Visualização de Dados e Introdução a SQL [24E4_1]

Prof. Elberth Moraes

Elias Matos Martins

TESTE DE PERFORMANCE - 3

25 de novembro de 2024

CONTEXTO DO PROJETO

Fabricação de Carros

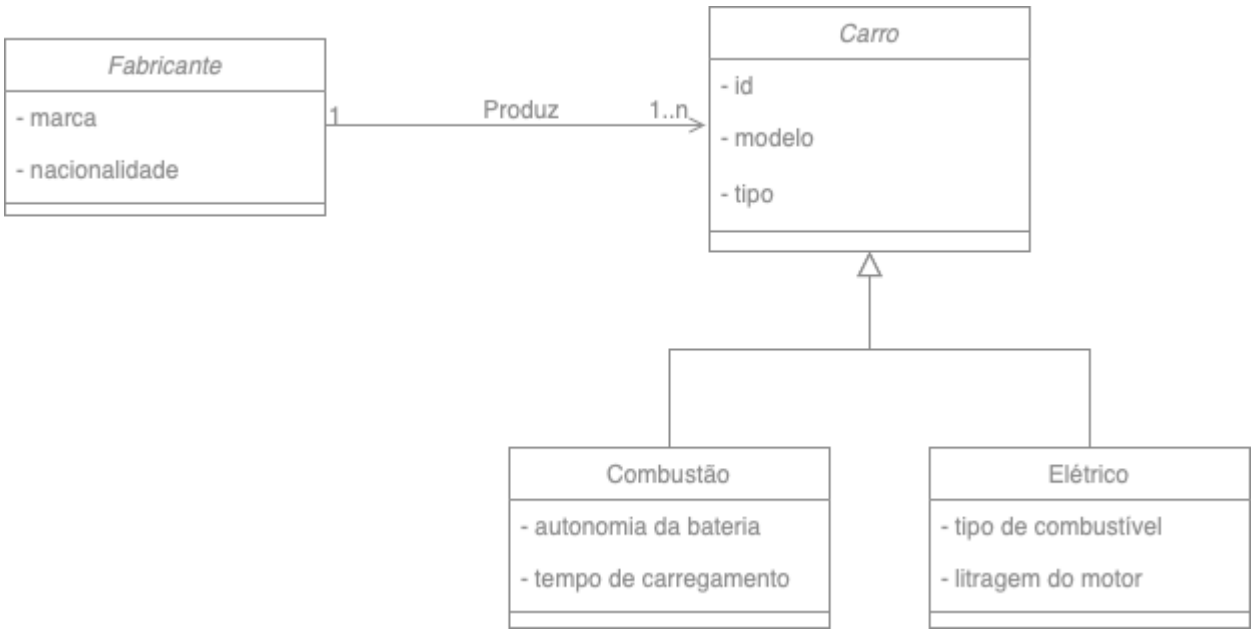
O objetivo deste projeto é monitorar os carros produzidos por diferentes fabricantes, categorizando-os em dois tipos principais (Elétricos e a Combustão). O dashboard fornecerá insights sobre a produção total, desempenho de vendas por fabricante, e características dos veículos, permitindo a análise do mercado e da eficiência produtiva.

Estrutura e Campos de Dados

Classes e Relacionamentos

1. **Fabricante**: contém atributos como marca e nacionalidade e representa os fabricantes responsáveis pela produção dos carros. Cada um pode oferecer diversos modelos
2. **Carro**: representa as características gerais de um veículo.
    - **Elétrico: i**ncluem características específicas como autonomia da bateria e tempo de carregamento
    - **Combustão: t**ipo de combustível e litragem do motor

Relação: Cada carro pertence a um fabricante, mas um fabricante pode produzir múltiplos carros.



CRIÇÃO DA BASE DE INFORMAÇÕES EM PLANILHA

Para darmos continuidade ao projeto de visualização e análise de dados, vamos criar uma planilha de dados em formato CSV que servirá como base de informações para o dashboard. A base será composta por um conjunto de dados desnormalizados, utilizando os campos definidos anteriormente, como nome e nacionalidade do fabricante; modelo, tipo de motor e etc.

