UC 2 -IMPLEMENTAR BANCO DE DADOS

Programador de Sistemas

27 de março de 2025

Prof Esp. Cesar Ricardo Velasque Trindade

Aula 3: Structured Query Language (SQL) - Linguagem de Consulta Estruturada: conceitos, sintaxe.

Senac Fecomércio Sesc

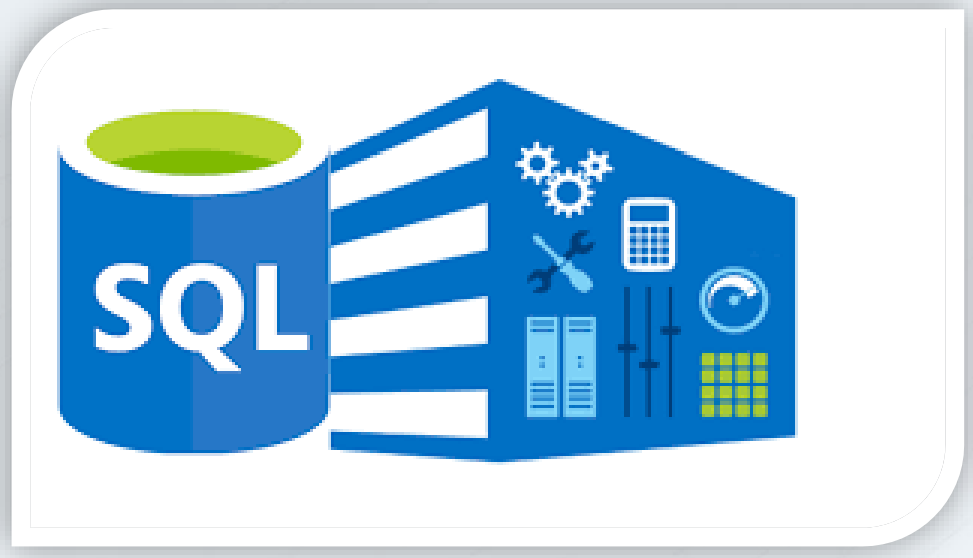
Tipos de dados

- O SQL suporta tipos de campos que podem ser agrupados em três categorias:

Tipos numéricos

Tipos de data e hora

Tipos string (caracteres)



SQL

TIPOS NUMÉRICOS

Tipos numéricos

TINYINT

Um inteiro muito pequeno. A faixa deste inteiro com sinal é de -128 até 127. A faixa sem sinal é de 0 até 255.
O intervalo de valores que pode armazenar depende de se ele é definido como UNSIGNED ou não. Se for UNSIGNED, ele só pode armazenar valores não negativos.
Comumente usado para armazenar valores pequenos, como códigos de identificação, indicadores booleanos (0 para falso e 1 para verdadeiro) ou códigos de status.

```sql
CREATE TABLE exemplo (
    id TINYINT,
    valor TINYINT UNSIGNED
);

INSERT INTO exemplo (id, valor) VALUES (-10, 100), (5, 200), (0, 50);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "valor". A coluna "id" é definida como TINYINT, o que significa que pode armazenar valores inteiros de -128 a 127. A coluna "valor" é definida como TINYINT UNSIGNED, o que significa que pode armazenar valores inteiros de 0 a 255.

Tipos numéricos

SMALLINT

Usado para armazenar valores inteiros em situações onde o intervalo de valores pode exceder o que o TINYINT pode suportar, mas um INT é considerado excessivo em termos de espaço de armazenamento.
A faixa do inteiro com sinal é de -32768 até 32767. A faixa sem sinal é de 0 a 65535

```sql
CREATE TABLE exemplo (
    id SMALLINT,
    quantidade SMALLINT UNSIGNED
);

INSERT INTO exemplo (id, quantidade) VALUES (-1000, 30000), (5000, 15000), (0, 2000);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "quantidade". A coluna "id" é definida como SMALLINT, o que significa que pode armazenar valores inteiros de -32768 a 32767. A coluna "quantidade" é definida como SMALLINT UNSIGNED, o que significa que pode armazenar valores inteiros de 0 a 65535.

