SQL

Voltar para a lista de artigos

Artigos

28th Nov 2023 | 9 minutos de leitura

Explicações sobre SQL JOINs: 5 exemplos claros de SQL INNER JOIN para iniciantes

Martyna Sławińska

Sql Join | Join | Joins

Índ

- *Está procurando uma explicação clara das junções em SQL? Confira estes cinco exemplos de SQL INNER JOIN!*

No SQL, `INNER JOIN` s pode ser um pouco difícil para os iniciantes dominarem. Mas, quando você começar a trabalhar com elas, verá que são muito úteis! Vamos discutir cinco exemplos de SQL `INNER JOIN` s. Mas, primeiro, vamos fazer uma rápida revisão do motivo pelo qual os JOINs são importantes.

- Nos bancos de dados relacionais, os dados são organizados e armazenados em tabelas. Cada tabela representa um tipo específico de informação. Mas, muitas vezes, você precisa analisar dados de diferentes tabelas simultaneamente. E é aí que os JOINs entram em cena.

Recomendamos que você pratique a união de tabelas seguindo nosso Cláusulas JOIN em SQL curso interativo. Ele inclui 93 exercícios que abrangem os seguintes

tópicos:

- Tipos de SQL JOIN.
- JOINs múltiplos.
- Self-joins, ou seja, união de uma tabela com ela mesma.
- JOINs não-equi.

Então, você está pronto para ver alguns exemplos de `INNER JOIN`? Vamos começar!

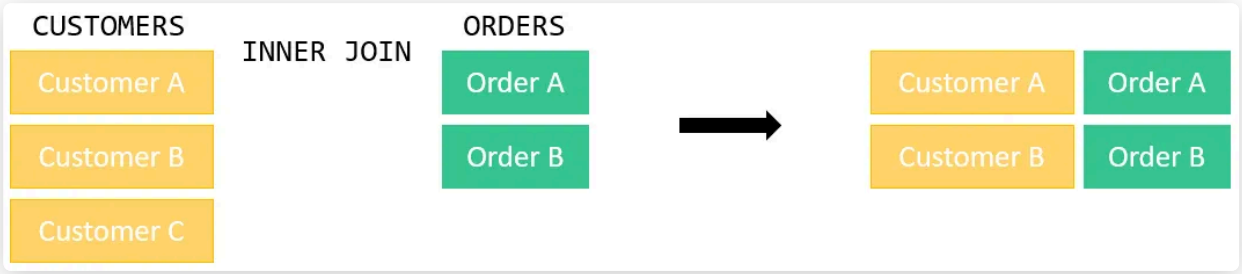
Unindo tabelas: Apresentando as operações SQL JOIN

Cláusulas JOIN em SQL combinam dados de duas ou mais tabelas com base em valores de coluna correspondentes. Por exemplo, você pode mesclar as informações dos clientes com os pedidos que eles fizeram ou vincular cada produto a seus fornecedores.

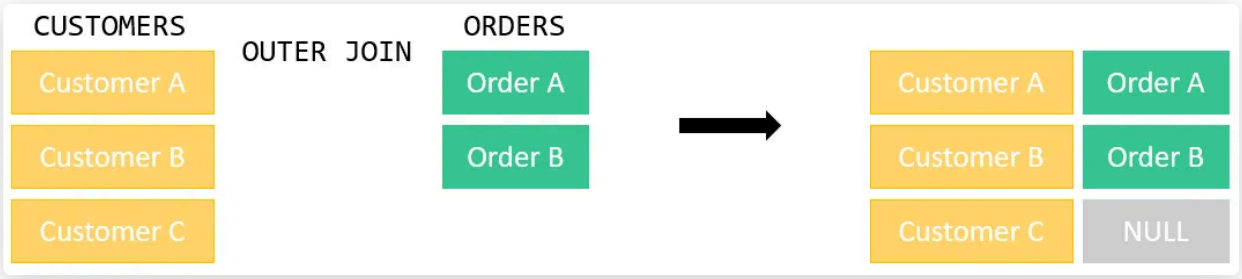
> LearnSQL.com.br é uma plataforma online projetada para ajudá-lo a dominar o SQL. LearnSQL.com.br permite que você escolha entre uma trilha completa de aprendizado, mini-trilhas para aprimorar habilidades específicas e cursos individuais. Você também pode selecionar o dialeto SQL (SQL Padrão, Microsoft SQL Server ou PostgreSQL) que melhor atende às suas necessidades.

Cláusulas JOIN em SQL As operações SQL JOIN permitem unir dados de diferentes tabelas e extrair informações significativas para seu caso de uso específico. Confira esta Folha de consulta SQL JOIN para ter uma visão geral dos diferentes tipos de JOINs.

