Comandos DQL (JOIN)

Estudo Dirigido

R.A.: 1221121608

Nome: Lucas Alexandre Ferreira Alves

Em SQL, JOINs são operações usadas para combinar dados de duas ou mais tabelas com base em uma condição especificada, criando um conjunto de resultados que contém informações de ambas as tabelas relacionadas. Os JOINs são amplamente utilizados em consultas SQL para recuperar dados relacionados entre diferentes tabelas de um banco de dados. Existem vários tipos de JOINs em SQL, são eles:

- **`JOIN (INNER JOIN):`** Retorna apenas as linhas que têm valores correspondentes nas duas tabelas sendo unidas. Neste exemplo, estamos combinando as tabelas Pedido e Cliente com base na coluna `id_cliente`. A consulta retorna o ID do pedido e o nome do cliente para as linhas que possuem valores correspondentes em ambas as tabelas.

```sql
SELECT id_pedido, nome_cliente
FROM Pedido
JOIN Cliente ON Pedido.id_cliente = Cliente.id_cliente;
```

- **`LEFT JOIN (LEFT OUTER JOIN)`**: Retorna todas as linhas da tabela à esquerda e as linhas correspondentes da tabela à direita. Se não houver correspondência, os valores **`NULL`** são retornados para as colunas da tabela à direita. Neste exemplo, estamos fazendo um **`LEFT JOIN`** entre as tabelas Cliente e Pedido. A consulta retorna o nome do cliente e o id do pedido para todas as linhas da tabela Cliente e apenas as linhas correspondentes da tabela Pedido. Se não houver correspondência, o valor **`NULL`** será retornado para o id do pedido.

```sql
SELECT id_pedido, nome_cliente
FROM Pedido
LEFT JOIN Cliente ON Pedido.id_cliente = Cliente.id_cliente;
```

- **RIGHT JOIN (RIGHT OUTER JOIN):** Retorna todas as linhas da tabela à direita e as linhas correspondentes da tabela à esquerda. Se não houver correspondência, os valores **NULL** são retornados para as colunas da tabela à esquerda. Neste exemplo, estamos fazendo um **RIGHT JOIN** entre as tabelas Pedido e Cliente. A consulta retorna o id do pedido e o nome do cliente para todas as linhas da tabela Pedido e apenas as linhas correspondentes da tabela Cliente. Se não houver correspondência, o valor **NULL** será retornado para o nome do cliente.

```
SELECT id_pedido, nome_cliente
FROM Pedido
RIGHT JOIN Cliente ON Pedido.id_cliente = Cliente.id_cliente;
```

- **FULL JOIN (FULL OUTER JOIN)**: Retorna todas as linhas de ambas as tabelas sendo unidas. Se não houver correspondência, os valores **NULL** são retornados para as colunas sem correspondência. Neste exemplo, estamos fazendo um **FULL JOIN** entre as tabelas Cliente e Pedido. A consulta retorna o nome do cliente e o id do pedido para todas as linhas de ambas as tabelas. Se não houver correspondência, os valores **NULL** serão retornados para as colunas sem correspondência.

```
SELECT id_pedido, nome_cliente
FROM Pedido
FULL JOIN Cliente ON Pedido.id_cliente = Cliente.id_cliente;
```

- **CROSS JOIN:** Retorna o produto cartesiano de duas tabelas, ou seja, todas as combinações possíveis entre as linhas das duas tabelas. Não há necessidade de uma condição de junção para ser especificada. Neste exemplo, estamos fazendo um **CROSS JOIN** entre as tabelas Cliente e Produto. A consulta retorna todas as combinações possíveis entre as linhas das duas tabelas, não sendo necessário especificar uma condição de junção.

```
SELECT nome_cliente, nome_produto
FROM Cliente
CROSS JOIN Produto;
```

Lembrando que esses são apenas exemplos básicos para ilustrar o uso dos diferentes tipos de JOINs. É importante adaptar as consultas às estruturas das tabelas e às colunas que estão sendo usadas para fazer a junção. Além desses, existem outros tipos de JOINs, como JOINs auto-relacionais e JOINs

usando subconsultas (como JOINs correlacionados e JOINs com EXISTS), que oferecem maior flexibilidade na realização de junções entre tabelas. Os JOINs são fundamentais para a manipulação de dados em bancos de dados relacionais, permitindo a obtenção de informações combinadas de várias tabelas para consultas e análises complexas.

