Prof. Gean
Disciplina: Banco de Dados

Tutorial MySQL

Aula 01 - MySQL – Como criar um Banco de Dados

Podemos criar um novo banco de dados com o comando a seguir:

```
CREATE DATABASE [IF NOT EXISTS] nome_BD;
```

Onde nome_BD é o nome que queremos dar ao banco de dados. O elemento **IF NOT EXISTS** é opcional. Ele previne o erro de tentar criar um banco de dados que já existe no servidor. Não é possível ter dois bancos de dados com o mesmo nome.

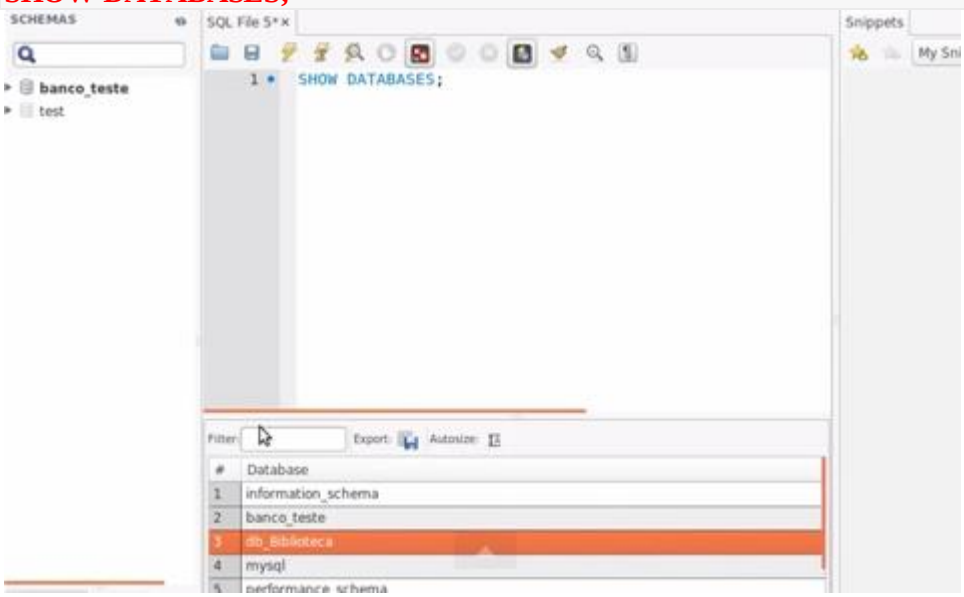
Exemplo – Vamos criar um banco de dados de nome db_Biblioteca, que será utilizado nos demais exemplos do curso:

```
CREATE DATABASE db_Biblioteca;
```

Verificar Banco de Dados

Podemos verificar os bancos de dados existentes com o comando SHOW DATABASES;:

```
SHOW DATABASES;
```



Verificando o banco de dados criado no MySQL

Comando USE

O comando USE instrui o SGBDR a utilizar o banco de dados especificado para rodar os comandos.

Sintaxe:

```
USE nome_banco_de_dados;
```

Para visualizar o banco de dados selecionado no momento use o comando:

```
SELECT DATABASE();
```

Excluir Banco de Dados

Podemos excluir um banco de dados existente com o comando DROP DATABASE;:

```
DROP DATABASE [IF EXISTS] nome_BD;
```

Aula 02 – MySQL – Constraints (Restrições) Primary Key, FK, Default, etc

SQL Constraints (Restrições) no MySQL

- As Restrições são regras aplicadas nas colunas de uma tabela.
- São usadas para limitar os tipos de dados que são inseridos.
- Podem ser especificadas no momento de criação da tabela (CREATE) ou após a tabela ter sido criada (ALTER)

As principais constraints são as seguintes:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- DEFAULT

NOT NULL

- A constraint NOT NULL impõe a uma coluna a NÃO aceitar valores NULL.
- Ou seja, a constraint NOT NULL obriga um campo a sempre possuir um valor.
- Deste modo, não é possível inserir um registro (ou atualizar) sem entrar com um valor neste campo.

