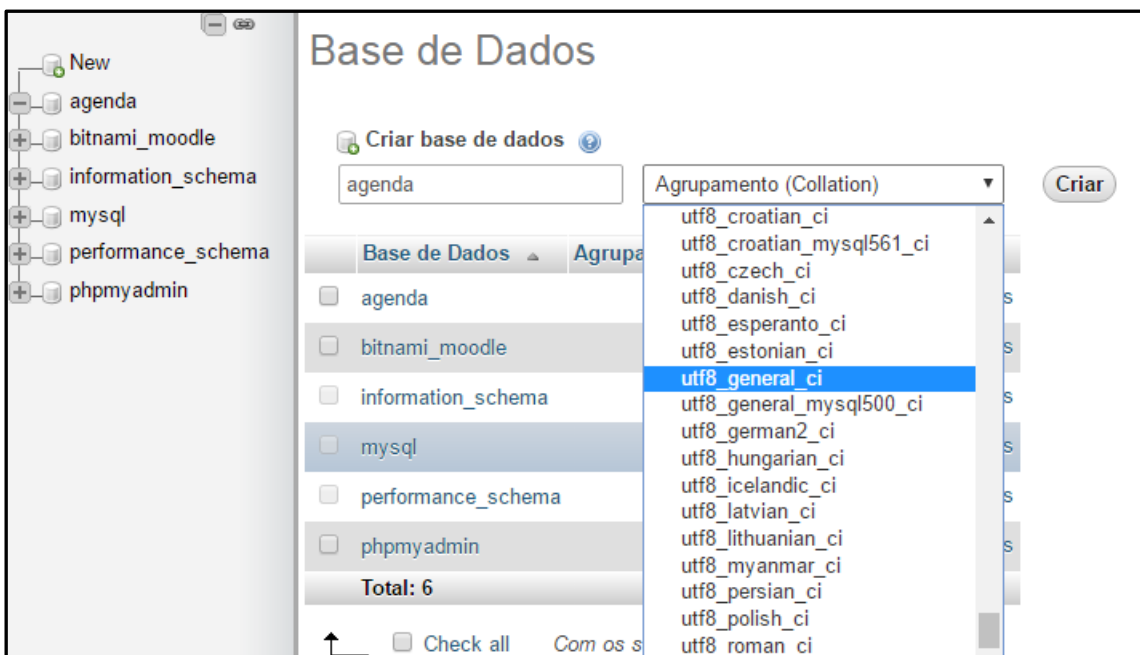
Com os dois serviços ativos, entre no navegador de sua preferência e digite o endereço IP 127.0.0.1. O Xampp executará o arquivo “index.php”, localizado na pasta htdocs. Aparecerá uma página com um menu na parte superior, onde deve ser selecionada a opção phpMyAdmin (última no canto superior direito).



A seguir será mostrada a tela de criação e gerenciamento de banco de dados. Para criar um novo banco, clique na opção “New” do menu vertical localizado na coluna mais à esquerda da tela, conforme imagem abaixo:



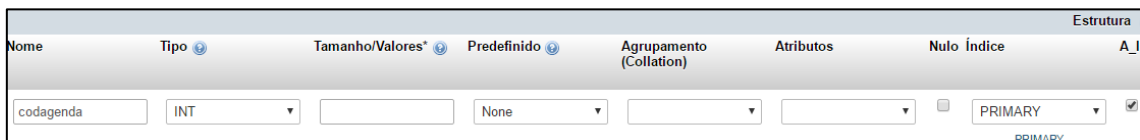
Essa escolha fará aparecer a tela de criação de banco de dados propriamente dita. Deverão ser informados o nome do banco (no caso abaixo, agenda) e o agrupamento, que indica o conjunto de caracteres usados pelo banco. O ideal é o conjunto “utf8\_general\_ci”. Agora, basta clicar no botão “criar” para que o phpMyAdmin envie os comandos necessários à criação do banco para o MySQL.



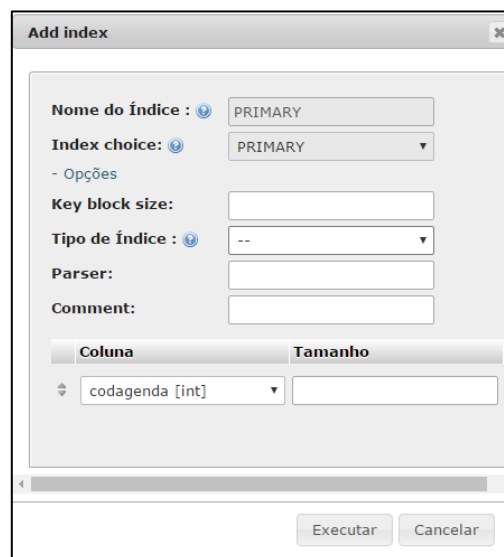
A próxima tela possibilitará a criação de uma tabela. Vale lembrar que a etapa anterior apenas criou o banco, que ainda se encontra sem nenhum conteúdo. Vamos criar uma tabela chamada “agenda”, com sete colunas (campo de dados). Para tanto,

basta informar o nome da tabela e modificar o número de colunas, que por padrão é igual a quatro. Após clicar o botão “executar” a nova tabela será criada, e teremos acesso à tela de cadastramento de campos da tabela.

