Alejandro J Bautista

Formula1 DB

Project Description

Managing Formula 1 with MySQL

- This relational database is designed to manage and store detailed information related to Formula 1, one of the most prestigious and followed motor racing competitions in the world. Using MySQL, a relational database management system, this project allows for efficient and structured organization and analysis of historical and current competition data.

Objective

- The main objective of this database is to provide a solid and flexible structure to record and query information about Formula 1 seasons, including race details, teams, drivers, engineers, sponsors, and results. The database is designed to facilitate data analysis, statistics generation, and retrieval of relevant information for fans, journalists, and sports analysts.

Key Components

1. Seasons and Races:
- The database stores information about different Formula 1 seasons and the races held in each season, including specific race data such as date and location.

2. Teams and Drivers:
- Teams competing in Formula 1 are registered along with the drivers who are part of each team. The database allows tracking the career paths of drivers and their associations with different teams over time.

3. Engineering Teams and Sponsors:
- The database also includes information about the engineering teams behind the teams' cars and the sponsors supporting them. This information is crucial for understanding the dynamics and financing behind each team.

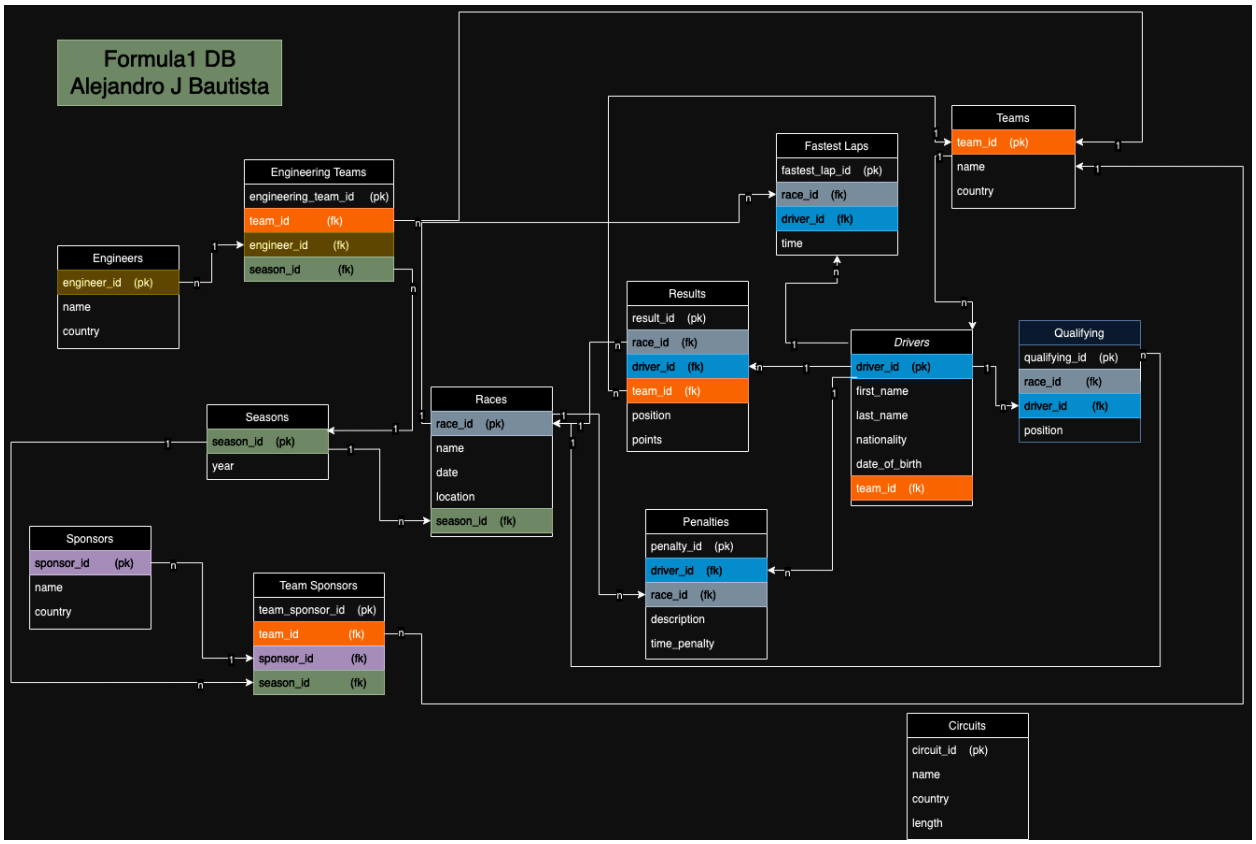
4. Results and Statistics:
- Race results are recorded, including final positions of drivers, points earned, fastest laps, and any penalties received. This enables detailed tracking of driver and team performance throughout the season.

Technology Used

- SQL (Structured Query Language): Language used to manage and manipulate the database, allowing for creation, updating, querying, and deletion of data.
- MySQL: Relational database management system (RDBMS) chosen to implement this project due to its scalability and widespread adoption in the industry.

