My SQL – Manipulando o Banco de Dados

Definições de um banco

O banco de dados é um repositório que serve para armazenar dados que podem ser consultados, fica armazenado em um disco rígido, como um HD ou SSD, por exemplo. Um banco de dados ocupa espaço em disco e conseguimos ir para um diretório e visualizar arquivos que representam o banco de dados.

Nos bancos há diversas ***entidades***, que ***são estruturas que organizam como os dados são armazenados***. Uma das principais entidades é a ***tabela, podendo conter várias em um banco de dados***.

Em uma analogia, uma ***tabela*** é como se fosse uma planilha no Excel — que há ***colunas e linhas***. Mas, diferente da planilha de Excel, que ao gerar uma nova conseguimos visualizar uma série de colunas e linhas em branco, ***no momento de criação da tabela é preciso já estabelecer as*** **definições** ***do que ela abrangerá***.

Algumas dessas definições é a quantidade e categoria de cada campo. O ***campo*** seria a coluna, então podemos ter, por exemplo, campos da categoria ***texto***, ***número***, que podemos ter com ***casas decimais*** ou ***inteiros***, ***lógico*** (verdadeiro ou falso), **binário**, que há bites armazenados que podem representar uma imagem ou algum outro arquivo diferente do formato texto e assim por diante. Portando, ao criar uma **tabela**, é necessário já definir quantos e as **categorias de cada coluna**.

Os **valores** de uma mesma coluna não podem ser de grupos diferentes, isto é, se o **campo foi estabelecido como numérico**, **podemos apenas a armazenar números nesse campo**. Se for incluído algo que fuja muito, tipo um texto, o banco de dados retorna erro.

Já as **linhas das tabelas**, são chamadas **registros**. Este, diferente dos campos, possui número infinito - a **depender do espaço em disco disponível para o banco de dados expandir**. Inclusive, ao gerar um banco de dados podemos determinar políticas de crescimento ou o limite máximo que ele pode ampliar.

Chave Primária

Outro conceito importante referente a tabela, é a ***chave primária*** (*primary key*). No momento de criar uma tabela, ***não obrigatoriamente***, podemos estabelecer uma chave primária, isso significa que ***os valores de um campo específico não podem se repetir em uma linha***.

Por exemplo, vamos supor que a tabela seja a "tabela_cadastro_clientes" e temos na primeira coluna o "CPF" dos clientes e na terceira o "Nome". Se escolhermos o CPF como chave primária, isso quer dizer que não podemos ter dois registros (linhas) na tabela que tenham os mesmos valores na coluna "CPF". Isso faz sentido, já que ninguém possui CPF idênticos, já o nome é possível repetir, por não ser considerado chave primária.

Já se tivermos uma ***chave primária composta***, o que ***não pode repetir é a combinação entre as colunas***. Portando, ***chave primária são os valores de campos ou combinação entre campos*** — chave primária composta — que não ***podem se repetir nos registros da tabela***.

Chave Estrangeira

No banco de dados podemos ter várias tabelas, cada uma possui fragmento da informação armazenada, podendo essas tabelas se relacionarem através da ***chave estrangeira*** (*foreign key*). Como na "tabela_cadastro_clientes" que um campo é o "CPF" e o outro é "Nome" e na outra tabela que representa as vendas de produto para cada cliente, "tabela_vendas", e nesta também temos o campo "CPF".

Ao criarmos uma chave estrangeira entre os campos "CPF" da "tabela_vendas" e da "tabela_cadastro_clientes", isso significa que teremos uma ligação entre elas. Como mencionado em vídeos anteriores, o SQL surgiu da necessidade de manipular e armazenar dados em um banco de dados relacional, que possui relações entre as tabelas, isto é, possuem chaves estrangeiras. Isso faz com o que a informação tenha ***integridade***, visto que não seria possível ter um cliente comprando um produto sem estar pré-cadastrado na tabela de clientes.

Antes dos bancos de dados relacionais, eram utilizados os ***transacionais***. Em outras palavras, não possuíam ligação entre tabelas, podendo assim, registrar o CPF de um cliente na "tabela_vendas" que não esteja, necessariamente, pré-cadastrado na "tabela_cadastro_clientes". Isso gera um problema de integridade do dado, por isso, os bancos de dados relacionais surgiram para melhorar a qualidade da informação armazenada.

Índice

Nas tabelas, podem ser encontrados os ***índices***. Eles permitem encontrar informações da tabela de maneira mais rápida. Como exemplo, vamos imaginar que queremos obter todos os clientes que começam com o nome Thiago, se não tivermos um índice, o banco de dados relacional terá de percorrer registro por registro até encontrar o nome solicitado pela primeira vez e, após encontrar, informar e seguir buscando até o término das linhas para verificar se há mais de um cliente com esse nome.

