UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

SCC0541 - Laboratório de Base de Dados (2025) – 1º Semestre 2025

Prof. Caetano Traina Junior

Projeto Prático - P3
Banco de Dados para Fórmula 1

Nome: Eduardo Ribeiro Rodrigues NUSP: 13696679
Nome: Gustavo Blois NUSP: 13688162
Nome: Jade Bortot de Paiva NUSP: 11372883
Nome: Matheus Godoy Bolsarini NUSP: 9896201

Sumário

1. DDL

Para esta parte do projeto, foi feito um script para a criação de 15 tabelas divididas em três principais categorias, das quais são:

- **Entidades do domínio Fórmula 1:**
    - Circuits
      Guarda as informações referentes ao circuito em si, armazenando, resumidamente, seu nome e localidade, dentre outros atributos pertinentes.
    - Constructors
      Guarda as informações dos construtores.
    - Drivers
      Guarda as informações dos pilotos.
    - Seasons
      Guarda as temporadas de competição.
    - Races
      Guarda as informações de cada corrida ocorrida em uma temporada

- **Tabelas de relacionamento e estatísticas:**
    - DriverStandings
      Guarda qual a classificação que os pilotos ficaram em cada corrida.
    - LapTimes
      Guarda o tempo de cada volta de cada piloto.
    - PitStops
      Guarda as paradas que os pilotos fizeram durante a corrida.
    - Qualifying
      Guarda os resultados das sessões classificatórias.
    - Status
      Guarda o status do piloto no final da corrida.
    - Results
      Guarda os resultados finais das corridas.

- **Tabelas de apoio geográfico:**
    - Countries
      Guarda as informações sobre os países.
    - Airports
      Guarda os dados sobre os aeroportos.
    - Goecities5k
      Guarda os dados geográficos de cidades.

Foram definidas as chaves primárias e sempre que necessário as chaves estrangeiras para garantir a integridade referencial que consta no MER-X. Para a criação das tabelas, seguimos o formato demonstrado na figura 1 abaixo.

```sql
CREATE TABLE Races (
    RaceId INT NOT NULL,
    Year INT NOT NULL,
    Round INT,
    CircuitId INT,
    Name TEXT NOT NULL,
    Date DATE NOT NULL,
    Time TIME,
    URL TEXT,
    fp1_date DATE,
    fp1_time TIME,
    fp2_date DATE,
    fp2_time TIME,
    fp3_date DATE,
    fp3_time TIME,
    quali_date DATE,
    quali_time TIME,
    sprint_date DATE,
    sprint_time TIME,

    CONSTRAINT PK_Races PRIMARY KEY (RaceId),
    CONSTRAINT FK_Races_Circuit FOREIGN KEY (CircuitId) REFERENCES
    Circuits(CircuitID) ON DELETE CASCADE,
    CONSTRAINT FK_Races_Season FOREIGN KEY (Year) REFERENCES Seasons
    (Year) ON DELETE CASCADE
);
```

Figura 1: Estrutura de criação da tabela Races

Dentro dos conjuntos de dados que recebemos para popular essas tabelas, notamos a presença de outras tabelas que não estavam mapeadas anteriormente. Decidimos por não adicioná-las, pois achamos pertinente seguir o mapeamento para depois, se necessário, adicionarmos as tabelas no nosso banco.

2. DML

Para preencher as tabelas criadas com os arquivos CSV e TSV, utilizamos o comando *\COPY,* disponível no PostgreSQL, com o devido tratamento para lidar com as aspas e valores nulos encontrados e definimos a codificação como UTF-8 para gerar compatibilidade no conjunto de dados, a figura 2 exemplifica o que foi feito abaixo.

```sql
SET search_path TO 'LabBD25-Grupo7'; --Mudando o Schema do public
para o criado para o grupo

SET client_encoding to 'UTF8'; --Mudando a codifiação de dados do
cliente para UTF8 para ter match com o servidor

COPY Circuits(CircuitID, CircuitRef, Name, Location, Country, Lat,
Lng, Alt, URL)
FROM 'C:/Users/mathe/Desktop/LAB BD 25/DADOS LAB BD 25 - F1/circuits.
csv'
DELIMITER ',' CSV HEADER QUOTE '"' NULL '\N';
```