- In summary, this relational database for Formula 1, implemented using MySQL, offers a solution for comprehensive management of information for one of the most complex and exciting sports competitions in the world. 🏁

E-R Diagram



Tables and Relationships

1️⃣ Drivers:

- driver_id (PK)
- first_name
- last_name
- nationality
- date_of_birth
- team_id (FK)

Relationships:

- (1-n) with Teams
- (1-n) with Results
- (1-n) with Fastest_Laps
- (1-n) with Qualifying
- (1-n) with Penalties

2️⃣ Teams:

- team_id (PK)
- name
- country

Relationships:

- (1-n) with Drivers
- (1-n) with Results
- (m-n) with Engineers through Engineering Teams
- (m-n) with Sponsors through Team_Sponsors

3️⃣ Seasons:

- season_id (PK)
- year

Relationships:

- (1-n) with Races
- (1-n) with Team_Constructors
- (1-n) with Team_Sponsors

4️⃣ Circuits (Additional, not directly related in the simplified design):

- circuit_id (PK)
- name
- country
- length

5️⃣ Races:

- race_id (PK)
- name
- date
- location
- season_id (FK)

Relationships:

- (1-n) with Results
- (1-n) with Fastest_Laps
- (1-n) with Qualifying
- (1-n) with Penalties

6️⃣ Results:

- result_id (PK)
- race_id (FK)
- driver_id (FK)
- team_id (FK)
- position
- points

Relationships:

- (n-1) with Races
- (n-1) with Drivers
- (n-1) with Teams

7️⃣ Fastest_Laps:

- fastest_lap_id (PK)
- race_id (FK)
- driver_id (FK)
- time

Relationships:

- (n-1) with Races
- (n-1) with Drivers

8️⃣ Qualifying:

- qualifying_id (PK)
- race_id (FK)
- driver_id (FK)
- position

Relationships:

- (n-1) with Races
- (n-1) with Drivers

9️⃣ Penalties:

- penalty_id (PK)
- driver_id (FK)
- race_id (FK)
- description
- time_penalty

Relationships:

- (n-1) with Drivers
- (n-1) with Races

🔟 Engineers

- engineer_id (PK)
- name
- country

Relationships:

- (1-n) with Engineering Team

1️⃣1️⃣ Egineering_Team:

- engineering_team_id (PK)
- team_id (FK)
- engineer_id (FK)
- season_id (FK)

Relationships:

- (n-1) with Engineers
- (n-1) with Teams
- (n-1) with Seasons

1️⃣2️⃣ Sponsors:

- sponsor_id (PK)
- name
- country

Relationships:

- (1-n) with Team_Sponsors

1️⃣3️⃣ Team_Sponsors:

- team_sponsor_id (PK)
- team_id (FK)
- sponsor_id (FK)
- season_id (FK)

Relationships:

- (n-1) with Sponsors
- (n-1) with Teams
- (n-1) with Seasons

Github Link: https://github.com/Alexjav129/Formula1DB-Bautista

Views (4)

VIEW 1: Team and Driver Info

- Objective: Provides detailed information about teams and their associated drivers.
- Tables: Teams (teams) and Drivers (drivers).
- Description: This view combines data from the teams table and the drivers table to show the names, nationalities, and birthdates of each driver along with their respective team's name and country.

VIEW 2: Race Results with Driver and Team Information

- Objective: Offers comprehensive race result details including driver and team information.
- Tables: Races (races), Results (results), Drivers (drivers), Teams (teams), Seasons (seasons).
- Description: This view integrates data from multiple tables to display race names, dates, locations, along with driver names, team names, and specific race results such as positions and points scored.

VIEW 3: Teams and Their Engineers

- Objective: Presents teams alongside their engineering staff for a given season.
- Tables: Teams (teams), Engineering Team (engineering_team), Engineers (engineers), Seasons (seasons).
- Description: This view combines information from the teams, engineering_team, engineers, and seasons tables to show each team's name and country alongside the names and countries of their engineers for a specified season.

VIEW 4: Teams and Their Sponsors

- Objective: Displays teams and their sponsor relationships for a particular season.
- Tables: Teams (teams), Team Sponsors (team_sponsors), Sponsors (sponsors), Seasons (seasons).
- Description: This view utilizes data from the teams, team_sponsors, sponsors, and seasons tables to showcase each team's name and country alongside the names and countries of their sponsors for a given season.

These views provide consolidated information based on the relationships between teams, drivers, engineers, sponsors, races, and results within the database structure.

Functions (2)

Function 1: Calculate Points Per Race

- **Objective:** This function calculates the points awarded based on the finishing position in a race, adhering to Formula 1 championship points rules.
- **Manipulated Data/Tables:** No specific tables are manipulated; it takes an integer parameter (position) and computes points based on conditional logic.
- **Description:** The function calculate_points_per_race determines the points awarded to drivers based on their finishing position in a race. It uses a series of conditional statements (IF and ELSE IF) to assign points according to Formula 1 scoring rules. Positions from 1st to 10th are assigned specific points (25, 18, 15, 12, 10, 8, 6, 4, 2, and 1 respectively), while positions below 10th receive no points.