Ponto de Partida

Antes de começar esse esse exercício, você deverá:

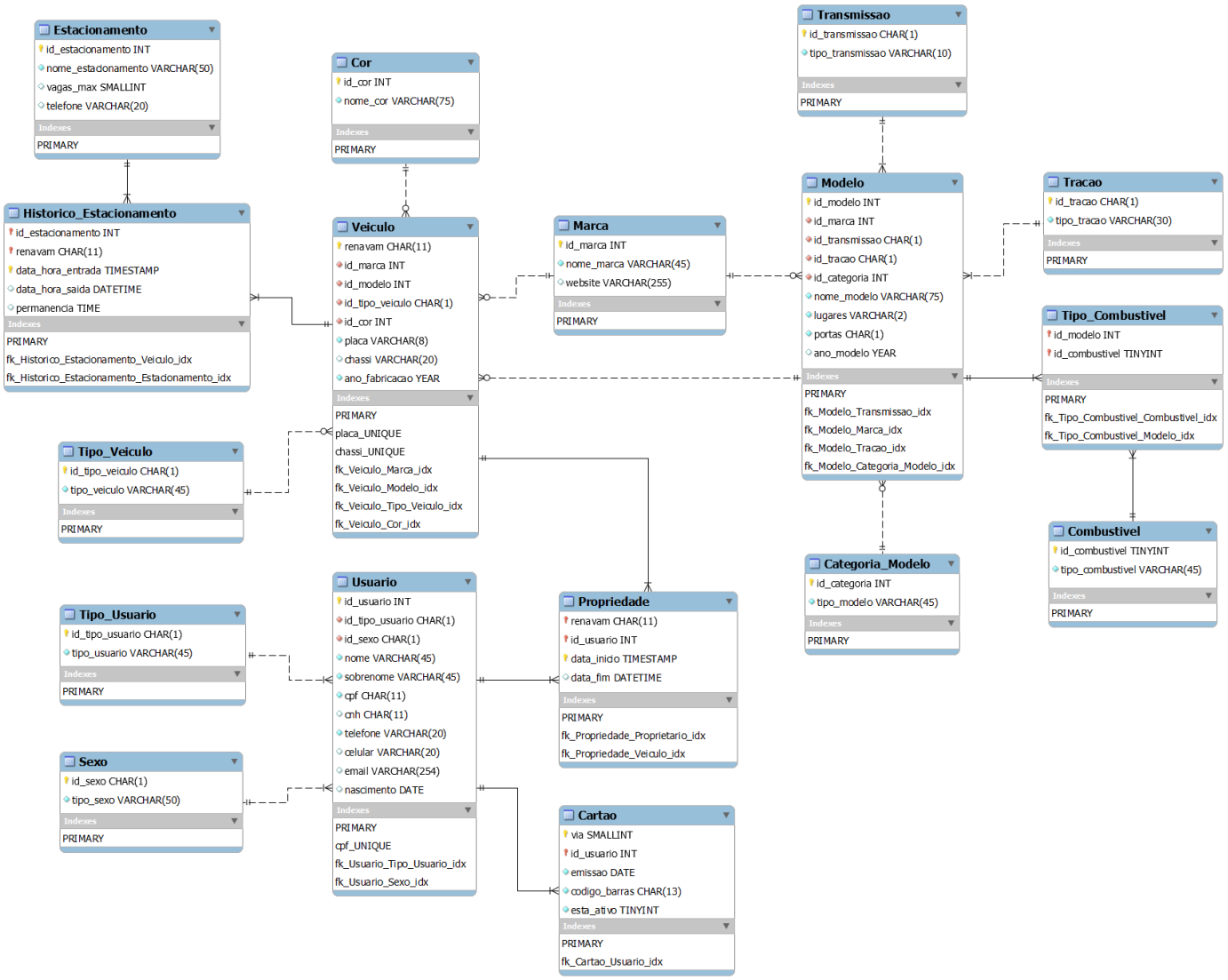
1. Pegar o arquivo **`estacionamento_universitario.sql`** https://raw.githubusercontent.com/profdiegoaugusto/banco-dados/master/mysql/linguagem-consulta-dados/data/estacionamento_universit%C3%A1rio.sql
2. Abrir o software MySQL
3. Executar o script para importar um novo banco de dados
4. Pegar o arquivo **`dml_estacionamento_universitario.sql`** https://raw.githubusercontent.com/profdiegoaugusto/banco-dados/master/mysql/linguagem-consulta-dados/data/dml_estacionamento.sql
5. Executar o script para inserir dados no banco de dados