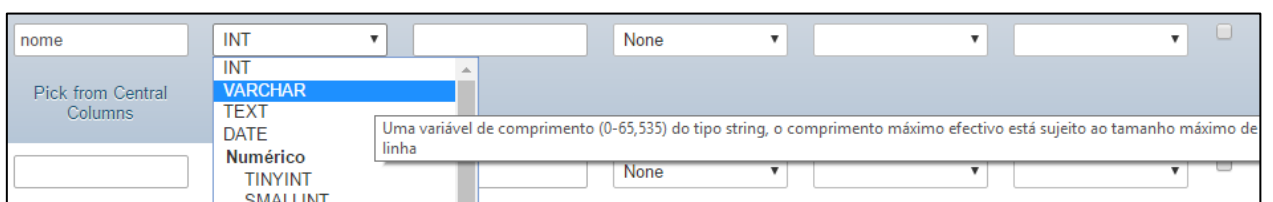
O número de linhas desta tela será igual a sete, uma linha para cada campo da tabela, conforme informado. Primeiramente, vamos criar a chave primária da tabela, que será o campo “codigo”. O nome do campo deve ser informado na primeira coluna, e deverão ser informados também o índice PRIMARY e marcado o *checkbox* A\_I (auto incremento). Isso fará com que este campo seja a chave primária da tabela e que seja automaticamente incrementado pelo MySQL a cada inserção sofrida pelo banco



Após selecionar o campo como PRIMARY aparecerá a seguinte tela, onde deverá ser confirmado o campo como chave primária, clicando no botão executar



Em sequência, será criado o campo nome. O tipo de dados utilizado será o VARCHAR, que é uma string de tamanho variável.



O campo nome também será usado para ordenar os registros da tabela (ordem alfabética). Para agilizar esse procedimento, é criado um índice para o campo.

O índice será do tipo UNIQUE, ou seja, não serão aceitos nomes duplicados na tabela. Observação: um campo do tipo VARCHAR recebe um tamanho máximo, no caso do campo nome, 80 caracteres. Mas o banco de dados só usará o número de caracteres necessários na hora de armazenar o conteúdo<sup>1</sup>.


Os outros campos da tabela serão criados a seguir, conforme tabela a seguir:

CAMPO	TIPO	TAMANHO	ÍNDICE
<b>codigo</b>	INT	Automático	PRIMARY
<b>nome</b>	VARCHAR	80	UNIQUE
<b>endereço</b>	VARCHAR	80	
<b>complemento</b>	VARCHAR	60	
<b>bairro</b>	VARCHAR	60	
<b>cidade</b>	VARCHAR	40	
<b>cep</b>	CHAR	8	


Como o campo “cep” tem tamanho fixo, não é necessário que ele seja do tipo VARCHAR, daí ter sido definido como CHAR (string de tamanho fixo). Após a criação da tabela, será mostrada a tela com a estrutura de campos da tabela

---



<sup>1</sup> Se o campo VARCHAR tiver menos que 255 caracteres, será adicionado um byte (caracter) para conter o tamanho efetivo do campo. Se o campo contiver mais de 255 caracteres, serão usados 2 bytes para armazenar o tamanho do campo. O tamanho máximo de um campo VARCHAR é de 65.535 caracteres. O tamanho máximo de um campo CHAR é de 255 caracteres.




Estrutura da tabela




Relation view

	#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Extra
<input type="checkbox"/>	1	codigo 	int(11)			Não	None	AUTO_INCREMENT
<input type="checkbox"/>	2	nome 	varchar(80)			Não	None	
<input type="checkbox"/>	3	endereco	varchar(80)			Não	None	
<input type="checkbox"/>	4	complemento	varchar(60)			Não	None	
<input type="checkbox"/>	5	bairro	varchar(60)			Não	None	
<input type="checkbox"/>	6	cidade	varchar(40)			Não	None	
<input type="checkbox"/>	7	cep	char(8)			Não	None	




☐ Check all


Com os seleccionados:




Procurar




Muda



Elimina



Primária



Único

Através dessa tela é possível gerenciar os campos, inclusive incluindo novos campos ou excluindo campos já inseridos, clicando nos links do lado direito da tela.

Para finalizar o processo de criação da tabela, é possível exportar os comandos de Data Definition Language (DDL) usados para criar a tabela agenda. Vale lembrar que são esses os comandos executados pelo banco de dados para criar a tabela, o phpMyAdmin é apenas uma interface entre o programador e o banco.