UNIQUE

- A restrição UNIQUE identifica de forma única cada registro em uma tabela de um banco de dados.
- As constraints UNIQUE e PRIMARY KEY garantem a unicidade em uma coluna ou conjunto de colunas.
- Uma constraint PRIMARY KEY automaticamente possui uma restrição UNIQUE definida, portanto não é necessário especificar essa constraint neste caso.
- É possível termos várias constraints UNIQUE em uma mesma tabela, mas apenas uma Chave Primária por tabela (lembrando que uma PK pode ser composta, ou seja, constituída por mais de uma coluna – mas ainda assim, será uma única chave primária).

PRIMARY KEY

- A restrição PRIMARY KEY (Chave Primária) identifica de forma única cada registro em uma tabela de banco de dados.

- As Chaves Primárias devem sempre conter valores únicos.
- Uma coluna de chave primária não pode conter valores NULL
- Cada tabela deve ter uma chave primária e apenas uma chave primária.

FOREIGN KEY

Uma FOREIGN KEY (Chave Estrangeira) em uma tabela é um campo que aponta para uma chave primária em outra tabela. Desta forma, é usada para criar os relacionamentos entre as tabelas no banco de dados.

Veja um exemplo de restrição Foreign Key aplicada:

```
CONSTRAINT fk_ID_Autor FOREIGN KEY (ID_Autor)
REFERENCES tbl_autores(ID_Autor)
```

Neste exemplo a chave primária está na tabela tbl_autores e uma chave estrangeira de nome ID_Autor foi criada na tabela atual, usando o nome fk_ID_Autor

DEFAULT

- A restrição DEFAULT é usada para inserir um valor padrão especificado em uma coluna.
- O valor padrão será adicionado a todos os novos registros caso nenhum outro valor seja especificado na hora de inserir dados.

Aula 03 - MySQL – Criação de Tabelas

Declaração CREATE TABLE – criando tabelas no MySQL

Para criar tabelas em um banco de dados, usamos a declaração **CREATE TABLE**. Veja a sintaxe abaixo:

```
CREATE TABLE [IF NOT EXISTS] nome_tabela (
    coluna tipo_dados constraints
    coluna tipo_dados constraints
    coluna tipo_dados constraints
    ...
);
```

Note que precisamos especificar, além do nome da tabela, os nomes das colunas que a comporão e também seus respectivos tipos de dados, além das eventuais *constraints*.

Tipos de Dados no MySQL

A tabela abaixo resume os tipos de dados mais comuns no MySQL, que podem ser usados na criação de tabelas, para estabelecer o tipo de cada coluna:

Tipo	Descrição

INT	Inteiros entre -2,147,483,648 e 2,147,483,647
TINYINT	Números inteiros de -128 a 127
SMALLINT	Números inteiros de -32768 a 32767
MEDIUMINT	Números inteiros de -8388608 a 8388607
BIGINT	Números entre -9,223,372,036,854,775,808 e 9,223,372,036,854,775,807
DECIMAL(M,D)	Ponto decimal com M dígitos no total (precisão) e D casas decimais (escala); o padrão é 10,0; M vai até 65 e D até 30.
FLOAT(M,D)	Ponto flutuante com precisão M e escala D; o padrão é 10,2; D vai até 24.
CHAR(M)	String que ocupa tamanho fixo entre 0 e 255 caracteres
BOOL / BOOLEAN	Valores binários 0 / 1; Na verdade, é um alias para o tipo TINYINT(1)
VARCHAR(M)	String de tamanho variável, com até 65535 M caracteres.
BLOB / MEDIUMBLOB/ TINYBLOB	Campo com tamanho máximo de 65535 caracteres binários; 'Binary Large Objects', são usados para armazenar grandes quantidades de dados, como imagens.
MEDIUMTEXT	Permite armazenar até 16.777.215 caracteres.
LONGTEXT	Permite armazenar até 4.294.967.295 caracteres.
DATE	Uma data de 01/01/1000 a 31/12/9999, no formato YYYY-MM-DD
DATETIME	Uma combinação de data e hora de 01/01/1000 00:00:00 a 31/12/9999 23:59:59, no formato YYYY-MM-DD HH:MM:SS
TIME	Hora apenas, no formato HH:MM:SS
YEAR(M)	Ano nos formatos de 2 ou 4 dígitos; Se forem 2 (YEAR(2)), ano vai de 1970 a 2069; para 4 (YEAR(4)), vai de 1901 a 2155. O padrão é 4.