1. Estacionamento Universitário

Uma Universidade deseja construir um sistema para facilitar e agilizar o controle de acesso de veículos aos seus cinco estacionamentos para prover mais segurança e comodidade para os seus usuários. O sistema deverá permitir que se cadastre todos os tipos de usuários (alunos, professores e funcionários), que receberão um cartão com um código de barras para sua identificação.

Cada usuário poderá solicitar o cadastramento de vários veículos com os quais utiliza os estacionamentos da universidade. Ao chegar a qualquer portão de acesso à universidade, o vigilante irá informar a placa do veículo e o usuário deverá passar o cartão magnético em um leitor de código de barras, e com isso, o sistema irá identificar se o veículo está relacionado com a identificação do usuário. Ao sair, o usuário simplesmente passa o seu cartão em outra leitora de código de barras.

O visitante (usuário não cadastrado) deverá pegar um cartão especial com os vigilantes. Através desses procedimentos, o sistema poderá fornecer dados de ocupação de cada estacionamento, além de permitir a consulta de quais os veículos estão, ou estiveram, dentro da universidade em um determinado dia e horário.


Estacionamento
id_estacionamento INT
nome_estacionamento VARCHAR(50)
vagas_max SMALLINT
telefone VARCHAR(20)
Indexes
PRIMARY
Cor
id_cor INT
nome_cor VARCHAR(75)
Indexes
PRIMARY
Transmissao
id_transmissao CHAR(1)
tipo_transmissao VARCHAR(10)
Indexes
PRIMARY
Historico_Estacionamento
id_estacionamento INT
renavam CHAR(11)
data_hora_entrada TIMESTAMP
data_hora_saida DATETIME
permanencia TIME
Indexes
PRIMARY
fk_Historico_Estacionamento_Veiculo_idx
fk_Historico_Estacionamento_Estacionamento_idx
Veiculo
renavam CHAR(11)
id_marca INT
id_modelo INT
id_tipo_veiculo CHAR(1)
id_cor INT
placa VARCHAR(8)
chassi VARCHAR(20)
ano_fabricacao YEAR
Indexes
PRIMARY
placa_UNIQUE
chassi_UNIQUE
fk_Veiculo_Marca_idx
fk_Veiculo_Modelo_idx
fk_Veiculo_Tipo_Veiculo_idx
fk_Veiculo_Cor_idx
Marca
id_marca INT
nome_marca VARCHAR(45)
website VARCHAR(255)
Indexes
PRIMARY
Modelo
id_modelo INT
id_marca INT
id_transmissao CHAR(1)
id_tracao CHAR(1)
id_categoria INT
nome_modelo VARCHAR(75)
lugares VARCHAR(2)
portas CHAR(1)
ano_modelo YEAR
Indexes
PRIMARY
fk_Modelo_Transmissao_idx
fk_Modelo_Marca_idx
fk_Modelo_Tracao_idx
fk_Modelo_Categoria_Modelo_idx
Tracao
id_tracao CHAR(1)
tipo_tracao VARCHAR(30)
Indexes
PRIMARY
Tipo_Combustivel
id_modelo INT
id_combustivel TINYINT
Indexes
PRIMARY
fk_Tipo_Combustivel_Combustivel_idx
fk_Tipo_Combustivel_Modelo_idx
Tipo_Veiculo
id_tipo_veiculo CHAR(1)
tipo_veiculo VARCHAR(45)
Indexes
PRIMARY
Categoria_Modelo
id_categoria INT
tipo_modelo VARCHAR(45)
Indexes
PRIMARY
Combustivel
id_combustivel TINYINT
tipo_combustivel VARCHAR(45)
Indexes
PRIMARY
Usuario
id_usuario INT
id_tipo_usuario CHAR(1)
id_sexo CHAR(1)
nome VARCHAR(45)
sobrenome VARCHAR(45)
cpf CHAR(11)
cnh CHAR(11)
telefone VARCHAR(20)
celular VARCHAR(20)
email VARCHAR(254)
nascimento DATE
Indexes
PRIMARY
cpf_UNIQUE
fk_Usuario_Tipo_Usuario_idx
fk_Usuario_Sexo_idx
Propriedade
renavam CHAR(11)
id_usuario INT
data_inicio TIMESTAMP
data_fim DATETIME
Indexes
PRIMARY
fk_Propriedade_Proprietario_idx
fk_Propriedade_Veiculo_idx
Tipo_Usuario
id_tipo_usuario CHAR(1)
tipo_usuario VARCHAR(45)
Indexes
PRIMARY
Sexo
id_sexo CHAR(1)
tipo_sexo VARCHAR(50)
Indexes
PRIMARY
Cartao
via SMALLINT
id_usuario INT
emissao DATE
codigo_barras CHAR(13)
esta_ativo TINYINT
Indexes
PRIMARY
fk_Cartao_Usuario_idx