Tipos numéricos

MEDIUMINT

Um inteiro de tamanho médio. A faixa com sinal é de -8388608 a 8388607. A faixa sem sinal é de 0 a 16777215.
Requer 3 bytes de armazenamento, o que o torna mais eficiente em termos de espaço do que um INT padrão, que requer 4 bytes.

```sql
CREATE TABLE exemplo (
    id MEDIUMINT,
    quantidade MEDIUMINT UNSIGNED
);

INSERT INTO exemplo (id, quantidade) VALUES (-5000000, 10000000), (2000000, 15000000);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "quantidade". A coluna "id" é definida como MEDIUMINT, o que significa que pode armazenar valores inteiros no intervalo de -8388608 a 8388607. A coluna "quantidade" é definida como MEDIUMINT UNSIGNED, o que significa que pode armazenar valores inteiros de 0 a 16777215.

Tipos numéricos

INT

INTEGER

Um inteiro de tamanho normal. A faixa com sinal é de -2147483648 a 2147483647. A faixa sem sinal é de 0 a 4294967295.
Ocupa 4 bytes de armazenamento

```sql
CREATE TABLE exemplo (
    id INT,
    quantidade INT
);

INSERT INTO exemplo (id, quantidade) VALUES (1001, 500), (2002, 1000), (3003, 750);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "quantidade". Ambas as colunas são definidas como INT. Os valores inseridos nas colunas "id" e "quantidade" são exemplos de números inteiros, como 1001, 2002, 3003, 500, 1000 e 750. Esses valores estarão dentro do intervalo suportado pelo tipo INT.

Tipos numéricos

BIGINT

Usado para armazenar números inteiros com um intervalo muito amplo. Ele ocupa 8 bytes de armazenamento.
Pode armazenar valores no intervalo de -9223372036854775808 a 9223372036854775807.
A faixa sem sinal é de 0 a 18446744073709551615.

```sql
CREATE TABLE exemplo (
    id BIGINT,
    populacao BIGINT
);

INSERT INTO exemplo (id, populacao) VALUES (1000000001, 300000000);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "populacao". Ambas as colunas são definidas como BIGINT. Os valores inseridos nas colunas "id" e "populacao" são exemplos de números inteiros muito grandes.

Tipos numéricos

FLOAT

Usado para armazenar números de ponto flutuante de precisão simples.
Pode armazenar números decimais com uma precisão limitada.
Útil para valores numéricos que podem incluir casas decimais e não requerem alta precisão.

```sql
CREATE TABLE exemplo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    preco FLOAT
);

INSERT INTO exemplo (preco) VALUES (10.99), (5.75), (8.50);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "preco". A coluna "preco" é definida como FLOAT e é usada para armazenar valores numéricos representando preços de produtos, por exemplo.

Os valores inseridos na coluna "preco" são números decimais, como 10.99, 5.75 e 8.50. Esses valores serão armazenados como números de ponto flutuante

Tipos numéricos

DOUBLE

DOUBLE PRECISION

Usado para armazenar números de ponto flutuante de precisão dupla.
Oferece uma capacidade maior de representar números decimais com alta precisão.
Comumente usado quando a precisão adicional é necessária para representar números decimais com alta precisão, como em cálculos científicos, financeiros ou de engenharia.

```sql
CREATE TABLE exemplo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    valor DOUBLE
);

INSERT INTO exemplo (valor) VALUES (123.456), (789.123), (456.789);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "valor". A coluna "valor" é definida como DOUBLE e é usada para armazenar valores numéricos de precisão dupla, como números decimais.

Os valores inseridos na coluna "valor" são exemplos de números decimais, como 123.456, 789.123 e 456.789.

Tipos numéricos

DECIMAL

NUMERIC

Usado para armazenar números decimais com precisão fixa.
Usado quando a precisão exata é necessária, como em aplicações financeiras.
Armazena números decimais com uma precisão exata especificada pelo usuário.
Não arredonda ou aproxima os valores. Ele armazena exatamente o valor especificado.

