01

Tipos de dados

Transcrição

[00:00] A entidade principal de um banco de dados é uma tabela, e uma tabela possui colunas, chamadas de campos. As colunas, ou campos, possuem tipos, e esses tipos não podem mudar. Se eu defino que uma coluna tem um determinado tipo, os valores de todas as linhas naquela coluna têm que ser do mesmo tipo.

[00:35] Quais são esses tipos? Existe uma gama enorme de tipos associados a campos das tabelas. Nós vamos mencionar no caso do MYSQL os principais. Não vou entrar muito em detalhes neles.

[00:53] Vamos começar pelos tipos numéricos. Temos dois tipos de números, inteiros e decimais. Dentro dos números inteiros, temos vários tipos que podem ser criados numa SQL. E esses tipos estão diretamente relacionados com o espaço que vou gastar para armazenar esse número dentro do banco de dados. E claro, maior espaço, maior conjunto de números possíveis a serem armazenados.

[01:32] Estou mostrando cinco tipos inteiros que podem ser armazenados em uma SQL. Temos o TINYINT, SMALLINT, MEDIUMINT, INT E BIGINT. Para armazenar um TINYINT, gasto 1 byte. Para armazenar um SMALLINT, gasto 2 bytes. Para armazenar um MEDIUMINT, gasto 3 bytes. Para armazenar um INT, gasto 4 bytes. E para armazenar um BIGINT, gasto 8 bytes. Quanto maior a quantidade de bytes que posso armazenar, maior o espaço de números com os quais posso trabalhar.

[02:16] Então, o TINYINT fica entre -128 e 127. Ou seja, se eu criar um número desse tipo e mandar armazenar nele, naquela coluna, o número 300, por exemplo, vou ter erro, porque ele não consegue armazenar o número 300. Já o SMALLINT fica entre -32768 e 32767. O MEDIUMINT fica entre -8388608 e 8388607. O INT fica entre -2147483648 e 2147483647. E o BIGINT fica entre -2xE63 e 2xE63-1.

[02:38] Eu tenho uma propriedade chamada UNSIGNED. Ele seria se o número tem ou não sinal. O sinal de menos de um número ocupa espaço. Então, se eu dizer para o MYSQL que determinado campo é do tipo inteiro e sem sinal, consigo ter um conjunto maior de números para armazenar nesse tipo de campo, porque não preciso armazenar dentro do computador o sinal. Ou pelo menos, mesmo que o número seja positivo, não preciso reservar um espaço para guardar o sinal.

[03:20] Por exemplo, se o TINYINT for do tipo UNSIGNED, eu posso armazenar de 0 a 255. No caso do SMALLINT, de 0 a 65535. O MEDIUMINT de 0 a 16777215. O INT de 0 a 4294967295. E o BIGINT de 0 a 2xE64-1. Eu ganho essa diferença de espaço.

[04:10] Agora vamos falar dos números decimais. Temos dois tipos. De precisão fixa ou ponto flutuante. O ponto flutuante significa que ele faz um arredondamento quando o número de casas decimais for maior do que o permitido no banco.

[04:37] Por exemplo, os dois tipos de valores decimais com ponto flutuante no MYSQL são o FLOAT e o DOUBLE. A diferença entre eles é o tamanho de armazenamento. O FLOAT trabalha com uma precisão simples de 4 bytes e o DOUBLE com uma precisão dupla de 8 bytes.

[04:58] Basicamente, o DOUBLE faz um arredondamento mais preciso com números com mais casas decimais. Se você quiser ter números muito exatos nos seus cálculos, você usa ele.

[05:14] Por exemplo, quando declaramos um número FLOAT ou DOABLE, podemos especificar o número de dígitos e o número de casas decimais. Coloquei como exemplo FLOAT(7,4). Estou dizendo que esse número FLOAT vai ter 7 dígitos, sendo que 4 casas decimais.

[05:39] Mas não fico limitado a gravar no banco de dados quatro casas decimais. Digamos que eu então peça para ele armazenar o número 999,00009. Ou seja, cinco casas decimais. O SQL vai arredondar isso para quatro casas decimais e me armazenar o número como 999,0001. Ele faz isso porque especifiquei que ele só poderia ter quatro casas decimais.