Vamos passar agora à criação de tabelas em nosso banco de dados db_biblioteca:

Criando tabelas no Banco de Dados

Primeiro vamos criar uma tabela para os livros, de nome **tbl_livro**. Usaremos o bloco de códigos a seguir:

```
USE db_Biblioteca;
CREATE TABLE IF NOT EXISTS tbl_livro (
ID_Livro SMALLINT  AUTO_INCREMENT PRIMARY KEY,
Nome_Livro VARCHAR(70) NOT NULL,
ISBN13 CHAR(13),
ISBN10 CHAR(10),
```

```
ID_Categoria SMALLINT,
ID_Autor SMALLINT NOT NULL,
Data_Pub DATE NOT NULL,
Preco_Livro DECIMAL(6,2) NOT NULL
);
```

Agora, vamos criar uma tabela para armazenar os dados dos autores:

```
CREATE TABLE tbl_autores (
ID_Autor SMALLINT PRIMARY KEY,
Nome_Autor VARCHAR(50) NOT NULL,
Sobrenome_Autor VARCHAR(60) NOT NULL
);
```

Uma para armazenar as categorias de livros:

```
CREATE TABLE tbl_categorias (
ID_Categoria SMALLINT PRIMARY KEY,
Categoria VARCHAR(30) NOT NULL
);
```

E, por fim, criamos uma tabela para armazenar os dados das editoras:

```
CREATE TABLE tbl_editoras (
ID_Editora SMALLINT PRIMARY KEY AUTO_INCREMENT,
Nome_Editora VARCHAR(50) NOT NULL
);
```

As tabelas criadas ainda não estão relacionadas entre si.

Aula 04 - MySQL – Auto Incremento de valores em colunas

Auto Incremento em colunas (AUTO_INCREMENT)

- O auto incremento permite que um número único seja gerado quando um novo registro é inserido em uma tabela.
- Em MYSQL trata-se da palavra chave AUTO_INCREMENT, cujo valor inicial padrão é 1, e se incrementa de 1 em 1.
- Para que o valor da coluna se inicie em 100, por exemplo, após criar a tabela e especificar a coluna que usará auto incremento execute o comando a seguir:

```
ALTER TABLE tabela AUTO_INCREMENT = 100;
```

- Ao inserir valores na tabela, não é necessário especificar o valor para a coluna de auto-incremento.
- Só é permitido usar uma coluna de auto incremento por tabela, geralmente do tipo inteiro.
- Necessita também da constraint NOT NULL (configurado automaticamente)

Auto Incremento – Exemplo

Vamos criar uma tabela de nome tbl_teste_incremento para estudarmos o uso do auto incremento em colunas. Use o código a seguir para isso:

```
CREATE TABLE tbl_teste_incremento (
Codigo SMALLINT PRIMARY KEY AUTO_INCREMENT,
Nome VARCHAR(20) NOT NULL
) AUTO_INCREMENT = 15;
```

