Alunos: Jonathan Neves Alencar

Lucas Alves da Costa

**Banco de Dados da Fórmula 1**

O nosso banco de dados organiza as informações básicas sobre as temporadas da fórmula 1, armazenando informações dos pilotos, pistas, equipes, grandes prêmios.

Ele consulta informações como a colocação de uma temporada e o pódio de cada grande prêmio.

As tabelas criadas foram: País, que tem como objetivo guardar informações de um país, como o id e o nome. Equipes, guardar informações nome da equipe, país que ela pertence. Pilotos, tem os dados como o nome do piloto, nacionalidade, equipe que representa. Pista guarda o país da pista e o nome da pista. Grande Prêmio que é responsável por reunir informações como pista que a corrida aconteceu, ano da corrida, e os três piloto do pódio. E a tabela pontos por corrida, que é responsável pela atribuição de pontos, dando 3 para o primeiro, 2 para o segundo, e 1 para o terceiro.

Uma view que criamos foi campeão da temporada, para visualizar o campeão de uma determinada temporada da fórmula 1. Foi escolhido a view ao invés de uma consulta porque um campeão de uma certa temporada é uma informação persistente.

**Consultas feitas**

* **Consulta de pontuação**: Visualizar o total de pontos de um piloto específico em uma temporada.

SELECT

ppc.piloto\_id,

gp.ano,

p.nome,

SUM(ppc.pontos) AS total\_pontos

FROM pontos\_por\_corrida ppc

JOIN grande\_premio gp ON ppc.grande\_premio\_id = gp.id

join piloto p on p.id = ppc.piloto\_id

WHERE p.nome = 'Fernando Alonso' AND gp.ano = 2022

GROUP BY ppc.piloto\_id, gp.ano, p.nome;

* **Listagem dos Campeões**: Verificar quais pilotos foram campeões em cada temporada.

SELECT

ano,

piloto\_id,

piloto\_nome,

total\_pontos

FROM (

SELECT

gp.ano,

ppc.piloto\_id,

p.nome AS piloto\_nome,

SUM(ppc.pontos) AS total\_pontos,

RANK() OVER (PARTITION BY gp.ano ORDER BY SUM(ppc.pontos) DESC) AS posicao

FROM pontos\_por\_corrida ppc

JOIN grande\_premio gp ON ppc.grande\_premio\_id = gp.id

JOIN piloto p ON ppc.piloto\_id = p.id

GROUP BY gp.ano, ppc.piloto\_id, p.nome

) AS ranking

WHERE posicao = 1

ORDER BY ano DESC;

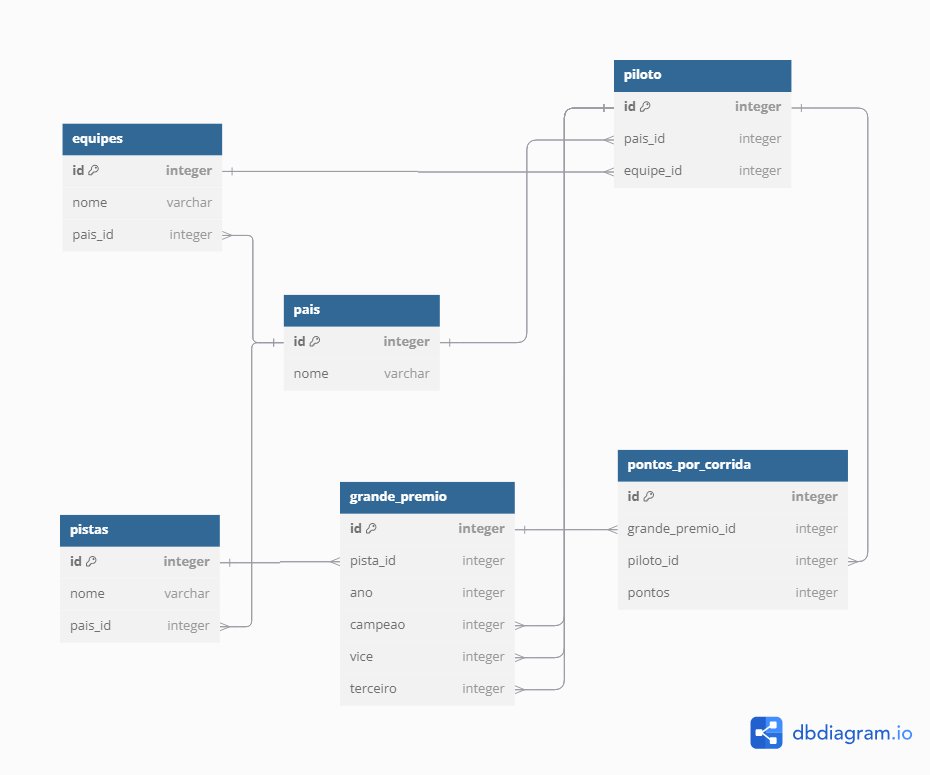
* **Rankings de Temporada**: A view campeao\_temporada fornece o piloto com a maior pontuação em uma temporada, simplificando a consulta de campeões e vice-campeões

