

[devmedia.com.br](https://www.devmedia.com.br)

Primeiros Passos no MySQL: Como criar um Banco e seus Comandos

Ricardo Arrigoni

9-12 minutos

Aprendendo a trabalhar com MySQL

Olá pessoal, no artigo de hoje vamos aprender a [trabalhar com o MySQL](#), para um melhor entendimento, iremos aprender como fazer pelo terminal do **MySQL**, utilizando de código puro SQL.

Antes de qualquer coisa, vamos fazer o [download do MySQL](#) e vamos entender o que ele é.

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a [linguagem SQL](#) (Structured Query Language ou Linguagem de Consulta Estruturada) como interface.

Saiba mais: [MySQL](#)

O MySQL atualmente é um dos maiores SGBD's do mundo, com mais de 10 milhões de instalações e vem sendo usado inclusive em projetos de grande porte em grandes empresas.

Entre essas empresas, estão: NASA, Friendster, Banco

Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S. Army, U.S. Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems, Google, entre outros.

Não é a toa que **o MySQL cresceu tanto assim nos últimos anos**, abaixo segue uma lista de coisas que fizeram esse SGBD crescer tanto e continuar crescendo cada vez mais.

- Portabilidade (suporta praticamente qualquer plataforma atual);
- Compatibilidade (existem drivers ODBC, [JDBC](#) e .NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, C#, Visual Basic, Python, Perl, PHP, ASP e Ruby
- Excelente desempenho e estabilidade;
- Pouco exigente quanto a recursos de hardware;
- Facilidade de uso;
- É um Software Livre com base na GPL (entretanto, se o programa que acessar o Mysql não for GPL, uma licença comercial deverá ser adquirida);
- Contempla a utilização de vários Storage Engines como MyISAM, InnoDB, Falcon, BDB, Archive, Federated, CSV, Solid...
- Suporta controle transacional;
- Suporta Triggers;
- Suporta Cursors (Non-Scrollable e Non-Updatable);
- Suporta Stored Procedures e Functions;
- Replicação facilmente configurável;

- Interfaces gráficas (MySQL Toolkit) de fácil utilização cedidos pela MySQL Inc.

Agora vamos entender um pouco a sintaxe do mysql. Primeiro vamos ver como fazemos para criar um banco de dados pelo terminal do mysql.

Criando o banco de dados

Listagem 1: Criando banco de dados

```
CREATE DATABASE bancodeteste;
```

Como podemos ver, a sintaxe é bem intuitiva e com um leve conhecimento da língua inglesa fica mais fácil ainda de entender.

Após criar o bando de dados, precisamos avisar ao mysql que vamos usá-lo, para isso basta escrevermos.

Listagem 2: Usando o banco de dados

Feito isso o nosso banco de dados está criado, faltando apenas criar a nossa tabela. Para isso vamos usar o comando CREATE TABLE.

Criando tabelas

Listagem 3: Criando tabelas

```
CREATE TABLE fornecedores(  
  codigo int(4) AUTO_INCREMENT,  
  nome varchar(30) NOT NULL,  
  email varchar(50),  
  PRIMARY KEY (codigo)  
);
```

Como podemos ver, nossa tabela está criada.

```
Database changed  
mysql> CREATE TABLE fornecedores(codigo int(4) auto_increment, nome varchar(30)
```

```
NOT NULL, email varchar(50), PRIMARY KEY(codigo));
Query OK, 0 rows affected (0.14 sec)

mysql> _
```

Figura 1: Tabela criada.

Agora vamos às explicações do que se tratam os comandos:

- **AUTO_INCREMENT** pode ser utilizado para automatizar um código que sirva de chave primária de uma tabela.
- **PRIMARY KEY** define a chave primária da tabela, isto é, o campo que serve como chave da tabela e que não pode ser repetido.
- **NOT NULL** define que um determinado campo seja de preenchimento obrigatório.

Agora já temos um banco de dados e uma tabela criada, com isso é possível manipular os dados do banco de dados.

INSERT

Agora vamos inserir alguma informação na nossa tabela, para isso vamos usar o comando **INSERT**, é bem simples também.

