

INSTITUTO POLITÉCNICO DE VIANA DO CASTELO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

ENGENHARIA INFORMÁTICA

ADMINISTRAÇÃO DE BASES DE DADOS

2022/2023

Trabalho Prático
Estatísticas de uma equipa de ciclismo

Alunos:

Alexandre Santos - 24585,
Rui Alves - 25343

Docente:

Marco Lima



**Instituto Politécnico
de Viana do Castelo**

3 de junho de 2023

Conteúdo

Lista de Figuras	v
1 Introdução	1
2 Modelação da Base de Dados	3
2.1 Modelo de dados	3
3 Criação e Configuração da Base de Dados	7
3.1 Criação da base de dados	7
3.2 Restrições de integridade	9
3.3 Tabela de logs	10
3.4 Inserção de dados	13
4 Queries SQL	15
4.1 Selects	15
4.2 Views	15
4.3 Triggers	16
4.4 Procedimentos	17
4.5 Stored Procedures	20
4.6 Function	21
4.7 Cursores	22
4.8 Outras funções	23
4.9 Pivot	24
5 Manutenção da Base de Dados	25
5.1 Filestream	25
5.2 Database Engine Tuning Advisor	29
5.3 Database Maintenance	38
5.4 Relatório da Base de Dados	43
6 Manutenção da Base de Dados (Pontos Opcionais)	49
6.1 Procedimento de inserção de dados em formato JSON	49
6.2 Procedimento que guarde dados em formato JSON	49
6.3 Trigger para notificação por email	50
7 Conclusões	55

Lista de Figuras

2.1	DER	4
2.2	Ciclista	4
2.3	Etapas	4
2.4	Tipo de estatística	5
2.5	Estatísticas	5
5.1	Alteração das propriedades do SQL SERVER	27
5.2	Habilitar sinalizadores de rastreamento específicos	27
5.3	Adicionar um novo parametro - T12324	27
5.4	Criação do FileGroup	27
5.5	Adicionar ficheiros FileStream	27
5.6	Criação de um novo FileGroup	28
5.7	Adicionar ficheiros FileStream ao novo FileGroup	28
5.8	Inserir ficheiros binarios na FileStream	28
5.9	Consulta com peso	29
5.10	tool > sql server profiler	29
5.11	Ligação á instância	30
5.12	Propriedades	30
5.13	Seleção de eventos	30
5.14	SQL:StmtCompleted	31
5.15	SQL:StmtCompleted Duration	31
5.16	Database Name	31
5.17	Trace a correr	32
5.18	Trace com consulta	32
5.19	Trace individual detalhado	32
5.20	Exportar o Trace	33
5.21	Sucesso na exportação	33
5.22	TraceID	33
5.23	Stop the Trace	33
5.24	Ver o Trace	34
5.25	Abrir o Database Engine Tuning Advisor	34
5.26	Conectar a instância	35
5.27	Selecionar as Tabelas	35
5.28	Gerar recomendações	35
5.29	Análise da BD	36
5.30	Resultados da análise	36
5.31	Report gerado	36
5.32	Guardar recomendações	36
5.33	Localização	37
5.34	Sucesso	37
5.35	Estimativa do plano de execução	37
5.36	Subtree Cost alto	37
5.37	Execução do plano	38

5.38	Otimização do Subtree Cost	38
5.39	Fragmentação das estatísticas	39
5.40	Abrir o Maintenance Plan Wizard	39
5.41	Maintenance Plan Wizard	39
5.42	New Job Schedule	40
5.43	Tarefas de manutenção	40
5.44	Tabela para verificação de integridade	40
5.45	Ordem das tarefas	41
5.46	Reorganização de índices	41
5.47	Atualização de estatísticas	41
5.48	Definir o backup	42
5.49	Opções do relatório e email	42
5.50	Resumo com sucesso de execução	42
5.51	Executar	42
5.52	Execução do plano	43
5.53	Resultado da execução	43
5.54	Finalização	44
5.55	Email enviado para o administrador	44
5.56	Criação de um relatório novo	45
5.57	Criar novo conjunto de dados	45
5.58	Janela de propriedades da fonte de dados	45
5.59	Compilação da instância e base de dados	46
5.60	Seleção das tabelas para o gráfico	46
5.61	Seleção entre colunas, linhas, valores e tipo de função	46
5.62	Definir layout do relatório	47
5.63	Resumo do relatório	47
5.64	Tentativa de criação de uma tabela com erro	47
5.65	Tentativa de criação de um gráfico com erro	48
6.1	Conversão JSON para SQL	50
6.2	Conversão SQL para JSON	50
6.3	Abrir Database Mail Configuration	50
6.4	Database Mail Configuration	51
6.5	Configuração de tarefas	51
6.6	Perfis	51
6.7	Criar perfil	51
6.8	Segurança do perfil	52
6.9	Parâmetros do sistema	52
6.10	Finalizar	52
6.11	Configuração do email completa	52
6.12	Envio de e-mail de teste	53
6.13	E-mail recebido	53
6.14	Propriedades do agente	53
6.15	Configuração do perfil	53
6.16	Email enviado através do trigger	53

Capítulo 1

Introdução

Com este trabalho, vamos desenvolver uma base de dados para gestão de estatísticas de uma equipa de ciclismo ao longo de um determinado número de anos. Começamos pela criação da base de dados, à criação de queries de consulta e alteração da BD como **triggers**, **procedimentos**, **Cursors**, até à implementação de ferramentas para monitorização, automatização, otimização e representação por gráficos dos dados.

Capítulo 2

Modelação da Base de Dados

2.1 Modelo de dados

Para o modelo de dados na nossa base de dados, utilizámos quatro tabelas:

- **Ciclista**, onde guardamos a informação do ciclista, e algumas das suas estatísticas;
- **tipoEst**, onde guardamos os tipos de estatísticas que serão armazenadas;
- **etapa**, onde guardamos a informação acerca das etaps, como aano, local de partida e chegada;
- **estatística**, onde é armazenado o valor de cada estatística por ciclista, numa determinada etapa de cada ano;

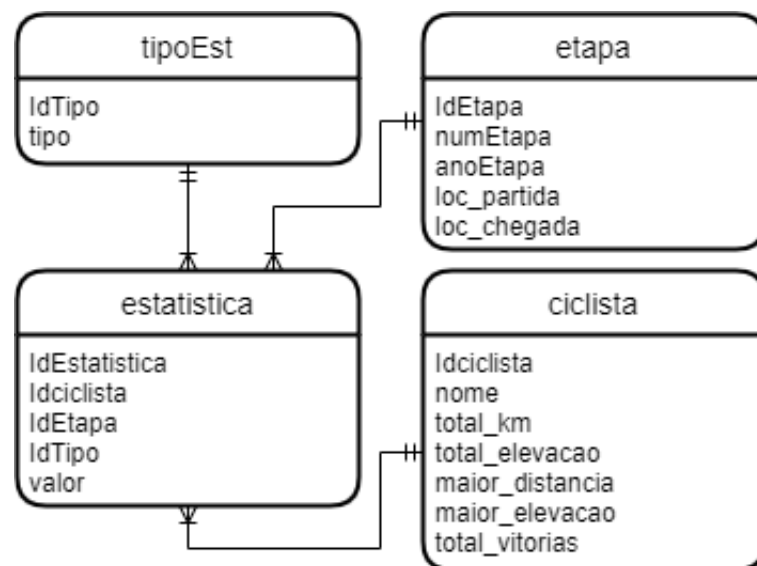


FIGURA 2.1: DER

	Idciclista	nome	total_km	total_elevacao	maior_distancia	maior_elevacao	total_vitorias
1	1	Artem	3492464.0	3492464.0	4999.0	4999.0	5700
2	2	Duarte Domingues	3579886.0	3579886.0	4997.0	4997.0	6080
3	3	Fábio Costa	3478274.0	3478274.0	5000.0	5000.0	5700
4	4	Frederico Figueiredo	3503072.0	3503072.0	4994.0	4994.0	5700
5	5	Julian Madrigal	3509402.0	3509402.0	4999.0	4999.0	5700
6	6	Luis Mendonça	3499094.0	3499094.0	5000.0	5000.0	5700
7	7	Mauricio Moreira	3540414.0	3540414.0	4995.0	4995.0	5700
8	8	Pedro Silva	3466839.0	3466839.0	4995.0	4995.0	5700
9	9	Rafael Reis	3499478.0	3499478.0	5000.0	5000.0	5700
10	10	Sergio Garcia	3555919.0	3555919.0	4998.0	4998.0	5700

