01

# Definições

**Transcrição**

[00:00] O objetivo deste treinamento, como eu tinha falado no vídeo de apresentação, é mostrar ao aluno o primeiro contato com o SQL. É para aquele aluno que nunca viu SQL na vida.

[00:14] Para você conhecer o SQL, ter uma noção de como ele funciona, a primeira coisa que você precisa saber é como funciona um banco de dados, porque a forma como ele é organizado está diretamente ligada à forma como vamos criar os comandos de SQL. É um pré-requisito.

[00:42] Estamos usando neste treinamento o MYSQL para mostrar a vocês como funciona a linguagem SQL. Eu poderia usar o Oracle, o SQL server, mas não, neste treinamento, por fazer parte de uma carreira mais completa que a Alura vai oferecer a vocês de MYSQL, temos como foco ensinar SQL para quem nunca viu SQL usando o MYSQL.

[01:12] Como eu falei, precisamos antes conhecer como é que se organiza um banco de dados. Os conceitos que vou mostrar neste vídeo podem até se aplicar, grande parte das vezes, em qualquer outro banco de dados. SQL Server, que é da Microsoft, Oracle, e assim por diante.

[01:36] Vamos começar falando desses conceitos. A primeira coisa importante é que a entidade maior é o próprio banco de dados. Ele é um repositório que armazena dados que podem ser recuperados. Esse banco de dados normalmente fica armazenado dentro de uma área de disco rígido, pode ser um disco normal, SSD. Ou seja, ele ocupa espaço em disco como um arquivo de Word, como uma planilha de Excel. Você consegue dentro do ambiente ir para um diretório específico e ver um ou mais arquivos que representam o banco de dados.

[02:25] Dentro do banco de dados, temos diversas entidades. São estruturas que organizam o armazenamento do dado dentro do banco de dados. A entidade principal, entre outras, é a tabela. Ou seja, um banco de dados tem dentro dele uma coleção de tabelas.

[02:53] O que é uma tabela? Fazendo uma analogia, é como se fosse uma planilha de Excel, onde tenho colunas e linhas. Só que diferente de uma planilha de Excel, em que você vê uma série de colunas e linhas em branco, na tabela, no momento em que ela é criada, já aplico sobre ela as definições sobre o que ela terá.

[03:25] Uma das definições que temos que fazer quando criamos uma tabela é dizer quantos campos ela terá e o tipo de cada campo. O campo seria a coluna. Posso ter, por exemplo, campos do tipo texto, número, campos com números com casas decimais ou números inteiros. Posso ter campos do tipo lógico, ou seja, verdadeiro ou falso. Posso ter campos do tipo binário, em que tenho bytes armazenados que representam, por exemplo, uma imagem, um outro arquivo com formato diferente do formato texto.

[04:13] Quando construo uma tabela, tenho que dizer quantos campos ela terá e qual o tipo de cada campo. Os valores do campo não podem ser de tipos diferentes. Ou seja, se eu digo que um campo é numérico, só posso ter números armazenados lá. Não posso em um campo numérico colocar textos. O próprio banco de dados vai dar erro.

[04:47] De maneira óbvia, cada linha da tabela é chamada de registro, ou linhas. Uma tabela tem número de campos limitados, mas pode ter infinitas linhas, ou infinitos registros. O número de registro não é limitador na tabela. O que limito é o número de campos.

[05:20] Claro que existe um limite máximo, que é o espaço em disco disponível para o meu banco de dados crescer. Inclusive, quando criamos um banco de dados, podemos colocar políticas de crescimento desse banco, ou um limite máximo.

[05:35] Voltando para as tabelas. Ela é uma entidade em que o dado fica armazenada, é composta por colunas e registros. As colunas são chamadas de campos, e tenho número de campos previamente definido. O que eu tenho dentro dos campos sempre possui o mesmo tipo. Já os registros, ou linhas, podem ter um número infinito.

[06:01] Existe outro conceito na tabela chamado “chave primária”. Não sou obrigado, a tabela não tem como obrigatoriedade ter uma chave primária, mas se eu disser que a tabela tem uma chave primária, estou dizendo que a combinação entre os campos não pode se repetir em uma linha.

[06:33] Um exemplo prático. Digamos que minha tabela fosse um cadastro de clientes, e tenho uma coluna. Posso dizer que ela é o CPF do cliente. Outra coluna posso dizer que é o nome do cliente. Quero fazer a seguinte definição na tabela: o CPF é chave primária. Se eu disse então que o CPF é chave primária, eu não vou poder ter na minha tabela duas linhas que tenham no valor da coluna CPF os mesmos valores. E isso tem sentido. Ninguém tem CPFs iguais. O nome, por exemplo, se estiver em uma coluna que não é chave primária, pode ser repetido.