A planilha incluirá 50 linhas de dados fictícios com os seguintes campos:

- **Fabricante:** Marca, Nacionalidade
- **Carro:** ID, Modelo, Tipo
- **Elétrico:** Autonomia da bateria, Tempo de carregamento
- **Combustão:** Tipo de combustível, Litragem do motor

A Planilha

Segue o link para um arquivo CSV contendo uma base de dados desnormalizada, projetada para facilitar a análise de carros e seus fabricantes. Cada linha representa um carro com informações como fabricante, nacionalidade, ID, modelo e tipo de veículo. Para veículos elétricos, incluem-se autonomia e tempo de carregamento; para os modelos a combustão, tipo de combustível e litragem do motor. Essa estrutura simplifica a exploração e visualização inicial dos dados, acelerando a geração de insights.

Base de dados para análise

DICIONÁRIO DE DADOS

Para facilitar o entendimento e análise dos dados, elaboramos o dicionário a seguir com a descrição detalhada dos campos presentes na planilha de controle de vendas. Cada campo foi classificado conforme seu tipo de dado e objetivo, proporcionando uma visão completa sobre a estrutura de dados utilizada.

TABELA FABRICANTE

Campo	Descrição	Tipo
ID	Identificador único do carro	Inteiro
Fabricante	Nome da empresa responsável pela produção do veículo	Texto
Nacionalidade	País de origem do fabricante	Texto
Tipo	Tipo do carro (Elétrico ou Combustão)	Texto
Autonomia	Quantidade de km que um carro elétrico pode rodar com uma carga completa	Texto
Tempo de carregamento	Tempo em horas para concluir o carregamento total do veículo quando descarregado.	Real
Tipo de combustível	Para os carros a combustão. Podem ser Gasolina, Diesel ou Flex	Texto
Litragem do motor	Para carros a combustão. Indica quantos litros de combustível o motor admite para gerar torque. Ex.: 1.5, 1.8, 2.0, etc	Real
Quantidade produzida	Indica a quantidade de unidades produzidas para cada modelo	Inteiro

Link para o dashboard de análise

SCRIPT SQL PARA CRIAÇÃO DO BANCO

Criação das Tabelas

```
C/C++
-- Criação da tabela Fabricantes
CREATE TABLE fabricantes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    nacionalidade VARCHAR(50) NOT NULL
);

-- Criação da tabela Carros
CREATE TABLE carros (
    id INT AUTO_INCREMENT PRIMARY KEY,
    modelo VARCHAR(100) NOT NULL,
    tipo VARCHAR(50) NOT NULL CHECK (tipo IN ('Elétrico', 'Combustão')),
    fabricante_id INT NOT NULL,
    autonomia_km INT, -- Aplicável somente para carros elétricos
    tempo_carregamento_h REAL, -- Aplicável somente para carros elétricos
    tipo_combustivel VARCHAR(20), -- Aplicável somente para carros a combustão
    litragem_motor REAL, -- Aplicável somente para carros a combustão
    quantidade_produzida INT NOT NULL,
    FOREIGN KEY (fabricante_id) REFERENCES fabricantes(id)
);
```

População da Tabela fabricantes

```
C/C++
INSERT INTO fabricantes (nome, nacionalidade)
VALUES
    ('Tesla', 'Estados Unidos'),
    ('Toyota', 'Japão'),
    ('Volkswagen', 'Alemanha'),
    ('Ford', 'Estados Unidos'),
    ('BMW', 'Alemanha');
```

População da Tabela carros

```
C/C++
INSERT INTO carros (modelo, tipo, fabricante_id, autonomia_km,
tempo_carregamento_h, tipo_combustivel, litragem_motor, quantidade_produzida)
VALUES
```

```
-- Carros elétricos
('Model S', 'Elétrico', 1, 600, 1.5, NULL, NULL, 5000),
('Model 3', 'Elétrico', 1, 500, 1.2, NULL, NULL, 8000),
('i3', 'Elétrico', 5, 300, 2.0, NULL, NULL, 2000),

-- Carros a combustão
('Corolla', 'Combustão', 2, NULL, NULL, 'Gasolina', 1.8, 10000),
('Passat', 'Combustão', 3, NULL, NULL, 'Diesel', 2.0, 6000),
('Mustang', 'Combustão', 4, NULL, NULL, 'Gasolina', 5.0, 2000),
('320i', 'Combustão', 5, NULL, NULL, 'Flex', 2.0, 3000);
```

Dicionário

- **Tabela** fabricantes**:** Contém a lista de fabricantes, com ID, nome e nacionalidade.
- **Tabela** carros**:** Armazena os dados dos carros, incluindo:
  - modelo: Nome do modelo do carro.
  - tipo: Indica se é “Elétrico” ou “Combustão”.
  - fabricante_id: Relaciona o carro ao fabricante por chave estrangeira.
  - autonomia_km e tempo_carregamento_h: Aplicáveis para carros elétricos.
  - tipo_combustivel e litragem_motor: Aplicáveis para carros a combustão.
  - quantidade_produzida: Total de unidades fabricadas.

