Banco de Dados I
Modelo Lógico - Parte 1

Gestão da Tecnologia da Informação – 2025.01

Sumário

1 Nomenclatura

A nomenclatura correta de tabelas e colunas no MySQL Workbench é essencial para garantir a organização, a clareza e a manutenção eficiente do banco de dados. Nomes bem escolhidos tornam a estrutura do banco mais intuitiva, facilitando o entendimento tanto para os desenvolvedores atuais quanto para futuros colaboradores. Boas práticas incluem utilizar nomes descritivos, consistentes e padronizados, evitando ambiguidades e abreviações excessivas. Uma nomeação clara também melhora a criação de consultas SQL, reduz erros de interpretação e torna a integração com outras aplicações mais segura e rápida.

Ao trabalhar com nomenclaturas, observe duas questões centrais: tabelas e colunas.

1.1 Tabelas

- Adote nomes no plural para nomear as tabelas
- Certifique-se de usar nomes significativos
- Padronize os nomes usando o mesmo formato (exemplo: minúsculas)
- Caso uma tabela tenha nome composto (exemplo: alunos matriculados) utilize o padrão *snake case*: `alunos_matriculados`
- Não utilize caracteres especiais

1.2 Colunas

- Adote nomes no singular para nomear as colunas
- Certifique-se de usar nomes significativos
- Padronize os nomes usando o mesmo formato (exemplo: minúsculas)
- Caso o nome seja composto (exemplo: data de nascimento) utilize o padrão *snake case*: `data_de_nascimento`
- Não utilize caracteres especiais

2 Tipos de dados

Cada coluna terá que possuir um tipo de dado próprio. Esse tipo irá variar de acordo com a natureza do dado armazenado. Por exemplo, se vai armazenar a idade de uma pessoa (exemplo: 25 anos) é diferente de armazenar a data de nascimento da pessoa (exemplo: 01/01/2000).

Existem diferentes tipos de dados para serem utilizados. No entanto, atualmente, centraremos o foco em três tipos principais: numéricos, *strings* e de data/hora.

2.1 Numéricos

No MySQL Workbench, os dados numéricos podem ser representados por vários tipos, entre eles os mais comuns são `INT` e `DECIMAL`.

2.1.1 INT

O tipo `INT` é utilizado para armazenar números inteiros, ou seja, sem casas decimais. É ideal para representar dados como identificadores, quantidades, idades, etc. O tipo `INT` pode receber valores entre -2147483648 a 2147483647, caso seja aceito valores negativos. O intervalo não assinado (positivo) é 0 a 4294967295. Além disso o tipo `INT` também permite que seja definido um intervalo máximo de números a serem armazenados, INT(n), no qual n significa o valor do intervalo máximo.

2.1.2 DECIMAL

O tipo `DECIMAL` é usado para armazenar números exatos com casas decimais fixas, sendo fundamental para valores financeiros ou situações em que a precisão é essencial. A precisão é Definida por dois parâmetros: DECIMAL(p, s), onde:

- p = precisão total (quantidade de dígitos).
- s = escala (quantidade de dígitos após o ponto decimal).

Exemplo: `DECIMAL(5,2)` pode armazenar valores de -999.99 até 999.99.

2.2 *Strings*

No MySQL Workbench, os tipos de dados `strings` são usados para armazenar cadeias de caracteres em bancos de dados, variando conforme o tamanho e a finalidade das informações. Os principais tipos textuais incluem `CHAR` e `VARCHAR` e suas variantes.

2.2.1 CHAR

O tipo `CHAR` é declarado utilizando um tamanho máximo de caracteres. Por exemplo, `CHAR(40)` significa que aquele campo é capaz de armazenar até 40 caracteres. Caso o valor armazenado no tipo `CHAR` seja menor que o especificado, automaticamente o restante do espaço será preenchido (ver Tabela 1). `CHAR` é utilizado para armazenar textos de tamanho fixo, ideal para campos que sempre possuem a mesma quantidade de caracteres, como códigos ou siglas.

2.2.2 VARCHAR

O tipo `VARCHAR` segue o mesmo funcionamento do `CHAR`. No entanto, não preenche o total do espaço caso existam menos caracteres que o limite estabelecido (ver Tabela 1). `VARCHAR` é empregado para textos de tamanho variável, sendo mais utilizado para armazenar informações como nomes ou descrições que podem variar em sua extensão.

