## 5.0 Manipulando tabelas

Um servidor de banco de dados é dividido em bases de dados com o intuito de separar as informações de domínios diferentes. Nessa mesma linha de raciocínio, podemos dividir os dados de uma base a fim de agrupá-los segundo as suas correlações. Essa separação é feita através de tabelas. Por exemplo, no sistema de um banco, é interessante separar o saldo e o limite de uma conta, do nome e CPF de um cliente. Então, poderíamos criar uma tabela para os dados relacionados às contas e outra para os dados relacionados aos clientes.

| Cliente | | |
|---|---|---|
| nome | idade | cpf |
| José | 27 | 31875638735 |
| Maria | 32 | 30045667856 |

| Conta | | |
|---|---|---|
| numero | saldo | limite |
| 1 | 1000 | 500 |
| 2 | 2000 | 700 |

***Tabela 5.0: Tabelas para armazenar os dados relacionados aos clientes e às contas***

Uma tabela é formada por **registros (linhas)** e os registros são formados por **campos (colunas)**. Por exemplo, considere uma tabela para armazenar as informações dos clientes de um banco. Cada registro dessa tabela armazena em seus campos os dados de um determinado cliente.

***Observação-5.0*:** Antes de utilizar os comandos abaixo, devemos selecionar uma base de dados em que você deseja trabalhar com a manipulação das tabelas. Para fazer essa seleção usamos o comando **USE**, estudado no capítulo 4 - tópico 4.3 . Todos as instruções abaixo foram escritas no MySQL Client.

### *5.1 Criando tabelas no MySQL Server*

As tabelas no MySQL Server são criadas através do comando **CREATE TABLE**. Na criação de uma tabela, é necessário definir quais são os nomes e os tipos das colunas. Abaixo temos o comando básico para a criação de uma tabela:

**CREATE TABLE**
*nome_da_tabela* **(***nome_campo1 tipo_campo1, nome_campo2 tipo_campo2***);**

**Legenda:**

- nome_da_tabela < nome da tabela que você pretende criar;
- nome_campo1 < nome do primeiro campo/coluna da tabela;
- tipo_campo1 < tipo de dado que o campo/coluna da tabela irá armazenar;





Vamos criar uma tabela de nome **Livro** e com os campos de nome **nome cod_livro** do tipo

```
mysql> CREATE TABLE Livro( cod_livro INTEGER, nome_livro VARCHAR(100) );
Query OK, 0 rows affected (0.09 sec)
```

**INTEGER** e **nome_livro** do tipo **VARCHAR.**

***Terminal 5.0: Criando uma tabela.***

Ainda, é possível criar uma tabela sem ativar/selecionar um banco de dados (**ver Observação-5.0 no início do capítulo**), contudo, é necessário informar no comando **CREATE TABLE** o nome do banco de dados e o nome da tabela, ficando o comando CREATE TABLE dessa forma:

**CREATE TABLE** *nome_do_banco*.*nome_da_tabela*
**(***nome_campo1 tipo_campo1, nome_campo2 tipo_campo2***);**

**Legenda:**

nome_do_banco < nome do banco onde será criado a tabela;

- < separa o nome do banco de dados e o nome da tabela que será criada;

Vamos criar a mesma tabela Livro criada no exemplo anterior, mas agora sem selecionar a base de dados

```
mysql> CREATE TABLE livraria.Livro( cod Integer, nome_livro varchar(100) );
Query OK, 0 rows affected (0.08 sec)
```

(**ver Observação-5.0 no início do capítulo**).

***Terminal 5.1: Cria na base de dados livraria a tabela Livro.***

### *5.2 Listando tabelas no MySQL Server*

As tabelas de uma base de dados podem ser listadas através do comando:

**SHOW TABLES;**

Antes de utilizar esse comando(**ver Observação-5.0 no início do capítulo**).

***Terminal 5.2: Listando as tabelas de uma base de dados.***

```
mysql> USE livraria;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+--------------------+
| Tables_in_livraria |
+--------------------+
| Livro              |
+--------------------+
1 row in set (0.00 sec)
```

## *5.3 Alterando uma tabela no MySQL Server*

Podemos alterar a estrutura de uma tabela e suas propriedades com o comando **ALTER TABLE**.