Figura 2: Tratamento de aspas, nulos e compatibilidade de dados

Alguns campos que armazenam variáveis de tempo foram definidas com o tipo *VARCHAR,* devido o formato dos dados não ser compatível com o formato aceito pelo tipo *TIME,* como *MM:SS.MS.*
Outro ocorrido nas inserções foi a omissão do *NULL '\N'* por incompatibilidade com o conteúdo dos arquivos.
Também criamos um script para exclusão ordenada das tabelas porque vimos a necessidade de reinicializar o banco e esse script facilita muito esse processo.

```
-- Remover todas as tabelas do esquema F1 (ordem correta para evitar
erros de FK)

SET search_path TO 'LabBD25-Grupo7'; --Mudando o Schema do public
para o criado para o grupo

DROP TABLE IF EXISTS
    Results,
    Qualifying,
    PitStops,
    LapTimes,
    DriverStandings,
    Races,
    Drivers,
    Constructors,
    Circuits,
    Seasons,
    Status,
    Airports,
    Countries,
    Geocities5k
CASCADE; --para garantir que mesmo se houver alguma dependência
residual, a tabela será removida.
```

Figura 3: Script para realizar a exclusão ordenada das tabelas do banco inteiro.

3. Resultados de população

Para conferir se as inserções correram para todos os dados, estamos realizando a contagem de registros de cada tabela seguindo o mesmo escopo, como na figura 4:

```sql
SET search_path TO 'LabBD25-Grupo7'; --Mudando o Schema do public para o criado para o grupo

SELECT * FROM (
SELECT 'Circuits' AS table_name, COUNT(*) AS total_rows FROM Circuits
UNION ALL
SELECT 'Constructors', COUNT(*) FROM Constructors
UNION ALL
SELECT 'Drivers', COUNT(*) FROM Drivers
UNION ALL
SELECT 'Seasons', COUNT(*) FROM Seasons
UNION ALL
SELECT 'Races', COUNT(*) FROM Races
UNION ALL
SELECT 'DriverStandings', COUNT(*) FROM DriverStandings
UNION ALL
SELECT 'LapTimes', COUNT(*) FROM LapTimes
UNION ALL
SELECT 'PitStops', COUNT(*) FROM PitStops
UNION ALL
SELECT 'Qualifying', COUNT(*) FROM Qualifying
UNION ALL
SELECT 'Status', COUNT(*) FROM Status
UNION ALL
SELECT 'Results', COUNT(*) FROM Results
UNION ALL
SELECT 'Countries', COUNT(*) FROM Countries
UNION ALL
SELECT 'Airports', COUNT(*) FROM Airports
UNION ALL
SELECT 'Geocities5k', COUNT(*) FROM Geocities5k) AS TODO ORDER BY 2 DESC;
```

Figura 4: Consulta para contagem de registros de cada tabela

Executando esse script para a contagem de registros de todas as tabelas, temos os seguintes valores:

	table_name text	total_rows bigint
1	LapTimes	589081
2	Airports	82796
3	DriverStandings	34863
4	Geocities5k	29828
5	Results	26759
6	PitStops	11371
7	Qualifying	10494
8	Races	1125
9	Drivers	861
10	Countries	249
11	Constructors	212
12	Status	139
13	Circuits	77
14	Seasons	75

Figura 5: Contagem de linhas das relações

4. Servidor

O servidor postgres está hospedado numa vps da digital ocean, com o sistema operacional debian 12, 1 GB de memória RAM e 25 GB de memória secundária.

A versão do postgres utilizada é a 15.12. Foi escolhido o Postgresql por ser uma alternativa opensource ao oracle-sql e com features de QoL úteis como o suporte à inserções em tabelas usando o Copy e o tipo de dados Text, e a versão 15 ao invés da 17 por ser a versão padrão no gerenciador de pacotes apt.

O servidor possui os schemas Public e LabBD25-Grupo7, o banco de dados labbd, e a role a13688162. O servidor possui serviços apenas nas portas 5432 (postgres) e 22 (ssh). A role “postgres” pode ser acessada apenas pelo root, de dentro do servidor.