FIGURA 2.2: Ciclista

	IdEtapa	numEtapa	anoEtapa	loc_partida	loc_chegada
1	1	0	2005	BILBAO	BILBAO
2	2	1	2005	VITORIA-GASTEIZ	SAN SEBASTIÁN
3	3	2	2005	AMOREBIETA-ETXANO	BAYONNE
4	4	3	2005	DAX	BAYONNE
5	5	4	2005	PAU	NOGARO
6	6	5	2005	TARBES	LARUNS
7	7	6	2005	MONT-DE-MARSIN	CAUTARETS-CAMBASQUE
8	8	7	2005	LIBOURNE	BORDÉUS
9	9	8	2005	SAINT-LÉONARD-DE-NOBLAT	LIMOGES
10	10	9	2005	VULCANIA	PUY DE DÔME
11	11	10	2005	CLERMONT-FERRAND	ISSOIRE
12	12	11	2005	ROANNE	MOULINS
13	13	12	2005	CHÂTILLON-SUR-CHALARONNE	BELLEVILLE-EN-BEAUJOLAIS
14	14	13	2005	ANNEMASSE	GRAND COLOMBIER

FIGURA 2.3: Etapa

	IdTipo	tipo
1	1	Posicao
2	2	Km prova
3	3	Acumulado positivo
4	4	Acumulado negativo
5	5	Tempo prova
6	6	Velocidade maxima
7	7	Velocidade media
8	8	Calorias
9	9	Ritmo cardiaco maximo
10	10	Ritmo cardiaco medio
11	11	Sprints ganhos
12	12	Montanhas ganhas
13	13	Quedas durante a prova
14	14	Troca de bicicleta
15	15	Troca de rodas

FIGURA 2.4: Tipo de estatística

	IdEstatistica	Idciclista	IdEtapa	IdTipo	valor
1	1	1	1	1	28.0
2	2	1	2	1	46.0
3	3	1	3	1	174.0
4	4	1	4	1	137.0
5	5	1	5	1	84.0
6	6	1	6	1	14.0
7	7	1	7	1	66.0
8	8	1	8	1	94.0
9	9	1	9	1	91.0
10	10	1	10	1	69.0
11	11	1	11	1	65.0
12	12	1	12	1	41.0
13	13	1	13	1	118.0
14	14	1	14	1	85.0
15	15	1	15	1	53.0
16	16	1	16	1	31.0

FIGURA 2.5: Estatísticas

Capítulo 3

Criação e Configuração da Base de Dados

3.1 Criação da base de dados

```
CREATE DATABASE estatisticasVoltaPT
ON
PRIMARY ( NAME = estatisticasVoltaPT,
          FILENAME = 'C:\abdtrabalho\estatisticasVoltaPT.mdf'),
          FILEGROUP estatisticasVoltaPTFS CONTAINS FILESTREAM(
            NAME = estatisticasVoltaPTFS,
            FILENAME = 'C:\abdtrabalho\estatisticasVoltaPTFS')
LOG ON (
          NAME = estatisticasVoltaPTLOG,
          FILENAME = 'C:\abdtrabalho\estatisticasVoltaPTLOG.ldf')
GO
```

Tipos de estatística:

```
CREATE TABLE tipoEst (
    IdTipo int PRIMARY KEY IDENTITY(1,1),
    tipo varchar(25)
);
```

Etapas:

```
CREATE TABLE etapa (
    IdEtapa int PRIMARY KEY IDENTITY(1,1),
    numEtapa int,
    anoEtapa decimal(4),
    loc_partida varchar(25),
    loc_chegada varchar(25)
);
```

Ciclista:

```
CREATE TABLE ciclista (
    Idciclista int PRIMARY KEY IDENTITY(1,1),
    nome varchar(25),
    total_km decimal(15,1),
    total_elevacao decimal(15,1),
    maior_distancia decimal(10,1),
    maior_elevacao decimal(10,1),
    total_vitorias int
);
```

Estatística:

```
CREATE TABLE estatistica (  
    IdEstatistica int PRIMARY KEY IDENTITY(1,1),  
    Idciclista int ,  
    IdEtapa int ,  
    IdTipo int ,  
    valor decimal(15,1),  
    CONSTRAINT FK_Idciclista FOREIGN KEY (Idciclista)  
        REFERENCES ciclista(Idciclista),  
    CONSTRAINT FK_IdEtapa FOREIGN KEY (IdEtapa)  
        REFERENCES etapa(IdEtapa),  
    CONSTRAINT FK_IdTipo FOREIGN KEY (IdTipo)  
        REFERENCES tipoEst(IdTipo)  
);
```

3.2 Restrições de integridade

Os valores devem estar dentro dos intervalos corretos:

```
ALTER TABLE estatistica
```

```
ADD CONSTRAINT checkValor CHECK (valor >= 0);
```

```
ALTER TABLE estatistica
```

```
ADD CONSTRAINT checkIdTipo1 CHECK (IdTipo > 0);
```

```
ALTER TABLE estatistica
```

```
ADD CONSTRAINT checkIdTipo2 CHECK (IdTipo < 16);
```

3.3 Tabela de logs

Criação da tabela de logs:

```
CREATE TABLE LogInteracoes (  
    numLog INT IDENTITY(1,1) PRIMARY KEY,  
    Tabela VARCHAR(100),  
    Operacao VARCHAR(20),  
    DataHora DATETIME,  
    Utilizador VARCHAR(50),  
    idRegisto INT  
);
```

Triggers para atualização da tabela de logs:

Ciclista:

```
CREATE TRIGGER tr_ciclista ON ciclista  
AFTER INSERT, UPDATE, DELETE  
AS  
BEGIN  
    DECLARE @operacao VARCHAR(20)  
    IF EXISTS (SELECT * FROM inserted)  
    BEGIN  
        IF EXISTS (SELECT * FROM deleted)  
            SET @operacao = 'UPDATE'  
        ELSE  
            SET @operacao = 'INSERT'  
    END  
    ELSE  
        SET @operacao = 'DELETE'  
  
    INSERT INTO LogInteracoes (Tabela, Operacao, DataHora, Utilizador, idRegisto)  
    SELECT 'ciclista', @operacao, GETDATE(), SUSER_SNAME(),  
        Idciclista FROM inserted UNION ALL  
    SELECT 'ciclista', @operacao, GETDATE(), SUSER_SNAME(),  
        Idciclista FROM deleted  
END
```

Tipos de estatística:

```
CREATE TRIGGER tr_tipoEst ON tipoEst  
AFTER INSERT, UPDATE, DELETE  
AS  
BEGIN  
    DECLARE @operacao VARCHAR(20)  
    IF EXISTS (SELECT * FROM inserted)  
    BEGIN  
        IF EXISTS (SELECT * FROM deleted)  
            SET @operacao = 'UPDATE'  
        ELSE  
            SET @operacao = 'INSERT'  
    END  
    ELSE
```



```

        SET @operacao = 'DELETE'

        INSERT INTO LogInteracoes (Tabela, Operacao, DataHora, Utilizador, idRegisto)
        SELECT 'tipoEst', @operacao, GETDATE(), SUSER_SNAME(),
            IdTipo FROM inserted UNION ALL
        SELECT 'tipoEst', @operacao, GETDATE(), SUSER_SNAME(),
            IdTipo FROM deleted
    END

