



# Desenvolvimento de Sistemas

---

## Índices (*index*)

### Introdução

Quando se faz consultas para buscar informações nos bancos de dados relacionais (SQL – *structured query language*), muitas é com o objetivo de encontrar uma informação específica, um valor em uma coluna e uma linha direta. Os índices podem ser utilizados para aumentar o desempenho dessas consultas, acelerar a busca e melhorar o desempenho do sistema como um todo.

Sem utilizar índices, o MySQL executa a busca iniciando do primeiro registro e “varrendo” a tabela até encontrar o valor ou chegar ao fim dos registros. Se seus bancos incluírem uma grande quantidade de registros, essa varredura pode demorar muito tempo, afetando o desempenho da aplicação como um todo.

Essa perda de desempenho no banco de dados afetará qualquer aplicação, seja ela *web*, *mobile* ou *desktop*, e, em último caso, pode afetar a produtividade geral do negócio. Imagine que um funcionário leve quinze segundos para buscar um registro e faça 25 consultas por hora. O tempo investido será de aproximadamente seis minutos e meio por hora, ou seja, o funcionário despende mais de 10% do seu dia de trabalho apenas esperando o sistema responder.

Em um mês, são três dias apenas aguardando o banco de dados dar uma resposta às consultas.

Programadores e administradores de bancos de dados precisam se esforçar para reduzir cada vez mais o tempo de resposta do sistema.

Mesmo sem o uso de índices, é possível tomar algumas atitudes para melhorar a *performance* das consultas e o tempo de resposta a elas. Veja a seguir algumas dicas de como fazer isso:

## Dica 1

Quando realizar uma consulta, selecione apenas as colunas que você realmente usará, assim a quantidade de dados que a consulta retorna será menor e, por isso, o retorno será mais rápido. Vale a pena avaliar quais colunas realmente são necessárias e listar apenas elas.

Por isso, evite utilizar:

```
SELECT * FROM tabela;
```

Em vez disso, utilize:

```
SELECT campo1 FROM tabela;
```

## Dica 2

Já na cláusula **WHERE**, a ordem dos filtros pode afetar a consulta e o desempenho dela, por isso prefira utilizar a seguinte ordem:

Números -> Data/hora -> Texto simples -> Texto extenso -> Binários

Essa ordem se aplica pelo nível de exatidão que é possível encontrar no campo, bem como pela complexidade do campo também. Quanto mais específico e simples for o dado da coluna, mais rápido o sistema faz uma comparação do valor e pode retornar a resposta.

Exemplo:

```
SELECT Campo1, Campo2, Campo3 WHERE Campo1 = 1 AND Campo2 = 'func' AND Campo3 = 0
```

Ainda na cláusula **WHERE**, é sempre melhor usar a igualdade como comparação, e se possível na ordem do exemplo anterior, em vez de filtrar por faixas como “maior que” ou “menor que”.

## Dica 3

A última dica é utilizar índices, pois eles ajudam o sistema gerenciador de banco de dados a localizar os registros com mais agilidade. Esse assunto será explorado com mais detalhes a partir de agora.

## Conceitos de consultas e índices

Para entender os índices no SQL e no MySQL, é preciso entender bem como as consultas no SQL funcionam.

As consultas são fundamentalmente executadas pelo comando **SELECT**, que, ao ser executado, efetuará uma ação chamada **TABLE SCAN**, ou seja, o sistema gerenciador de banco de dados vai **percorrer toda a tabela**, lendo item por item de cada linha e coluna.

Se você estiver utilizando filtros na cláusula **WHERE**, mesmo assim, o **TABLE SCAN** lerá todos os registros e devolverá na consulta somente os que atendam aos filtros; caso algum não atenda, a ação não devolverá esse registro, mas ele terá sido lido mesmo assim.

Perceba então que, se você tiver uma tabela com 7 milhões de registros e quiser apenas um registro, ainda assim o **TABLE SCAN** lerá a tabela inteira.

