**Tipos de Dados no SQL**

**SENAI ROBERTO MANGE**

**Giovanna Marques, Turma DSTB18**

**Tipos de Dados no MySQL:**

No MySQL, escolher o tipo de dado correto para cada coluna é muito importante para garantir a integridade dos dados, otimizar o desempenho do banco e economizar espaço de armazenamento. Os tipos de dados são classificados em quatro categorias principais.

**1. Tipos Numéricos:** Armazenam valores numéricos. A escolha do tipo ideal depende da faixa de valores que você precisa armazenar e se necessita de casas decimais.

**INT (ou INTEGER)**

É o tipo de inteiro mais comum. Armazena números inteiros que vão de aproximadamente -2.1 bilhões a 2.1 bilhões. Se definido como UNSIGNED (sem sinal), a faixa se torna de 0 a 4.2 bilhões, armazenamento: 4 bytes. Ele é Ideal para chaves primárias (id), contadores, quantidades e qualquer número inteiro que não exceda essa faixa. Por exemplo: id\_usuario, quantidade\_estoque.

**BIGINTD**

Usado para números inteiros muito grandes, quando o INT não é suficiente. A faixa é de aproximadamente -9 quintilhões a 9 quintilhões, armazenamento: 8 bytes, em sistemas que podem gerar um volume gigantesco de registros (bilhões ou mais), como logs de sistemas, transações financeiras em larga escala ou IDs de redes sociais.

**TINYINT**

O menor tipo de inteiro, ele armazena valores de -128 a 127. Se UNSIGNED, a faixa vai de 0 a 255, perfeito para dados que têm um número muito limitado e conhecido de estados, como colunas de status ou flags booleanas. Por exemplo, status\_ativo (onde 0 = inativo, 1 = ativo).

**DECIMAL (ou NUMERIC)**

Descrição: Armazena números decimais com precisão exata. É ideal para cálculos onde a precisão é crucial. Você define a precisão total de dígitos e o número de casas decimais. Ex: DECIMAL(10, 2) pode armazenar um número com até 10 dígitos, sendo 2 deles após a vírgula (de -99999999.99 a 99999999.99). Ele é essencial para valores monetários (preços, salários), cálculos financeiros, científicos ou qualquer situação onde erros de arredondamento não são aceitáveis. Por exemplo: preco\_produto, saldo\_conta.

**FLOAT e DOUBLE**

Descrição: Armazenam números decimais com precisão aproximada. DOUBLE tem o dobro da precisão do FLOAT. São mais rápidos para cálculos, mas podem levar a pequenos erros de arredondamento.

Quando usar: Em contextos onde a precisão absoluta não é a maior prioridade e o desempenho é importante. Comum em medições científicas ou cálculos estatísticos que já possuem uma margem de erro. Por exemplo: latitude, longitude, medições de sensores.

**2. Tipos de Texto e Caracteres: Armazenam desde um único caractere até um livro inteiro.**

**CHAR(tamanho)**

Ele armazena uma string de tamanho fixo. Se você definir CHAR(10) e inserir a palavra "oi", ele armazenará "oi" seguido de 8 espaços para completar os 10 caracteres. O tamanho máximo é de 255 caracteres. Essencial para dados que sempre terão o mesmo tamanho, como siglas de estados (UF CHAR(2)), códigos de CEP formatados (cep CHAR(9)) ou códigos de produtos padronizados.´

**VARCHAR(tamanho)**

armazena uma string de tamanho variável, até um limite máximo especificado. VARCHAR(100) pode armazenar até 100 caracteres, mas se você inserir a palavra "oi", ele usará apenas o espaço necessário para "oi" mais um pequeno controle de tamanho. É o tipo mais flexível e comum para textos. Ideal para nomes, e-mails, títulos, senhas, endereços, etc.

**TEXT**

Usado para armazenar textos longos. Não é necessário definir um tamanho máximo (o limite padrão é de 65.535 caracteres).Ele é perfeito para descrições detalhadas de produtos, artigos de blog, comentários de usuários, biografias ou qualquer texto que possa exceder o limite do VARCHAR.

**ENUM**

Descrição: Permite que uma coluna aceite apenas um valor de uma lista pré-definida de opções. Ex: ENUM('Ativo', 'Inativo', 'Pendente').

Quando usar: Ótimo para colunas com um conjunto pequeno e fixo de valores possíveis, como status\_pedido ou nivel\_associacao. Ajuda a manter a consistência dos dados.

**3. Tipos de Data e Hora: Armazenam informações temporais com diferentes níveis de detalhe.**

Armazena uma data no formato 'AAAA-MM-DD'. Para registrar apenas a data, sem se preocupar com a hora, como: data\_nascimento, data\_cadastro, data\_validade.

**TIME**

Armazena uma hora no formato 'HH:MM:SS'. Para registrar apenas a hora do dia ou a duração de um evento. Exemplos: horario\_agendamento, tempo\_de\_prova.

**DATETIME**

Armazena a combinação de data e hora no formato 'AAAA-MM-DD HH:MM:SS'. A faixa de valores é de '1000-01-01 00:00:00' a '9999-12-31 23:59:59'. Ideal para registrar o momento exato de um evento que não depende de fuso horário. Exemplos: data\_emprestimo, data\_publicacao\_artigo.

**TIMESTAMP**

Também armazena data e hora, mas com uma particularidade importante: ele é dependente do fuso horário (timezone). O valor é armazenado internamente como segundos desde a época Unix (1970-01-01 UTC) e é convertido para o fuso horário da sessão quando recuperado. Sua faixa é menor que a do DATETIME. Ele é Perfeito para registrar eventos que precisam ser consistentes globalmente, como data\_criacao\_registro ou data\_ultima\_modificacao.

**4. Tipos Binários: Armazenam dados brutos, como imagens, áudios, PDFs ou qualquer outro tipo de arquivo.**

**BLOB (Binary Large Object)**

É uma família de tipos (TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB) usada para armazenar grandes quantidades de dados binários diretamente no banco de dados.

Quando usar: Embora seja possível armazenar arquivos no banco de dados com BLOB, geralmente não é a prática recomendada. Armazenar arquivos no banco pode aumentar drasticamente seu tamanho, dificultar backups e diminuir o desempenho. A abordagem mais comum e eficiente é salvar o arquivo em um sistema de arquivos (no servidor ou em um serviço de armazenamento como Amazon S3) e guardar apenas o caminho (caminho do arquivo ou URL) em uma coluna VARCHAR no banco de dados. O uso de BLOB é mais indicado para casos específicos onde os dados binários são pequenos e precisam estar atomicamente ligados ao registro.