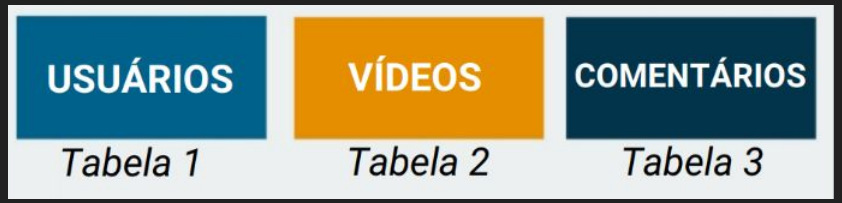
Este curso é de propriedade do aplicativo SoloLearn e foi traduzido por Ana Beatriz Augusto usando os recursos Reverso Context e Google Tradutor.

Introdução à base de dados

Um banco de dados é uma coleção de dados que é organizada de uma maneira que facilita o acesso, o gerenciamento e a atualização.

Um banco de dados é feito de tabelas que armazenam informações relevantes.

Por exemplo, você usaria um banco de dados se você fosse criar um site como o YouTube, que contém várias informações como vídeos, nomes de usuário, senhas e comentários.



Uma tabela armazena e mostra dados de forma estruturada por colunas e linhas, semelhante às planilhas do Excel.

Bancos de dados frequentemente contêm múltiplas tabelas, cada uma designada para um propósito. Por exemplo, imagine criar uma tabela que contenha nomes e números de telefone.

Primeiro, iríamos criar colunas chamadas Nome, Sobrenome e Telefone.

Cada tabela contém seus campos, baseado no que irá armazenar.

Nome	Sobrenome	Telefone
John	Smith	13 9 7155-5512
David	Williams	11 9 5691-9997
Chloe	Anderson	12 9 2010-7777
Emily	Adams	14 9 5663-3312
James	Roberts	15 9 7637-7729

Uma tabela tem um número específico de colunas mas pode conter quantas linhas quiser.

Uma chave primária é um campo na tabela que, unicamente, identifica os registros da tabela.

As principais características das chaves primárias são:
- Devem conter um valor único para cada linha.
- Não podem conter valores nulos.

Por exemplo, nossa tabela contém um registro para cada nome em uma lista telefônica. A coluna ID seria uma boa escolha de chave primária pois, há a chance de duas pessoas terem o mesmo nome.

ID	Nome	Sobrenome	Telefone
1	John	Smith	13 9 7155-5512
2	David	Williams	11 9 5691-9997
3	Chloe	Anderson	12 9 2010-7777
4	Emily	Adams	14 9 5663-3312
5	James	Roberts	15 9 7637-7729

- Cada tabela apenas pode ter UMA chave primária.
- O valor da chave primária deve ser diferente para cada linha.

Quando você entende o que é um banco de dados, saber o que é SQL fica fácil. SQL significa Structured Query Language (Linguagem Estruturada para Consultas)

SQL é usado para acessar e manipular um banco de dados. MySQL é um programa que entende SQL.

O SQL pode:
- inserir, atualizar ou deletar registros em um banco de dados.
- criar novos bancos de dados, tabelas, procedimentos armazenados, views.
- recuperar dados do banco de dados, etc.

SQL é de padrão ANSI, mas há diferentes versões da linguagem SQL. A maioria dos programas de bancos de dados SQL tem suas próprias propriedades adicionadas, mas todas elas suportam os comandos principais.

SELECT

A afirmação `SHOW` mostra informação que está no banco de dados e em suas tabelas. Essa ferramenta útil te deixa observar o conteúdo do seu banco de dados e te lembrar da estrutura de suas tabelas.

Por exemplo, o comando `SHOW DATABASES` lista os bancos de dados gerenciados pelo servidor.

Ao longo do tutorial estaremos usando MySQL e a ferramenta PHPMyAdmin para executar consultas SQL.
A maneira mais fácil de ter MySQL e PHPMyAdmin é instalar ferramentas gratuitas como XAMPP ou WAMP, que incluem o que é necessário.

O comando `SHOW TABLES` é usado para mostrar todas as tabelas no banco de dados MySQL selecionado.

`SHOW COLUMNS` mostra informação sobre as colunas de uma tabela específica.
O seguinte exemplo mostra as colunas em nossa tabela clientes:

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(60)	NO		NULL	
Sobrenome	varchar(60)	NO		NULL	
Cidade	varchar(30)	NO		NULL	
CodigoPostal	int(10)	NO		NULL	

`SHOW COLUMNS` mostra os seguintes valores para cada coluna da tabela:
Field: nome da coluna
Type: tipo de dado da coluna
Key: indica se a coluna é indexada
Default: valor padrão referido a coluna
Extra: pode conter qualquer informação adicional disponível sobre uma coluna específica

A afirmação `SELECT` é usada para selecionar dados de um banco de dados. O resultado é armazenado em uma tabela de resultados, que é chamada de result-set.

