Nesse vídeo, aprenderemos sobre as estruturas de JOIN, que permitem unir duas ou mais tabelas dentro de uma única consulta SQL. Basta que essas tabelas tenham um campo em comum.

Até agora, somente realizamos seleções de uma tabela por vez, no entanto, haverá ocasiões em que será necessário consultar informações que estão separadas, parte em uma tabela e parte em outra. Nesses contextos, aplicaremos os JOINs.

A seguir, veremos alguns exemplos, usando duas tabelas:

Na primeira tabela (**à *esquerda*** do vídeo), temos o nome de quatro clientes e seus respectivos números identificadores. Na segunda tabela (**à *direita***), são apresentados os números identificadores de quatro clientes e seus respectivos hobbies. Note que Pedro e Cláudia (números 4 e 6) não têm hobbies correspondentes na segunda tabela. Por outro lado, fotografia e artesanato são atividades atreladas a clientes que não estão cadastrados na primeira tabela (números 5 e 8). De resto, percebemos que João (identificador 1), gosta de ir à praia e que o hobby de Maria (identificador 3) é o futebol.

Então, o primeiro tipo de JOIN que conheceremos é o INNER JOIN:

Interface gráfica do usuário, Texto, chat ou mensagem de texto

O conteúdo gerado por IA pode estar incorreto.

amos entender por partes o que cada pedaço dessa consulta significa. Em TABELA\_ESQUERDA A, estamos declarando "A" como o *alias* da primeira tabela, à esquerda. Um processo semelhante é feito em TABELA\_DIREITA B. Assim, quando escrevemos SELECT A.NOME, B.HOBBY, vamos selecionar o campo "NOME" da tabela A e o campo "HOBBY" da tabela B. Ao final, temos a expressão ON A.IDENTIFICADOR = B.IDENTIFICADOR, na qual especificamos qual é campo em comum dessas tabelas. E o comando INNER JOIN é responsável por retornar somente os registros que têm correspondência nas duas tabelas. Nesse caso, são apenas João e Maria (identificadores 1 e 3).

Outro tipo de comando JOIN é o LEFT JOIN:

Texto

O conteúdo gerado por IA pode estar incorreto.

*Left*, em inglês, significa "esquerda" e, no nosso contexto, representa a primeira tabela, à esquerda, a que vem **antes** do comando JOIN. Com essa consulta, o MySQL trará **todos** os elementos da tabela A e somente os correspondentes da tabela B:

Assim, todos os clientes da primeira tabela estarão no retorno, inclusive aqueles que não têm hobby (Pedro e Cláudia). Nesses casos, a coluna "HOBBY" apresentará valor *null* (nulo).

Da mesma maneira que existe o LEFT JOIN, temos o RIGHT JOIN, que trará todos os elementos da tabela da **direita** (ou seja, a que vem **depois** do comando JOIN) e somente os correspondentes da esquerda:

Texto

O conteúdo gerado por IA pode estar incorreto.

Todos os hobbies da tabela B estarão no retorno, inclusive aqueles cujo identificador de cliente não aparece na outra tabela (fotografia e artesanato). Nesses casos, na coluna "NOME" aparecerá com o valor *null*.

Além disso, existe também o FULL JOIN:

Texto

O conteúdo gerado por IA pode estar incorreto.

Com esse comando, **todos** os elementos tanto de uma quanto da outra tabela aparecerão no resultado. Os registros que não tiverem correspondência, apresentarão a coluna com valor nulo:

Portanto, a consulta nos informa que João gosta de praia; Maria curte futebol; Pedro e Cláudia não tem hobby; e não há clientes cadastrados que gostem de fotografia ou artesanato.

Por fim, temos o CROSS JOIN (junção cruzada), que retorna o produto cartesiano das duas tabelas. Ele tem uma estrutura ligeiramente diferente das anteriores, sem nenhuma condição de união. Ou seja, quando não especificamos qual é o campo que liga as tabelas, o MySQL fará o CROSS JOIN:

Texto

O conteúdo gerado por IA pode estar incorreto.

Ou seja, o resultado mostrará todas as combinações possíveis. Como temos quatro clientes na tabela A e quatro hobbies na tabela B, o resultado apresentará 16 combinações.

Agora, vamos fazer alguns exemplos práticos no MySQL Workbench. Criaremos um novo script e iniciaremos consultando a tabela de vendedores e a de notas fiscais:

Texto

O conteúdo gerado por IA pode estar incorreto.

Analisando o retorno, verificamos que a primeira tabela apresenta dados dos vendedores e a segunda mostra informações de notas fiscais. Elas têm um campo em comum: "MATRICULA". Por meio dessa relação, é possível criar uma consulta que junte informações de ambas:

Texto

O conteúdo gerado por IA pode estar incorreto.

Os campos em comum usados no JOIN não precisam ter o mesmo nome. O importante é que tenham o **mesmo conteúdo** para que a relação entre tabelas seja viável.

O retorno mostrará todos os campos da tabela A e todos os campos da tabela B, unidos em um só resultado. Será possível ver os dados de cada nota fiscal emitida, **junto** das informações do vendedor associado a ela.

Com essas duas tabelas, também podemos usar outros comandos com o JOIN, como o GROUP BY. Por exemplo, se nossa meta for descobrir quantas notas fiscais cada vendedor emitiu:

Texto

O conteúdo gerado por IA pode estar incorreto.

O mesmo resultado será obtido se fizéssemos um CROSS JOIN entre essas tabelas e filtrássemos (com WHERE) apenas os registros que têm correspondência no número da matrícula:

Texto

O conteúdo gerado por IA pode estar incorreto.

Entre essas duas opções, recomendo o uso da primeira, porque com o INNER JOIN é mais fácil de compreender as junções (principalmente quando ficam complexas) e também por ser a forma mais moderna. A segunda opção era comum há uns 20 anos, quando estruturas como INNER JOIN, LEFT JOIN e RIGHT JOIN ainda não existiam, e há quem ainda opte por ela, mas eu particularmente prefiro o INNER JOIN.

Assim, nesse vídeo fizemos um apanhado dos tipos de JOINe nos aprofundamos um pouco, na prática, no uso do INNER JOIN.