Um `INNER JOIN` em SQL combina linhas de várias tabelas combinando seus valores de coluna comuns. Ao aplicar um `INNER JOIN` para vincular os dados do cliente aos pedidos correspondentes, você verá uma lista de clientes que fizeram pelo menos um pedido.



Além de `INNER JOIN`, o SQL também oferece outros tipos de junções: `LEFT JOIN`, `RIGHT JOIN`, e `FULL JOIN`. Essas uniões são chamadas de `OUTER JOIN`s. Diferentemente de uma operação `INNER JOIN`, uma operação `OUTER JOIN` pode listar linhas de ambas as tabelas, mesmo que não haja correspondência.



Vamos dar uma olhada em um exemplo para entender melhor a natureza de `SQL INNER JOIN`s.

Aqui está a tabela `customers` que armazena as informações do cliente:

customer_id	first_name	last_name	email
1	Cory	Castillo	cc@email.com
2	Ellie	Potts	ep@email.com
3	Jack	Greer	jg@email.com

E aqui está a tabela `orders` tabela que armazena todos os pedidos feitos pelos clientes:

order_id	customer_id	order_date	order_amount
27	1	6/6/2023	100.00
28	2	7/7/2023	150.00

order_id	customer_id	order_date	order_amount
29	1	8/8/2023	20.00

A coluna comum (mostrada em rosa) na qual as colunas `customers` e `orders` são unidas é a coluna `customer_id` .

Agora, aqui está a consulta:

Code

```
SELECT
  c.customer_id,
  c.email,
  o.order_date,
  o.order_amount
FROM customers AS c
INNER JOIN orders AS o
ON c.customer_id = o.customer_id;
```

Essa consulta une a tabela `customers` (com o alias `AS c`) e a tabela `orders` (com o pseudônimo `AS o`). Esses aliases de tabela fornecem uma maneira clara de informar ao banco de dados a tabela de origem de cada coluna (usando a sintaxe `table_alias.column_name`).

Esses aliases são empregados na cláusula `ON` para estabelecer a coluna para unir tabelas. Também os usamos na instrução `SELECT` , em que duas colunas são selecionadas da tabela de **clientes** (`c.customer_id and c.email`) e outras duas colunas da tabela `orders` (`o.order_date and o.order_amount`).

Aqui está o resultado da consulta:

customer_id	email	order_date	order_amount
1	cc@email.com	6/6/2023	100.00
2	ep@email.com	7/7/2023	150.00
1	cc@email.com	8/8/2023	20.00

Você pode observar que o cliente com o ID 3 não aparece na tabela resultante. Esse cliente específico ainda não fez nenhum pedido; portanto, ele não tem

nenhum registro correspondente na tabela `orders` na tabela.

A palavra-chave `INNER JOIN` é intercambiável com a palavra-chave `JOIN`. Em outras palavras, quando você usa `JOIN`, o banco de dados a interpreta como uma notação abreviada para `INNER JOIN`.

Para saber mais sobre `INNER JOIN` s, consulte nosso artigo What Is an Inner Join in SQL?

Exemplos de SQL INNER JOIN na prática

Vamos explorar alguns exemplos do SQL `INNER JOIN` feitos sob medida para iniciantes. Antes de começar a ver os exemplos, confira este artigo sobre como praticar Cláusulas JOIN em SQL.

Exemplo 1: Vincular livros a autores

Você tem duas tabelas que armazenam informações sobre livros e autores. Junte essas tabelas para ver a lista de livros e seus autores.

Aqui está a tabela `books` tabela:

book_id	title	publication_year	author_id
1	Frankenstein	1818	22
2	The Time Machine	1895	23
3	The Martian	2011	24
4	2001: A Space Odyssey	1968	25
5	Dune	1965	26

A coluna `book_id` identifica cada livro de forma exclusiva. E a coluna `author_id` atribui um autor a cada livro.

E aqui está a `authors` tabela:

author_id	author_name
22	Mary Shelley
23	H. G. Wells
24	Andy Weir
25	Arthur C. Clarke
26	Frank Herbert

A coluna `author_id` identifica exclusivamente cada autor. E a coluna `author_name` armazena os nomes completos dos autores.