```

Etapas:

```

CREATE TRIGGER tr_etapa ON etapa
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @operacao VARCHAR(20)
    IF EXISTS (SELECT * FROM inserted)
    BEGIN
        IF EXISTS (SELECT * FROM deleted)
            SET @operacao = 'UPDATE'
        ELSE
            SET @operacao = 'INSERT'
    END
    ELSE
        SET @operacao = 'DELETE'

    INSERT INTO LogInteracoes (Tabela, Operacao, DataHora, Utilizador, idRegisto)
    SELECT 'etapa', @operacao, GETDATE(), SUSER_SNAME(),
        IdEtapa FROM inserted UNION ALL
    SELECT 'etapa', @operacao, GETDATE(), SUSER_SNAME(),
        IdEtapa FROM deleted
END

```

Estatística:

```

-- estatistica
CREATE TRIGGER tr_estatistica ON estatistica
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @operacao VARCHAR(20)
    IF EXISTS (SELECT * FROM inserted)
    BEGIN
        IF EXISTS (SELECT * FROM deleted)
            SET @operacao = 'UPDATE'
        ELSE
            SET @operacao = 'INSERT'
    END
    ELSE
        SET @operacao = 'DELETE'

    INSERT INTO LogInteracoes (Tabela, Operacao, DataHora, Utilizador, idRegisto)
    SELECT 'estatistica', @operacao, GETDATE(), SUSER_SNAME(),

```

```
        IdEstatistica FROM inserted UNION ALL
SELECT  'estatistica', @operacao, GETDATE(), SUSER_SNAME(),
        IdEstatistica FROM deleted
END
```

3.4 Inserção de dados

Tipos de estatística:

```
insert into tipoEst (tipo) values ('Posicao', 'Km prova');
insert into tipoEst (tipo) values ('Acumulado positivo');
insert into tipoEst (tipo) values ('Acumulado negativo');
insert into tipoEst (tipo) values ('Tempo prova');
insert into tipoEst (tipo) values ('Velocidade maxima');
insert into tipoEst (tipo) values ('Velocidade media');
insert into tipoEst (tipo) values ('Calorias');
insert into tipoEst (tipo) values ('Ritmo cardiaco maximo');
insert into tipoEst (tipo) values ('Ritmo cardiaco medio');
insert into tipoEst (tipo) values ('Sprints ganhos');
insert into tipoEst (tipo) values ('Montanhas ganhas');
insert into tipoEst (tipo) values ('Quedas durante a prova');
insert into tipoEst (tipo) values ('Troca de bicicleta');
insert into tipoEst (tipo) values ('Troca de rodas');
```

Ciclista:

```
insert into ciclista (nome, total_km, total_elevacao, maior_distancia,
    maior_elevacao, total_vitorias) values
    ('Artem', '0', '0', '0', '0', '0');
insert into ciclista (nome, total_km, total_elevacao, maior_distancia,
    maior_elevacao, total_vitorias) values
    ('Duarte Domingues', '0', '0', '0', '0', '0');
insert into ciclista (nome, total_km, total_elevacao, maior_distancia,
    maior_elevacao, total_vitorias) values
    ('Fábio Costa', '0', '0', '0', '0', '0');
insert into ciclista (nome, total_km, total_elevacao, maior_distancia,
    maior_elevacao, total_vitorias) values
    ('Frederico Figueiredo', '0', '0', '0', '0', '0');
insert into ciclista (nome, total_km, total_elevacao, maior_distancia,
    maior_elevacao, total_vitorias) values
    ('Julian Madrigal', '0', '0', '0', '0', '0');
insert into ciclista (nome, total_km, total_elevacao, maior_distancia,
    maior_elevacao, total_vitorias) values
    ('Luís Mendonça', '0', '0', '0', '0', '0');
insert into ciclista (nome, total_km, total_elevacao, maior_distancia,
    maior_elevacao, total_vitorias) values
    ('Mauricio Moreira', '0', '0', '0', '0', '0');
...
```

Etapas:

```
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
    values (0, 2005, 'BILBAO', 'BILBAO');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
    values (1, 2005, 'VITORIA-GASTEIZ', 'SAN SEBASTIÁN');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
    values (2, 2005, 'AMOREBIETA-ETXANO', 'BAYONNE');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
```

```
values (3, 2005, 'DAX', 'BAYONNE');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
values (4, 2005, 'PAU', 'NOGARRO');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
values (5, 2005, 'TARBES', 'LARUNS');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
values (6, 2005, 'MONT-DE-MARSIN', 'CAUTARETS-CAMBASQUE');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
values (7, 2005, 'LIBOURNE', 'BORDÉUS');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
values (8, 2005, 'SAINT-LÉONARD-DE-NOBLAT', 'LIMOGES');
insert into etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
values (9, 2005, 'VULCANIA', 'PUY DE DÔME');
...
```

Estatística:

```
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 1, 1, 28);
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 2, 1, 46);
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 3, 1, 174);
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 4, 1, 137);
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 5, 1, 84);
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 6, 1, 14);
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 7, 1, 66);
insert into estatistica (idciclista, IdEtapa, IdTipo, Valor)
values (1, 8, 1, 94);
...
```

Capítulo 4

Queries SQL

4.1 Selects

1. Total de KMs por ciclista por ano:

```
SELECT c.nome, e.anoEtapa, SUM(est.valor) AS total_km
FROM ciclista c
JOIN estatistica est ON c.Idciclista = est.Idciclista
JOIN etapa e ON est.IdEtapa = e.IdEtapa
WHERE est.IdTipo = 2
GROUP BY c.nome, e.anoEtapa
ORDER BY c.nome, e.anoEtapa;
```

2. Soma do número de vitórias num determinado ano:

```
SELECT c.Idciclista, c.nome,
SUM(CASE WHEN e.valor = 1 AND e.IdTipo = 1 THEN 1 ELSE 0 END) AS NumVitorias
FROM estatistica e
INNER JOIN ciclista c ON e.Idciclista = c.Idciclista
GROUP BY c.Idciclista, c.nome;
```

3. Média de KMs de cada etapa durante os anos todos:

```
SELECT e.numEtapa, AVG(est.valor) AS media_km
FROM etapa e
INNER JOIN estatistica est ON e.IdEtapa = est.IdEtapa
WHERE est.IdTipo = 2
GROUP BY e.numEtapa;
```

4.2 Views

1. View complexa, dando permissão de leitura exclusivamente a um utilizador “**WebAPP**” a ser criado: 1.1. Criar user “**WebAPP**” com permissão de leitura apenas:

```
CREATE LOGIN WebAPP WITH PASSWORD = '123456';
CREATE USER WebAPP FOR LOGIN WebAPP;
```

1.2. Criar a view:

```
CREATE VIEW PodiosCorredores AS
SELECT Corredor, numEtapa, anoEtapa, loc_partida, loc_chegada, TipoEstatistica, ValorEstatistica
FROM (
    SELECT c.nome AS Corredor, e.numEtapa, e.anoEtapa, e.loc_partida, e.loc_chegada, t.tipo
    AS TipoEstatistica, es.valor AS ValorEstatistica,
```

```

    ROW_NUMBER() OVER (PARTITION BY c.nome, e.numEtapa ORDER BY c.nome, e.numEtapa) AS
    FROM ciclista c
    JOIN estatistica es ON c.Idciclista = es.Idciclista
    JOIN etapa e ON es.IdEtapa = e.IdEtapa
    JOIN tipoEst t ON es.IdTipo = t.IdTipo
) AS Podios
WHERE RowNumber <= 3;