[07:33] Quando eu tiver uma chave primária composta, o que não pode se repetir é a combinação das duas colunas. Então, chave primária são os campos cujas combinações não podem se repetir dentro das linhas da tabela ou dos registros da tabela.

[07:59] Tenho muitas tabelas dentro de um banco de dados. Cada tabela tem uma parte da informação, ou melhor, uma parte do dado guardado. Mas essas tabelas podem se relacionar, ter uma ligação entre elas. Vou dar outro exemplo.

[08:23] Digamos que eu tenha uma tabela com cadastro de clientes, em que tenho um campo com CPF e outro com nome do cliente. E tenho outra tabela que representa as vendas de produtos para aquele cliente. É claro que na tabela que representa as vendas também vou ter um campo dizendo o CPF do cliente que comprou o produto.

[08:50] Se eu crio uma chave estrangeira entre o campo CPF, que está na tabela que mostra as vendas e o campo CPF que está no cadastro de clientes, isso significa que há uma ligação entre essas duas tabelas. E aí não vou poder ter um CPF de cliente na tabela de vendas que não tenha sido previamente cadastrado na tabela de clientes.

[09:25] Quando falei no vídeo da aula passada sobre a história do SQL, eu disse que ele veio da necessidade de poder manipular dados em um banco de dados relacional. Um banco de dados relacional é aquele em que tenho essas ligações, chamadas chave estrangeira. Por isso o chamamos de relacional, por ter relações entre as tabelas.

[09:52] Isso faz com que o dado tenha uma integridade. Eu não vou poder colocar um cliente comprando um produto que ele não possa estar previamente cadastrado na tabela de clientes.

[10:09] Antes dos bancos de dados relacionais, os bancos de dados eram transacionais. Eles não tinham essa ligação entre as tabelas. Isso fazia com que eu pudesse registrar na tabela de venda de cliente o CPF de um cliente que não está previamente cadastrado. Isso cria um problema de integridade no meu dado. Por isso os bancos de dados relacionais vieram para melhorar a qualidade da informação que é armazenada dentro do banco de dados.

[10:40] Uma tabela também pode ter um índice. O índice é um instrumento que faz com que eu consiga achar elementos na minha tabela de maneira mais rápida. Imagine que quero achar todos os clientes com o nome Victorino. Se eu não tiver um índice, o banco de dados relacional vai ter que percorrer linha a linha testando. Vai chegar uma hora em que ele vai achar um cliente que se encaixa, mas ele vai continuar procurando até o final da tabela, porque pode ser que haja mais de um cliente com o nome Victorino.

[11:40] Se eu tenho um índice para o campo do nome, ele já faz um algoritmo interno que, de certa forma, ordena alfabeticamente os elementos da coluna nome. Eu sei que o Victorino está no final, porque começa com V. Então não preciso percorrer todas as linhas. Vou direto para lá. E a partir daquele momento procuro o nome. A busca fica muito mais rápida.

[12:17] O índice serve para facilitar nossa busca. Uma observação: quando temos uma chave estrangeira, automaticamente o banco de dados cria índices nesses campos que se inter-relacionam, para facilitar a vida dele e poder, por exemplo, ao cadastrar um cliente na tabela de vendas, o banco de dados internamente vai ter que checar, conferir se aquele cliente tem na tabela de cadastro do cliente.

[12:58] Para ele achar esse cara rápido, é interessante que a tabela original tenha índice. Por isso, o próprio banco de dados se compromete a criar o índice.

[13:07] Tenho dentro do banco de dados várias tabelas, compostas por linhas e colunas, ou campos e registros. Essas tabelas têm chaves estrangeiras, chaves primárias e podem ter índices também. Podemos associar um grupo de tabelas ao que chamamos de esquema.

[13:38] O esquema é como se fosse um assunto. Posso ter um banco de dados com mil, duas mil, três mil tabelas, mas quero dizer que um grupo de tabelas tem a ver com vendas. Outro grupo tem a ver com produção. Tabelas de esquemas diferentes podem muito bem se relacionar. Transformar esse esquema é apenas uma maneira mais fácil de agrupar as tabelas por determinado assunto. É muito mais uma questão de organização.

[14:12] O banco de dados também tem uma coisa chamada “view”. É um grupamento de tabelas. Vamos ver neste curso que temos uma coisa chamada query, ou seja, consulta. Posso fazer uma consulta no banco de dados e essa consulta simplesmente não vai pegar somente informação de uma tabela, mas também informações de duas, três, ou mil tabelas ao mesmo tempo. Claro que vou ligar essas tabelas através das chaves estrangeiras.