Se quiser ver os títulos dos livros e os nomes dos autores, você pode unir essas tabelas com base em sua coluna comum (a coluna `author_id` em ambas as tabelas). Basta selecionar `title` from `books` e `author_name` from `authors` :

Code

```
SELECT
  b.title,
  a.author_name
FROM books AS b
INNER JOIN authors AS a
ON b.author_id = a.author_id;
```

Esta é a tabela resultante:

title	author_name
Frankenstein	Mary Shelley
The Time Machine	H. G. Wells
The Martian	Andy Weir
2001: A Space Odyssey	Arthur C. Clarke
Dune	Frank Herbert

Observe que a cada livro é atribuído um autor e a cada autor é atribuído um livro, pois não há valores duplicados na coluna comum `author_id`. Examine o próximo exemplo para ver o que acontece se a coluna comum tiver valores duplicados.

Exemplo 2: Atribuir produtos a categorias

Você tem duas tabelas que armazenam produtos e categorias de produtos. Junte-se a essas tabelas para ver a lista de produtos com suas categorias.

Aqui está a tabela `products` tabela:

product_id	product_name	category_id
1	Apple	22
2	Orange	22
3	Potato	23
4	Carrot	23
5	Chocolate	24

A coluna `product_id` identifica cada produto de forma exclusiva. E a coluna `category_id` atribui uma categoria a cada produto.

E aqui está a `categories` tabela:

category_id	category_name
22	Fruits
23	Vegetables
24	Snacks

A coluna `category_id` identifica exclusivamente cada categoria. E a coluna `category_name` armazena os nomes completos das categorias.

Se quiser ver os produtos e suas categorias, você pode unir essas tabelas com base em sua coluna comum, que é a coluna `category_id`. Aqui está a consulta:

Code

```
SELECT
  p.product_name,
  c.category_name
FROM products AS p
INNER JOIN categories AS c
ON p.category_id = c.category_id;
```

Esta é a tabela resultante:

product_name	category_name
Apple	Fruits
Orange	Fruits
Potato	Vegetables
Carrot	Vegetables
Chocolate	Snacks

As categorias com IDs 22 e 23 são atribuídas a dois produtos cada; seus nomes aparecem duas vezes na tabela de saída.

Exemplo 3: Listar médicos e pacientes com o mesmo primeiro nome

Você tem duas tabelas que armazenam informações sobre médicos e pacientes. Junte-se a essas tabelas para ver a lista de médicos e pacientes que compartilham o mesmo primeiro nome.

Aqui está a tabela `doctors` tabela:

doctor_id	first_name	last_name
1	Samantha	Monroe

doctor_id	first_name	last_name
2	Melvin	Ferrell
3	Albie	Blake
4	Rose	Bernard
5	Loui	Peterson

A coluna `doctor_id` identifica cada médico de forma exclusiva. As outras duas colunas armazenam o nome e o sobrenome dos médicos.

E aqui está a `patients` tabela:

patient_id	first_name	last_name
23	Ben	Woodward
24	Samantha	Thomson
25	Kate	Donovan
26	Albie	Vasquez
27	Loui	Chen

A coluna `patient_id` identifica exclusivamente cada paciente. As outras duas colunas armazenam o nome e o sobrenome dos pacientes.

Se quiser ver a lista de médicos e pacientes que compartilham o mesmo primeiro nome, você pode unir essas tabelas com base na coluna `first_name` .

Code

```
SELECT d.*, p.*
FROM doctors AS d
INNER JOIN patients AS p
ON d.first_name = p.first_name;
```

Observe que d.* define todas as colunas da tabela `doctors` e `p.*` define todas as colunas da tabela `patients` da tabela.

Esta é a tabela resultante:

doctor_id	first_name	last_name	patient_id
1	Samantha	Monroe	24
3	Albie	Blake	26
5	Loui	Peterson	27

Graças à operação `INNER JOIN`, você pode listar apenas os pares de médico e paciente em que os primeiros nomes são iguais.

Exemplo 4: Combinar pessoas com apartamentos adequados

Você tem duas tabelas que armazenam informações sobre apartamentos disponíveis para aluguel e pessoas procurando apartamentos que se encaixem em sua faixa de preço. Junte essas tabelas para ver quais apartamentos podem ser alugados por qual pessoa.

Aqui está a tabela `apartments` tabela:

apartment_id	rent
1	1000
2	700
3	500

A coluna `apartment_id` identifica exclusivamente cada apartamento. E a coluna `rent` armazena o valor do aluguel mensal.

E aqui está a `persons` tabela:

person_id	email	max_rent
23	ab@email.com	900
24	cd@email.com	600
25	ef@email.com	3000

A coluna `person_id` identifica exclusivamente cada pessoa que está procurando um apartamento para alugar. A coluna `email` armazena os e-mails dessas pessoas e a coluna `max_rent` armazena o valor máximo de aluguel mensal que elas podem pagar pelo apartamento.