Uma consulta pode recuperar informação de colunas selecionadas ou de todas as colunas da tabela. Para criar uma afirmação `SELECT` simples, especifique o(s) nome(s) da(s) coluna(s) que você precisa.

Sintaxe da afirmação `SELECT` do SQL:

```
SELECT colunas
FROM tabela
```

- *colunas* inclui uma ou mais colunas de cada dado recuperado
- *tabela* é o nome de cada tabela de onde a informação é recuperada.

Abaixo está os dados da nossa tabela clientes:

ID	Nome	Sobrenome	Cidade	CodigoPostal
1	John	Smith	Nova York	10199
2	David	Williams	Los Angeles	90052
3	Chloe	Anderson	Chicago	60607
4	Emily	Adams	Houston	77201
5	James	Roberts	Filadelfia	19104

A seguinte afirmação SQL seleciona o `Nome` da tabela clientes:

```
SELECT Nome FROM clientes
```

Nome
John
David
Chloe
Emily
James

Uma afirmação `SELECT` retorna zero ou mais linhas de uma ou mais tabelas do banco de dados.

Regras de sintaxe SQL

SQL permite executar múltiplas consultas ou comandos ao mesmo tempo.

A seguinte afirmação SQL seleciona as colunas `Nome` e `Cidade` da tabela clientes:

```
SELECT Nome FROM clientes;
SELECT Cidade FROM clientes;
```

Resultado:

Nome
John
David
Chloe
Emily
James

Cidade
Nova York
Los Angeles
Chicago
Houston
Filadelfia

Lembre-se de terminar cada afirmação SQL com ponto e vírgula para indicar que a afirmação está completa e pronta para ser interpretada. Nesse tutorial, usaremos ponto e vírgula no fim de cada afirmação SQL.

SQL não diferencia entre letras maiúsculas e minúsculas.
As seguintes afirmações são equivalentes e irão mostrar o mesmo resultado:

```
select Cidade from clientes;
SELECT Cidade FROM cleintes;
sElEct Cidade From clientes;
```

É comum praticar a escrita de todos os comandos SQL em letras maiúsculas.

Uma única afirmação SQL pode ser colocada em uma ou mais linhas. Além disso, múltiplas afirmações SQL podem ser combinadas em uma única linha.

Espaços em branco e múltiplas linhas são ignoradas no SQL. Por exemplo, a consulta a seguir está absolutamente correta.

```
SELECT        Cidade
FROM clientes;
```

Entretanto, é recomendado evitar linhas e espaços em branco desnecessários.

Espaçamento apropriado e indentação separa os comandos fazendo com que suas afirmações SQL fiquem fáceis de ler e editar.

Selecionando múltiplas colunas

Como citado anteriormente, a afirmação `SELECT` do SQL retorna registros de tabelas no seu banco de dados SQL.

Você pode selecionar múltiplas colunas de uma vez. Apenas liste os nomes das colunas separadas por vírgulas:

```
SELECT Nome, Sobrenome, Cidade
FROM clientes;
```

Resultado:

Nome	Sobrenome	Cidade
John	Smith	Nova York
David	Williams	Los Angeles
Chloe	Anderson	Chicago
Emily	Adams	Houston
James	Roberts	Filadelfia

Não coloque uma vírgula depois do último nome da coluna.

Para retornar toda a informação contida em sua tabela, coloque um asterisco (*) depois do comando SELECT, invés de escrever cada nome das colunas separadamente.

A seguinte afirmação SQL seleciona todas as colunas da tabela clientes:

```
SELECT * FROM clientes;
```

Resultado:

ID	Nome	Sobrenome	Cidade	CodigoPostal
1	John	Smith	Nova York	10199
2	David	Williams	Los Angeles	90052
3	Chloe	Anderson	Chicago	60607
4	Emily	Adams	Houston	77201
5	James	Roberts	Filadelfia	19104

Em SQL, o asterisco significa "todos".

DISTINCT e LIMIT

Quando você tem registros duplicados em uma tabela, faz sentido mostrar apenas registros únicos, invés de mostrar os duplicados.

O comando `DISTINCT` do SQL é usado em conjunto com o `SELECT` para eliminar todos os registros duplicados e mostrar apenas os registros únicos.