```sql
CREATE TABLE exemplo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    preco DECIMAL(10, 2)
);

INSERT INTO exemplo (preco) VALUES (123.45), (789.12), (456.78);
```

Neste exemplo, criamos uma tabela chamada "exemplo" com duas colunas: "id" e "preco". A coluna "preco" é definida como DECIMAL(10, 2), o que significa que pode armazenar números decimais com até 10 dígitos, dos quais 2 são para a parte decimal.

Os valores inseridos na coluna "preco" são exemplos de números decimais, como 123.45, 789.12 e 456.78

SQL

TIPOS DE DATA E HORA

Tipos de data e hora

DATE

Usado para armazenar datas sem informações de tempo. Ele segue o formato "YYYY-MM-DD" (ano-mês-dia) e pode armazenar datas no intervalo de '1000-01-01' a '9999-12-31'.
Útil para armazenar datas de nascimento, datas de eventos, datas de transações e assim por diante.
Amplamente utilizado em aplicativos de bancos de dados onde datas precisam ser armazenadas e manipuladas, como em sistemas de reservas, calendários e aplicativos de gerenciamento de tarefas.

Neste exemplo, criamos uma tabela chamada "eventos" com três colunas: "id", "nome_evento" e "data_evento". A coluna "data_evento" é definida como DATE e é usada para armazenar a data do evento. Os valores inseridos nessa coluna estão no formato 'YYYY-MM-DD', como '2000-12-25', '2023-03-18' e '2024-07-10'.

```sql
CREATE TABLE eventos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome_evento VARCHAR(100),
    data_evento DATE
);

INSERT INTO eventos (nome_evento, data_evento) VALUES
    ('Aniversário', '2000-12-25'),
    ('Reunião', '2023-03-18'),
    ('Conferência', '2024-07-10');
```

Tipos de data e hora

TIMESTAMP

Usado para armazenar um carimbo de data/hora que representa um ponto específico no tempo.
Usado frequentemente para registrar o momento em que uma linha foi inserida ou atualizada em uma tabela.
Segue o formato 'YYYY-MM-DD HH:MM:SS' (ano-mês-dia hora:minuto:segundo).

```sql
CREATE TABLE registros (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    data_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO registros (nome) VALUES
    ('Usuário1'),
    ('Usuário2'),
    ('Usuário3');
```

Neste exemplo, criamos uma tabela chamada "registros" com três colunas: "id", "nome" e "data_registro". A coluna "data_registro" é definida como TIMESTAMP e é configurada para se auto-atualizar sempre que uma nova linha é inserida na tabela, usando o valor atual do servidor MySQL. Os valores inseridos nessa coluna serão registrados automaticamente com a data/hora da inserção.

Tipos de data e hora

TIME

Usado para armazenar valores de tempo sem data.
O formato para um valor TIME é 'HH:MM:SS' (hora:minuto:segundo).
Comumente usado em sistemas de gerenciamento de bancos de dados para armazenar informações de tempo, como durações de tarefas, horários de funcionamento de estabelecimentos, entre outros.

Neste exemplo, criamos uma tabela chamada "tarefas" com três colunas: "id", "descricao" e "duracao". A coluna "duracao" é definida como TIME e é usada para armazenar a duração estimada de cada tarefa. Os valores inseridos nessa coluna estão no formato 'HH:MM:SS', como '02:30:00' para uma reunião de equipe com duração de 2 horas e 30 minutos.

```sql
CREATE TABLE tarefas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    descricao VARCHAR(100),
    duracao TIME
);

INSERT INTO tarefas (descricao, duracao) VALUES
    ('Reunião de equipe', '02:30:00'),
    ('Treino na academia', '01:15:00'),
    ('Almoço', '00:45:00');
```

Tipos de data e hora

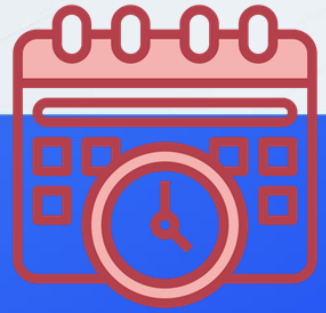
YEAR

Armazena apenas valores de ano, sem incluir informações de mês ou dia.
É útil para armazenar anos em aplicações onde apenas o ano é relevante, como datas de eventos anuais, anos de nascimento, entre outros.
O intervalo de valores que um YEAR pode armazenar é de 1901 a 2155, ou o valor especial '0000', que representa um ano não especificado ou nulo.