Se quiser ver quais apartamentos podem ser alugados por qual pessoa, você pode unir essas tabelas com base nas colunas `rent` e `max_rent`:

Code

```
SELECT
  p.person_id,
  p.email,
  a.apartment_id,
  a.rent AS apartment_rent
FROM apartments AS a
INNER JOIN persons AS p
ON a.rent < p.max_rent;
```

Aqui, a correspondência deve ser feita entre a coluna de aluguel da tabela `apartments` e a coluna `max_rent` da tabela `persons` tabela, onde `rent < max_rent`.

Essa é a tabela resultante:

person_id	email	apartment_id	apartment_ren
25	ef@email.com	1	1000
23	ab@email.com	2	700
25	ef@email.com	2	700
23	ab@email.com	3	500
24	cd@email.com	3	500
25	ef@email.com	3	500

Ao usar a cláusula ON com a condição que contém o sinal `<`, várias linhas da `apartments` foram atribuídas a várias linhas da tabela `persons` tabela, e vice-versa. E você pode ver quais apartamentos são adequados para alugar para quais pessoas.

Pratique seu conhecimento em SQL com nosso curso de Curso de Práticas em SQL.

Exemplo 5: unir tabelas de continentes, países e cidades

Desta vez, você tem três tabelas que armazenam informações sobre continentes, países e cidades. Junte essas tabelas para saber qual cidade pertence a qual país e qual país pertence a qual continente.

Aqui está a tabela `continents` que armazena todos os continentes:

continent_id	continent_name
1	Asia
2	Africa
3	North America
4	South America
5	Antarctica
6	Europe
7	Australia

A coluna `continent_id` identifica cada continente de forma exclusiva. E a coluna `continent_name` armazena seu nome completo.

Aqui está a tabela `countries` tabela que armazena os países selecionados:

country_id	country_name	continent_id
23	United States	3
24	Brazil	4
25	South Africa	2
26	Japan	1

country_id	country_name	continent_id
27	Poland	6

A coluna `country_id` identifica exclusivamente cada país. A coluna `country_name` armazena seu nome completo. E a coluna `continent_id` armazena o continente em que o país está localizado.

E aqui está a tabela `cities` tabela que armazena as cidades selecionadas:

city_id	city_name	country_id
33	Rio de Janeiro	24
34	New York	23
35	Tokyo	26
36	Warsaw	27
37	Cape Town	25

A coluna `city_id` identifica exclusivamente cada cidade. A coluna `city_name` armazena seu nome completo. E a coluna `country_id` armazena o país onde a cidade está localizada.

Se quiser ver quais cidades, países e continentes estão juntos, você pode unir essas três tabelas com base em suas colunas comuns:

Code

```
SELECT
  cont.continent_name,
  coun.country_name,
  ci.city_name
FROM continents AS cont
INNER JOIN countries AS coun
ON cont.continent_id = coun.continent_id
INNER JOIN cities AS ci
ON coun.country_id = ci.country_id;
```

A tabela `continent` é unida à tabela `countries` em sua coluna comum (`continent_id`). E a tabela `countries` é unida à tabela `cities` em sua coluna comum (`country_id`).

Esta é a tabela resultante:

continent_name	country_name	city_name
Asia	Japan	Tokyo
Africa	South Africa	Cape Town
North America	United States	New York
South America	Brazil	Rio de Janeiro
Europe	Poland	Warsaw

As três tabelas são unidas em suas colunas comuns e o comando `SELECT` lista os nomes de continente, país e cidade com base na união entre as colunas de ID.

Confira este artigo sobre junção de três ou mais tabelas no SQL para obter mais exemplos de junções de várias tabelas.

Sua vez de praticar exemplos de INNER JOIN em SQL

Ao longo deste artigo, nos aprofundamos nas complexidades do `INNER JOIN` em SQL, desde os conceitos básicos até cenários mais avançados. Usamos várias condições `JOIN` e até unimos três tabelas. Como vimos, o poder do `INNER JOIN` está em sua capacidade de conectar conjuntos de dados sem problemas, o que nos permite extrair insights valiosos.

> LearnSQL.com.br oferece uma solução completa para tudo em SQL, cobrindo conceitos básicos a avançados em uma única plataforma. LearnSQL.com.br é especificamente voltado para SQL. Oferece 10+ cursos interativos de SQL que variam em dificuldade do iniciante ao avançado e desafios mensais de SQL para praticar suas habilidades em SQL.

Você precisa de mais exemplos de SQL `INNER JOIN` ? Experimente todos os tipos de JOINs seguindo nosso curso sobre Cláusulas JOIN em SQL que mencionamos anteriormente. Você terá toda a prática que precisa para se sentir confortável ao escrever JOINs.