2.2.3 Exemplo

Valor	CHAR(4)	VARCHAR(4)
''	' '	''
'ab'	'ab '	'ab'
'abcd'	'abcd'	'abcd'
'abcdefgh'	'abcd'	'abcd'

Tabela 1: Fonte: https://dev.mysql.com/doc/refman/8.4/en/char.html

Diferente do exposto na última linha, pode ser que seja gerado um erro ao armazenar valores inválidos no campo.

2.3 Data/Hora

No MySQL, os tipos de dados `DATE`, `TIME` e `DATETIME` são usados para armazenar informações relacionadas a datas e horários, cada um com características distintas.

2.3.1 DATE

O tipo `DATE` armazena apenas a data, sem informações sobre o horário. Ele segue o formato `YYYY-MM-DD`, onde:

- `YYYY` representa o ano (com 4 dígitos),
- `MM` representa o mês (com 2 dígitos),
- `DD` representa o dia (com 2 dígitos).

Exemplo: `'2025-04-28'`.

2.3.2 TIME

O tipo `TIME` armazena apenas a hora, sem a data. Ele é utilizado para armazenar uma quantidade de tempo ou um horário específico, no formato `HH:MM:SS`, onde:

- `HH` representa as horas (com 2 dígitos),
- `MM` representa os minutos (com 2 dígitos),
- `SS` representa os segundos (com 2 dígitos).

Exemplo: `'14:30:00'`.

2.3.3 DATETIME

O tipo `DATETIME` combina a data e o horário em um único valor, no formato `YYYY-MM-DD HH:MM:SS`. Ele é útil quando se deseja registrar um momento exato, incluindo tanto a data quanto o horário. Exemplo: `'2025-04-28 14:30:00'`.

3 Integridade

A integridade dos dados é um princípio fundamental em bancos de dados, garantindo que as informações armazenadas sejam corretas, consistentes e confiáveis ao longo do tempo. Ela é mantida através de restrições aplicadas nas tabelas, que evitam erros, duplicidades e inconsistências.

Além disso, também existe a integridade de domínio, a qual controla que os dados inseridos estejam dentro de um conjunto válido de valores, como tipos corretos (números, datas, textos) e regras específicas.

Ao modelar as colunas no MySQL Workbench, é possível definir a integridade em diferentes níveis. Inicialmente, pode-se observar as regras gerais para garantir a integridade do domínio da aplicação, como ilustrado abaixo:

Sigla	**Significado**	**Descrição**
PK	Primary Key	Define a chave primária da tabela. Garante unicidade.
NN	Not Null	Indica que a coluna não pode conter valores nulos. O valor deve ser obrigatoriamente informado.
UQ	Unique	Garante que todos os valores da coluna sejam únicos. Permite múltiplos valores nulos. Gera erro ao tentar inserir duplicatas.
BIN	Binary	Coluna armazena cadeias de bytes (binárias), e não texto. As comparações e ordenações são feitas com base nos valores numéricos dos bytes.
UN	Unsigned	Define a coluna como sem sinal (unsigned), ou seja, somente aceita valores positivos, a partir de zero.
ZF	Zerofill	Preenche os números com zeros à esquerda até atingir o comprimento definido. Exemplo: 5 em uma coluna INT(4) vira 0005.
AI	Auto Increment	Usado geralmente em chaves primárias para gerar identificadores únicos automaticamente a cada nova inserção.
G	Generated Column	Coluna gerada automaticamente a partir de expressões envolvendo outras colunas. Pode ser virtual (calculada) ou armazenada (persistida).

Referências

- https://dev.mysql.com/doc/refman/8.4/en/date-and-time-types.html
- https://dev.mysql.com/doc/refman/8.4/en/numeric-types.html
- https://dev.mysql.com/doc/refman/8.4/en/char.html
- https://dev.mysql.com/doc/refman/8.0/en/create-table.html
- https://www.tutorialspoint.com/what-do-column-flags-mean-in-mysql-workbench
- https://ort.com.br/blog/2024/04/17/o-que-significam-esses-codigos-no-mysql-pk-nn-uq-b-un-zf-ai-e-g/