```sql
CREATE TABLE eventos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome_evento VARCHAR(100),
    ano_evento YEAR
);

INSERT INTO eventos (nome_evento, ano_evento) VALUES
    ('Olimpíadas', '2024'),
    ('Exposição de arte', '2022'),
    ('Conferência internacional', '2023');
```

Neste exemplo, criamos uma tabela chamada "eventos" com três colunas: "id", "nome_evento" e "ano_evento". A coluna "ano_evento" é definida como YEAR e é usada para armazenar o ano em que ocorrerá o evento. Os valores inseridos nessa coluna estão no formato de 4 dígitos, como '2024', '2022' e '2023'.



SQL

TIPOS STRING

Tipos String

CHAR

Usado para armazenar cadeias de caracteres de comprimento fixo.
É preciso especificar um tamanho máximo para o campo.
Sempre ocupará esse tamanho, preenchendo os espaços restantes com espaços em branco, se necessário.
Se a string armazenada for menor que o tamanho máximo definido para o campo CHAR, o restante do espaço será preenchido com espaços em branco.
Comumente usado para armazenar valores de tamanho fixo, como códigos de identificação, abreviações ou campos que sempre terão o mesmo comprimento.

Neste exemplo, criamos uma tabela chamada "produtos" com duas colunas: "codigo" e "nome". A coluna "codigo" é definida como CHAR(5), o que significa que sempre ocupará 5 caracteres de comprimento, mesmo que a string inserida seja menor. Os valores inseridos nesta coluna estão no formato 'A123', 'B456' e 'C789', onde o campo CHAR sempre ocupará 5 caracteres.

```sql
CREATE TABLE produtos (
    codigo CHAR(5),
    nome VARCHAR(100)
);

INSERT INTO produtos (codigo, nome) VALUES
    ('A123', 'Produto 1'),
    ('B456', 'Produto 2'),
    ('C789', 'Produto 3');
```

Tipos String

VARCHAR

Usado para armazenar cadeias de caracteres de comprimento variável.
É preciso especificar um tamanho máximo para o campo VARCHAR, mas ele só ocupará o espaço necessário para armazenar a string.
Comumente usado para armazenar valores de texto de tamanho variável, como nomes de pessoas, endereços de e-mail, descrições de produtos e assim por diante.

```sql
CREATE TABLE clientes (
    nome VARCHAR(50),
    email VARCHAR(100)
);

INSERT INTO clientes (nome, email) VALUES
    ('João da Silva', 'joao@example.com'),
    ('Maria Oliveira', 'maria@example.com'),
    ('Pedro Santos', 'pedro@example.com');
```

Neste exemplo, criamos uma tabela chamada "clientes" com duas colunas: "nome" e "email". Ambas as colunas são definidas como VARCHAR, onde "nome" tem um tamanho máximo de 50 caracteres e "email" tem um tamanho máximo de 100 caracteres. Os valores inseridos nestas colunas são strings de comprimento variável, como nomes e endereços de e-mail. O campo VARCHAR alocará espaço apenas para o tamanho necessário para armazenar cada string.

Tipos String

TEXT

Usado para armazenar strings de texto de comprimento variável, que podem ser muito grandes.
Pode armazenar grandes blocos de texto, como conteúdo de artigos, postagens de blog, comentários, entre outros.
Frequentemente usado em aplicações web para armazenar conteúdo de texto longo, como postagens de blog, descrições de produtos, mensagens de e-mail e assim por diante.

```sql
CREATE TABLE posts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(100),
    conteudo TEXT
);

INSERT INTO posts (titulo, conteudo) VALUES
    ('Postagem 1', 'Este é o conteúdo da postagem 1.'),
    ('Postagem 2', 'Este é o conteúdo da postagem 2. Ela pode ser muito mais longa do que a postagem 1.'),
    ('Postagem 3', 'Este é o conteúdo da postagem 3. Ela também pode ser bastante longa.');
```

Neste exemplo, criamos uma tabela chamada "posts" com três colunas: "id", "titulo" e "conteudo". A coluna "conteudo" é definida como TEXT e é usada para armazenar o conteúdo das postagens. Os valores inseridos nessa coluna podem ser de comprimento variável, permitindo armazenar grandes blocos de texto, como o conteúdo de postagens de blog.

Tipos String

ENUM

Usado para definir um conjunto de valores possíveis para uma coluna. Permite escolher apenas um valor de uma lista predefinida de opções. Restringe os valores que podem ser armazenados na coluna às opções predefinidas, o que pode ajudar a garantir a integridade dos dados.

Neste exemplo, criamos uma tabela chamada "alunos" com três colunas: "id", "nome" e "genero". A coluna "genero" é definida como ENUM e especifica duas opções possíveis: 'Masculino' e 'Feminino'. Quando inserimos registros na tabela, podemos escolher apenas uma das opções pré-definidas para o gênero de cada aluno. Isso garante que apenas valores válidos sejam inseridos na coluna "genero".