```

1.3. Dar permissão:

```
GRANT SELECT ON PodiosCorredores TO WebAPP;
```

1.4. Chamar a view:

```
SELECT * from PodiosCorredores;
```

4.3 Triggers

1. Um trigger que seja disparado sempre que é criado, alterado ou eliminado um registo. Deve ser capaz de atualizar outra tabela.

```

CREATE TRIGGER soma_valores
ON estatistica
AFTER INSERT
AS
BEGIN
    -- Atualização do total_km para cada ciclista
    UPDATE c
    SET total_km = c.total_km + i.valor
    FROM ciclista c
    INNER JOIN inserted i ON c.IdCiclista = i.IdCiclista
    WHERE i.IdTipo = 2;

    -- Atualização do total_elevacao para cada ciclista
    UPDATE c
    SET total_elevacao = c.total_elevacao + i.valor
    FROM ciclista c
    INNER JOIN inserted i ON c.IdCiclista = i.IdCiclista
    WHERE i.IdTipo = 3;

    -- Atualização da maior_distancia para cada ciclista
    UPDATE c
    SET maior_distancia = CASE
        WHEN i.valor > c.maior_distancia THEN i.valor
        ELSE c.maior_distancia
    END
    FROM ciclista c
    INNER JOIN inserted i ON c.IdCiclista = i.IdCiclista
    WHERE i.IdTipo = 2;

    -- Atualização da maior_elevacao para cada ciclista
    UPDATE c

```

```

SET maior_elevacao = CASE
    WHEN i.valor > c.maior_elevacao THEN i.valor
    ELSE c.maior_elevacao
END
FROM ciclista c
INNER JOIN inserted i ON c.IdCiclista = i.IdCiclista
WHERE i.IdTipo = 3;

-- Atualização do total_vitorias para cada ciclista
UPDATE c
SET total_vitorias = c.total_vitorias + 1
FROM ciclista c
INNER JOIN inserted i ON c.IdCiclista = i.IdCiclista
WHERE i.valor = 1 AND i.IdTipo = 1;
END;

```

2. Um trigger de validação de dados antes da sua inserção.

```

CREATE TRIGGER validacao_insert
ON estatistica
INSTEAD OF INSERT
AS
BEGIN
    -- Verificar se os campos estão vazios
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE ISNULL(Idciclista, '') = '' OR ISNULL(IdEtapa, '')
            = '' OR ISNULL(IdTipo, '') = '' OR ISNULL(valor, '') = ''
    )
    BEGIN
        RAISERROR ('Todos os campos devem ser preenchidos.', 16, 1);
        ROLLBACK TRANSACTION;
        RETURN;
    END;

    -- Inserir os dados válidos na tabela
    INSERT INTO estatistica (Idciclista, IdEtapa, IdTipo, valor)
    SELECT Idciclista, IdEtapa, IdTipo, valor
    FROM inserted;
END;

```

4.4 Procedimentos

1. Classificar como trepador e velocista com "Sim" ou "Não":

```

CREATE PROCEDURE classificarCiclistasPorAno
AS
BEGIN
    IF OBJECT_ID('tempdb..#Classificacoes') IS NOT NULL
        DROP TABLE #Classificacoes;

```

```

CREATE TABLE #Classificacoes (
    IdCiclista INT,
    AnoEtapa DECIMAL(4,0),
    Trepador VARCHAR(3),
    Velocista VARCHAR(3)
);

DECLARE @AnoEtapa DECIMAL(4,0);
SET @AnoEtapa = (SELECT MIN(anoEtapa) FROM etapa);

WHILE @AnoEtapa IS NOT NULL
BEGIN
    INSERT INTO #Classificacoes (IdCiclista, AnoEtapa, Trepador, Velocista)
    SELECT c.IdCiclista,
           @AnoEtapa,
           CASE WHEN montanhas.Cnt > 5 THEN 'Sim' ELSE 'Não' END,
           CASE WHEN sprints.Cnt > 5 THEN 'Sim' ELSE 'Não' END
    FROM ciclista c
    LEFT JOIN (
        SELECT es.IdCiclista, COUNT(*) AS Cnt
        FROM estatistica es
        INNER JOIN etapa e ON es.IdEtapa = e.IdEtapa
        WHERE es.IdTipo = 12 AND e.anoEtapa = @AnoEtapa
        GROUP BY es.IdCiclista
    ) montanhas ON c.IdCiclista = montanhas.IdCiclista
    LEFT JOIN (
        SELECT es.IdCiclista, COUNT(*) AS Cnt
        FROM estatistica es
        INNER JOIN etapa e ON es.IdEtapa = e.IdEtapa
        WHERE es.IdTipo = 13 AND e.anoEtapa = @AnoEtapa
        GROUP BY es.IdCiclista
    ) sprints ON c.IdCiclista = sprints.IdCiclista;

    SET @AnoEtapa = (SELECT MIN(anoEtapa) FROM etapa WHERE anoEtapa > @AnoEtapa);
END;

SELECT c.nome, cl.AnoEtapa, cl.Trepador, cl.Velocista
FROM ciclista c
LEFT JOIN #Classificacoes cl ON c.IdCiclista = cl.IdCiclista
ORDER BY cl.AnoEtapa DESC;

DROP TABLE #Classificacoes;
END;

EXEC classificarCiclistasPorAno;

```

2. Validar a inserção de dados na tabela estatística:

```

CREATE PROCEDURE classificarCiclistasPorAno
AS
BEGIN
    IF OBJECT_ID('tempdb..#Classificacoes') IS NOT NULL

```



```
DROP TABLE #Classificacoes;

CREATE TABLE #Classificacoes (
    IdCiclista INT,
    AnoEtapa DECIMAL(4,0),
    Trepador VARCHAR(3),
    Velocista VARCHAR(3)
);

DECLARE @AnoEtapa DECIMAL(4,0);
SET @AnoEtapa = (SELECT MIN(anoEtapa) FROM etapa);

WHILE @AnoEtapa IS NOT NULL
BEGIN
    INSERT INTO #Classificacoes (IdCiclista, AnoEtapa, Trepador, Velocista)
    SELECT c.IdCiclista,
        @AnoEtapa,
        CASE WHEN montanhas.Cnt > 5 THEN 'Sim' ELSE 'Não' END,
        CASE WHEN sprints.Cnt > 5 THEN 'Sim' ELSE 'Não' END
    FROM ciclista c
    LEFT JOIN (
        SELECT es.IdCiclista, COUNT(*) AS Cnt
        FROM estatistica es
        INNER JOIN etapa e ON es.IdEtapa = e.IdEtapa
        WHERE es.IdTipo = 12 AND e.anoEtapa = @AnoEtapa
        GROUP BY es.IdCiclista
    ) montanhas ON c.IdCiclista = montanhas.IdCiclista
    LEFT JOIN (
        SELECT es.IdCiclista, COUNT(*) AS Cnt
        FROM estatistica es
        INNER JOIN etapa e ON es.IdEtapa = e.IdEtapa
        WHERE es.IdTipo = 13 AND e.anoEtapa = @AnoEtapa
        GROUP BY es.IdCiclista
    ) sprints ON c.IdCiclista = sprints.IdCiclista;

    SET @AnoEtapa = (SELECT MIN(anoEtapa) FROM etapa WHERE anoEtapa > @AnoEtapa);
END;

SELECT c.nome, cl.AnoEtapa, cl.Trepador, cl.Velocista
FROM ciclista c
LEFT JOIN #Classificacoes cl ON c.IdCiclista = cl.IdCiclista
ORDER BY cl.AnoEtapa DESC;

DROP TABLE #Classificacoes;

EXEC classificarCiclistasPorAno;
```

4.5 Stored Procedures

1. Inserir:

```
CREATE PROCEDURE sp_InserirDados
    @nome varchar(25),
    @total_km decimal(15,1),
    @total_elevacao decimal(15,1),
    @maior_distancia decimal(10,1),
    @maior_elevacao decimal(10,1),
    @total_vitorias int
AS
BEGIN
    INSERT INTO ciclista (nome, total_km, total_elevacao,
        maior_distancia, maior_elevacao, total_vitorias)
    VALUES (@nome, @total_km, @total_elevacao, @maior_distancia,
        @maior_elevacao, @total_vitorias);
END
```

2. Atualizar:

```
CREATE PROCEDURE sp_AtualizarDados
    @IdCiclista int,
    @nome varchar(25),
    @total_km decimal(15,1),
    @total_elevacao decimal(15,1),
    @maior_distancia decimal(10,1),
    @maior_elevacao decimal(10,1),
    @total_vitorias int
AS
BEGIN
    UPDATE ciclista
    SET nome = @nome,
        total_km = @total_km,
        total_elevacao = @total_elevacao,
        maior_distancia = @maior_distancia,
        maior_elevacao = @maior_elevacao,
        total_vitorias = @total_vitorias
    WHERE IdCiclista = @IdCiclista;
END
```