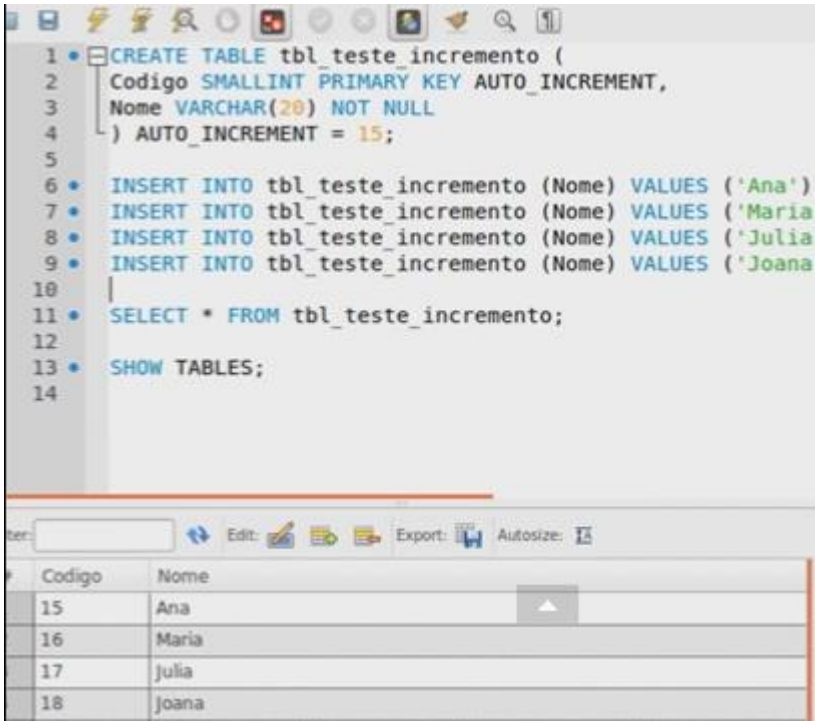
Agora, vamos inserir dados aleatórios na tabela para realizarmos o teste. Não se preocupe se não entender o código, pois esses comandos serão estudados nas próximas aulas:

```
INSERT INTO tbl_teste_incremento (Nome) VALUES ('Ana');
INSERT INTO tbl_teste_incremento (Nome) VALUES ('Maria');
INSERT INTO tbl_teste_incremento (Nome) VALUES ('Julia');
INSERT INTO tbl_teste_incremento (Nome) VALUES ('Joana');
```

E, finalmente, verificamos se o auto invremento funcionou executando uma consulta na tabela:

```
SELECT * FROM tbl_teste_incremento;
```

Veja o resultado no MySQL Workbench:



Verificar o valor atual do auto incremento

Podemos verificar o valor de incremento mais atual armazenado em uma tabela no banco de dados com o comando a seguir:

```
SELECT MAX (nome_coluna)
FROM tabela
```

Exemplo no banco de dados db_biblioteca:

```
SELECT MAX(ID_Livro)
```

```
FROM tbl_livro;
```

Alterar o próximo valor no campo de Auto-Incremento

Para alterar o valor de incremento do próximo registro a ser armazenado em uma tabela, use o comando a seguir:

```
ALTER TABLE tabela AUTO_INCREMENT = valor;
```

Ex.:- Para usar o valor 90 a partir do próximo registro:

```
ALTER TABLE tbl_teste_incremento
AUTO_INCREMENT = 90;
```

Aula 05 MySQL – Alterar Tabelas – comando ALTER TABLE

Alterar Tabelas e Colunas no MySQL

- É possível alterar a estrutura de uma tabela no MySQL após ter sido criada, acrescentando ou excluindo atributos (campos)

Usamos para isso o comando **ALTER TABLE**

Excluir colunas: ALTER TABLE – DROP

```
ALTER TABLE nome-tabela
DROP COLUMN nome-coluna;
```

Exemplo – Excluindo a coluna ID_autor da tabela tbl_livro:

```
ALTER TABLE tbl_livro
DROP COLUMN ID_autor;
```

Pode-se excluir uma chave primária também:

```
ALTER TABLE tabela
DROP PRIMARY KEY;
```

Adicionar colunas: ALTER TABLE – ADD

Sintaxe:

```
ALTER TABLE tabela
ADD  coluna tipo_dados constraints;
```

Exemplo no banco de dados db_biblioteca: Vamos adicionar a coluna id_autor na tabela tbl_livro, com a constraint de chave estrangeira (a chave primária está na tabela tbl_autores):

```
ALTER TABLE tbl_livro
ADD  ID_Autor  SMALLINT NOT NULL;
```

Vamos adicionar a chave estrangeira agora:

```
ALTER TABLE tbl_livro
ADD CONSTRAINT fk_ID_Autor
FOREIGN KEY (ID_Autor)
REFERENCES tbl_autores (ID_autor)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

Continuando os exemplos, vamos adicionar à tabela tbl_livro a coluna id_editora:

```
ALTER TABLE tbl_livro
ADD  ID_editora  SMALLINT NOT NULL;
```

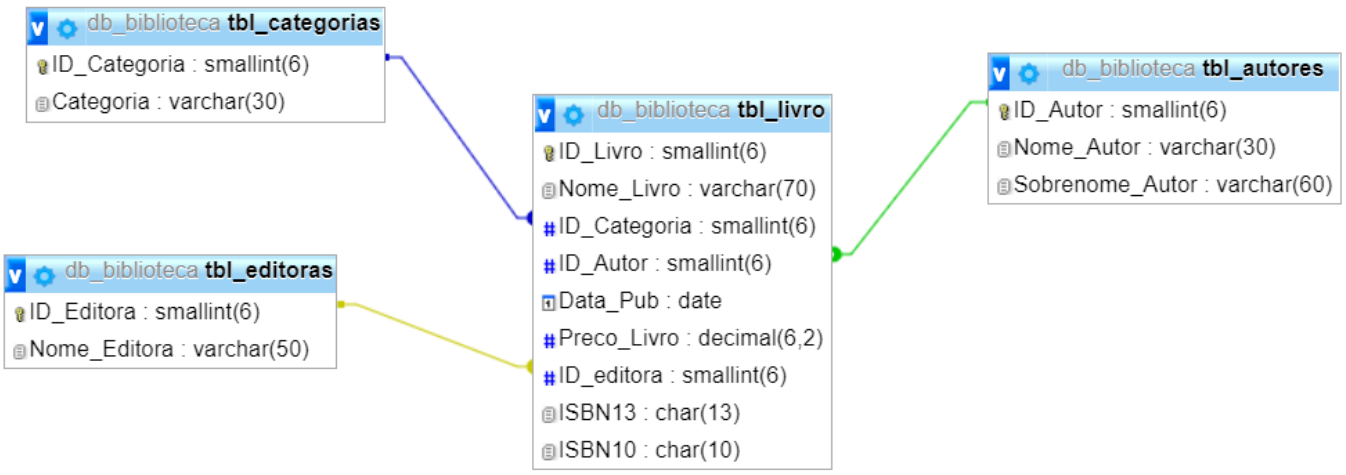
E fazê-la chave estrangeira da tabela tbl_editoras:

```
ALTER TABLE tbl_Livro
ADD CONSTRAINT fk_id_editora
FOREIGN KEY (ID_editora)
REFERENCES tbl_editoras (ID_editora)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

Também vamos criar o relacionamento entre a tabela de livros e a tabela de categorias:

```
ALTER TABLE tbl_Livro
ADD CONSTRAINT fk_id_categoria
FOREIGN KEY (ID_Categoria)
REFERENCES tbl_categorias (ID_Categoria)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

Após criar os relacionamentos entre as tabelas, teremos a estrutura mostrada no diagrama entidade-relacionamento a seguir:



Alterar uma tabela

Podemos alterar colunas, por exemplo mudando o tipo de dados:

```
ALTER TABLE tbl_Livro
MODIFY COLUMN ID_Autor SMALLINT;
```

ALTER TABLE – ADD PK

Podemos também alterar uma coluna, adicionando constraints, como uma chave primária. Veja como ficaria a adição da restrição de chave primária a uma coluna de nome ID_Cliente em uma tabela fictícia chamada Clientes:

```
ALTER TABLE Clientes
ADD PRIMARY KEY (ID_Cliente)
```

Obs. A coluna ID_Cliente deve existir antes de ser transformada em chave primária.

A coluna ID_Cliente receberá a Constraint (restrição) PRIMARY KEY, e passará a ser a chave primária da tabela.

Excluir Constraints

Pode-se excluir uma chave primária (ou outra constraint):

```
ALTER TABLE nome-tabela
DROP PRIMARY KEY;
```

DROP TABLE – Excluir tabela

Podemos excluir uma tabela com o comando DROP TABLE

Sintaxe:

```
DROP TABLE nome_tabela;
```

Exemplo: excluindo a tabela Clientes (fictícia):

```
DROP TABLE Clientes;
```