1. Selecione o banco de dados (esquema) **`bd_estacionamento`**.

```
USE bd_estacionamento;
```

2. Selecione todas as informações dos estacionamentos

```
SELECT * FROM historico_estacionamento;
```

3. Usando JOIN Implícito, selecione a `placa, ano_fabricacao e tipo` dos veículos

```
SELECT veiculo.placa, veiculo.ano_fabricacao, veiculo.id_tipo_veiculo
FROM veiculo, historico_estacionamento
WHERE veiculo.renavam = historico_estacionamento.renavam
```

4. Usando JOIN Implícito, selecione a `placa, nome_marca e ano_fabricacao` dos veículos ordenados de maneira crescente pelo ano de fabricação

```
SELECT veiculo.placa, marca.nome_marca, veiculo.ano_fabricacao
FROM veiculo, historico_estacionamento, marca
WHERE veiculo.renavam = historico_estacionamento.renavam
  AND veiculo.id_marca = marca.id_marca
ORDER BY veiculo.ano_fabricacao;
```

5. Usando JOIN Implícito, selecione o `id_usuario, nome, sobrenome, id_sexo, telefone e papel` dos usuário, em ordem alfabética por nome

```
SELECT u.id_usuario, u.nome, u.sobrenome, u.id_sexo, u.telefone, t.papel
FROM usuario u, tipo_usuario t
WHERE u.id_tipo_usuario = t.id_tipo_usuario
ORDER BY u.nome;
```

6. Usando JOIN Implícito, selecione o `id_usuario, nome, sobrenome, cpf juntamente com todas as informações do Cartão de Estacionamento`

```
SELECT u.id_usuario, u.nome, u.sobrenome, u.cpf, c.*
FROM usuario u, cartao_estacionamento c
WHERE u.id_usuario = c.id_usuario;
```

7. Usando JOIN Implícito, selecione o `renavam, placa, chassi, nome_marca, nome_modelo, ano_fabricacao` dos veículo ordenados de maneira decrescente por ano de fabricação.

```
SELECT v.renavam, v.placa, v.chassi, m.nome_marca, mo.nome_modelo,
v.ano_fabricacao
FROM veiculo v, marca m, modelo mo
WHERE v.id_marca = m.id_marca
  AND v.id_modelo = mo.id_modelo
ORDER BY v.ano_fabricacao DESC;
```