Veja um exemplo com a cláusula **WHERE** procurando o **código 3**:

	Código	Nome
Codigo = 3? Não. Desconsiderar	1	José
Codigo = 3? Não. Desconsiderar	2	Paulo
Codigo = 3? Sim. Retornar registro	3	Marta
Codigo = 3? Não. Desconsiderar	4	Felipe

Esse exemplo, para que ficasse visualmente mais claro, tem apenas quatro registros e, por isso, o resultado do **TABLE SCAN** será bem rápido. Imagine, porém, se, em vez de quatro registros, tivessem 4 mil ou 4 milhões de registros.

Para aperfeiçoar o desempenho dessas consultas, serão utilizados os índices, que são objetos do banco de dados, os quais melhoram a organização e aceleram as consultas nesses bancos.

Ao criar um índice em uma coluna, o banco de dados ordenará a tabela por essa coluna e, a partir disso, sempre que se utilizar filtros sobre essa coluna, as consultas serão feitas a partir de uma busca binária, que é um algoritmo de busca que divide os dados na metade sucessivamente.

Analise a seguinte tabela:



Id	Produto
1	Leite
2	Arroz
3	Café
4	Macarrão
5	Feijão
6	Aveia
7	Goiabada
8	Lentilha
9	Batata
10	Pão
11	Polvilho
12	Tapioca

Supondo que você esteja procurando a **Id 11**, lembre-se de que o campo **Id** nesse exemplo já está ordenado para demonstrar os índices.

Primeiramente, o sistema gerenciador de banco de dados dividirá a coluna ao meio e verificará se a **Id 11** está na metade de baixo ou na metade de cima. Neste caso, a **Id** está na metade de baixo, então o sistema não avaliará as da metade de cima – já reduzindo o tempo do **TABLE SCAN** pela metade –, mas fará essa verificação repetidas vezes.

A primeira divisão será:



Id	Produto
7	Goiabada



8	Lentilha
9	Batata
10	Pão
11	Polvilho
12	Tapioca

A segunda divisão será:

Id	Produto
10	Pão
11	Polvilho
12	Tapioca

Observe que o número de registros para o **TABLE SCAN** será muito reduzido e, consequentemente, o desempenho da consulta aumentará muito.

Esse exemplo baseia-se na estrutura de dados chamada “árvore de busca binária” do MySQL, sendo mais especificamente uma árvore B, ou árvore balanceada, mas existem outros tipos de índices que o MySQL pode criar embasado em outras estruturas de dados.

Para relembrar alguns conceitos, você pode revisar os conteúdos relativos à **estrutura de dados** na unidade curricular sobre algoritmos deste curso.

Se você tiver dúvida sobre quais as colunas em que devem ser criados os índices, a dica é sempre usá-los em colunas envolvidas em cláusulas **WHERE** e **JOINS** de consultas.

# Comandos de criação e manipulação



No MySQL, é muito fácil criar índices, pois eles podem ser criados no momento da definição das tabelas ou posteriormente, a partir dos comandos próprios do MySQL.

## Criando índices concomitantemente à criação da tabela

Para criar um índice no momento da definição da tabela, utiliza-se a palavra reservada **INDEX** seguida do nome do campo que será o *index* dentro de parênteses.

Como exemplo, será criada uma tabela “Cliente” com os campos **Id** do tipo inteiro e **Nome** do tipo texto. A criação do índice com a tabela utilizará o seguinte código.

```
CREATE TABLE CLIENTE
(
    Id INT,
    Nome VARCHAR(50),
    INDEX (Id)
);
```

Observe um segundo exemplo, agora com a tabela “Editora”. O campo **Id** está marcado como chave primária e o campo **NomeEditora** será usado como índice, pois, futuramente, será preciso pesquisar a editora pelo nome para se cadastrar os livros no sistema.



```
CREATE TABLE Editora
(
  Id SMALLINT PRIMARY KEY AUTO_INCREMENT,
  NomeEditora VARCHAR(40) NOT NULL,
  INDEX (NomeEditora)
);
```

Veja que, quando se cria uma tabela com uma chave primária, o MySQL automaticamente torna esse campo um tipo especial de **índice primário**, que define a ordem dos dados da tabela.