[06:20] Esse é o número tipo ponto flutuante.

[06:25] Decimais fixos. Tenho o DECIMAL ou NUMERIC. São iguais. Ele é definido em números de dígitos que o número pode ter, que é até 65 dígitos. Estou contanto tanto os números inteiros quanto decimais. Posso ter um número com um número inteiro e 64 casas decimais, ou posso ter um número com 64 números inteiros e uma casa decimal. O limite é 65 dígitos.

[07:05] Quando especificamos o número de casas decimais e o número de dígitos, já estamos especificando para ele o conjunto máximo e mínimo de dados. Por exemplo, se eu digo que um número é DECIMAL(5,2), ele só pode ser armazenado entre -999,99 e 999,99. Especifiquei o tamanho. E ele não vai fazer a aproximação. Vai gravar exatamente o número que está sendo colocado dentro do banco de dados.

[07:48] Outro tipo numérico é o BIT, que armazena valores de até 64 bytes. O BIT representa um byte. Por exemplo, se eu falar que o campo é do tipo BIT(1), posso armazenar ou 1 ou zero. Agora, se o campo é BIT(2), posso armazenar 01, 10, 00, 11. Se eu tiver um número de até 64 BITS, tenho um tamanho de 64 caracteres de um, pode ser gravado 1 ou 0 e representa um número binário.

[08:46] O BIT é muito usado também quando temos campos do tipo lógicos. Normalmente consideramos o 1 verdadeiro e o 0 falso.

[08:59] Os campos numéricos têm alguns atributos. Vou mencionar os mais principais. Já mencionei o SIGNED e o UNSIGNED, que diz se ele terá número negativo ou não. O ZEROFILL preenche com zeros o que estiver faltando do número. Ou seja, se eu tenho um campo do tipo INT(4), se eu mandar armazenar o valor 5, ele vai armazenar 0005.

[09:47] O AUTO_INCREMENT é um campo numérico que automaticamente vai criar uma sequência numérica. Eu nunca incluo dados nele. É o banco sozinho que vai gerar esse numérico. Por exemplo, a tabela está vazia, tenho um campo desse tipo, eu insiro um valor, ele vai colocar o valor 1 para esse campo. Quando eu inserir o segundo valor, ele vai colocar o valor 2. E assim por diante.

[10:28] Cada campo novo que eu for colocando vai crescendo de 1. Essa sequência que começa no 1 ou no 0. A propriedade de começar no 1 ou no 0 é uma propriedade que posso especificar no AUTO_INCREMENT, bem como o valor de incremento que aplico. Posso crescer 0, 5, 10, 20, ao invés de um em um.

[10:59] Os erros do tipo OUT OF RANGE acontecem quando tentamos gravar dentro da base de dados um valor que está fora do espaço permitido.

[11:30] Agora vamos falar dos campos de data e hora. Temos cinco principais.

[11:43] O campo DATE armazena simplesmente um dia. Normalmente armazenamos data dentro do MYSQL como ano, mês e dia com traços no meio. O campo DATE vai do dia 1000-01-01 até 9999-12-31.

[12:07] Já o DATETIME é muito parecido com DATE, mas além de armazenar a data, armazeno também a hora. Em algumas situações é importante gravar a hora, principalmente quando temos campos do tipo LOG, que quero saber a hora em que alguém fez determinada ação dentro do meu sistema.

[12:31] O TIMESTAMP é muito parecido, só que tem duas características principais. É um range menor, que vai de 1970-01-01 a 2038-01-19, e tem uma característica que é o fuso horário. Por isso o range fica menor, tenho que armazenar mais coisa.

[12:54] Você pode dizer inicialmente que é um range muito pequeno, mas eu não acho, porque para que situações você vai armazenar fuso horários? Para sistemas, por exemplo, de agendas, para agendar reuniões, supondo que na sua empresa você tem funcionários que vivem em países diferentes, então seu sistema tem que tratar o fuso horário. E ninguém vai marcar uma reunião hoje até 2038. Se você quer realmente gravar datas com horas, o fuso horário não é importante, a não ser nessa situação, então você usa o DATETIME.

