Sabemos que a entidade principal do banco de dados é a tabela, que possui colunas (campos) de diferentes categorias (tipos) e que, ao longo do mesmo campo, todos os valores devem ser do tipo estabelecido na criação da tabela.

Vamos entender agora quais são os principais tipos no caso do MySQL.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* Propriedade UNSIGNED: Não permite sinal no número. Por isso, o conjunto de valores válidos aumentam.

Iniciando pelo tipo NUMÉRICOS, temos duas categorias: INTEIROS e DECIMAIS. Nos números inteiros há diversos tipos que podem ser criados no MySQL e estão diretamente relacionados com o espaço que vou precisar para armazenar determinado número no banco de dados. Quanto maior o espaço, maior o conjunto de números possíveis a serem armazenados.

Observando a tabela, na coluna "TIPO" temos cinco categorias de números inteiros que podem ser armazenados no MySQL, o **TINYINT**, **SMALLINT**, **MEDIUMINT**, **INT** E **BIGINT**. No segundo campo, **Valor em Bytes** temos o número de bytes gastos para armazenar um número de determinado tipo, como o TINYINT que para armazenar é necessário 1 byte, o SMALLINT que gasta 2 bytes e assim sucessivamente.

Quanto maior a quantidade de bytes que posso armazenar, maior o espaço de números com os quais posso trabalhar. Então, por exemplo, o TINYINT fica entre -128 e 127, se for criado um número dessa categoria e enviado para armazenar o número 300, será retornado um erro, visto que esse tipo não suporta.

Temos também uma propriedade chamada UNSIGNED. É um atributo para números inteiros no MySQL utilizado para a definição de números positivos (sem sinal), visto que o sinal de menos (-) ocupa espaço de armazenamento. Então, se informar para o MySQL que uma coluna específica é do tipo inteiro e sem sinal, o conjunto de números para armazenar nesse tipo de campo é maior, visto que não é preciso armazenar no computador os sinais e até mesmo de números positivos, já que não é reservado um espaço para arquivar o sinal.

Exemplificando, se o TINYINT for do tipo UNSIGNED, consigo guardar de 0 a 255 e se for com sinal do -128 a 127, perceba haver uma diferença de espaços com o ganho do sinal.

Falando um pouco sobre os números DECIMAIS, temos dois tipos: de **PRECISÃO FIXA** e **PONTO FLUTUANTE**. Este significa que será realizado um arredondamento quando o número de casas decimais for superior ao número permitido pelo banco de dados. No ponto flutuante também temos dois tipos de valores decimais, o **FLOAT**, que trabalha com uma precisão simples de 4 bytes e o **DOUBLE**, com precisão dupla de 8 bytes. A diferença entre ambos é o tamanho de espaço de armazenamento.

O DOUBLE é mais usado para quando queremos ter números mais exatos nos cálculos, uma vez que ele faz arredondamentos mais precisos com números com mais casas decimais. Quando declaramos um número FLOAT ou DOUBLE, é possível especificar o número de dÍgitos e casas decimais.

Como exemplo, quando declaramos esse tipo, podemos especificar da seguinte forma FLOAT (7,4) isso significa que esse número do tipo FLOAT vai ter sete dígitos, sendo quatro casas decimais. Podemos inserir um número de casas decimais para além de quatro, como no caso do número 999,00009 com cinco casas decimais. O MySQL, atendendo a especificação, vai arredondar esse número para quatro casas decimais, armazenando o 999,0001.

Nos DECIMAIS FIXOS, temos o **DECIMAL** ou **NUMERIC**, são análogos e sua definição é a quantidade máxima de dígitos que o número pode ter, no caso é 65 dígitos. Isso para números INTEIROS e DECIMAIS, ou seja, posso ter um número com um número inteiro e 64 casas decimais, ou posso ter um número com 64 números inteiros e apenas uma casa decimal.

Quando especificamos o número de casas decimais e a quantidade de dígitos, já estamos indicando o conjunto máximo e mínimo de dados. Vamos supor que temos DECIMAL(5,2), isto é, o número é decimal com 5 dígitos e 2 casas decimais, só poderá armazenar entre -999,99 e 999,99, tal como especificamos no tamanho. Não será feita nenhuma aproximação, apenas vai ser gravado o número exato digitado para ser guardado no banco de dados.

Outro tipo numérico é o BIT, que armazena valores de até 64 bytes. Explicando um pouco sobre o BIT, se o campo for do tipo **BIT(1)**, é possível armazenar 1 ou 0. Se for **BIT(2)** é viável armazenar 01,10,00 e 11. Agora, se tiver um número de até 64 BITS, tenho o tamanho de 64 caracteres que podem ser gravado de 1 ou 0 que representa um número binário. Em campos do tipo LÓGICO, também é usado o BIT, sendo 1 (um) verdadeiro e 0 (zero) falso.

Os campos numéricos possuem alguns atributos. Já foi mencionado o SIGNED e UNSIGNED, que representam se haverá número negativo ou não, mas, agora vamos entender um pouco sobre o ZEROFILL, que preenche com zeros o que estiver faltando do número. Por exemplo, se tiver um campo do tipo INT(4) e pedir para solicitar o valor 5, será gravado 0005.