Em seguida são criados mais índices, como o campo **NomeEditora**, que será, portanto, um índice secundário.

## Criando índices após a criação da tabela

Quando se cria uma tabela, pode-se não saber inicialmente para qual campo o índice será necessário, uma vez que ainda não foram definidos quais campos serão usados em consultas (**SELECT**) ou junções (**JOIN**) com outras tabelas.

Nesse caso, é preciso usar um comando DDL (*data definition language*) para adicionar um índice a uma tabela já criada.

A sintaxe do comando são as palavras reservadas **CREATE INDEX** seguidas do nome do índice, depois a expressão **ON**, que indica onde esse índice será criado, e finalmente o nome da tabela seguida do nome do campo dentro dos parênteses, especificando qual será o índice.

Observe:

```
CREATE INDEX nome_do_indice ON nome_da_tabela(campo)
```

Confira o exemplo com a tabela “Cliente”, sem índice:



```
CREATE TABLE cliente
(
    Id INT
    Nome Varchar(50)
);
```

Agora, acrescenta-se o índice à tabela:

```
CREATE INDEX idx_id ON cliente(Id)
```

Já que o índice está sendo criado após a criação da tabela, é preciso dar um nome para ele. Convencionalmente, utiliza-se um prefixo como “idx\_” ou “index\_”, seguido do nome do campo.

Considere o exemplo da tabela “Editora”:

```
CREATE TABLE Editora
(
    Id SMALLINT PRIMARY KEY AUTO_INCREMENT,
    NomeEditora VARCHAR(40) NOT NULL,
);
```

Agora veja a criação do índice:

```
CREATE INDEX idx_NomeEditora ON Editora(NomeEditora);
```

Vale lembrar que, eventualmente, será preciso criar o índice invertido, ou seja, em ordem decrescente. Para isso, basta adicionar a palavra reservada **DESC** junto ao nome do campo.

Atualizando o exemplo anterior, o comando seria:

```
CREATE INDEX idx_NomeEditora ON Editora(NomeEditora DESC);
```

## Comandos da aplicação de índices

### Visualização de índices

Para visualizar os índices tidos na tabela, pode-se utilizar o comando:

```
SHOW INDEX FROM <nome da tabela>
```

Veja um exemplo:

```
SHOW INDEX FROM Cliente;
```

ou

```
SHOW INDEX FROM Editora;
```

O resultado será aproximadamente este:

#	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Co
1	Editora	0	PRIMARY	1	Id	A	0				BTREE	
2	Editora	1	idx_NomeEditora	1	NomeEditora	D	0				BTREE	

Figura 1 – Resultado da consulta **SHOW INDEX** no MySQL Workbench

A figura mostra a aba “Output” do MySQL Workbench apresentando o resultado do comando. Na primeira linha, destacam-se as colunas Table = “Editora”, Key\_name = “PRIMARY”, Column\_name = “id”. Na segunda linha, as mesmas colunas contêm o seguinte: Table = “Editora”, Key\_name = “idx\_NomeEditora” e Column\_name = “NomeEditora”. Há ainda outras colunas com detalhes dos índices.

## Removendo índices

Caso o índice não faça mais sentido na tabela, por algum erro de definição, de execução ou por alguma mudança de processos, e você deseje removê-lo, o comando é muito simples. Para isso, utilize as palavras reservadas **DROP INDEX** seguidas do nome do índice, da palavra reservada **ON** e do nome da tabela na qual está o índice.

**DROP INDEX <indice> ON <tabela>**

Veja o exemplo:

```
DROP INDEX idx_NomeEditora ON Editora;
```

## Teste de índices

Agora que você já entendeu os principais conceitos sobre os índices, poderá testar a busca com eles. Para isso, revise a definição da tabela “Editoras” com uma chave primária, que automaticamente será o seu índice primário.