[14:49] Depois que consigo juntar as tabelas e criar um resultado para essa consulta, posso transformar essa consulta em uma view. Ou seja, a view tem um comportamento igual ao de uma tabela, só que por detrás dela já tem uma consulta construída fazendo algum tipo de regra de negócio para agrupar informações, juntar coisas.

[15:18] Usando o SQL, eu trato a visão como se fosse uma tabela já existente. Só que ela na verdade é lógica. Às vezes a view tem uma performance não muito boa, se ela estiver baseada em comandos de SQL muito rebuscados, muito custosos.

[15:38] Como eu disse, temos, por exemplo, dentro do comando SQL, comandos para fazer uma consulta na base de dados. Quando faço essa consulta, vou dizer nela em que tabelas quero buscar informação. Se eu buscar informações em uma tabela somente, é muito simples. Se quero buscar informações em duas ou mais tabelas, tenho que fazer uma coisa chamada “join”, que junta as tabelas através de um critério.

[16:13] Quando faço essa consulta, juntando tabelas, posso inclusive implementar o que chamamos de filtros. Por exemplo, se eu quiser só clientes do sexo masculino, ou clientes que vivem na região Sul. Ser do sexo masculino ou da região Sul são critérios de filtro, que estou fazendo sobre a minha consulta, que pode ler dados direto de uma única tabela ou de várias tabelas se relacionando entre si. E claro, posso depois transformar aquela consulta em uma view.

[16:49] O banco de dados também tem internamente o que chamamos de “procedures”. Lembra que eu falei na introdução ao MYSQL que a linguagem SQL não é uma linguagem bem estruturada? Ela não tem dentro dela comandos de condição, repetição. Mas os construtores de bancos de dados, MYSQL, Oracle, SQL Server criaram linguagens que não estão mais respeitando o padrão ANSI, mas linguagens proprietárias que permitem com que consigamos, usando comandos de SQL, fazer algum tipo de lógica estruturada, através de ifs, whiles e vários comandos de repetição. É como se eu fizesse um programa em uma linguagem nativa do banco de dados utilizando comandos de SQL.

[17:49] Dentro das procedures, posso também ter funções. As funções são cálculos que faço com os campos. Normalmente, ela entra com o parâmetro campos. E depois posso usar essa função dentro de um comando de consulta.

[18:10] O próprio banco de dados, no caso o MYSQL, já tem um catálogo de funções para tudo quanto é coisa. Tirar espaços em branco do registro, transformar maiúscula em minúsculas, fazer cálculos complexos, como por exemplo de datas,

quantos dias tenho entre a data A e B, cálculos numéricos. Tenho diversas funções já prontas. Mas, posso, se quiser, construir minha própria função e usá-la como uma função qualquer do banco de dados.

[18:48] Um banco de dados também tem uma coisa chamada “trigger”. É um aviso, um alerta que programo caso alguma coisa aconteça no banco de dados ou na tabela. Por exemplo, “me avise caso alguém insira algo na tabela” ou “me avise caso alguém delete informações em uma tabela”. Esse aviso não é uma mensagem que vou ver. Ele seria o seguinte: posso fazer um processo, que pode ser uma função, uma procedure ou um único comando SQL, que seja executado quando a condição da trigger, por exemplo, for satisfeita.

[19:40] Vamos dar um exemplo. Digamos que eu tenha duas tabelas. Uma de clientes e uma de taxas. Todo cliente cadastrado preciso criar uma taxa na tabela de taxas para ele. E claro que no momento em que cadastro o cliente, preciso criar uma taxa com valor zero na tabela de taxas. Posso, por exemplo, ter uma trigger que diz “no momento em que você incluir alguém na tabela de clientes, vá na tabela de taxas e insira aquele código do cliente que você colocou, mais uma taxa default”.

[20:20] Ou seja, estou garantindo que toda vez que incluir um cliente novo, ele automaticamente vai ter um valor de taxa, mesmo que a taxa seja zero. Depois faço uma revisão. Isso é o trigger. Posso tê-lo para várias situações que acontecem dentro do banco de dados.

[20:41] O banco de dados tem esses componentes todos. Tabelas, views, procedures e funções. É um pouco disso que você precisa saber sobre banco de dados. Espero que essa explanação rápida que eu dei te estimule a procurar, por exemplo, na internet, mais informações sobre o que é um banco de dados. Isso é extremamente importante para que você entenda a linguagem SQL. E, por consequência, conhecer também o MYSQL.