Listagem 4: Inserindo Dados

```
INSERT INTO fornecedores(codigo, nome, email)
VALUES (null, "Ricardo",
"ricoarrigoni@gmail.com") ;
INSERT INTO fornecedores(codigo, nome, email)
VALUES (null, "João", "joao@gmail.com") ;
INSERT INTO fornecedores(codigo, nome, email)
VALUES (null, "Maria", "maria@gmail.com") ;
```

```
Database changed
mysql> CREATE TABLE fornecedores(codigo int(4) auto_increment, nome varchar(30)
NOT NULL, email varchar(50), PRIMARY KEY(codigo));
Query OK, 0 rows affected (0.14 sec)
```

```
mysql> INSERT INTO fornecedores(codigo, nome, email) VALUES (null, "Ricardo", "ricoarrigoni@gmail.com");
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO fornecedores(codigo, nome, email) VALUES (null, "João", "joao@gmail.com");
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO fornecedores(codigo, nome, email) VALUES (null, "Maria", "maria@gmail.com");
Query OK, 1 row affected (0.06 sec)

mysql> _
```

Figura 2: Dados inseridos na tabela.

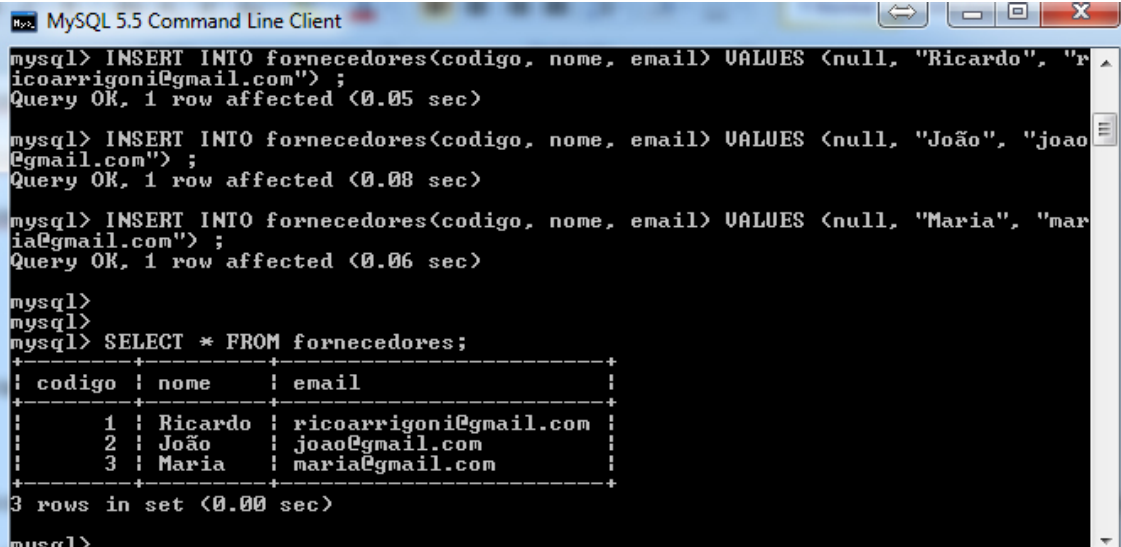
SELECT

Agora que nossa tabela está com alguns registros inseridos nela, nós vamos usar o comando SELECT pra poder selecionar e buscar esses registros. Continuaremos no mesmo terminal e iremos digitar o seguinte código:

Listagem 5: Usando o comando SELECT

```
SELECT * FROM fornecedores;
```

O Resultado desse SELECT nós vemos na Figura 5.



```
mysql> INSERT INTO fornecedores(codigo, nome, email) VALUES (null, "Ricardo", "ricoarrigoni@gmail.com");
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO fornecedores(codigo, nome, email) VALUES (null, "João", "joao@gmail.com");
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO fornecedores(codigo, nome, email) VALUES (null, "Maria", "maria@gmail.com");
Query OK, 1 row affected (0.06 sec)

mysql>
mysql>
mysql> SELECT * FROM fornecedores;
+----+-----+-----+
| codigo | nome  | email                |
+----+-----+-----+
| 1      | Ricardo | ricoarrigoni@gmail.com |
| 2      | João   | joao@gmail.com        |
| 3      | Maria  | maria@gmail.com       |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Figura 5: Resultado do SELECT global.

Como podemos perceber, todos os registros da tabela foram retornados. Isso se deu porque o uso do SELECT *

faz com que a consulta retorne todos os valores da tabela.

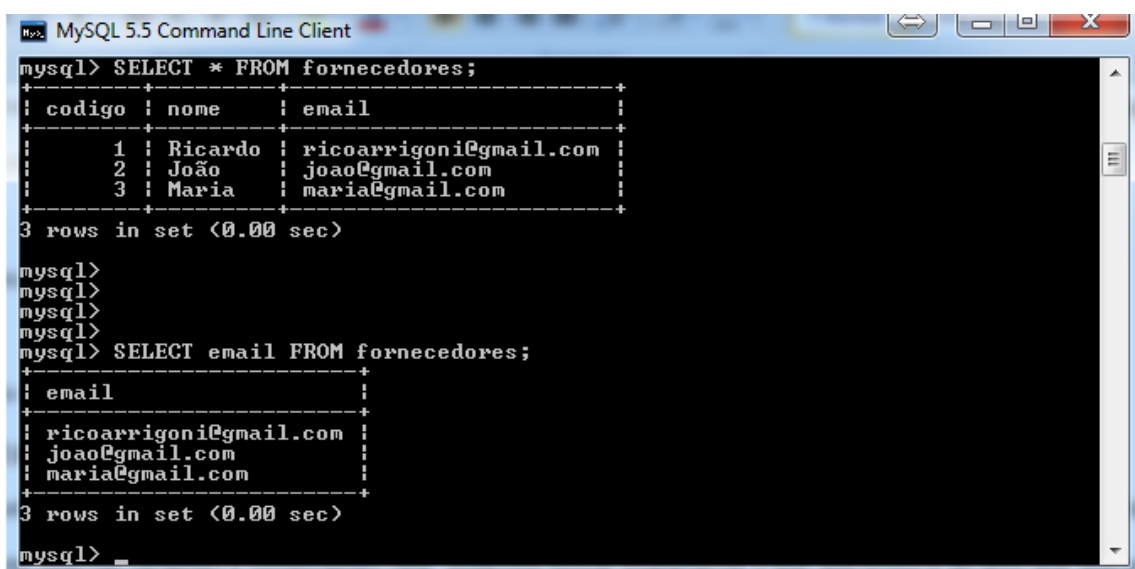
Mas o comando SELECT permite diversas variações e combinações nele, podemos buscar exatamente o que queremos e do jeito que queremos, por exemplo.

Se quisermos buscar apenas o e-mail do fornecedor nós podemos, basta fazer da seguinte forma:

Listagem 6: Usando o SELECT específico

```
SELECT email FROM fornecedores;
```

Nesse caso ele irá exibir somente o que está no campo email do registro.



```
mysql> SELECT * FROM fornecedores;
+----+-----+-----+
| codigo | nome  | email                |
+----+-----+-----+
| 1      | Ricardo | ricoarrigoni@gmail.com |
| 2      | João   | joao@gmail.com        |
| 3      | Maria  | maria@gmail.com       |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql>
mysql>
mysql>
mysql> SELECT email FROM fornecedores;
+-----+
| email                |
+-----+
| ricoarrigoni@gmail.com |
| joao@gmail.com        |
| maria@gmail.com       |
+-----+
3 rows in set (0.00 sec)