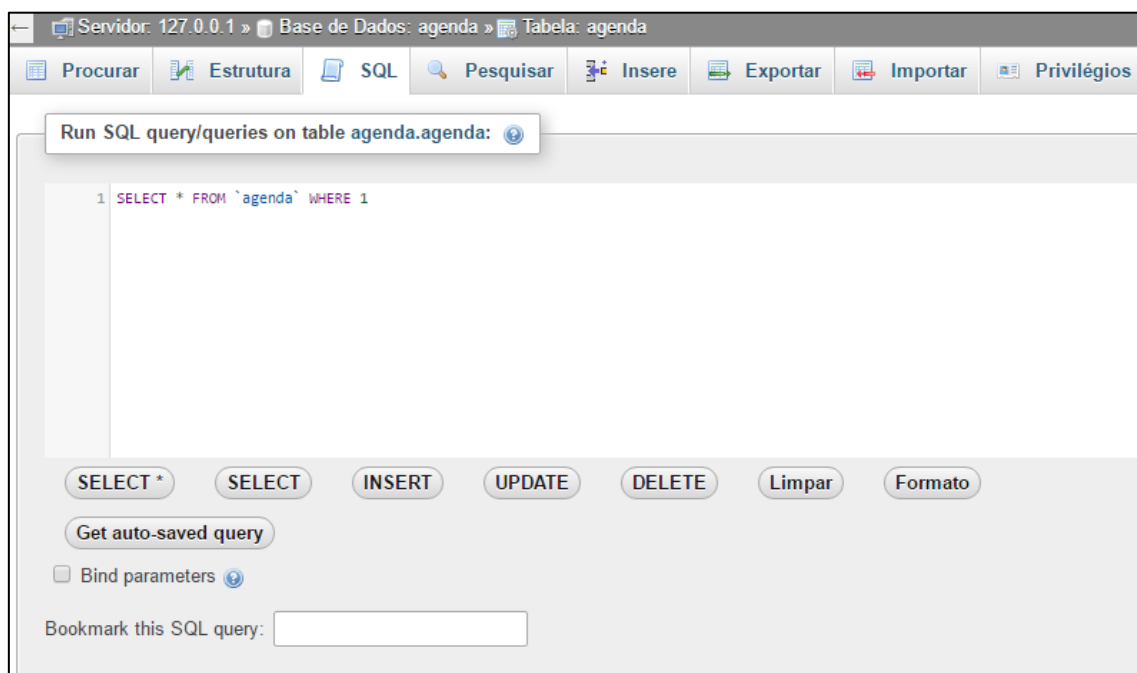
Para ter acesso a estes comandos, deve-se clicar no botão “Exportar”, e marcar as seguintes opções:

- Export method – personalizado;
- Saída – Ver resultado como texto;
- Opções específicas do formato – marcar “estrutura”;
- Object creation options- marcar as opções:
  - Add drop table / trigger statement
  - Add create table statement
  - If not exists...
  - Auto-increment value
  - Enclose table and column names with backquotes

Após executar o comando, será gerado uma lista com os comandos de SQL-DDL necessários para criar a tabela agenda:

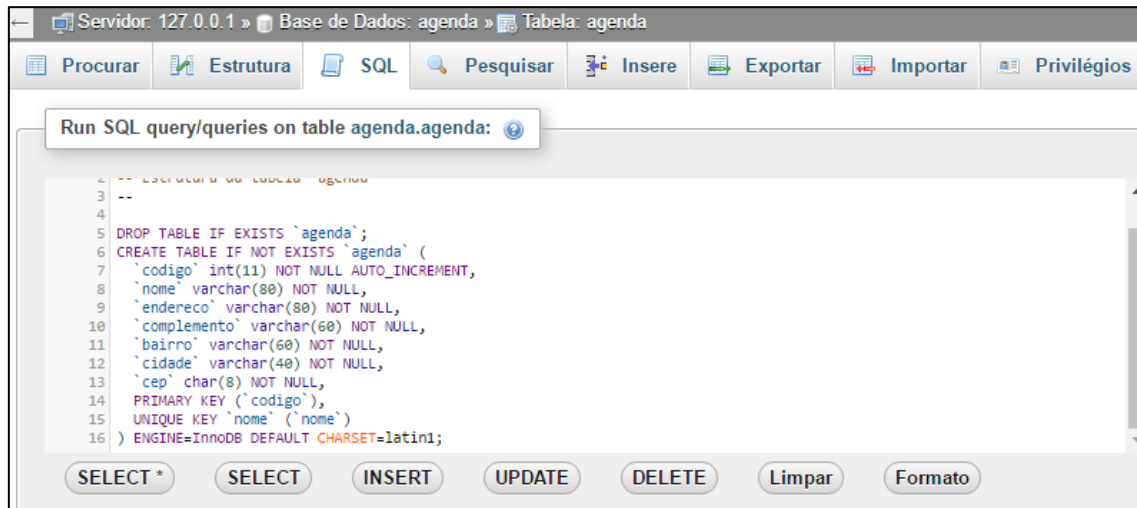
```
--
-- Estrutura da tabela `agenda`
--
DROP TABLE IF EXISTS `agenda`;
CREATE TABLE IF NOT EXISTS `agenda` (
  `codigo` int(11) NOT NULL AUTO_INCREMENT,
  `nome` varchar(80) NOT NULL,
  `endereco` varchar(80) NOT NULL,
  `complemento` varchar(60) NOT NULL,
  `bairro` varchar(60) NOT NULL,
  `cidade` varchar(40) NOT NULL,
  `cep` char(8) NOT NULL,
  PRIMARY KEY (`codigo`),
  UNIQUE KEY `nome` (`nome`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Esses comandos podem ser copiados para um editor de texto, como o Sublime, a fim de que sejam salvos em um arquivo com extensão “sql” (por exemplo, agenda.sql). Dessa forma, toda vez que for necessário criar essa tabela, bastará executar tal arquivo dentro do phpMyAdmin para que a tabela agenda seja recriada VAZIA dentro do banco de dados. Essa opção é muito útil durante o desenvolvimento do sistema. Para executar uma sequência de comandos SQL dentro do phpMyAdmin, selecione a tabela desejada e depois clique no botão SQL do menu da tabela, o que fará com que a seguinte tela seja exibida:

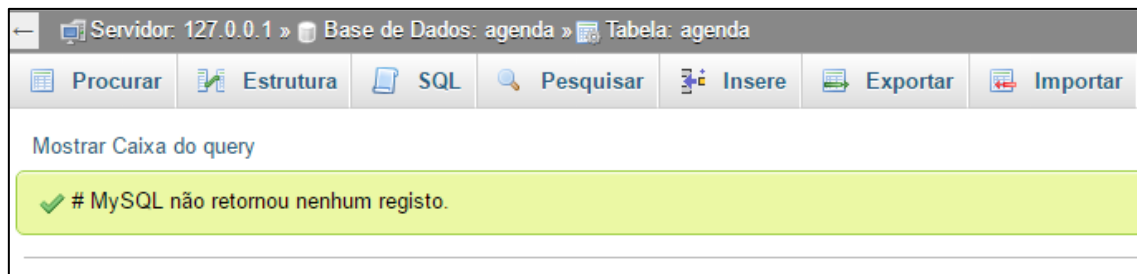


A caixa de texto no meio da tela pode ser usada para enviar qualquer comando SQL para o banco de dados. Para executar o script de criação da tabela agenda, basta

copiar os comandos (Ctrl+C/Ctrl+V) que foram gravados no arquivo mencionado acima (agenda.sql). O conteúdo da tela será o seguinte:

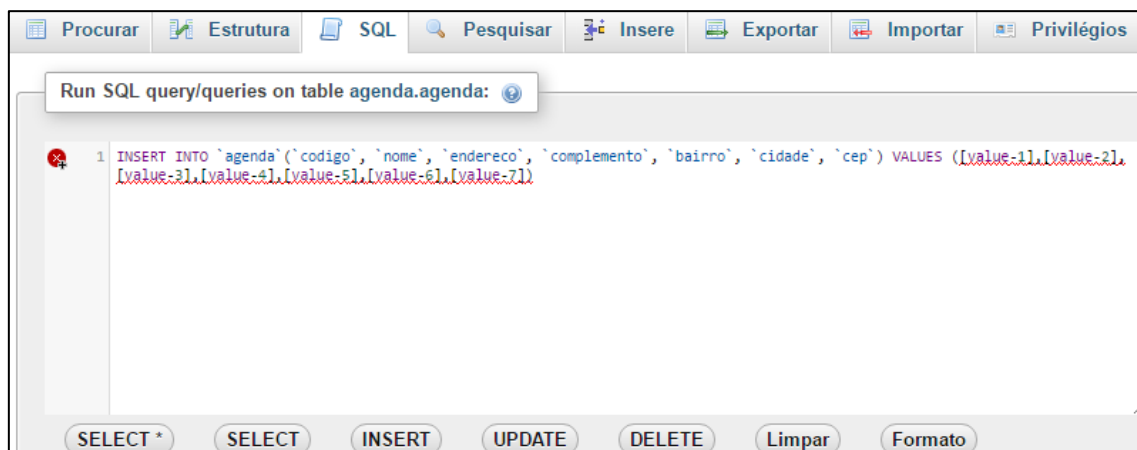


Para executar essa sequência de comandos, basta clicar no botão “Executar”, localizado no canto inferior direito da tela. Se os comandos SQL forem executados sem erro, aparecerá uma mensagem de sucesso na tela:



Essa tela também pode ser usada para incluir/alterar/consultar/excluir dados da tabela, através dos comandos SQL Insert, Select, Update e Delete.

Para inserir um registro na tabela, clique no botão INSERT, para obter o seguinte comando:



Nesse ponto, o comando está incompleto, daí o “x” no círculo vermelho. Siga os seguintes passos antes de executar o comando de inclusão no banco:

- Dentro do primeiro parêntese, apague o campo ‘codigo’, já que ele foi definido como auto incrementável, ou seja, o próprio MS SQL vai gerar o seu valor;
- O parâmetro VALUES informa entre parênteses o conteúdo de cada campo. Como todos são do tipo string (VARCHAR ou CHAR), devem vir entre aspas simples. Os valores devem ser informados na mesma ordem em que o nome dos campos aparecem. Por exemplo:

```
1 INSERT INTO `agenda`(`nome`, `endereço`, `complemento`, `bairro`, `cidade`, `cep`) VALUES ('Nelson da Silva', 'Rua A', '100', 'Centro', 'Valença', '27600000')
```

Agora o comando está sintaticamente correto, por isso o sinal de erro desapareceu. Para inserir esse registro na tabela, clique novamente no botão “Executar”.

Após a inclusão, será mostrada uma mensagem de sucesso, e logo abaixo o comando SQL que foi executado. Para incluir mais um registro, clique no link “Edita”, para tornar a mostra o comando e incluir novo registro.

Para consultar os registros incluídos, digite o comando “SELECT \* FROM `agenda` ORDER BY nome” e clique em “Executar”. Os registros serão exibidos da seguinte forma:

✓ A mostrar registros de 0 - 3 (4 total, A consulta demorou 0.0005 segundos.) [nome: MARIA ANTONIA... - PEDRO ANTONIO]

```
SELECT * FROM `agenda` ORDER BY nome
```

☐ Mostrar tudo | Número de registros: 25 | Filtrar registros:

Ordenar por chave: Nenhum

+ Opções

	codigo	nome	endereço	complemento	bairro	cidade	cep
<input type="checkbox"/>	3	Maria Antonia	Rua B	300	Centro	Pinheiral	27197000
<input type="checkbox"/>	4	Maria Luiza	Rua B	400	Centro	Pinheiral	27197000
<input type="checkbox"/>	1	Nelson da Silva	Rua A	100	Centro	Valença	27600000
<input type="checkbox"/>	2	Pedro Antonio	Rua B	200	Centro	Pinheiral	27197000

☐ Check all | Com os seleccionados:

☐ Mostrar tudo | Número de registros: 25 | Filtrar registros:



O parâmetro “order by nome” faz com que a tabela seja exibida em ordem alfabética de nome, e não na ordem da chave primária.

**Criando uma tabela com telefones** – Para cada registro na tabela “agenda” podemos ter diversos números de telefones, tanto fixos como celulares. Para cadastrar esses novos registros, usaremos uma nova tabela, chamada “telefone”, com a seguinte estrutura:







CAMPO	TIPO	TAMANHO	ÍNDICE
codfone	INT	Automático	PRIMARY
codend	INT	Automático	INDEX
telefone	CHAR	11	
tipofone	INT	Automático	

Essa tabela tem um relacionamento do tipo “um para n” com a tabela “agenda”, ou seja, para cada registro da tabela “agenda” podem existir nenhum, um ou mais registros na tabela “telefone”. Além disso, os registros da tabela “telefone” são relacionados com o registro na tabela “agenda” através do campo “codend”, conforme visto acima.

Esse campo que relaciona duas tabelas é conhecido como **chave estrangeira**:

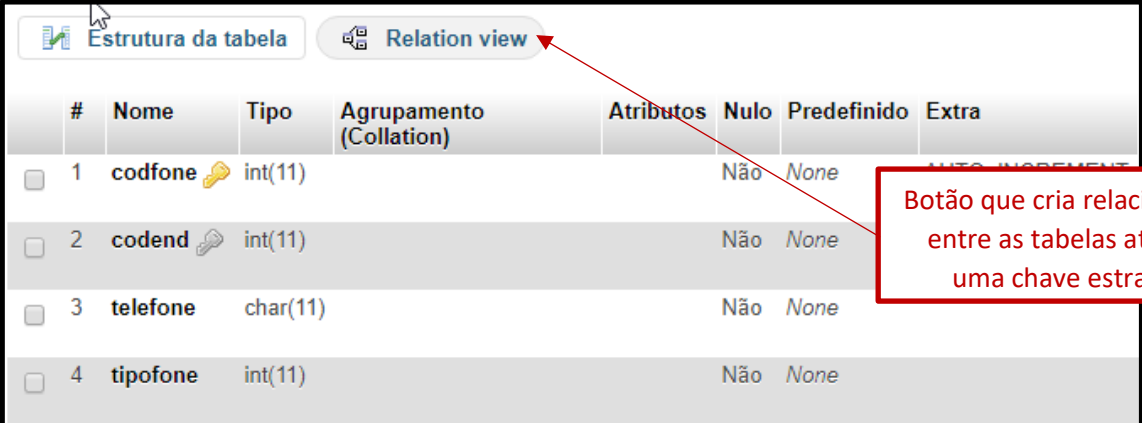
Chave estrangeira (foreign key) é o campo que estabelece o relacionamento entre duas tabelas. Assim, uma coluna (campo) corresponde à mesma coluna que é a chave primária de outra tabela. Dessa forma, deve-se especificar na tabela que contém a chave estrangeira quais são essas colunas e à qual tabela está relacionada.<sup>2</sup>

No caso, a chave primária da tabela “agenda” (campo “código”) se corresponde com o campo “codend” da tabela “telefone”, fazendo a ligação entre as duas tabelas. Caso queira, os dois campos podem ter o mesmo nome. Contudo, para que essa ligação funcione, é necessário criar um índice para o campo que será chave estrangeira antes de configurá-lo como tal. Esse índice deve ser do tipo INDEX (e não UNIQUE, como o que criamos para o campo “nome” na tabela agenda). O nome do índice será “codend”, conforme figura a seguir:

Índices									
Acções	Nome da chave	Tipo	Único	Pacote	Coluna	Quantidade	Agrupamento (Collation)	Nulo	Comentário
  	PRIMARY	BTREE	Sim	Não	codfone	0	A	Não	
  	codend	BTREE	Não	Não	codend	0	A	Não	

<sup>2</sup> Site da DEVMEDIA. Disponível em <<https://www.devmedia.com.br/breve-conceito-de-foreign-key/17426>>. Acesso em 26 abr. 2018.

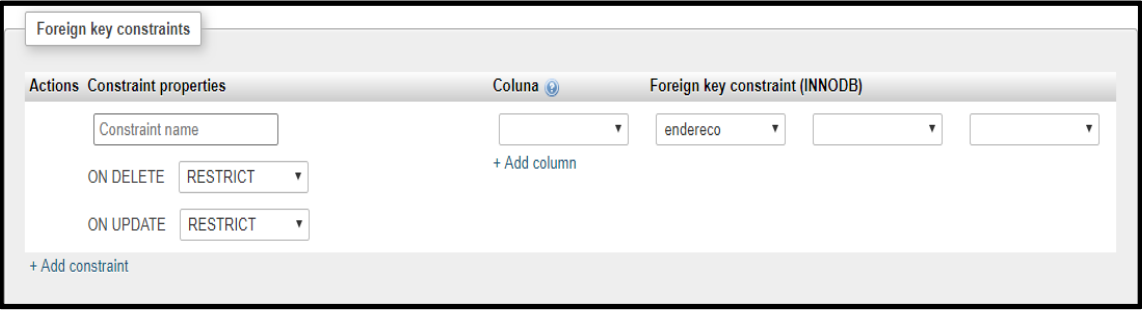
A estrutura da tabela “telefone” será a seguinte:



#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Extra
1	codfone	int(11)			Não	None	PRIMARY KEY
2	codend	int(11)			Não	None	
3	telefone	char(11)			Não	None	
4	tipofone	int(11)			Não	None	

Botão que cria relacionamento entre as tabelas através de uma chave estrangeira

O próximo passo será clicar no botão “**Relation view**”, visto na imagem acima, para criar o relacionamento entre as duas tabelas. Após, aparecerá uma área na tela chamada “**Foreign key constraints**”. Essa área é usada para criar a restrição (constraint) que será necessária para criar a chave estrangeira na tabela telefone:



Foreign key constraints

Actions Constraint properties

Constraint name

ON DELETE RESTRICT

ON UPDATE RESTRICT

+ Add constraint

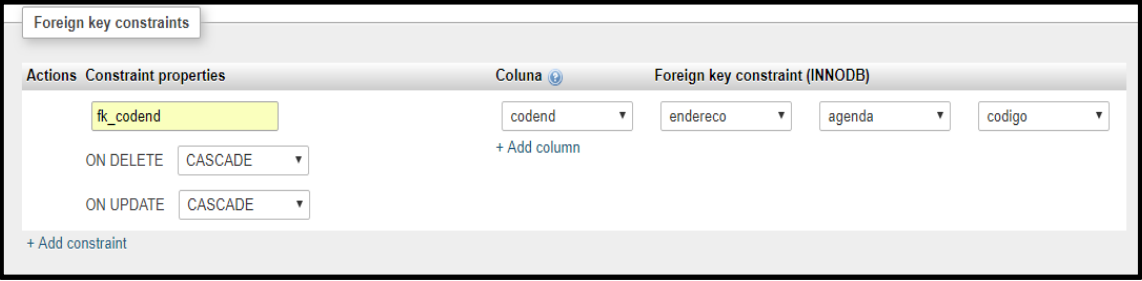
Coluna

Foreign key constraint (INNODB)

endereço

+ Add column

Na caixa de texto “**Constraint name**” deverá ser informado o nome da *constraint* associada à chave estrangeira, que será ‘fk\_codend’. Os outros valores devem ser preenchidos da seguinte maneira:



Foreign key constraints

Actions Constraint properties

fk\_codend

ON DELETE CASCADE

ON UPDATE CASCADE

+ Add constraint

Coluna

Foreign key constraint (INNODB)

codend

+ Add column

1) ON DELETE – CASCADE: quando ocorrer uma exclusão de registro na tabela “agenda”, todos os registros relacionados ao excluído que existirem na tabela “telefone” serão AUTOMATICAMENTE excluídos;

2) ON UPDATE – CASCADE: quando a chave primária de um registro da tabela “agenda” for alterada, os registros a ela relacionados na tabela “telefone” terão o campo “codend” alterado AUTOMATICAMENTE;

3) Combo box “codend”: informa o campo da tabela “telefone” que será usado como chave estrangeira. Somente aparecem os campos com índices previamente criados;

4) Combo box com “endereço” selecionado: indica o nome do banco de dados onde a tabela principal do relacionamento está inserida. Nesse exemplo, as duas tabelas pertencem ao banco de dados “endereço” (O seu banco de dados pode ter outro nome);

5) Combo box com “agenda” selecionado: indica qual a tabela principal do relacionamento (onde a chave primária está localizada);

6) Combo box com “código” selecionado: indica qual o campo da tabela “agenda” será usado no relacionamento com a chave estrangeira de “telefone”. No caso, escolhemos a chave primária da tabela “agenda”. De novo, só aparecem os campos que tenham índices previamente criados.

Para confirmar, basta clicar no botão “Guarda”. Após, é interessante gerar os comandos SQL de criação da tabela (ver acima). O resultado será o seguinte:




```
--  
-- Estrutura da tabela `telefone`  
--  
DROP TABLE IF EXISTS `telefone`;  
CREATE TABLE `telefone` (  
  `codfone` int(11) NOT NULL,  
  `codend` int(11) NOT NULL,  
  `telefone` char(11) NOT NULL,  
  `tipofone` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
--  
-- Indexes for table `telefone`  
--  
ALTER TABLE `telefone`  
  ADD PRIMARY KEY (`codfone`),  
  ADD KEY `fk_codend` (`codend`);  
  
--  
-- AUTO_INCREMENT for table `telefone`  
--  
ALTER TABLE `telefone`  
  MODIFY `codfone` int(11) NOT NULL AUTO_INCREMENT;  
--  
-- Constraints for dumped tables  
--  
--  
-- Limitadores para a tabela `telefone`  
--  
ALTER TABLE `telefone`  
  ADD CONSTRAINT `fk_codend` FOREIGN KEY (`codend`) REFERENCES `agenda` (`codigo`) ON DELETE CASCADE ON UPDATE CASCADE;
```

O último comando cria a chave estrangeira e a exclusão e alteração automáticas.

**Criando uma tabela com tipos de telefone** – Na tabela “telefone” existe o campo “tipofone”, onde é guardado o código de tipo de telefone. Esse código deverá ser a chave primária de uma tabela onde constem todos os tipos de telefones necessários, como “celular”, “casa”, “comercial”, etc. Essa tabela se chamará “fonetipo”, e sua estrutura será a seguinte:

Estrutura da tabela		Relation view					
#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Extra
1	codtipofone	int(11)			Não	None	AUTO_INCREMENT
2	nometipofone	varchar(20)			Não	None	

O campo “codtipofone” será a chave primária auto-incrementada pelo MySQL, e o campo “nometipofone” terá um índice do tipo UNIQUE, conforme figura:

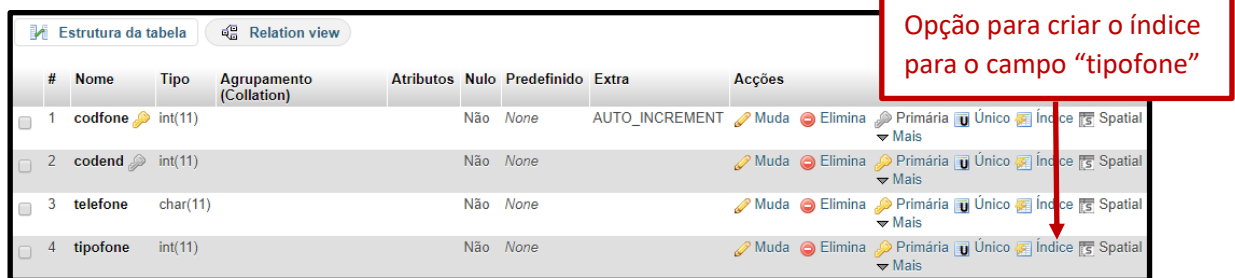
Índices						
Acções	Nome da chave	Tipo	Único	Pacote	Coluna	
 <a href="#">Edita</a>  <a href="#">Elimina</a>	PRIMARY	BTREE	Sim	Não	codtipofone	
 <a href="#">Edita</a>  <a href="#">Elimina</a>	nometipofone	BTREE	Sim	Não	nometipofone	

A tabela “fonetipo” deverá ter seus registros cadastrados antes de começar o cadastramento de telefones na tabela “telefone”. Sempre que um registro for inserido na tabela “telefone”, deverá ser informado no campo “tipofone” um valor já cadastrado de “codtipofone” na tabela “fonetipo”. Simetricamente, não deverá ser permitido excluir um registro da tabela “fonetipo” caso sua chave primária já tenha sido utilizada em pelo menos um registro da tabela “telefone”.

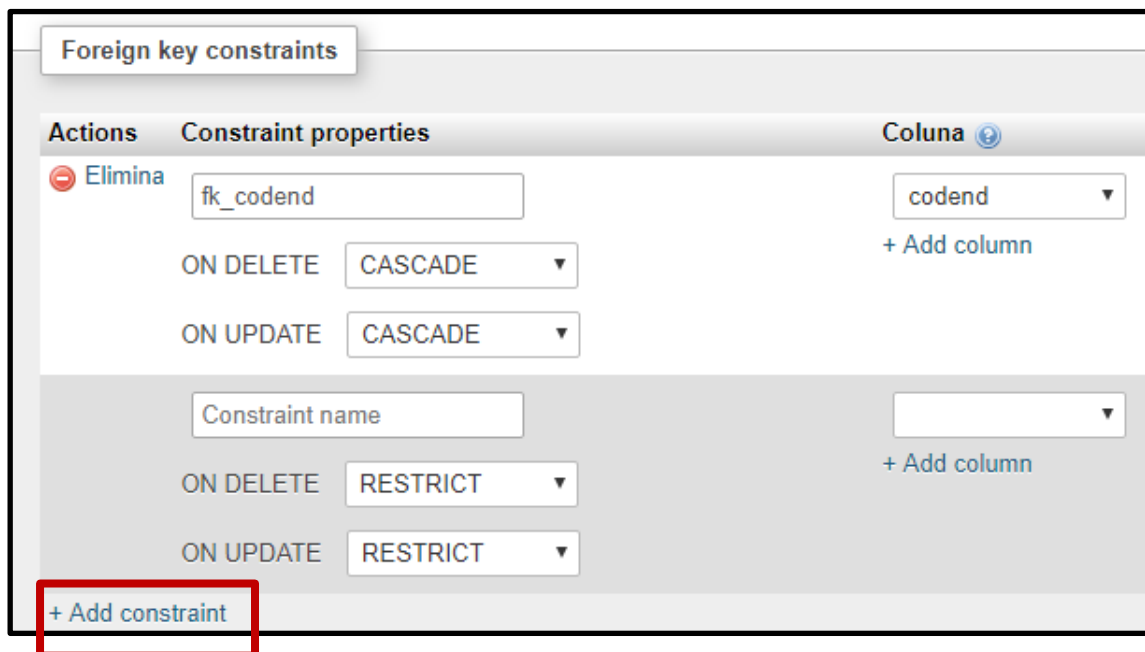
O relacionamento descrito acima é chamado de “Pai-filho”, onde a tabela pai é “fonetipo” e a tabela filho é “telefone”. Nessa situação, o banco de dados só permite a exclusão de um registro “pai” se ele não tiver “filhos”. Caso contrário, será enviada uma mensagem de erro MySQL nº 1451, como a descrita a seguir:

Erro
Comando SQL:
DELETE FROM `fonetipo` WHERE codtipofone=1
Mensagens do MySQL :
#1451 - Cannot delete or update a parent row: a foreign key constraint fails (`endereco`.`telefone`, CONSTRAINT `fk_tipofone` FOREIGN KEY (`tipofone`) REFERENCES `fonetipo` (`codtipofone`))

Para esse controle funcionar, novamente vamos precisar de uma chave estrangeira na tabela “telefone”, nesse caso usaremos o campo “tipofone”. Inicialmente, criamos um índice do tipo INDEX para o campo “tipofone”, clicando na opção “Índice” do phpMyAdmin:



Após a criação, aparecerá o ícone da chave ao lado do nome do campo, indicando que o índice foi criado. O próximo passo é clicar no botão “**Relation view**” para definir o campo como chave estrangeira. Logo após, é necessário clicar no link “**+ Add Constraint**”, que aparece abaixo do canto inferior esquerdo da janela “**Foreign key constraints**”:



Agora, devemos informar a relação de chave estrangeira entre o campo “tipofone” da tabela filho “telefone” e o campo “codtipofone” na tabela pai “fonetipo”:



A principal mudança nessa definição de chave estrangeira em relação a que foi feita anteriormente é mudança dos valores dos combo boxes ON DELETE e ON

UPDATE, que passam a ter o valor RESTRICT. Esse valor faz com que o MySQL verifique a existência de registros “filhos” no momento da exclusão de um registro “pai”. Sempre que existir pelo menos um “filho” a mensagem de erro nº 1451 será enviada como resposta ao SQL de exclusão (comando DELETE).

Após essa inclusão de chave estrangeira, os comandos de criação da tabela “telefone” passam a ser os seguintes:

```
DROP TABLE IF EXISTS `telefone`;
CREATE TABLE `telefone` (
  `codfone` int(11) NOT NULL,
  `codend` int(11) NOT NULL,
  `telefone` char(11) NOT NULL,
  `tipofone` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Indexes for table `telefone`
--
ALTER TABLE `telefone`
  ADD PRIMARY KEY (`codfone`),
  ADD KEY `fk_codend` (`codend`),
  ADD KEY `tipofone` (`tipofone`);

--
-- AUTO_INCREMENT for table `telefone`
--
ALTER TABLE `telefone`
  MODIFY `codfone` int(11) NOT NULL AUTO_INCREMENT;

--
-- Limitadores para a tabela `telefone`
--
ALTER TABLE `telefone`
  ADD CONSTRAINT `fk_codend` FOREIGN KEY (`codend`) REFERENCES `agenda` (`codigo`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `fk_tipofone` FOREIGN KEY (`tipofone`) REFERENCES `fonetipo` (`codtipofone`);
```