8. Usando JOIN, selecione o `renavam, placa, chassi, nome_marca, nome_modelo, nome_cor, id_tipo_veiculo, ano_fabricacao` dos veículo ordenados de maneira decrescente por ano de fabricação.

```
SELECT v.renavam, v.placa, v.chassi, m.nome_marca, mo.nome_modelo,
c.nome_cor, v.id_tipo_veiculo, v.ano_fabricacao
FROM veiculo v
JOIN marca m ON v.id_marca = m.id_marca
JOIN modelo mo ON v.id_modelo = mo.id_modelo
JOIN cor c ON v.id_cor = c.id_cor
JOIN tipo_veiculo tv ON v.id_tipo_veiculo = tv.id_tipo_veiculo
ORDER BY v.ano_fabricacao DESC;
```

9. Usando JOIN, selecione id_modelo, nome_marca, nome_modelo, tipo_combustivel dos modelos de veículos

```
SELECT mo.id_modelo, mo.nome_modelo, m.nome_marca, c.tipo_combustivel
FROM modelo mo
JOIN marca m ON mo.id_marca = m.id_marca
JOIN tipo_combustivel tc ON mo.id_modelo = tc.id_modelo
JOIN combustivel c ON tc.id_combustivel = c.id_combustivel
```

10. Em cada um dos estacionamentos quais veículos entraram e saíram?

```
SELECT e.nome_estacionamento, he.renavam, he.data_hora_entrada,
he.data_hora_saida
FROM estacionamento e
LEFT JOIN historico_estacionamento he ON e.id_estacionamento =
he.id_estacionamento;
```

11. Qual é a média de tempo de permanência dos veículos em cada um dos estacionamentos

```
SELECT e.nome_estacionamento, AVG(he.permanencia) AS media_permanencia
FROM estacionamento e
LEFT JOIN historico_estacionamento he ON e.id_estacionamento =
he.id_estacionamento
GROUP BY e.nome_estacionamento;
```

12. Mostre quais estacionamentos possuem veículos e quantos veículos estão estacionados em cada um deles

```
SELECT e.nome_estacionamento, COUNT(he.id_estacionamento) AS
total_veiculos_estacionados
FROM estacionamento e
LEFT JOIN historico_estacionamento he ON e.id_estacionamento =
he.id_estacionamento
GROUP BY e.nome_estacionamento;
```

13. Usando JOIN, selecione id_usuario, nome, sobrenome, telefone, papel, placa, chassi, data_inicio, data_fim dos proprietários de veículos

```
SELECT u.id_usuario, u.nome, u.sobrenome, u.telefone, tu.papel, v.placa,
v.chassi, p.data_inicio, p.data_fim
FROM usuario u
JOIN propriedade p ON u.id_usuario = p.id_usuario
JOIN veiculo v ON p.renavam = v.renavam
JOIN tipo_usuario tu ON u.id_tipo_usuario = tu.id_tipo_usuario;
```

14. Usando JOIN, selecione renavam, placa, chassi, nome_marca, nome_modelo, lugares, portas, nome_cor, tipo_transmissao, tipo_tracao, tipo, categoria, tipo_combustivel, ano_modelo, ano_fabricacao dos veículos

```
SELECT v.renavam, v.placa, v.chassi, m.nome_marca, mo.nome_modelo, mo.lugares,
mo.portas, c.nome_cor, t.tipo_transmissao, tr.tipo_tracao, 'Tipo de Veículo' AS
tipo_veiculo, cm.categoria, 'Tipo de Combustível' AS tipo_combustivel,
mo.ano_modelo, v.ano_fabricacao
FROM veiculo v
JOIN marca m ON v.id_marca = m.id_marca
JOIN modelo mo ON v.id_modelo = mo.id_modelo
JOIN cor c ON v.id_cor = c.id_cor
JOIN transmissao t ON mo.id_transmissao = t.id_transmissao
JOIN tracao tr ON mo.id_tracao = tr.id_tracao
JOIN categoria_modelo cm ON mo.id_categoria = cm.id_categoria;
```

15. Usando LEFT JOIN, Quais tipos de veículos não estão estacionados?

```
SELECT tv.tipo
FROM tipo_veiculo tv
LEFT JOIN veiculo v ON tv.id_tipo_veiculo = v.id_tipo_veiculo
WHERE v.id_tipo_veiculo IS NULL;
```