3. Apagar:

```
CREATE PROCEDURE sp_RemoverDados
    @IdCiclista int
AS
BEGIN
    DELETE FROM ciclista
    WHERE IdCiclista = @IdCiclista;
END
```

4. Controlo de Transação e Tratamento de Erros:

```

CREATE PROCEDURE sp_TransacaoExemplo
    @IdEtapa int,
    @IdCiclista int,
    @IdTipo int,
    @valor decimal(15, 1)
AS
BEGIN
    BEGIN TRANSACTION;

    BEGIN TRY
        -- Inserir dados na tabela estatistica
        INSERT INTO estatistica (IdEtapa, IdCiclista, IdTipo, valor)
        VALUES (@IdEtapa, @IdCiclista, @IdTipo, @valor);

        -- Atualizar dados na tabela ciclista
        UPDATE ciclista
        SET total_km = total_km + @valor,
            total_vitorias = total_vitorias + 1
        WHERE IdCiclista = @IdCiclista;
        COMMIT;
    END TRY
    BEGIN CATCH
        -- Lidar com erros
        IF @@TRANCOUNT > 0
            ROLLBACK;
        THROW;
    END CATCH;
END

```

4.6 Function

A function que desenvolvemos retorna a quantidade de vitórias de um determinado ciclista.

```

CREATE FUNCTION calcularVitorias(@idCiclista INT)
RETURNS INT
AS
BEGIN
    DECLARE @vitorias INT;

    SELECT @vitorias = COUNT(*)
    FROM estatistica
    WHERE Idciclista = @idCiclista
    AND IdTipo = 1
    AND valor = 1;

    RETURN @vitorias;
END;

DECLARE @idCiclista INT;
SET @idCiclista = 1; -- mudar pelo id do ciclista que queremos

```

```

SELECT nome, dbo.calcularVitorias(IdCiclista) AS vitorias
FROM ciclista
WHERE IdCiclista = @idCiclista;

```

4.7 Cursores

1. Read only, que retorna ano, número de etapa, nome de ciclista e o tempo de prova correspondente:

```

DECLARE @AnoEtapa DECIMAL(4);
DECLARE @NumEtapa INT;
DECLARE @Nome VARCHAR(30);
DECLARE @TempoProva DECIMAL(10,2);
DECLARE @Horas INT;
DECLARE @Minutos INT;

DECLARE @Result TABLE (
    AnoEtapa DECIMAL(4),
    NumEtapa INT,
    Nome VARCHAR(30),
    TempoProva VARCHAR(5)
)

DECLARE Clas_ciclista CURSOR LOCAL STATIC READ_ONLY FOR
SELECT TP.anoEtapa,
       TP.numEtapa,
       TC.nome,
       TE.valor
FROM estatistica TE
INNER JOIN ciclista TC ON TE.Idciclista = TC.Idciclista
INNER JOIN etapa TP ON TE.IdEtapa = TP.IdEtapa
WHERE TE.IdTipo = 5
ORDER BY TP.anoEtapa, TP.numEtapa, TC.nome;

OPEN Clas_ciclista
FETCH NEXT FROM Clas_ciclista INTO @AnoEtapa, @NumEtapa, @Nome, @TempoProva

WHILE @@FETCH_STATUS = 0
BEGIN
    SET @Horas = FLOOR(@TempoProva / 60);
    SET @Minutos = @TempoProva % 60;

    INSERT INTO @Result (AnoEtapa, NumEtapa, Nome, TempoProva)
    VALUES (@AnoEtapa, @NumEtapa, @Nome, RIGHT('0' + CONVERT(VARCHAR(2), @Horas), 2)
    + ':' + RIGHT('0' + CONVERT(VARCHAR(2), @Minutos), 2))

    FETCH NEXT FROM Clas_ciclista INTO @AnoEtapa, @NumEtapa, @Nome, @TempoProva
END

CLOSE Clas_ciclista

```

```
DEALLOCATE Clas_ciclista
```

```
SELECT * FROM @Result  
Clas_ciclista
```

2. Update, que atualiza a tabela ciclista com os dados da tabela estatísticas:

```
DECLARE @Idciclista INT;  
DECLARE @Valor DECIMAL(15, 1);  
  
DECLARE cursorAtualizar CURSOR FOR  
SELECT Idciclista, valor  
FROM estatistica;  
  
OPEN cursorAtualizar;  
  
FETCH NEXT FROM cursorAtualizar INTO @Idciclista, @Valor;  
  
WHILE @@FETCH_STATUS = 0  
BEGIN  
    UPDATE ciclista  
    SET total_km = total_km + @Valor,  
        total_elevacao = total_elevacao + @Valor,  
        maior_distancia = CASE WHEN @Valor >  
        maior_distancia THEN @Valor ELSE maior_distancia END,  
        maior_elevacao = CASE WHEN @Valor >  
        maior_elevacao THEN @Valor ELSE maior_elevacao END,  
        total_vitorias = total_vitorias + 1  
    WHERE Idciclista = @Idciclista;  
  
    FETCH NEXT FROM cursorAtualizar INTO @Idciclista, @Valor;  
END;  
  
CLOSE cursorAtualizar;  
DEALLOCATE cursorAtualizar;
```

4.8 Outras funções

1. Row Number: numerar linhas sequencialmente

```
SELECT ROW_NUMBER() OVER (ORDER BY Idciclista) AS Numeracao, *  
FROM ciclista;
```

2. Rank: numerar linhas sequencialmente

```
SELECT RANK() OVER (ORDER BY total_km DESC) AS Classificacao, *  
FROM ciclista;
```

3. Dense Rank: obter a classificação dos ciclistas por total de vitórias

```
SELECT DENSE_RANK() OVER (ORDER BY total_vitorias DESC) AS Classificacao, *  
FROM ciclista;
```

4. Partition By: obter a média das estatísticas por tipo para cada ciclista e ano

```
SELECT c.nome, e.anoEtapa, te.tipo, AVG(es.valor) AS media_valor
FROM ciclista c
INNER JOIN estatistica es ON c.Idciclista = es.Idciclista
INNER JOIN etapa e ON es.IdEtapa = e.IdEtapa
INNER JOIN tipoEst te ON es.IdTipo = te.IdTipo
GROUP BY c.nome, e.anoEtapa, te.tipo
ORDER BY e.anoEtapa, c.nome, te.tipo;
```

4.9 Pivot

Calcular a média de valores de estatística por tipo para cada ciclista com pivot:

```
SELECT *
FROM (SELECT Idciclista, IdTipo, valor FROM estatistica) AS src
PIVOT (AVG(valor) FOR IdTipo IN ([1], [2], [3], [4], [5],
[6], [7], [8], [9], [10], [11], [12], [13], [14], [15])) AS pivoted
ORDER BY Idciclista;
```

Capítulo 5

Manutenção da Base de Dados

5.1 Filestream

Com o filestream surgiram dificuldades, pois, com a versão atual do SQL Server não é possível implementar o filestream após criação da base de dados, deve ser criado no início.

```
CREATE DATABASE estatisticasVoltaPT
ON
PRIMARY ( NAME = estatisticasVoltaPT,
          FILENAME = 'C:\abdtrabalho\estatisticasVoltaPT.mdf'),
          FILEGROUP estatisticasVoltaPTFS CONTAINS FILESTREAM(
          NAME = estatisticasVoltaPTFS,
          FILENAME = 'C:\abdtrabalho\estatisticasVoltaPTFS')
LOG ON (
          NAME = estatisticasVoltaPTLOG,
          FILENAME = 'C:\abdtrabalho\estatisticasVoltaPTLOG.ldf')
GO
```

