Automatizações do DATABASE da Fórmula 1

O banco original possuia duas tabelas chamadas constructor_standings e constructor_results que geravam qual equipe tinha ganhado mais pontos e consequentemente teria vencido, então criamos a view **pontos_construtores_db** para simplificar o processo e dizer quais são as equipes com as melhores pontuações.

```
CREATE VIEW pontos_construtores_db AS
SELECT
    c.name AS constructor_name,
    SUM(r.points) AS ranking_constructors
FROM
    constructor_db c
JOIN
    drivers_db d ON d.constructorId = c.constructorId
JOIN
    results_db r ON r.driverId = d.driverId
GROUP BY
    c.name;
```

A view **nacionalidades_vencedoras** foi criada para ranquear as 5 nacionalidades dos pilotos com mais pontos, facilitando a busca pelas nacionalidades que mais se destacam na Fórmula 1.

```
CREATE VIEW dominancia_equipes_circuitos_db AS
select cons.constructorid, cons.name AS equipe, cir.circuitid, cir.name AS circuito, cir.country AS país,
COUNT (CASE WHEN r.position = 1 THEN 1 END) AS vitorías, COUNT (CASE WHEN r.position <= 3 THEN 1 END)
AS podios, SUM(r.points) AS total_pontos, MIN (r.position) AS melhor_posicao_no_circuito,
MAX (CASE WHEN r.position = 1 THEN s.year END) AS ultimo_ano_vitoria
FROM results_db r
JOIN drivers_db d ON d.driverid = r.driverid
JOIN constructor_db cons ON cons.constructorid = d.constructorid
JOIN races_db ra ON ra.raceid = r.raceid
JOIN circuits_db cir ON cir.circuitid = ra.circuitid
JOIN seasons_db s ON s.seasonid = ra.seasonid
GROUP BY cons.constructorid, cons.name, cir.circuitid, cir.name, cir.country;
```

A view **dominância_equipes_circuitos_db** facilita a visualização das equipes que dominam quais circuitos, com base em vitórias, pódios, pontos e histórico de resultados.

```sql
CREATE VIEW dominancia_equipes_circuitos_db AS
SELECT
    cons.constructorid,
    cons.name AS equipe,
    cir.circuitid,
    cir.name AS circuito,
    cir.country AS pais,

    COUNT(CASE WHEN r.position = 1 THEN 1 END) AS vitorias,
    COUNT(CASE WHEN r.position <= 3 THEN 1 END) AS podios,
    SUM(r.points) AS total_pontos,
    MIN(r.position) AS melhor_posicao_no_circuito,
    MAX(CASE WHEN r.position = 1 THEN s.year END) AS ultimo_ano_vitoria

FROM results_db r
JOIN drivers_db d ON d.driverid = r.driverid
JOIN constructor_db cons ON cons.constructorid = d.constructorid
JOIN races_db ra ON ra.raceid = r.raceid
JOIN circuits_db cir ON cir.circuitid = ra.circuitid
JOIN seasons_db s ON s.seasonid = ra.seasonid

GROUP BY
    cons.constructorid,
    cons.name,
    cir.circuitid,
    cir.name,
    cir.country;
```

O banco original possuía uma tabela chamada drivers_standings que mostrava o desempenho final do piloto em uma corrida, para facilitar o processo criamos a view **pontos_pilotos** para mostrar os pontos finais do piloto em uma corrida.

```sql
CREATE OR REPLACE VIEW pontos_pilotos AS
SELECT
    d.name AS driver_name,
    SUM(r.points) AS ranking_drivers,
    d.nationality AS nationality_driver
FROM
    drivers_db d
JOIN
    results_db r ON r.driverId = d.driverId
GROUP BY
    d.name, d.nationality;
```