16. Usando LEFT JOIN, mostre todas as informações das tabelas Tipo_Veiculo, Veiculo e Historico_Estacionamento

```
SELECT tv.*, v.*, he.*
FROM tipo_veiculo tv
LEFT JOIN veiculo v ON tv.id_tipo_veiculo = v.id_tipo_veiculo
LEFT JOIN historico_estacionamento he ON v.renavam = he.renavam;
```

17. Usando LEFT JOIN, Quais tipos de veículos não estão parados no estacionamento?

```
SELECT tv.tipo
FROM tipo_veiculo tv
LEFT JOIN veiculo v ON tv.id_tipo_veiculo = v.id_tipo_veiculo
LEFT JOIN historico_estacionamento he ON v.renavam = he.renavam
WHERE he.id_estacionamento IS NULL;
```

18. Usando LEFT JOIN, mostre todas informações das cores e dos veículos, ordenado de maneira alfabética por nome da cor

```
SELECT c.*, v.*
FROM cor c
LEFT JOIN veiculo v ON c.id_cor = v.id_cor
ORDER BY c.nome_cor;
```

19. Usando RIGHT JOIN, quais os tipos de combustíveis que não estão presentes nos modelos de veículos?

```
SELECT tc.*
FROM tipo_combustivel tc
RIGHT JOIN modelo mo ON tc.id_modelo = mo.id_modelo
WHERE mo.id_modelo IS NULL;
```

20. Usando RIGHT JOIN, mostre todas as informações das tabelas Modelo, Tipo_Combustivel e Combustível.

```
SELECT mo.*, tc.*, c.*
FROM modelo mo
RIGHT JOIN tipo_combustivel tc ON mo.id_modelo = tc.id_modelo
RIGHT JOIN combustivel c ON tc.id_combustivel = c.id_combustivel;
```

21. Usando RIGHT JOIN, selecione o nome e website de todas as marcas que não possuem veículos (RIGHT)

```
SELECT m.nome_marca, m.website
FROM marca m
RIGHT JOIN veiculo v ON m.id_marca = v.id_marca
WHERE v.id_marca IS NULL;
```

22. Usando NATURAL JOIN, Mostre id_usuario, id_tipo_usuario, nome, sobrenome, id_sexo, telefone, placa, data_hora_entrada, data_hora_saida dos proprietários de veículos não saíram do estacionamento

```
SELECT u.id_usuario, u.id_tipo_usuario, u.nome, u.sobrenome, u.id_sexo,
u.telefone, v.placa, he.data_hora_entrada, he.data_hora_saida
FROM usuario u
INNER JOIN propriedade p ON u.id_usuario = p.id_usuario
INNER JOIN veiculo v ON p.renavam = v.renavam
LEFT JOIN historico_estacionamento he ON v.renavam = he.renavam
WHERE he.data_hora_saida IS NULL;
```

23. Usando NATURAL JOIN, selecione as informações dos proprietários de veículos com final de placa 5

```
SELECT u.id_usuario, u.id_tipo_usuario, u.nome, u.sobrenome, u.id_sexo,
u.telefone, v.placa
FROM usuario u
INNER JOIN propriedade p ON u.id_usuario = p.id_usuario
INNER JOIN veiculo v ON p.renavam = v.renavam
WHERE SUBSTRING(v.placa, -1) = '5';
```

24. Usando CROSS JOIN, mostre uma lista com o relacionamento do id e nomes de todas as marcas com o id e nome de todos os tipos de veículos.

```
SELECT U.id_usuario, U.id_tipo_usuario, U.nome, U.sobrenome, U.id_sexo,
U.telefone, V.placa, HE.data_hora_entrada, HE.data_hora_saida
FROM Usuario U
NATURAL JOIN Veiculo V
LEFT JOIN Historico_Estacionamento HE ON V.renavam = HE.renavam
WHERE HE.data_hora_saida IS NULL AND HE.data_hora_entrada IS NOT NULL;
```