Depois de criar a base de dados, criamos a tabela e inserimos os dados:

```
-- Create table for filestream
CREATE TABLE [dbo].[FileStreamTablevoltaPT] (
    [Idciclista] INT,
    [FSID] UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE,
    [FSDescription] VARCHAR(50),
    [FSBLOB] VARBINARY(MAX) FILESTREAM NULL,
    FOREIGN KEY (Idciclista) REFERENCES ciclista(Idciclista)
);

-- Inserir dados binários numa tabela FILESTREAM
INSERT into dbo.FileStreamTablevoltaPT VALUES
( 1, newid(), 'Ciclista_1',(select * from
OPENROWSET(BULK N'C:\ciclistas\ciclista_1.jpg',SINGLE_BLOB) as FS))

INSERT into dbo.FileStreamTablevoltaPT VALUES
( 2, newid(), 'Ciclista_2',(select * from
OPENROWSET(BULK N'C:\ciclistas\ciclista_2.jpg',SINGLE_BLOB) as FS))

INSERT into dbo.FileStreamTablevoltaPT VALUES
( 3, newid(), 'Ciclista_3',(select * from
OPENROWSET(BULK N'C:\ciclistas\ciclista_3.jpg',SINGLE_BLOB) as FS))
```

```
INSERT into dbo.FileStreamTablevoltaPT VALUES  
( 4, newid(), 'Ciclista_4',(select * from  
OPENROWSET(BULK N'C:\ciclistas\ciclista_4.jpg',SINGLE_BLOB) as FS))
```

```
INSERT into dbo.FileStreamTablevoltaPT VALUES  
( 5, newid(), 'Ciclista_5',(select * from  
OPENROWSET(BULK N'C:\ciclistas\ciclista_5.jpg',SINGLE_BLOB) as FS))
```

```
INSERT into dbo.FileStreamTablevoltaPT VALUES  
( 6, newid(), 'Ciclista_6',(select * from  
OPENROWSET(BULK N'C:\ciclistas\ciclista_6.jpg',SINGLE_BLOB) as FS))
```

```
INSERT into dbo.FileStreamTablevoltaPT VALUES  
( 7, newid(), 'Ciclista_7',(select * from  
OPENROWSET(BULK N'C:\ciclistas\ciclista_7.jpg',SINGLE_BLOB) as FS))
```

```
INSERT into dbo.FileStreamTablevoltaPT VALUES  
( 8, newid(), 'Ciclista_8',(select * from  
OPENROWSET(BULK N'C:\ciclistas\ciclista_8.jpg',SINGLE_BLOB) as FS))
```

```
INSERT into dbo.FileStreamTablevoltaPT VALUES  
( 9, newid(), 'Ciclista_9',(select * from  
OPENROWSET(BULK N'C:\ciclistas\ciclista_9.jpg',SINGLE_BLOB) as FS))
```

```
INSERT into dbo.FileStreamTablevoltaPT VALUES  
( 10, newid(), 'Ciclista_10',(select * from  
OPENROWSET(BULK N'C:\ciclistas\ciclista_10.jpg',SINGLE_BLOB) as FS))
```