Você pode escrever o código a seguir no seu MySQL para testar:

```
CREATE TABLE Editora (  
    Id SMALLINT PRIMARY KEY AUTO_INCREMENT,  
    NomeEditora VARCHAR(40) NOT NULL  
);
```

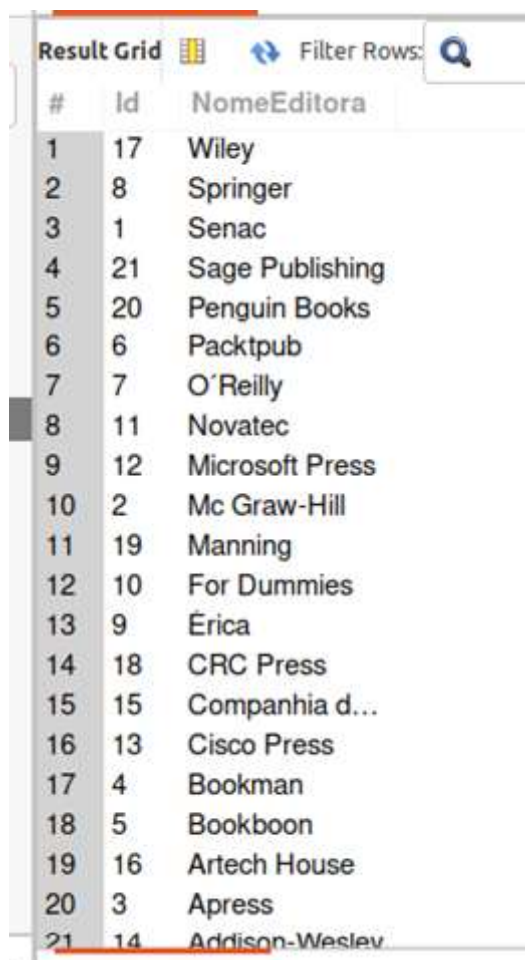
Na sequência, serão adicionadas algumas editoras à tabela:

```
INSERT INTO Editora (NomeEditora)  
VALUES  
( 'Senac'),  
( 'Mc Graw-Hill'),  
( 'Apress'),  
( 'Bookman'),  
( 'Bookboon'),  
( 'Packtpub'),  
( 'O´Reilly'),  
( 'Springer'),  
( 'Érica'),  
( 'For Dummies'),  
( 'Novatec'),  
( 'Microsoft Press'),  
( 'Cisco Press'),  
( 'Addison-Wesley'),  
( 'Companhia das Letras'),  
( 'Artech House'),  
( 'Wiley'),  
( 'CRC Press'),  
( 'Manning'),  
( 'Penguin Books'),  
( 'Sage Publishing');
```

Agora, faça uma consulta simples no banco:

```
SELECT * FROM Editora;
```

O resultado será algo como:



#	Id	NomeEditora
1	17	Wiley
2	8	Springer
3	1	Senac
4	21	Sage Publishing
5	20	Penguin Books
6	6	Packtpub
7	7	O'Reilly
8	11	Novatec
9	12	Microsoft Press
10	2	Mc Graw-Hill
11	19	Manning
12	10	For Dummies
13	9	Érica
14	18	CRC Press
15	15	Companhia d...
16	13	Cisco Press
17	4	Bookman
18	5	Bookboon
19	16	Artech House
20	3	Apress
21	14	Addison-Wesley

Figura 2 – Resultado da consulta sobre a tabela “Editora”

: A figura mostra a aba “Output” do MySQL Workbench, com os dados de Id e nome de todas as tabelas inseridas, conforme informações do script de povoamento.

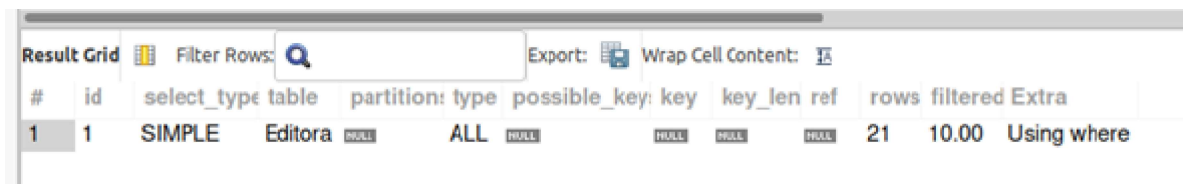
Observe neste momento uma consulta mais específica, buscando a editora Wiley:

```
SELECT * FROM Editora WHERE NomeEditora = 'Wiley';
```