Agora, se tivermos um índice para o campo "Nome" temos um algoritmo interno - como se já estivesse ordenado alfabeticamente os elementos da coluna "Nome". Isto é, já

sabemos que o nome Thiago consta nos últimos registros com a inclusão do índice, não precisando percorrer todas as linhas até encontrar o primeiro nome com a letra **V**.

O índice permite, neste caso, que a busca se inicie nas linhas que os nomes começam com a letra **V** e a partir disso, procurar o nome Victorino, tornando a busca mais rápida. Portanto, o índice serve para facilitar e agilizar a procura.

Quando temos uma chave estrangeira, automaticamente o banco de dados cria índices nos campos que se interrelacionam, para que seja viável, por exemplo, ao cadastrar um cliente na "tabela_vendas" o banco de dados, internamente, verifique se o cliente consta na "tabela_cadastro_clientes" e para encontrar rápido é proveitoso que a tabela original já possua índice.

Schema

Já sobre ***schemas*** é o conjunto de tabelas que representam o mesmo assunto. As tabelas de esquemas diferentes podem se relacionar, transformar em *Schemas* é apenas uma forma de agrupar as tabelas por tema, sendo mais utilizado no sentido de organização.

View

Os bancos possuem a ***view*** um agrupamento de tabelas. É uma consulta e ela pode me retornar não apenas as informações de uma determinada tabela, mas de duas os mais através das chaves estrangeiras. Após conseguir unir duas ou mais tabelas e gerar um resultado para essa consulta, podemos transformar-lá em uma **view**.

Isso significa que a view possui um comportamento similar a tabela, mas que por trás dela já há uma consulta estabelecida com as regras de negócio para agrupar as informações solicitadas.

Join

Também temos o ***Join*** que uni as tabelas através de um critério, ao elaborar essa consulta que junta tabelas podemos definir filtros, tal como clientes apenas no sexo masculino e/ou que moram apenas na região Sul. Essa consulta que aplicamos o filtro pode ser uma view.

Trigger

No banco tem o ***triggger*** que é um aviso programado caso algo ocorra no banco de dados ou tabela. Como, se quisermos ser avisados caso alguém realize alguma alteração ou delete informações nas tabelas. Este aviso poder ser uma função, uma *procedure* ou um único comando SQL, que será executado quando a condição da *trigger* for satisfeita.

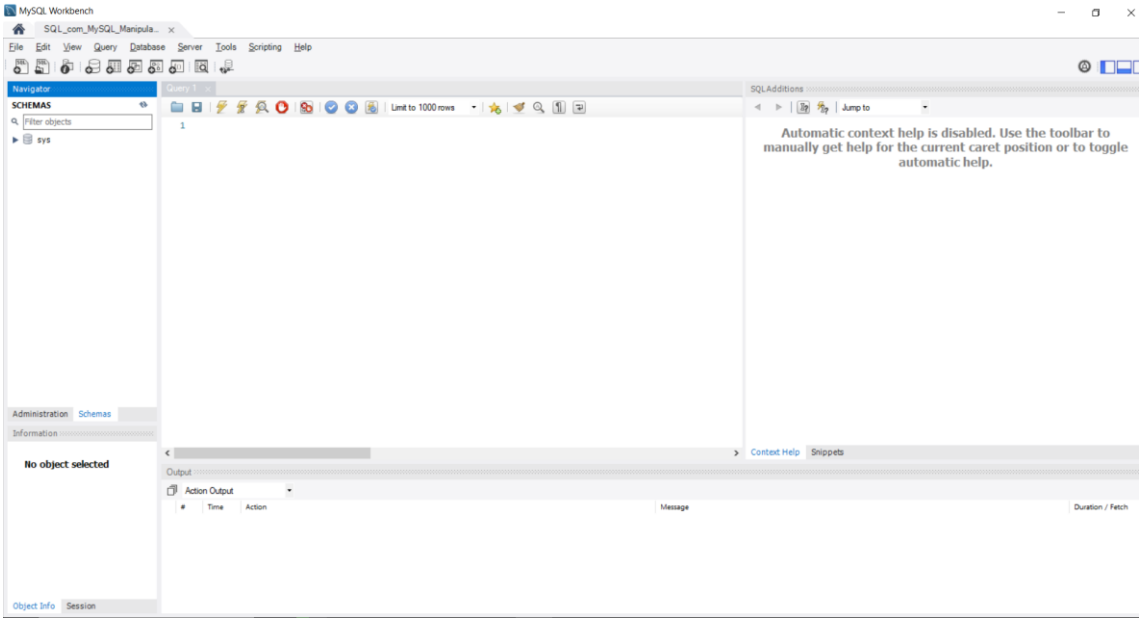
Por exemplo, se tivermos duas tabelas "tabela_clientes" e a "tabela_taxas". Todo cliente cadastrado é preciso ir à tabela taxas e criar uma taxa com o valor zero na "tabela_taxas". Podemos ter uma *trigger* que toda vez que criado um cadastro na tabela

de cliente, irá à tabela de taxas inserir o código do cliente e mais uma taxa *default* (padrão).