CONSULTAS SQL

Consulta básica com SELECT e WHERE

Requisito funcional: Obter os carros do tipo “Elétrico” fabricados por empresas dos Estados Unidos.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante, nacionalidade
FROM carros
WHERE tipo = 'Elétrico' AND nacionalidade = 'Estados Unidos';
```

Consulta com ORDER BY

Requisito funcional: Listar todos os carros ordenados por modelo em ordem crescente.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante
FROM carros
ORDER BY modelo ASC;
```

Consulta única com DISTINCT

Requisito funcional: Identificar todas as nacionalidades distintas dos fabricantes na base.

Consulta SQL:

```
C/C++
SELECT DISTINCT nacionalidade FROM fabricantes;
```

Consulta com WHERE e operadores de comparação

Requisito funcional: Encontrar os carros elétricos com autonomia maior que 400 km.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante, autonomia
FROM carros
WHERE tipo = 'Elétrico' AND autonomia > 400;
```

Consulta com BETWEEN

Requisito funcional: Listar carros a combustão com litragem do motor entre 1.5 e 2.0 litros.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante, litragem_motor
FROM carros
WHERE tipo = 'Combustão' AND litragem_motor BETWEEN 1.5 AND
2.0;
```

Consulta com IN

Requisito funcional: Obter os carros fabricados pela Toyota, Ford ou Tesla.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante
FROM carros
WHERE fabricante IN ('Toyota', 'Ford', 'Tesla');
```

Consulta com LIKE para busca parcial

Requisito funcional: Buscar carros cujo modelo começa com “Model”.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante
FROM carros
WHERE modelo LIKE 'Model%';
```

Combinação de AND e OR

Requisito funcional: Encontrar carros elétricos com autonomia maior que 300 km ou carros a combustão que usam gasolina.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante, tipo, autonomia,
tipo_combustivel
FROM carros
WHERE (tipo = 'Elétrico' AND autonomia > 300) OR (tipo =
'Combustão' AND tipo_combustivel = 'Gasolina');
```

Filtro com IS NULL

Requisito funcional: Listar carros cujo tempo de carregamento não foi informado.

Consulta SQL:

```
C/C++
SELECT id, modelo, fabricante, tempo_carregamento
FROM carros
WHERE tempo_carregamento IS NULL;
```

Uso de COALESCE

Requisito funcional: Substituir valores nulos em “tempo de carregamento” por “0 horas”.

Consulta SQL:

```
C/C++
SELECT id, modelo, COALESCE(tempo_carregamento, 0) AS
tempo_carregamento
FROM carros;
```

Agrupamento com GROUP BY

Requisito funcional: Contar o número de modelos fabricados por cada fabricante.

Consulta SQL:

```
C/C++
SELECT fabricante, COUNT(*) AS total_modelos
FROM carros
GROUP BY fabricante;
```

Soma (SUM) com agrupamento

Requisito funcional: Calcular o total de carros produzidos por cada tipo (Elétrico ou Combustão).

Consulta SQL:

```
C/C++
SELECT tipo, SUM(quantidade_produzida) AS total_producao
FROM carros
GROUP BY tipo;
```

Contagem de registros com COUNT

Requisito funcional: Contar quantos carros existem na base.

Consulta SQL:

```
C/C++
SELECT COUNT(*) AS total_carros
FROM carros;
```

Cálculo de valores mínimo (MIN) e máximo (MAX)

Requisito funcional: Identificar a menor e a maior autonomia entre carros elétricos.

Consulta SQL:

```
C/C++
SELECT MIN(autonomia) AS menor_autonomia, MAX(autonomia) AS
maior_autonomia
FROM carros
WHERE tipo = 'Elétrico';
```

Cálculo de média (AVG) com arredondamento (ROUND)

Requisito funcional: Calcular a média de litragem dos motores de carros a combustão.

Consulta SQL:

```
C/C++
SELECT ROUND(AVG(litragem_motor), 2) AS media_litragem
FROM carros
WHERE tipo = 'Combustão';
```

Filtro de agregações com HAVING

Requisito funcional: Listar fabricantes com mais de 1000 carros produzidos no total.

Consulta SQL:

```C/C++
SELECT fabricante, SUM(quantidade_produzida) AS total_producao
FROM carros
GROUP BY fabricante
HAVING SUM(quantidade_produzida) > 1000;
```