mysql> _
```

Figura 3: SELECT específico.

ORDER BY

Se em nossa consulta sql nós quisermos que os registros retornados venham ordenados, nós podemos usar o comando ORDER BY, basta dizer pelo que você quer ordenar que ele traz o registro ordenado.

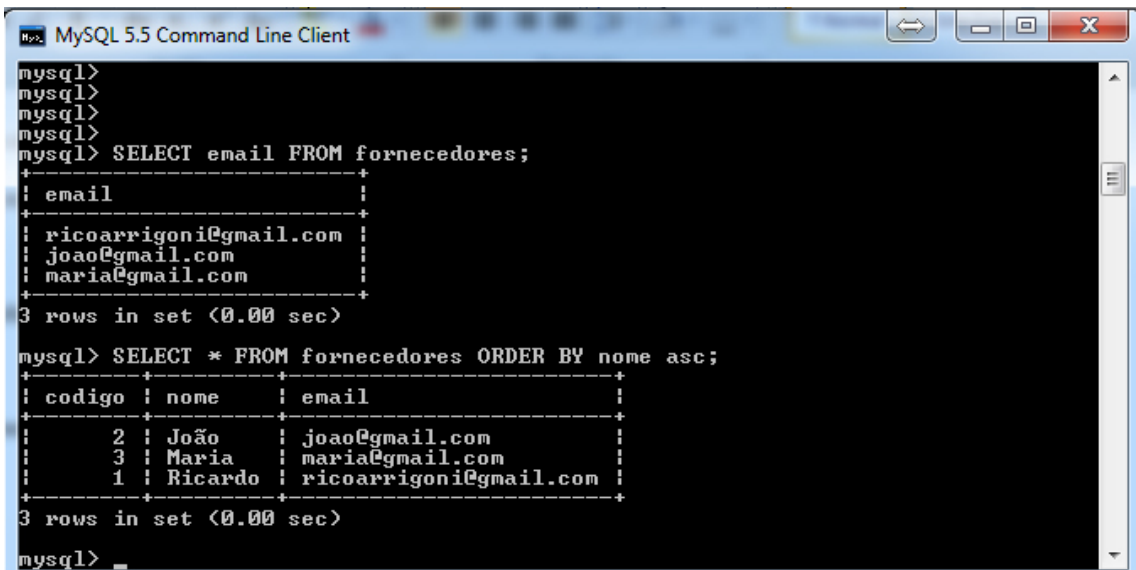
Listagem 7: SELECT usando ORDER BY

```
SELECT * FROM fornecedores ORDER BY nome asc;
```

Dessa forma, os registros retornados virão ordenados pelo campo nome em ordem alfabética. Mas como eu sei que virá em ordem alfabética?

Quando eu uso o termo “asc” eu digo que quero em ordem ascendente, se eu quiser fazer um SELECT que venha em ordem descendente, basta utilizarmos o “desc”.

O resultado da nossa busca pode ser visto aqui:



```

mysql>
mysql>
mysql>
mysql>
mysql> SELECT email FROM fornecedores;
+-----+
| email |
+-----+
| ricoarrigoni@gmail.com |
| joao@gmail.com |
| maria@gmail.com |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM fornecedores ORDER BY nome asc;
+-----+-----+-----+
| codigo | nome | email |
+-----+-----+-----+
| 2 | João | joao@gmail.com |
| 3 | Maria | maria@gmail.com |
| 1 | Ricardo | ricoarrigoni@gmail.com |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

Figura 4: SELECT ordenado por ordem alfabética.

UPDATE

Para alterar um registro, usamos o comando UPDATE, com ele é possível editar os campos de sua tabela e colocar outro valor neles.