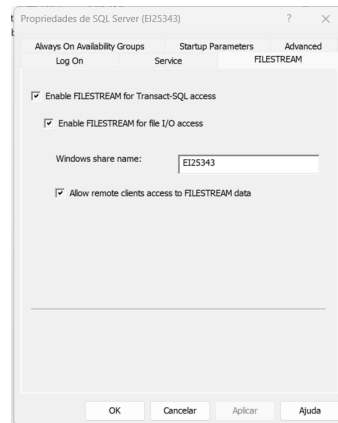



FIGURA 5.1: Alteração das propriedades do SQL SERVER

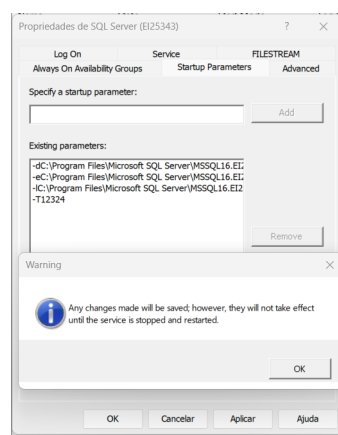


FIGURA 5.2: Habilitar sinalizadores de rastreamento específicos

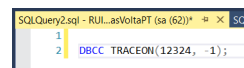


FIGURA 5.3: Adicionar um novo parametro - T12324

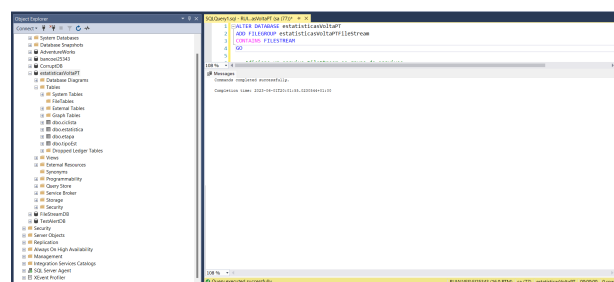


FIGURA 5.4: Criação do FileGroup

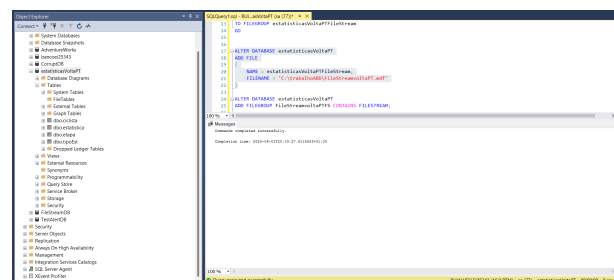


FIGURA 5.5: Adicionar ficheiros FileStream

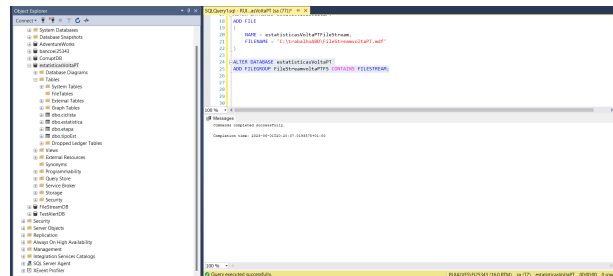


FIGURA 5.6: Criação de um novo FileGroup

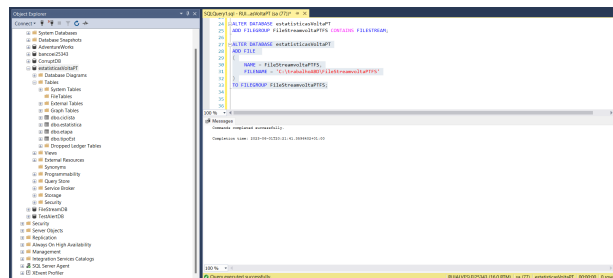


FIGURA 5.7: Adicionar ficheiros FileStream ao novo FileGroup

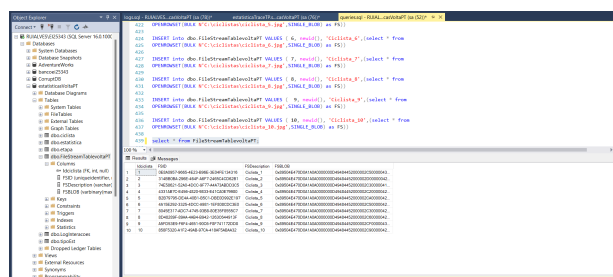


FIGURA 5.8: Inserir ficheiros binarios na FileStream

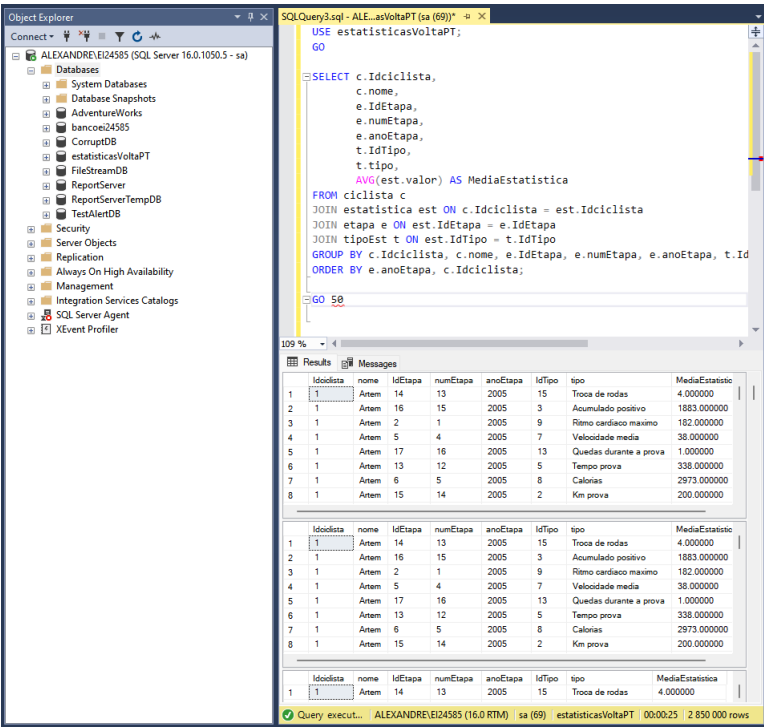


FIGURA 5.9: Consulta com peso

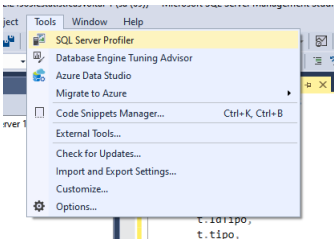


FIGURA 5.10: Tool > SQL Server Profiler

5.2 Database Engine Tuning Advisor

TRACE

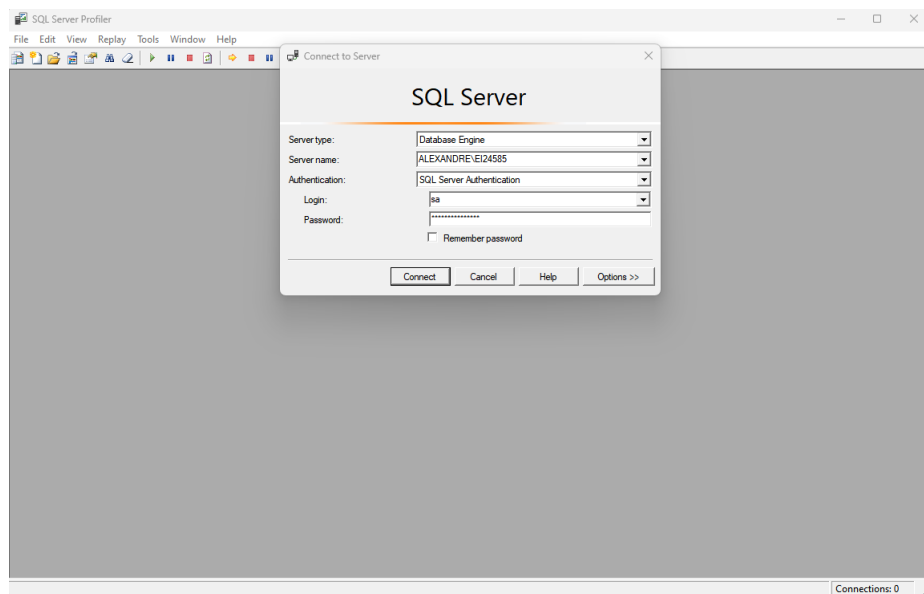