**ALTER TABLE** *nome_da_tabela* **RENAME** novo_*nome_tabela*;

### 5.3.1 Renomeando uma tabela com o comando ALTER TABLE.

**Legenda:**

nome_da_tabela < nome da tabela que você deseja alterar;

novo_nome_tabela < novo nome da tabela;

```
mysql> ALTER TABLE Livro RENAME livros;
Query OK, 0 rows affected (0.00 sec)
```

**Exemplo:**

***Terminal 5.3: Alterando o nome da tabela.***

### 5.3.2 Adicionando uma nova coluna com o comando ALTER TABLE.

**ALTER TABLE** *nome_da_tabela* **ADD** *nome_do_novo_campo tipo_do_novo_campo*;

**Legenda:**

nome_da_tabela < nome da tabela que você deseja adicionar o novo campo;

*nome_do_novo_campo* < nome do campo a ser adicionado a tabela;

*tipo_do_novo_campo < tipo do campo que sera adicionado a tabela;*

**Exemplo:**

```
mysql> ALTER TABLE Livro ADD paginas INTEGER;
Query OK, 0 rows affected (0.00 sec)
```

***Terminal 5.4: Adicionando uma coluna.***





### 5.3.3 Removendo uma coluna com o comando ALTER TABLE.

**ALTER TABLE** *nome_da_tabela* **DROP COLUMN** *nome_da_coluna_deletada;*

**Legenda:**

nome_da_tabela < nome da tabela que você deseja adicionar o novo campo;

*nome_da_coluna_deletada* < nome da coluna a ser deletada na tabela;

**Exemplo:**

```
mysql> ALTER TABLE Livro DROP COLUMN paginas;
Query OK, 0 rows affected (0.00 sec)
```

***Terminal 5.5: Removendo uma coluna.***

### 5.3.4 Renomeando um campo com o comando ALTER TABLE.

**ALTER TABLE** *nome_da_tabela* **CHANGE** *nome_campo novo_nome_campo tipo*;

**Legenda:**

nome_da_tabela < nome da tabela que você deseja alterar;

*nome_campo* < nome do campo que será alterado;

*novo_nome_campo < novo nome do campo;*

*tipo < tipo do campo;*

```
mysql> ALTER TABLE Livro CHANGE cod id_livro INTEGER;
```

**Exemplo:**

***Terminal 5.6: Renomeando campo cod da tabela Livro para id_livro***

## *5.4 Deletando uma tabela no MySQL Server*

**DROP TABLE** *nome_da_tabela;*

Se uma tabela não for mais desejada, ela pode ser removida através do comando **DROP TABLE**.

**Legenda:**

*nome_da_tabela < nome da tabela que você deseja deletar;*

```
mysql> DROP TABLE Livro;
Query OK, 0 rows affected (0.00 sec)
```

***Terminal 5.7: Eliminando tabela***

## *5.5 Saiba mais...*

### 5.5.1 Tipos de dados do MySQL

Ao criar uma tabela você deverá especificar o tipo de dados a ser armazenado nela. O MySQL possui três tipos de dados básicos: **numéricos, data/hora e string**.

**Tipos de dados numéricos**

| TIPO | INTERVALO | bytes | DESCRIÇÃO |
|---|---|---|---|
| TINYINT[(M)] | -127 a 128; ou 0 a 255 | 1 | inteiros muitos pequenos |
| BIT | | | o mesmo que TINYINT |
| BOOL | | | o mesmo que TINYINT |
| SMALLINT[(M)] | -32768 a 32767 | 2 | inteiros pequenos |
| MEDIUMINT[(M)] | -8388608 a 8388607; ou 0 a 16777215 | 3 | inteiros de tamanho médio |
| INT[(M)] | -213 a 231-1; ou 0 a 232-1 | 4 | inteiros regulares |
| INTEGER[(M)] | | | o mesmo que INT |
| BIGINT[(M)] | -263 a 263-1; ou 0 a 264-1 | 8 | inteiros grandes |
| FLOAT(precisão) | depende da precisão | variável | números de ponto flutuante de precisão simples ou dupla |
| FLOAT[(M,D)] | 1.175494351E-38 a ±3.402823466E+38 | 4 | números de ponto flutuante de precisão simples. O mesmo que FLOAT(4) |
| DOUBLE[(M,D)] | ±1.7976931348623157 E +308 a ±2.2250738585072014 E -308 | 8 | números de ponto flutuante de precisão dupla. O mesmo que FLOAT(8) |
| DOUBLE | | | O mesmo que DOUBLE[(M,D)] |
| PRECISION[(M,D)] | | | O mesmo que DOUBLE[(M,D)] |
| REAL[(M,D)] | | | O mesmo que DOUBLE[(M,D)] |
| DECIMAL[(M,D)] | variável | M+2 | número de ponto flutuante armazenado como char |
| NUMERIC[(M,D)] | | | O mesmo que DECIMAL |
| DEC[(M,D)] | | | O mesmo que DECIMAL |