Outro atributo é o AUTO\_INCREMENT. Esta propriedade é o próprio banco de dados que gera uma sequência numérica automaticamente. Se tiver uma tabela vazia com um campo desse tipo e for inserido um valor nessa coluna será atribuído o **valor 1**, se incluirmos mais um valor será gerado o **valor 2** e assim por diante. Essa sequência se inicia no 1 ou no 0, definido no próprio AUTO\_INCREMENT, bem como, o valor do incremento que quero aplicar, podendo ser 0, 5, 10, 15, 20 ao invés de 1, 2, 3, 4.

O OUT OF RANGE é um erro que acontece quando tentamos gravar na base de dados um valor que está fora do espaço permitido.

Vamos compreender um pouco agora sobre os campos de DATA e HORA, temos cinco principais:

O campo DATE armazena um dia, no formato ano, mês e dia com traços entre eles, vai dia **1000-01-01** até **9999-12-31**. Já o DATETIME, guarda data e hora. A hora é importante principalmente quando tempos campos do tipo LOG, que possui o horário em que alguém fez alguma ação específica no sistema.

Seguindo, o TIMESTAMP é bem semelhante ao o **DATETIME**, contudo possui duas características principais: tem um *range* menor, que vai de **1970-01-01** a **2038-01-19** e possui fuso horário. Por isso o range fica menor, para armazenar mais informações.

Por mais que esse range de datas pareça pequeno, vale lembrar que utilizamos fuso horário para sistemas, por exemplo, de agendas para marcar reuniões em empresas que tiver funcionários(as) em diversos países. Por isso, o range até 2038 não é tão pequeno quanto parece, visto que ninguém irá marcar uma reunião de hoje até o ano de 2038.

O TIME armazena somente o horário e tem um range, de -838:59:59 a 839:59:59. Normalmente, usamos só para gravar uma hora no relógio que vai de meia-noite às onze e cinquenta e nove da noite do dia seguinte. Por isso, não é preciso ter 838 horas gravadas.

No campo YEAR é guardado somente o ano de 1901 a 2155. Podendo ser de duas ou quatro casas decimais, mas, normalmente utilizamos o DATE com uma data de primeiro de janeiro do ano que quero armazenar. Nesse treinamento ainda aprenderemos que existem funções do tipo data que conseguem extrair de um campo DATE, DATETIME ou TIMESTAMP o ano específico.

Falando um pouco do tipo STRING, são as cadeias de caracteres, os textos. Temos algumas categorias, começando pelo **CHAR** e o **VARCHAR**, ambos possuem o tamanho limite de 255 caracteres, a principal diferença são os espaços.

Caso tivermos um campo CHAR(4) e o utilizamos para armazenar as letras "aa", no banco de dados será guardado dois espaços vazios, já que no campo foi estabelecido o caractere de tamanho quatro, ficaria " aa". Isto é, esse campo ocupa maior espaço em disco, visto que grava os espaços vazios que, em algumas situações, pode ser desnecessário. Já quando nos referimos ao VARCHAR(4), se quisermos armazenar "aa", será guardado somente dois caracteres sem gastar espaço a mais no disco.

Temos também as STRINGS do tipo binário, o BINARY e o VARBINARY. Ambos possuem o mesmo conceito do CHAR e VARCHAR, um armazena os espaços e o outro não. O que os difere é que o armazenamento não é em caractere e sim em bytes, ou seja, o tamanho máximo é expresso em binário.

Outros campos do tipo STRING são o **BLOB** e o **TEXT**. No primeiro temos as variações de tamanho máximos **TINYBLOB**, **BLOB**, **MEDIUMBLOB** e **LONGBLOB** e no TEXT também, temos o **TINYTEXT**, **TEXT**, **MEDIUMTEXT** e **LONGTEXT**.

O BLOB é binário, então, em um LONGBLOB é possível guardar um grande binário no banco. Exemplo, posso gravar bytes de um arquivo no Word, ou os bytes de uma foto no banco de dados. Já o TEXT vai ser usado para armazenar textos.

Outro campo é o ENUM, é como se definíssemos opções. Por exemplo, se tivermos um campo chamado "SIZE" do tipo ENUM, na declaração é necessário inserir as opções que serão armazenadas nesse campo, no caso Size ENUM ('x-small', 'small', 'médium', 'large', 'x-large'), só conseguimos guardar no campo "SIZE" os textos nos parênteses.

Dois atributos importantes dos campos do tipo STRING são os SET e o COLLATE, estão relacionados com as cadeias de caracteres que serão usados para armazenar o texto. Caso queira guardar texto em alguma língua diferente é preciso selecionar na definição de campo o SET e o COLLATE especificando a língua utilizada, assim, será inserido no campo uma **tabela ASCII** maior ou menor dependendo do tipo que você está usando.

Finalmente, há alguns campos que aparecem mais nas versões atuais do MySQL, como o SPACIAL, para visualizar mapas. O tipo POINT grava o ponto usando a latitude e longitude no banco MYSQL que fornece algum determinado local, como, restaurante, supermercado ou ponto turístico. Outro tipo é o LINESTRING, que representa uma linha. O GEOMETRY e POLYGON, que representam áreas no mapa.

Temos outros tipos, mas mencionei esses para dar um apanhado geral das possibilidades que temos para criar tabelas e quais as opções para definir o tipo de coluna.