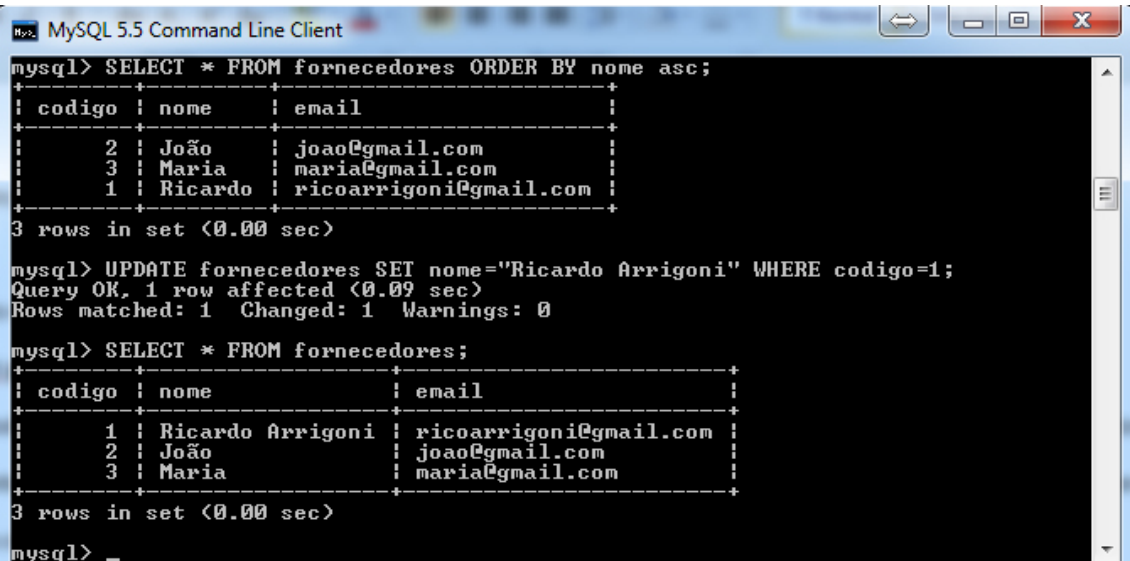
Listagem 8: Editando registros

```
UPDATE fornecedores SET nome="Ricardo Arrigoni"
WHERE codigo=1;
```

Após esse comando, o nome “Ricardo” será alterado para “Ricardo Arrigoni”, vamos novamente dar um SELECT na tabela para poder ver o registro alterado

Listagem 9: SELECT para ver os dados atualizados

SELECT * FROM fornecedores;



```
mysql> SELECT * FROM fornecedores ORDER BY nome asc;
+----+-----+-----+
| codigo | nome      | email                |
+----+-----+-----+
| 2      | João      | joao@gmail.com       |
| 3      | Maria     | maria@gmail.com      |
| 1      | Ricardo   | ricoarrigoni@gmail.com |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql> UPDATE fornecedores SET nome="Ricardo Arrigoni" WHERE codigo=1;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM fornecedores;
+----+-----+-----+
| codigo | nome          | email                |
+----+-----+-----+
| 1      | Ricardo Arrigoni | ricoarrigoni@gmail.com |
| 2      | João          | joao@gmail.com       |
| 3      | Maria         | maria@gmail.com      |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Figura 5: Editando registros.

Como vimos no exemplo anterior, estamos usando uma cláusula chama de WHERE em nosso SQL, essa cláusula é responsável por definir qual registro específico da tabela vai ser afetado pelo comando SQL.

DELETE

Por último e não menos importante, vamos falar do comando DELETE. Ele é o responsável por remover todo e qualquer registro do bando de dados.

Nota:

Uma vez executado, esse comando não é reversível, portanto tome bastante cuidado ao deletar algum registro de seu banco de dados.

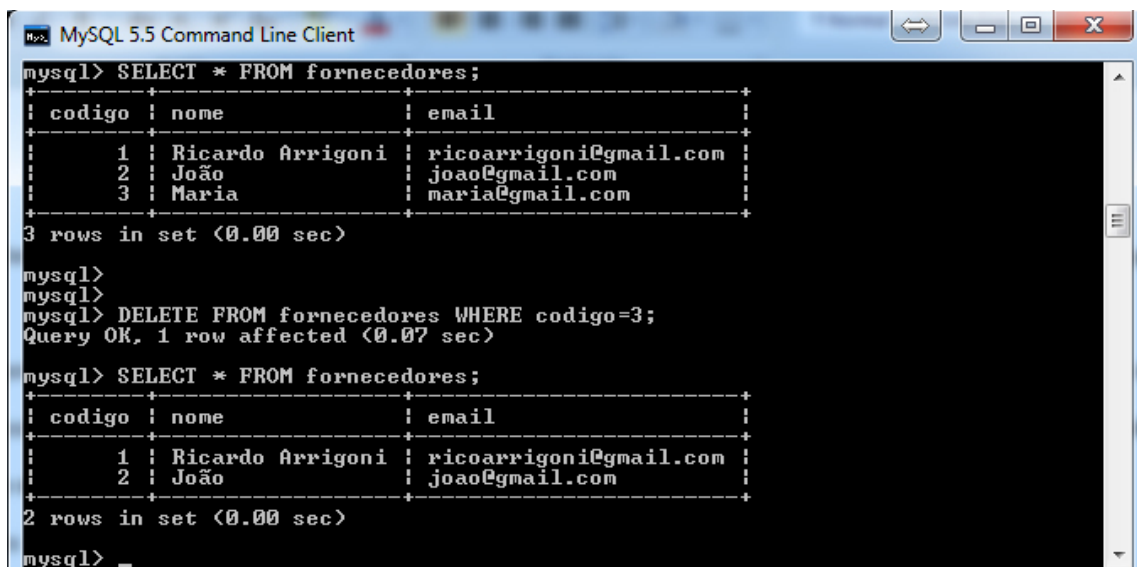
Agora que já sabemos o que ele faz e que devemos tomar muito cuidado ao utilizá-lo, vamos ver como usar ele em nosso exemplo.

Listagem 10: Deletando dados