**OBSERVAÇÕES:**

- As opções entre colchetes (**[e]**) são opcionais;

- Dentre os tipos que se ajustam aos dados a serem inseridos, escolha sempre o de menor tamanho;

- Para dados do tipo inteiro você pode usar a opção **UNSIGNED** para especificar inteiros positivos ou zero;

- **M** especifica o tamanho máximo de exibição;

- **D** especifica o número de casas decimais. O valor máximo de D é 30 ou M-2;

- Tanto para números inteiros como para números de ponto flutuante você pode especificar a opção **ZEROFILL** que preenche os números com zeros iniciais. Colunas especificadas com ZEROFILL são automaticamente configuradas como UNSIGNED;

**Tipos de dados data/hora**

| TIPO | INTERVALO | DESCRIÇÃO |
|---|---|---|
| DATE | 1000-01-01 a 9999-12-31 | data. Exibido como YYYY-MM-DD |
| TIME | -838:59:59 a 838:59:59 | hora. Exibido como HH:MM:SS |
| DATETIME | 1000-01-01 00:00:00 a 9999-12-31 23:59:59 | data e hora. Exibido como YYYY-MM-DD HH:MM:SS |
| TIMESTAMP[(M)] | 1970-01-01 00:00:00 a algum momento em 2037. Depende do limite do sistema operacional | registro de data e hora útil para transações. Os formatos de exibição podem ser: TIMESTAMP YYYYMMDDHHMMSS TIMESTAMP(14) YYYYMMDDHHMMSS TIMESTAMP(12) YYMMDDHHMMSS TIMESTAMP(10) YYMMDDHHMM TIMESTAMP(8) YYYYMMDD TIMESTAMP(6) YYMMDD TIMESTAMP(4) YYMM TIMESTAMP(2) YY |
| YEAR[(2)] | 70 a 69 (1970 a 2069) | ano |
| YEAR[(4)] | 1901 a 2155 | ano |





**Tipos de dados string**

| TIPO | INTERVALO | DESCRIÇÃO |
|---|---|---|
| [NATIONAL] CHAR(M) [BINARY] | 0 a 255 caracteres | string de comprimento fixo M. NATIONAL especifica que o conjunto de caracteres padrão (ANSI SQL) será utilizado. BINARY especifica que os dados devem ser tratados de modo a não haver distinção entre maiúsculas e minúsculas (o padrão é distinguir). |
| CHAR | 1 | o mesmo que CHAR(1) |
| [NATIONAL] | 1 a 255 | string de comprimento variável |
| VARCHAR(M) [BINARY] | variável | string de tamanho variável. O mesmo que [BINARY]. |
| TINYBLOB | 0 a 28 - 1 (255) | BLOB pequeno |
| TINYTEXT | 0 a 28 - 1 (255) | TEXT pequeno |
| BLOB | 0 a 216 - 1 (65535) | BLOB normal |
| TEXT | 0 a 216 - 1 (65535) | TEXT normal |
| MEDIUMBLOB | 0 a 224 - 1 (16777215) | BLOB médio |
| MEDIUMTEXT | 0 a 224 - 1 (16777215) | TEXT médio |
| LONGBLOB | 0 a 232 - 1 (4294967295) | BLOB longo |
| LONGTEXT | 0 a 232 - 1 (4294967295) | TEXT longo |
| ENUM('valor1','valor2',...) | 0 a 65535 | armazenam um dos valores listados ou NULL |
| SET('valor1','valor2',...) | 0 a 64 | armazenam um ou mais dos valores listados ou NULL |

**OBSERVAÇÕES:**

- CHAR e VARCHAR armazenam strings de comprimento fixo e variável respectivamente. VARCHAR trabalha mais lento.
- TEXT e BLOB armazenam textos grandes ou objetos binários (figuras, som, etc.). TEXT diferencia maiúsculas de minúsculas.

## 5.5.2 Tipo de tabelas/ Storage Engine/ Motor de Armazenamento do MySQL

O MySQL possui uma característica um pouco diferente dos outros sistemas gerenciadores de banco de

dados, uma vez que no MySQL é possível escolher o tipo da tabela no momento da criação da mesma.