```sql
CREATE TABLE alunos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    genero ENUM('Masculino', 'Feminino')
);

INSERT INTO alunos (nome, genero) VALUES
    ('João', 'Masculino'),
    ('Maria', 'Feminino'),
    ('Alex', 'Masculino');
```

Tipos String

SET

Usado para armazenar um conjunto de valores que podem ser escolhidos em uma lista predefinida de opções.
Semelhante ao tipo ENUM, mas permite escolher mais de uma opção da lista.
Permite a seleção de múltiplos valores.

```sql
CREATE TABLE pedidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    produtos VARCHAR(100),
    extras SET('Molho', 'Queijo', 'Bacon', 'Cebola', 'Tomate')
);

INSERT INTO pedidos (produtos, extras) VALUES
    ('Hambúrguer', 'Queijo,Molho'),
    ('Pizza', 'Queijo,Tomate'),
    ('Sanduíche', 'Bacon,Cebola');
```

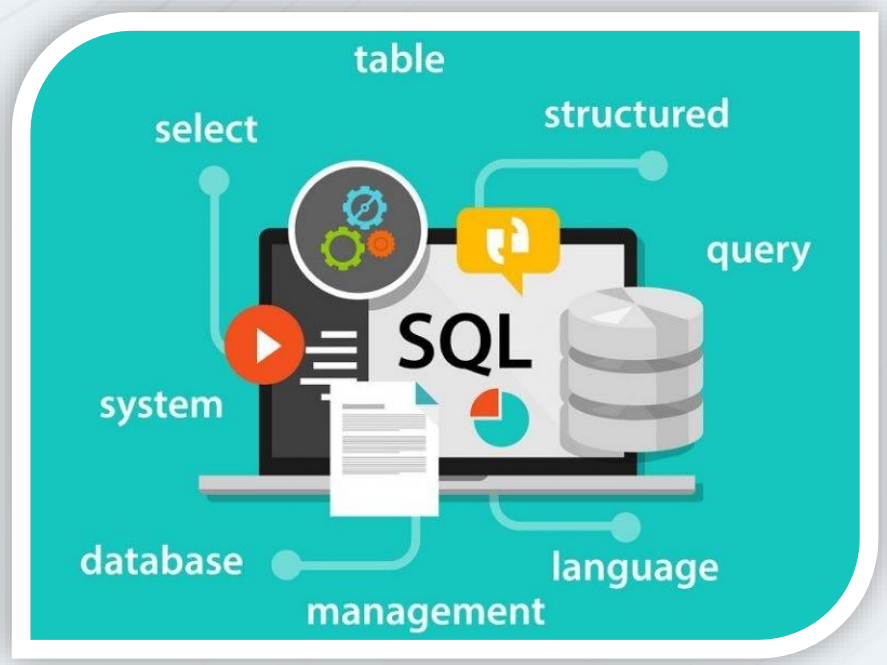
Neste exemplo, criamos uma tabela chamada "pedidos" com três colunas: "id", "produtos" e "extras". A coluna "extras" é definida como SET e especifica cinco opções possíveis: 'Molho', 'Queijo', 'Bacon', 'Cebola' e 'Tomate'. Quando inserimos registros na tabela, podemos escolher várias opções para os extras de cada pedido, separando-as por vírgula. Por exemplo, o primeiro pedido inclui queijo e molho como extras.

Escolhendo o tipo de dado correto para cada atributo da tabela

Para um uso mais eficiente do armazenamento, tente usar o tipo mais adequado em todos os casos.

Por exemplo, se um campo de inteiro for usado para valores em uma faixa entre 1 e 99999, MEDIUMINT UNSIGNED é o melhor tipo.

Representação precisa de valores monetários é um problema comum. Use o tipo DECIMAL.
Se a precisão não é tão importante, o tipo DOUBLE pode ser satisfatório.



Senac Fecomércio Sesc

SC.SENAC.BR