[13:37] O TIME armazena somente a hora. Ele tem um range pequeno, de -838:59:59 a 839:59:59. Normalmente, usamos só para gravar uma hora dentro do relógio que vai de meia-noite a onze e cinquenta e nove da noite do dia seguinte. Então não tem necessidade de ter 838 horas gravadas.

[14:09] YEAR. Nesse campo você pode armazenar somente o ano, tendo de 1901 a 2155. Posso ter YEAR de dois ou de quatro. De dois vai armazenar o ano com duas casas decimais. YEAR de quatro com quatro casas decimais. Esse tipo de campo acho desnecessário até existir. Normalmente usamos o campo DATE com uma data primeiro de janeiro no ano que quero armazenar.

[14:45] Lá na frente vou mostrar que existem funções do tipo data que conseguem extrair de um campo DATE, DATETIME ou TIMESTAMP o ano específico.

[14:58] Vamos falar dos tipos STRINGS. São as cadeias de caracteres. Texto. Tenho dois tipos. O CHAR e o VARCHAR, cujo tamanho máximo são 255 caracteres. A diferença entre eles é os espaços. Por exemplo, se eu digo que tenho um campo VARCHAR(4) e quero armazenar a letra “aa”, ele vai armazenar no banco e vai criar dois espaços vazios “ aa", porque o tamanho do caractere é quatro e eu o mandei gravar somente dois. O que faltar para o tamanho do campo, ele preenche com vazios.

[15:53] Ou seja, o CHAR gasta mais espaço em disco, porque gasta espaço para gravar os vazios, que podem ser desnecessários. Pode ser porque podemos encontrar situações onde gravar o vazio é importante.

[16:13] Já o VARCHAR, se eu mando armazenar “aa", mesmo se o VARCHAR for quatro, ele vai armazenar realmente somente dois caracteres e não vai gastar espaço com outros dois caracteres vazios.

[16:28] Também tenho os strings do tipo binários. O BINARY e o VARBINARY. Os dois têm o mesmo conceito do CHAR e VARCHAR. Um armazena espaços vazios e o outro não. A diferença entre eles é que eu gravo os bytes daquilo que estou armazenando. O tamanho máximo não é caracteres, mas bytes.

[17:05] Tenho também dois outros campos, BLOB e TEXT. No BLOB, tenho TINYBLOB, BLOB, MEDIUMBLOB e LONGBLOB. No TEXT a mesma coisa. Isso tem a ver com os tamanhos máximos. O BLOB é binário, então em um LONGBLOB posso gravar um grande binário dentro do banco. Por exemplo, posso gravar os bytes de um arquivo Word, ou os bytes de uma foto. Já o TEXT vai ser usado para armazenar textos.

[18:03] Outro campo é o ENUM. É como se eu definisse opções. Por exemplo, digo que meu campo se chama size e ele é do tipo ENUM. Quando declaro o campo do tipo ENUM, tenho que colocar dentro dele opções. E só consigo armazenar nesse campo os textos x-small, small, médium, large e x-large. Ou seja, só consigo gravar dentro da minha base de dados essas cinco opções.

[18:45] Dois atributos importantes dos campos do tipo texto são o SET e COLLATE. São muito parecidos com aqueles parâmetros que falei quando vimos banco de dados. Tem a ver com as cadeias de caracteres que vou usar para armazenar o texto. Posso estar querendo armazenar o texto em alfabeto, por exemplo, russo. Ou posso, por exemplo, armazenar um texto com ideogramas japoneses. Tenho que selecionar na definição do campo o SET e o COLLATE específicos da língua que estou usando, porque aí ele vai colocar dentro do campo uma tabela ask maior ou menos dependendo do tipo que você está usando.

[19:46] Alguns campos que só apareceram nas versões mais novas do SQL, os SPACIAL. Hoje em dia, todo mundo quer ver mapa, e preciso gravar pontos nele. Então, posso gravar o ponto através da latitude e longitude dentro do banco MYSQL. Isso seria do tipo POINT. Mas tenho também LINESTRING, que representa uma linha, GEOMETRY e POLYGON, que representam áreas dentro do mapa.

[20:57] Tenho outros tipos, mas quis só mencionar esses para dar um apanhado geral das possibilidades que você tem quando cria uma tabela para definir o tipo de coluna.