Function 2: Determine Champion Status

- **Objective:** This function determines whether a given championship count qualifies an individual as a champion or not.
- **Manipulated Data/Tables:** No specific tables are manipulated; it takes an integer parameter (championship) and returns a status based on its value.
- **Description:** The function determine_champion_status evaluates whether the input championship count qualifies an individual as a champion. It returns a string indicating "Champion" if the input is greater than zero, otherwise "Not a Champion". This function operates in a NO SQL context, meaning it doesn't interact with database tables but rather performs a simple conditional check and returns a result based on the input parameter.

These functions are designed to encapsulate specific logic for calculating race points and determining championship status based on provided inputs, enhancing the database's capability to handle Formula 1-related calculations and status determinations.

Stored Procedures (2)

Stored Procedure 1: Get Number of Races per Season

- **Objective/Benefit:** This stored procedure is designed to provide a summary of the number of races held each season. It aggregates the count of races per year, offering a quick overview of the racing activity across different seasons.
- **Interacting Tables:**
    - seasons
    - races
- **Description:** The stored procedure sp_get_races_per_season retrieves the total number of races for each season. It performs a LEFT JOIN between the seasons and races tables on the season_id column. The procedure then groups the results by the year column from the seasons table and counts the number of races (race_id) for each year. This information is useful for understanding the distribution of races over different seasons and can help in analyzing the growth or changes in the racing calendar.

```SQL
CALL sp_get_races_per_season();
```

Stored Procedure 2: Get Race Results for a Driver

- **Objective/Benefit:** This stored procedure provides detailed race results for a specific driver, allowing for the analysis of a driver's performance over time. It fetches and organizes the race results in a clear and concise manner.
- **Interacting Tables:**
    - results
    - races
    - drivers
- **Description:** The stored procedure sp_get_driver_results retrieves all race results for a specified driver. It accepts a driver ID as an input parameter and joins the results, races, and drivers tables to collect relevant data. The procedure selects the race name, race date, the driver's finishing position, and the points earned in each race. The results are ordered by the race date in descending order, providing a chronological overview of the driver's performance. This procedure is beneficial for analyzing a driver's historical performance and can be used for reporting and statistical analysis.

```SQL
CALL sp_get_driver_results(1);
CALL sp_get_driver_results(10);
CALL sp_get_driver_results(20);
```

These stored procedures enhance the functionality of the Formula 1 database by providing essential summary and detailed data retrieval capabilities, which are crucial for performance analysis and reporting.

Triggers (2)

Trigger 1: Audit Races Trigger

- **Objective/Benefit:** This trigger ensures that any insertion into the `races` table is logged in the `racesTriggerAudit` table. This is crucial for maintaining an audit trail of changes, providing a historical record of all additions to the `races` table.
- **Interacting Tables:**
  - `races`
  - `racesTriggerAudit`
- **Description:** The `afterInsertRacesTrigger` is an `AFTER INSERT` trigger that activates whenever a new record is inserted into the `races` table. When this occurs, the trigger copies the details of the new race (including `race_id`, `name`, `date`, `location`, and `season_id`) into the `racesTriggerAudit` table. Additionally, it logs the type of DML operation ('INSERT'), the current date and time (`ChangeDate`), and the user who made the change (`UserChange`). This ensures that all new race entries are recorded for audit purposes.

```SQL
-- Insert data into Races to trigger the audit
INSERT INTO races (name, date, location, season_id) VALUES
('British Grand Prix2023', '2023-10-16', 'Silverstone', 4),
('Monaco Grand Prix2023', '2023-11-28', 'Monaco', 4),
('Italian Grand Prix2023', '2023-12-03', 'Monza', 4);
```

Trigger 2: Audit Results Trigger

- **Objective/Benefit:** This trigger ensures that any insertion into the `results` table is logged in the `resultsTriggerAudit` table. This is essential for tracking all changes made to the race results, thereby maintaining a comprehensive audit trail.
- **Interacting Tables:**
  - `results`
  - `resultsTriggerAudit`

- **Description:** The `afterInsertResultsTrigger` is an `AFTER INSERT` trigger that activates whenever a new record is inserted into the `results` table. When triggered, it copies the details of the new result (including `result_id`, `race_id`, `driver_id`, `team_id`, `position`, and `points`) into the `resultsTriggerAudit` table. Additionally, it logs the type of DML operation ('INSERT'), the current date and time (`ChangeDate`), and the user who made the change (`UserChange`). This trigger ensures that all new results entries are recorded for audit purposes, providing a detailed log of all race results additions.

```SQL
Insert data into Results to trigger the audit
INSERT INTO results (race_id, driver_id, team_id, position,
points) VALUES
(1, 1, 1, 1, 25),
(1, 2, 1, 2, 18),
(2, 3, 2, 1, 25),
(2, 4, 2, 2, 18),
(3, 5, 3, 1, 25);
```

These triggers help maintain the integrity and accountability of the data within the Formula 1 database by providing automatic logging of significant table changes.