A função **vencedor_corrida** foi criada para facilitar a pesquisa do vencedor de uma corrida ao inserir uma pista específica de uma determinada temporada pelo seu ano no calendário.

```
CREATE OR REPLACE FUNCTION vencedor_corrida (ano INTEGER, pista VARCHAR)
RETURNS VARCHAR AS $$
DECLARE
    nome_piloto_vencedor VARCHAR;
BEGIN
    SELECT d.name
    INTO nome_piloto_vencedor
    FROM results_db r
    JOIN races_db ra ON r.raceid = ra.raceid
    JOIN seasons_db s ON ra.seasonid = s.seasonid
    JOIN circuits_db c ON ra.circuitid = c.circuitid
    JOIN drivers_db d ON r.driverid = d.driverid
    WHERE s.year = ano
      AND c.name = pista
      AND r.position = 1
    LIMIT 1;

    RETURN nome_piloto_vencedor;
END;
$$ LANGUAGE plpgsql;
```

A função **total_pilotos** foi criada para facilitar a busca de quantos pilotos diferentes já correram por uma determinada equipe.

```
CREATE FUNCTION total_pilotos (equipe VARCHAR)
RETURNS INTEGER AS $$
DECLARE
qnt INTEGER;
BEGIN
SELECT COUNT (DISTINCT r.driverid)
INTO qnt
FROM results_db r
JOIN constructor_db c ON r.constructorid = c.constructorid
WHERE c.name = equipe;
RETURN qnt;
END;
$$ LANGUAGE plpgsql;
DROP FUNCTION IF EXISTS total_pilotos(varchar) CASCADE;
```

A função **total_vitorias_piloto** serve para descobrir quantas vitórias um piloto tem no banco de dados e retornar essa informação em formato de texto.

```sql
CREATE OR REPLACE FUNCTION total_vitorias_piloto (nome_piloto_param VARCHAR)
RETURNS VARCHAR AS $$
DECLARE
    qtd_vitorias INT;
BEGIN
    SELECT COUNT(*)
    INTO qtd_vitorias
    FROM drivers_db d
    JOIN results_db r ON r.driverid = d.driverid
    WHERE d.name = nome_piloto_param -- ERRO 1: Faltava o sinal de igual (=)
      AND r.position = 1;

    IF qtd_vitorias = 0 THEN
        RETURN 'Esse piloto não possui vitórias.';
    ELSE
        RETURN 'Esse piloto possui ' || qtd_vitorias || ' vitória(s).';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

A procedure **estatisticas_piloto_temporada** facilita a busca de estatísticas completas de um piloto em uma temporada.

```
CREATE OR REPLACE PROCEDURE estatisticas_piloto_temporada (
    IN ano_param INTEGER,
    IN nome_piloto_param VARCHAR
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_driverid INT;
    total_corridas INT DEFAULT 0;
    vitorias INT DEFAULT 0;
    podios INT DEFAULT 0;
    pontos DECIMAL DEFAULT 0.0;
    melhor_posicao INT DEFAULT 0;
    media_posicao DECIMAL DEFAULT 0.0;
    posicao_final INT DEFAULT 0;
BEGIN
    SELECT driverid
    INTO v_driverid
    FROM drivers_db
    WHERE name = nome_piloto_param;

    IF v_driverid IS NULL THEN
        RAISE NOTICE 'Piloto não encontrado.';
        RETURN;
    END IF;

    SELECT COUNT(*)
    INTO total_corridas
    FROM results_db res
    JOIN races_db r ON r.raceid = res.raceid -- Corrigido typo: raceId -> raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND res.driverid = v_driverid;

    SELECT COUNT(*)
    INTO vitorias
    FROM results_db res
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND res.driverid = v_driverid AND res.position = 1;
```

```
    SELECT COUNT(*)
    INTO podios
    FROM results_db res
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND res.driverid = v_driverid AND res.position <= 3;

    SELECT COALESCE(SUM(res.points), 0)
    INTO pontos
    FROM results_db res
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND res.driverid = v_driverid;

    SELECT MIN(res.position)
    INTO melhor_posicao
    FROM results_db res
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND res.driverid = v_driverid;

    SELECT COALESCE(AVG(res.position), 0)
    INTO media_posicao
    FROM results_db res
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND res.driverid = v_driverid;