Com isso, estamos garantindo que ao incluir um novo cliente, ele(a) já terá um valor de taxa, mesmo que seja zero. O *trigger* pode ser utilizado para diversas outras situações no banco de dados.

Criando um Banco de Dados

Depois de instalar o ***My SQL Server*** junto com o ***My SQL Workbench***, abra o workbench e você verá o ambiente do software.



A sintaxe para criar um banco no My SQL é o comando ***CREATE {DATABASE/SCHEMA}***. No My SQL database e schema são sinônimos.

Podemos acrescentar também o comando ***IF NOT EXISTS*** **db_name**, esse comando significa que se não for encontrado o nome atribuído ao banco de dados, será criado e caso exista, não será feita nenhuma execução.

Veja o código na documentação do My SQL.

13.1.12 CREATE DATABASE Statement

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_option] ...

create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name
  | COLLATE [=] collation_name
  | ENCRYPTION [=] {'Y' | 'N'}
}
```

Olhando para as informações em ***create_option***, se não for indicado o ***CHARACTER_SET***, como o UTF 8 ou UTF-16, o código utiliza o padrão (default).

- **UTF-8** - UCS Transformation Format 8 (formato de transformação UCS 8)

- **UTF-16**- 16-bit Unicode Transformation Format (formato de Transformação Unicode)

No nosso computador possuímos uma tabela interna chamada **ASCII** (da sigla *American Standard Code for Information Interchange*, "Código Padrão Americano para o Intercâmbio de Informação"), em que cada letra digitada no computador é convertida em um código que a representa.

O código ASCII pode diferir conforme o idioma, como na língua inglesa que não existe acento e nem cê-cedilha. Por isso, a tabela ASCII original não comporta esses tipos de caracteres especiais. Porém, há tabelas que integram esses caracteres especiais, então, ao digitar uma letra com acento será buscado na tabela o código correspondente.

Por esse motivo, em ***CHARACTER_SET*** vamos informar para o banco de dados o conjunto de caracteres permitidos, a depender do idioma. Por exemplo, se a construção do banco de dados são com informações em português, é necessário, portanto, que seja incluído caracteres da língua portuguesa.

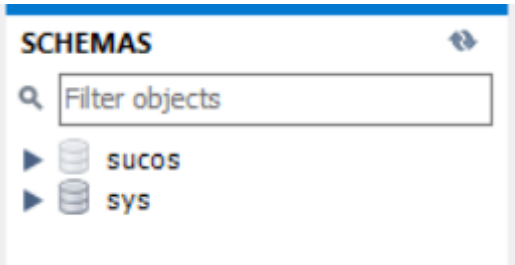
Apenas por questão de conhecimento, o ***COLLATE*** também específica o padrão desses conjuntos de caracteres a serem usados e o ***ENCRYPTION*** informa se o banco de dados será criptografado ou não.

Como no momento estamos aprendendo SQL, essa informação não é tão relevante e, em razão disso, vamos somente rodar o comando ***CREATE***.

Agora vamos criar o nosso banco no Workbench.

```
1 •   CREATE DATABASE Sucos;
2
3 •   CREATE DATABASE IF NOT EXISTS Sucos;
```

Depois de executar clique em ***refresh***, é um ícone com duas setas do lado da palavra ***SCHEMAS***.



Agora vamos ver onde esse banco está localizado fisicamente, em que parte no disco ele se encontra. Selecione a sua pasta de arquivos do próprio computador e procure o repositório do MySQL. No caso, o caminho é o ***C:\ProgramData\MySQL\MySQLServer 8.0*** do MySQL e note haver um arquivo nomeado ***my.ini***, este é lido pelo MySQL toda vez que ele é iniciado, ou seja, é o arquivo de inicialização.

No arquivo ***my.ini***. há uma série de variáveis de ambiente. Abrindo esse arquivo, clicando com o botão direito do mouse, em um editor de texto, conseguimos encontrar a variável ***datadir = C:/ProgramData/MySQL/MySQLServer 8.0/Data***, que mostra o ambiente onde o banco de dados está localizado.

Voltando para os diretórios no computador, na pasta "Data" note que temos um diretório chamado **sucos** agora, porém dentro não temos nenhum arquivo ainda. Mas, caso selecionarmos o repositório *world* é possível visualizar uma série de arquivos separados, as tabelas com a extensão ***.ibd***, isto é, cada tabela possui um registro separado.

Criamos o banco de dados e a partir de agora podemos prosseguir com o nosso treinamento, criando outros componentes no banco de dados **sucos**.

Deletando um Banco de Dados

A sintaxe utilizada para excluir um banco de dados é ***DROP DATABASE***. No My SQL, assim como na criação, podemos usar o ***IF EXISTS db_name*** para verificar se o banco existe no My SQL, se sim o banco será excluído, se não existir nada irá acontecer.

13.1.24 DROP DATABASE Statement

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```