O formato de armazenamento dos dados, bem como alguns recursos do banco de dados são dependentes do tipo de tabela escolhido. Para sabermos quais tipos de tabelas o MySQL instalado na sua máquina nos permite criar usamos o comando:

```
mysql> SHOW ENGINES;
+------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine     | Support | Comment                                                        | Transactions | XA   | Savepoints |
+------------+---------+----------------------------------------------------------------+--------------+------+------------+
| InnoDB     | YES     | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| MRG_MYISAM | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| BLACKHOLE  | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| CSV        | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| MEMORY     | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| FEDERATED  | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| ARCHIVE    | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| MyISAM     | DEFAULT | Default engine as of MySQL 3.23 with great performance         | NO           | NO   | NO         |
+------------+---------+----------------------------------------------------------------+--------------+------+------------+
8 rows in set (0.00 sec)
```

***Terminal 5.7: comando SHOW ENGINES;***

Perceba pela figura Terminal 5.7 que o MySQL nos fornece flexibilidade na forma de armazenamento dos nossos dados, isso é uma característica forte do MySQL. Desses Storage Engines suportados por meu MySQL, vamos explanar somente dois deles, no geral os mais usados, são eles o MyISAM e o INNODB.

- **MyISAM**

O tipo de tabela ou Storage Engine MyISAM é o mecanismo instalado e configurado como padrão pelo MySQL até a versão inferior a 5.5, justamente por apresentar um dos melhores resultados em todos os aspectos possíveis.

É um método de armazenamento muito rápido, de bom armazenamento em disco, sem restrições de uso de tipos de dados, que permite o uso de todos os recursos do MySQL, com exceção do suporte a transações

A menos que a necessidade seja o uso de transações, este é o método indicado para a maioria dos casos, desde pequenos até grandes bancos de dados. É o único tipo de tabela do MySQL que suporta buscas do tipo Fulltext Searches.

Outra característica é que o seu nível de bloqueio(LOCK / UNLOCK) é de tabela, não tão prático quanto os níveis mais aprofundados. Contudo, como não utiliza o suporte a transações, o número geral de solicitações de bloqueio é menor, mantendo a boa disponibilidade deste tipo de Storage Engine.

Recomendado para sistemas que priorizam velocidade na pesquisa de dados.





**Nome de uso:** MyISAM **Suporte a índices:** SIM **Suporte a transações:** Não
**Tipos de dados não suportados:** Nenhum

**Nível de bloqueio:** Tabela

- **InnoDB**

Já o tipo de tabela InnoDB passou a ser a padrão a partir da versão 5.5 do MySQL, substituindo a Storage Engine MyISAM que até então era a padrão. O InnoDB é o mais recomendado para grandes e complexos bancos de dados, pois além de oferecer todos os recursos disponibilizados pelo método MyISAM, ainda permite o uso de transações com as propriedades ACID (Atomicidades, Consistência, Isolamento e Durabilidade).

Desenvolvido pela Inobase Oy, além do armazenamento de dados e índices em disco, algumas dessas informações são armazenadas também na memória enquanto o servidor está ativo, fazendo com que o seu processamento seja ainda mais veloz.

Outra característica que o diferencia do método MyISAM é que seu nível de bloqueio

(LOCKS) é de linha, sendo mais eficiente e aumentando sua disponibilidade, pois apenas os registros comprometidos em uma transação são bloqueados, e não a tabela toda.

Recomendado para todos os tamanhos de banco de dados, especialmente os de grande porte, além de ser o ideal para sistemas em que os dados estão em constante mudanças.

**Nome de uso:** InnoDB

**Suporte a índices:** Sim

**Suporte a transações:** Sim (ACID)

**Tipos de dados não suportados:** Nenhum

**Nível de bloqueio:** Linha

## Exercícios Propostos

Crie uma base de dados com as seguintes características:

NOME DO BANCO: Vendas
CHARSET: latin1
COLLATE: latin1_general_ci

Selecione essa base de dados e crie uma tabelas com as seguintes características:

NOME DA TABELA: Vendedor

ENGINE: InnoDB

Campos:

cod – Integer – NOT NULL – AUTO_INCREMENT

nome – Varchar(50) – NOT NULL telefone – Varchar(15) – NOTNULL dataNascimento – DATE

Adicione um novo campo horaInicioTrabalho que será do tipo TIME a tabela criada na questão anterior.

Altere o campo **nome** para **nomeVendedor - varchar(60)** da tabela criada na questão 02;