    RAISE NOTICE '=== Estatísticas de % na temporada % ===', nome_piloto_param, ano_param;
    RAISE NOTICE 'Corridas disputadas: %', total_corridas;
    RAISE NOTICE 'Vitórias: %', vitorias;
    RAISE NOTICE 'Pódios: %', podios;
    RAISE NOTICE 'Pontos totais: %', pontos;
    RAISE NOTICE 'Melhor posição: %', melhor_posicao;
    RAISE NOTICE 'Média de posição: %', media_posicao;

END;
$$;
```

A procedure **fastest_laps_do_piloto** facilita a consulta rápida do histórico de voltas mais rápidas de um piloto.

```sql
CREATE OR REPLACE PROCEDURE fastest_laps_do_piloto (IN nome_piloto_param VARCHAR)
LANGUAGE plpgsql
AS $$
DECLARE
    r RECORD;
BEGIN
    FOR r IN
        SELECT s.year,
               ra.name AS corrida,
               c.name AS circuito,
               res.rank AS pos_fastest_lap
        FROM results_db res
        JOIN drivers_db d ON d.driverid = res.driverid
        JOIN races_db ra ON ra.raceid = res.raceid
        JOIN seasons_db s ON s.seasonid = ra.seasonid
        JOIN circuits_db c ON c.circuitid = ra.circuitid
        WHERE d.name = nome_piloto_param
          AND res.rank = 1
        ORDER BY s.year
    LOOP
        RAISE NOTICE 'Ano: %, Corrida: %, Circuito: %, Rank Volta: %',
                     r.year, r.corrida, r.circuito, r.pos_fastest_lap;
    END LOOP;

    IF NOT FOUND THEN
        RAISE NOTICE 'Nenhum registro de volta mais rápida para o piloto %.', nome_piloto_param;
    END IF;
END;
$$;
```

A procedure **estatisticas_equipe_temporada** facilita a busca de estatísticas completas de uma equipe em uma temporada.

```
CREATE OR REPLACE PROCEDURE estatisticas_equipe_temporada (
    IN ano_param INTEGER,
    IN nome_equipe_param VARCHAR
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_equipe_id INT;
    total_pontos DECIMAL DEFAULT 0;
    vitorias INT DEFAULT 0;
    podios INT DEFAULT 0;
    total_corridas INT DEFAULT 0;
    melhor_posicao INT DEFAULT 0;
    media_posicao DECIMAL DEFAULT 0;
    posicao_final INT DEFAULT 0;
    pilotos TEXT;
BEGIN
    SELECT constructorid INTO v_equipe_id
    FROM constructor_db
    WHERE name = nome_equipe_param;

    IF v_equipe_id IS NULL THEN
        RAISE NOTICE 'Equipe não encontrada.';
        RETURN;
    END IF;

    SELECT COALESCE(SUM(res.points), 0)
    INTO total_pontos
    FROM results_db res
    JOIN drivers_db d ON res.driverid = d.driverid -- Link via Driver
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND d.constructorid = v_equipe_id;

    SELECT COUNT(*)
    INTO vitorias
    FROM results_db res
    JOIN drivers_db d ON res.driverid = d.driverid
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND d.constructorid = v_equipe_id AND res.position = 1;
```

```
    SELECT COUNT(*)
    INTO total_corridas
    FROM results_db res
    JOIN drivers_db d ON res.driverid = d.driverid
    JOIN races_db r ON r.raceid = res.raceid -- Corrigido typo raceI
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND d.constructorid = v_equipe_id;

    SELECT MIN(res.position)
    INTO melhor_posicao
    FROM results_db res
    JOIN drivers_db d ON res.driverid = d.driverid
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND d.constructorid = v_equipe_id;

    SELECT COALESCE(AVG(res.position), 0)
    INTO media_posicao
    FROM results_db res
    JOIN drivers_db d ON res.driverid = d.driverid
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND d.constructorid = v_equipe_id;