```
DELETE FROM fornecedores WHERE codigo=3;
```

Saiba mais: [Comandos básicos em SQL - INSERT, UPDATE, DELETE e SELECT](#)

Depois de executar esse comando o nosso registro de `codigo=3` foi excluído do banco de dados e ele não poderá ser recuperado, ficamos então com apenas 2 registros em nossa tabela.



```
mysql> SELECT * FROM fornecedores;
+----+-----+-----+
| codigo | nome      | email                |
+----+-----+-----+
| 1      | Ricardo Arrigoni | ricoarrigoni@gmail.com |
| 2      | João         | joao@gmail.com        |
| 3      | Maria        | maria@gmail.com       |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql>
mysql> DELETE FROM fornecedores WHERE codigo=3;
Query OK, 1 row affected (0.07 sec)

mysql> SELECT * FROM fornecedores;
+----+-----+-----+
| codigo | nome      | email                |
+----+-----+-----+
| 1      | Ricardo Arrigoni | ricoarrigoni@gmail.com |
| 2      | João         | joao@gmail.com        |
+----+-----+-----+
2 rows in set (0.00 sec)

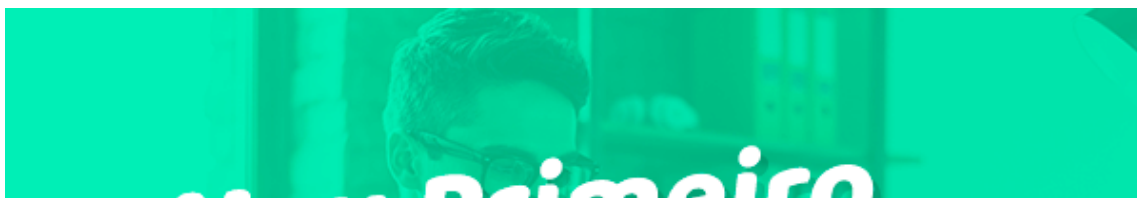
mysql>
```

Figura 6: Deletando dados da tabela.

Como pudemos ver, o MySQL é um banco de dados bem simples e fácil de se usar, além de ser bastante leve e o melhor de tudo, gratuito. Por essas e outras que ele cresce cada vez mais tanto no mundo acadêmico quanto no mundo corporativo.

Fico por aqui nesse tutorial, em breve mais tutoriais para os leitores do site da DevMedia, um abraço e até a próxima.

Confira também





Inicie agora sua carreira de programador por apenas
R\$49,90/mês

Ainda está em dúvida? Experimente a plataforma
durante 3 dias sem cartão. [Faça um teste grátis](#)

Benefícios

- Suporte em tempo real
- Certificado de autoridade
- Exercícios para praticar
- Estudo gamificado
- Planos de estudo para cada carreira de programador



Receba nossas novidades

Suporte ao aluno - Tire a sua dúvida.