Para ver os detalhes de como é feita essa consulta, você poderá utilizar a palavra reservada **EXPLAIN**:

```
EXPLAIN SELECT * FROM Editora WHERE NomeEditora = 'Wiley';
```

O resultado será algo como:



#	id	select_type	table	partition	type	possible_key	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	Editora		ALL					21	10.00	Using where

Figura 3 – Resultado do comando **EXPLAIN** para a consulta sem índice

Observe que, na coluna **rows**, a consulta teve que percorrer 21 linhas para achar o resultado solicitado e que está usando a cláusula **WHERE**, como indica na coluna **Extra**.

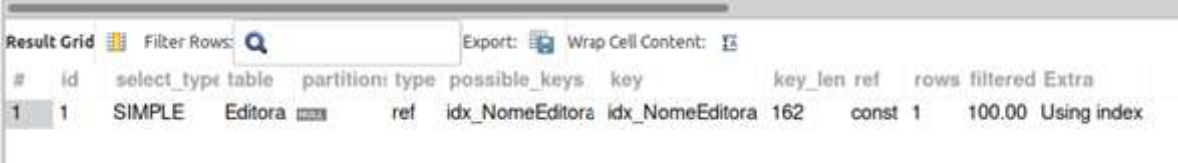
Agora, como feito anteriormente, acrescente um índice à coluna **NomeEditora**. Você pode utilizar o código a seguir para acrescentar esse índice.

```
CREATE INDEX idx_NomeEditora ON Editora(NomeEditora);
```

Dessa vez, execute novamente o **SELECT** com **EXPLAIN** para ver o comportamento da consulta que contém o índice:

```
EXPLAIN SELECT * FROM Editora WHERE NomeEditora = 'Wiley';
```

O resultado será:



#	id	select_type	table	partition	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	Editora		ref	idx_NomeEditora	idx_NomeEditora	162	const	1	100.00	Using index

Figura 4 – Resultado do comando EXPLAIN sobre consulta com índice

A figura mostra o resultado da consulta no MySQL Workbench com os seguintes dados:

```
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
----+-----+-----+ | id | select_type | table | partitions | type | possible_keys | key |
key_len | ref | rows | filtered | Extra | +---+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+ | 1 | SIMPLE | Editora | NULL | ref
| idx_NomeEditora | idx_NomeEditora | 162 | const | 1 | 100.00 | Using index | +---+-----
----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+ 1 row in set, 1 warning (0,00 sec)
```

Perceba que na coluna **rows** aparece somente uma coluna e na coluna **Extra** aparece a utilização de um *index*, por isso a eficiência da consulta é muito maior. No MySQL Workbench você pode rolar o painel à direita até **Query Stats** para ver o tempo que a consulta levou para executar.

### Que tal realizar alguns desafios?

Considere a seguinte tabela:



```
create table clientes ( codigo int(11) not null auto_increment, nome
varchar(100) not null, email varchar(100) not null, telefone varchar(100)
not null, primary key (codigo));
```

Supondo que as consultas a esses dados estão causando lentidão às respostas, quais índices poderiam ajudar a melhorar a *performance* do sistema?

Crie a tabela, alimente-a com dados e faça testes com e sem os índices para verificar a eficácia.

## Encerramento

Em um mundo onde a quantidade de dados armazenados em sistemas computacionais só aumenta, os acessos a esses conteúdos são exponencialmente maiores e também estão em pleno crescimento.

Assim, o acesso rápido a esses dados é um fator de extrema importância e que pode afetar não só o desempenho do sistema, mas também a produtividade e o faturamento das empresas e dos usuários dele.

Com o uso de índices e de boas práticas para otimizar o seu banco, será possível garantir que a *performance* dos seus aplicativos e sistemas não seja prejudicada pela lentidão no acesso e nas consultas, evitando assim qualquer prejuízo e avançando em direção ao sucesso dos seus sistemas.