A sintaxe básica do DISTINCT é a seguinte:

```
SELECT DISTINCT coluna1, coluna2
FROM tabela;
```

Veja a tabela clientes abaixo:

ID	Nome	Sobrenome	Cidade
1	John	Smith	Nova York
2	David	Williams	Los Angeles
3	Chloe	Anderson	Chicago
4	Emily	Adams	Houston
5	James	Roberts	Filadelfia
6	Andrew	Thomas	Nova York
7	Daniel	Harris	Nova York
8	Charlotte	Walker	Chicago
9	Samuel	Clark	San Diego
10	Anthony	Young	Los Angeles

Veja que há nomes duplicados de cidade. A seguinte afirmação SQL seleciona apenas valores distintos da coluna Cidade:

```
SELECT DISTINCT Cidade FROM clientes;
```

Isso mostraria o seguinte resultado. Registros duplicados foram removidos.

Cidade
Nova York
Los Angeles
Chicago
Houston
Filadelfia
San Diego

O comando DISTINCT apenas busca por valores únicos.

Por padrão, todos os resultados que satisfazem as condições especificadas na afirmação SQL são retornados. Entretanto, às vezes precisamos retornar apenas alguns registros.
No MySQL, isso é feito usando o comando `LIMIT`.

A sintaxe do `LIMIT` é a seguinte:

```
SELECT colunas
FROM tabela
LIMIT numero_de_registros;
```

Por exemplo, podemos retornar os primeiros cinco registros da tabela clientes.

```
SELECT ID, Nome, Sobrenome, Cidade
FROM clientes LIMIT 5;
```

Iríamos ter o seguinte resultado:

ID	Nome	Sobrenome	Cidade
1	John	Smith	Nova York
2	David	Williams	Los Angeles
3	Chloe	Anderson	Chicago
4	Emily	Adams	Houston
5	James	Roberts	Filadelfia

Você também pode escolher registros usando uma determinada ordem.
No exemplo a seguir, escolhemos quatro registros, começando da terceira posição:

```
SELECT ID, Nome, Sobrenome, Cidade
FROM clientes LIMIT 3, 4;
```

Retornado o seguinte resultado:

ID	Nome	Sobrenome	Cidade
4	Emily	Adams	Houston
5	James	Roberts	Filadélfia
6	Andrew	Thomas	Nova York
7	Daniel	Harris	Nova York

O motivo pelo qual é mostrado resultados a partir do ID número 4 (e não 3) é que o MySQL começa contando do zero, significando que a primeira linha começa do número 0 (e não 1).

Nome totalmente qualificado

Em SQL, você pode colocar o nome da tabela antes do nome da coluna fazendo a separação com um ponto final.
As seguintes afirmações são equivalentes:

```
SELECT Cidade FROM clientes;
SELECT clientes.Cidade FROM clientes;
```

O termo para a sintaxe mencionada acima é chamado de “nome totalmente qualificado" da coluna.

Essa forma de escrita é útil especialmente quando estamos trabalhando com múltiplas tabelas que podem ter mesmos nomes de colunas.

Order By

ORDER BY é usado com o SELECT para classificar os dados retornados.

O exemplo a seguir classifica nossa tabela clientes pela coluna Nome.

```
SELECT * FROM clientes
ORDER BY Nome;
```

Como pode ver, as linhas estão ordenadas de maneira alfabética pela coluna Nome. Por padrão, o comando ORDER BY classifica os resultados em ordem crescente.

Resultado:

ID	Nome	Sobrenome	Cidade
6	Andrew	Thomas	Nova York
10	Anthony	Young	Los Angeles
8	Charlotte	Walker	Chicago
3	Chloe	Anderson	Chicago
7	Daniel	Harris	Nova York
2	David	Williams	Los Angeles
4	Emily	Adams	Houston
5	James	Roberts	Filadélfia
1	John	Smith	Nova York
9	Samuel	Clark	San Diego

Classificando múltiplas colunas

Quando estiver usando `ORDER BY` com mais de uma coluna, separe a lista de colunas com vírgulas. Aqui está a tabela clientes mostrando os seguintes registros:

ID	Nome	Sobrenome	Idade
1	John	Smith	35
2	David	Smith	23
3	Chloe	Anderson	27
4	Emily	Adams	34
5	James	Roberts	31
6	Andrew	Thomas	45
7	Daniel	Harris	30

Classificando por `Sobrenome` e `Idade`:

```
SELECT * FROM clientes
ORDER BY Sobrenome, Idade;
```

Essa afirmação `ORDER BY` retorna o seguinte resultado:

ID	Nome	Sobrenome	Idade
4	Emily	Adams	34
3	Chloe	Anderson	27
7	Daniel	Harris	30
5	James	Roberts	31
2	David	Smith	23
1	John	Smith	35
6	Andrew	Thomas	45

Como temos dois Smiths, eles irão ser ordenados pela coluna `Idade` em ordem crescente.

O comando `ORDER BY` começa a classificar na sequência das colunas. Irá classificar pela primeira coluna listada e então pela segunda, assim em diante.

Este curso é de propriedade do aplicativo SoloLearn e foi traduzido por Ana Beatriz Augusto usando os recursos Reverso Context e Google Tradutor.