    INTO pilotos
    FROM results_db res
    JOIN drivers_db d ON d.driverid = res.driverid
    JOIN races_db r ON r.raceid = res.raceid
    JOIN seasons_db s ON r.seasonid = s.seasonid
    WHERE s.year = ano_param AND d.constructorid = v_equipe_id;

    RAISE NOTICE '===== Estatísticas da equipe % na temporada % =====', nome_equipe_param, ano_param;
    RAISE NOTICE 'Pontos totais: %', total_pontos;
    RAISE NOTICE 'Vitórias: %', vitorias;
    RAISE NOTICE 'Pódios: %', podios;
    RAISE NOTICE 'Total de corridas disputadas: %', total_corridas;
    RAISE NOTICE 'Melhor posição obtida: %', melhor_posicao;
    RAISE NOTICE 'Média de posição: %', media_posicao;
    RAISE NOTICE 'Pilotos na temporada: %', pilotos;

END;
$$;
```

A trigger **valida_coordenada** foi criada com o objetivo de validar as coordenadas inseridas na tabela circuits_db. Ela é importante pois impede a inserção de circuitos com latitudes ou longitudes impossíveis, e evita que dados malformados corrompam a integridade da aplicação que consumiria esses dados.

```sql
CREATE OR REPLACE FUNCTION valida_coordenada()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.lat IS NOT NULL AND (NEW.lat < -90 OR NEW.lat > 90) THEN
        RAISE EXCEPTION 'Input Inválido: Latitude % está fora do range permitido (-90 a 90).', NEW.lat;
    END IF;

    IF NEW.lng IS NOT NULL AND (NEW.lng < -180 OR NEW.lng > 180) THEN
        RAISE EXCEPTION 'Input Inválido: Longitude % está fora do range permitido (-180 a 180).', NEW.lng;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trg_valida_coordenada ON public.circuits_db;

CREATE TRIGGER trg_valida_coordenada
BEFORE INSERT OR UPDATE ON public.circuits_db
FOR EACH ROW
EXECUTE FUNCTION valida_coordenada();
```

A trigger **impede_resultados_futuros** foi criada com o objetivo de manter a integridade da tabela results_db, evitando erros de lógica e a impedindo a adulteração dos resultados. Ela evita que ocorra a inserção de dados futuros na tabela results_db, pois não faz lógica colocar resultados de uma corrida em que ainda não aconteceu.

```sql
CREATE OR REPLACE FUNCTION impede_resultados_futuros()
RETURNS TRIGGER AS $$
DECLARE
    race_data DATE;
BEGIN
    SELECT "date" INTO race_data
    FROM public.races_db
    WHERE raceid = NEW.raceid;

    IF race_data > CURRENT_DATE THEN
        RAISE EXCEPTION 'Lógica Temporal: Não é possível inserir resultados para uma corrida futura (Data: %).', race_data;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_impede_resultados_futuros
BEFORE INSERT OR UPDATE ON public.results_db
FOR EACH ROW
EXECUTE FUNCTION impede_resultados_futuros();
```

A trigger **atualiza_fastestlaptime** faz a atualização automática da volta mais rápida de um corredor. Ao inserir ou alterar uma volta na tabela laptimes_db, verifica se é um tempo recorde e atualiza no results_db o fastestlaptime. Foi feito com o objetivo de automatizar e melhorar a coerência das informações nas tabelas.

```sql
CREATE OR REPLACE FUNCTION atualiza_fastestlaptime()
RETURNS TRIGGER AS $$
DECLARE
    best_time_atual INTERVAL;
BEGIN
    SELECT fastestlaptime INTO best_time_atual
    FROM public.results_db
    WHERE raceid = NEW.raceid AND driverid = NEW.driverid;

    IF best_time_atual IS NULL OR NEW.milliseconds < best_time_atual THEN

        UPDATE public.results_db
        SET fastestlaptime = NEW.milliseconds
        WHERE raceid = NEW.raceid AND driverid = NEW.driverid;

    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_atualiza_fastestlaptime
AFTER INSERT OR UPDATE ON public.laptimes_db
FOR EACH ROW
EXECUTE FUNCTION atualiza_fastestlaptime();
```