SELECT ano, piloto\_id, nome, total\_pontos

FROM campeao\_temporada

ORDER BY ano, total\_pontos DESC;

**Modelagem do banco de dados**



**Criação das Tabelas**

CREATE TABLE pais (

    id SERIAL PRIMARY KEY,

    nome VARCHAR(50) NOT NULL

);

CREATE TABLE equipes (

    id SERIAL PRIMARY KEY,

    nome VARCHAR(50) NOT NULL,

    pais\_id INT REFERENCES pais(id) NOT NULL

);

CREATE TABLE piloto (

    id SERIAL PRIMARY KEY,

    nome VARCHAR(50) NOT NULL,

    pais\_id INT REFERENCES pais(id) NOT NULL,

    equipe\_id INT REFERENCES equipes(id)

);

CREATE TABLE pistas (

    id SERIAL PRIMARY KEY,

    nome VARCHAR(50) NOT NULL,

    pais\_id INT REFERENCES pais(id) NOT NULL

);

CREATE TABLE grande\_premio (

    id SERIAL PRIMARY KEY,

    pista\_id INT REFERENCES pistas(id) NOT NULL,

    ano INT NOT NULL,

    campeao INT REFERENCES piloto(id) NOT NULL,

    vice INT REFERENCES piloto(id) NOT NULL,

    terceiro INT REFERENCES piloto(id) NOT NULL

);

CREATE TABLE pontos\_por\_corrida (

    id SERIAL PRIMARY KEY,

    grande\_premio\_id INT REFERENCES grande\_premio(id),

    piloto\_id INT REFERENCES piloto(id),

    pontos INT NOT NULL

);

CREATE VIEW campeao\_temporada AS

SELECT ano, piloto\_id, nome, total\_pontos FROM (

    SELECT

        gp.ano,

        p.id AS piloto\_id,

        p.nome,

        SUM(ppc.pontos) AS total\_pontos,

        RANK() OVER (PARTITION BY gp.ano ORDER BY SUM(ppc.pontos) DESC) AS posicao

    FROM grande\_premio gp

    JOIN pontos\_por\_corrida ppc ON gp.id = ppc.grande\_premio\_id

    JOIN piloto p ON ppc.piloto\_id = p.id

    GROUP BY gp.ano, p.id, p.nome

) sub

WHERE posicao = 1;

**Index HASH**

CREATE INDEX ON pontos\_por\_corrida USING HASH (piloto\_id);

O funcionamento do index Hash é exatamente da mesma forma que aprendemos essa estrutura de dados HASH em estrutura de dados 1 (AED1), ele a partir de uma entrada, por exemplo um número, aplica uma função hash, essa função gerará um valor associado ao número dado de entrada, esse valor é chamado de valor hash e é armazenado em uma tabela de dispersão, é o índice dessa tabela. Exemplo, aplicando a função hash para id = 1 da coluna piloto\_id

hash(1) = 1 \* 3 % 7 = 3

Isso significa que na tabela hash, basta acessar o índice 3 da tabela de dispersão que acharemos diretamente o piloto com id = 1. Caso fosse um índice de árvore b+, para achar o id=1, seria necessário percorrer uma árvore nó a nó, fazendo muitas verificações, e caso o índice fosse uma tabela sequencial a busca iria verificar posição a posição o que também é muito ineficiente, daí o índice hash é muito mais direto ao ponto, basta aplicar a função hash que teremos o índice buscado. A sua única desvantagem é que ele não funciona para buscas por intervalo, exemplo: id > 4, ele não funciona porque a função não ordena os dados, logo não necessariamente após o valor hash de id =3, será o valor hash de id =4.

Daí sabendo que HASH é ideal para operações de igualdade, operação que é realizada repetidas vezes para o cálculo tanto para saber quem foi o campeão da temporada, quanto para se informar sobre a classificação de uma temporada.

**Cenário de Utilização**

CREATE VIEW campeao\_temporada AS

SELECT ano, piloto\_id, nome, total\_pontos FROM (

    SELECT

        gp.ano,

        p.id AS piloto\_id,

        p.nome,

        SUM(ppc.pontos) AS total\_pontos,

        RANK() OVER (PARTITION BY gp.ano ORDER BY SUM(ppc.pontos) DESC) AS posicao

    FROM grande\_premio gp

    JOIN pontos\_por\_corrida ppc ON gp.id = ppc.grande\_premio\_id

    JOIN piloto p ON ppc.piloto\_id = p.id

    GROUP BY gp.ano, p.id, p.nome

) sub

WHERE posicao = 1;

É uma view onde me diz quem é o campeão daquela temporada. Fazendo uso repetidas vezes do index no momento da somatória dos pontos dos pilotos, por isso o index hash é adequado já que ele funciona rápido para buscas de valores exatos.