FIGURA 5.11: Ligação á instância

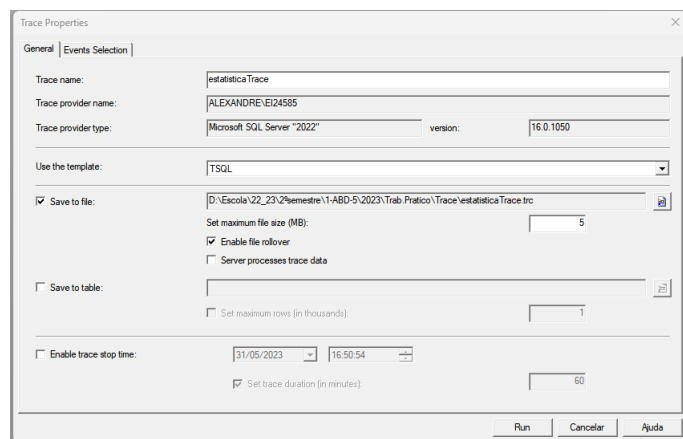


FIGURA 5.12: Propriedades

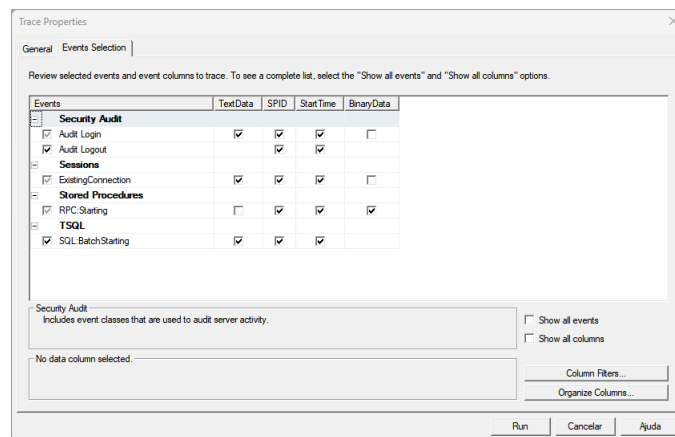


FIGURA 5.13: Seleção de eventos

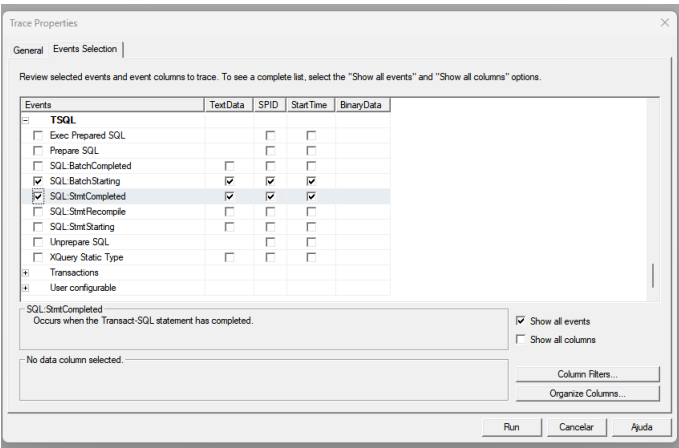


FIGURA 5.14: SQL:StmtCompleted

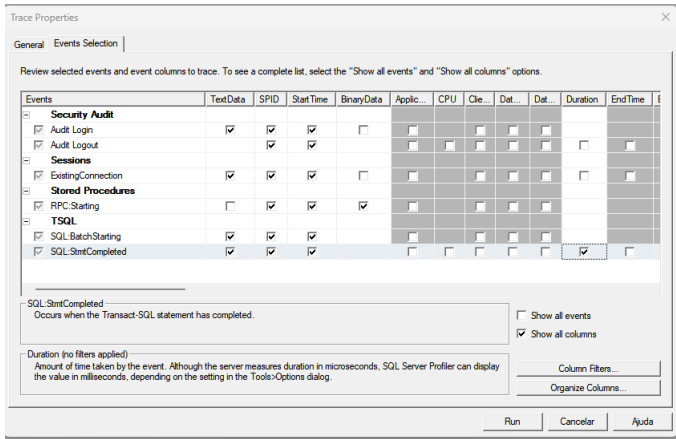


FIGURA 5.15: SQL:StmtCompleted Duration

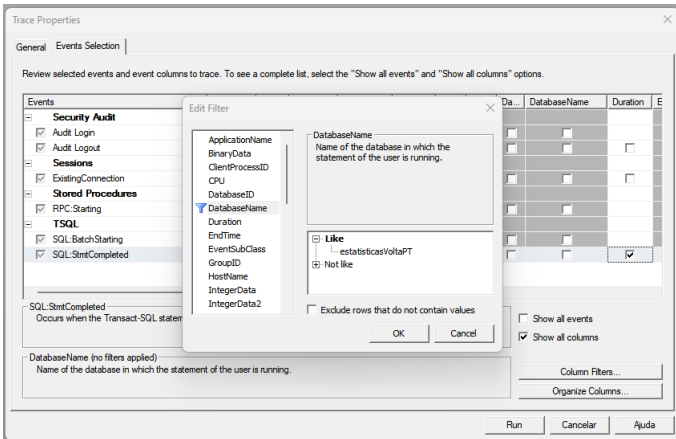


FIGURA 5.16: Database Name

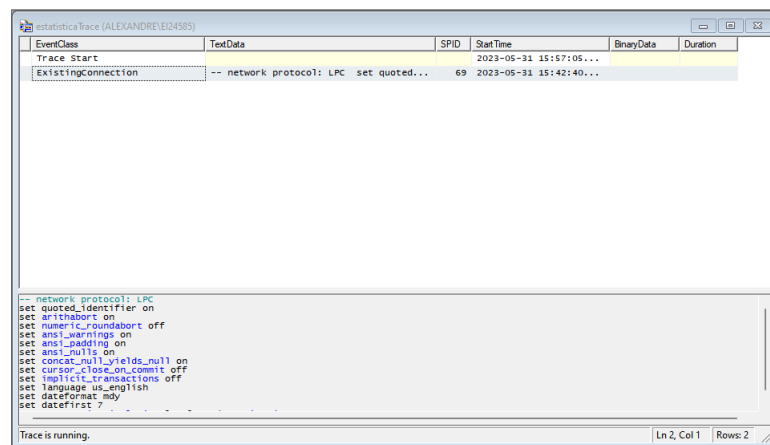


FIGURA 5.17: Trace a correr

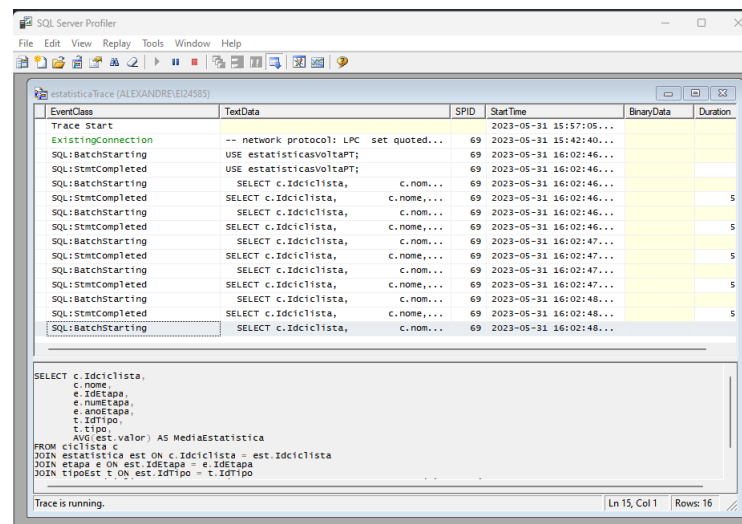


FIGURA 5.18: Trace com consulta

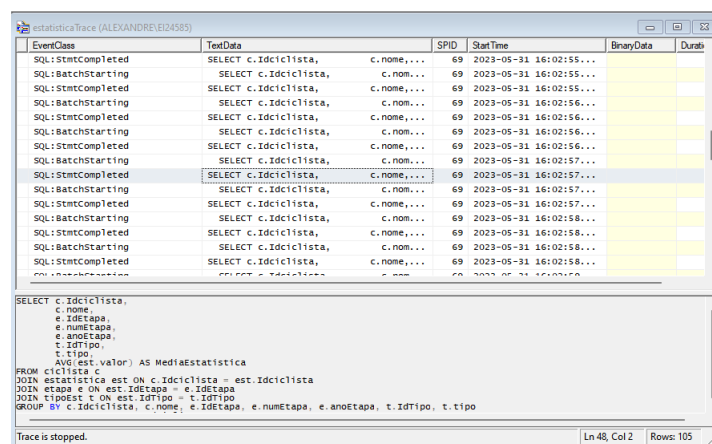


FIGURA 5.19: Trace individual detalhado

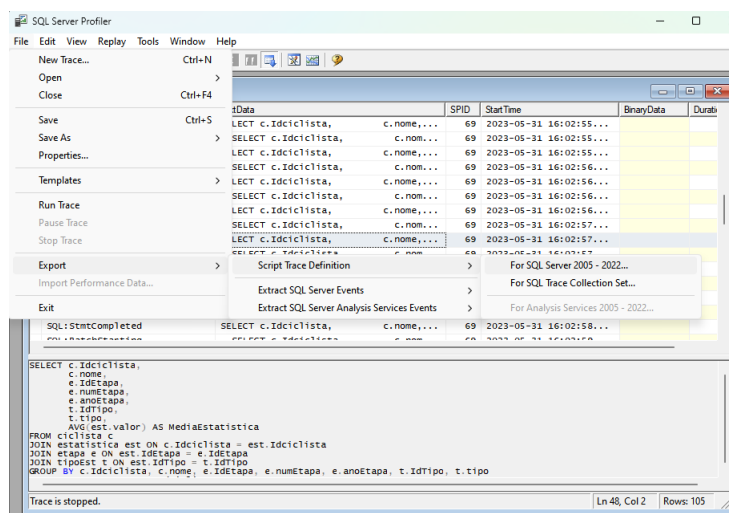


FIGURA 5.20: Exportar o Trace



FIGURA 5.21: Sucesso na exportação

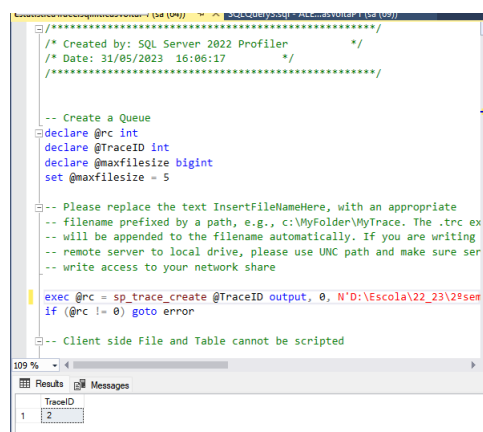


FIGURA 5.22: TraceID

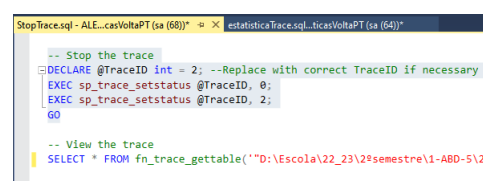


FIGURA 5.23: Stop the Trace

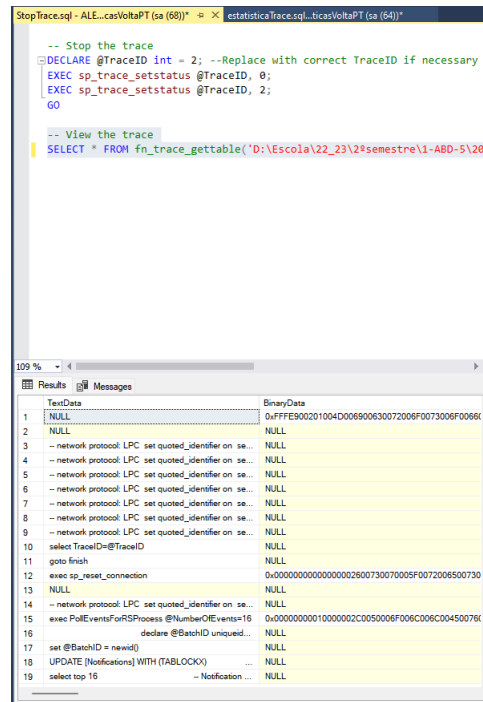


FIGURA 5.24: Ver o Trace

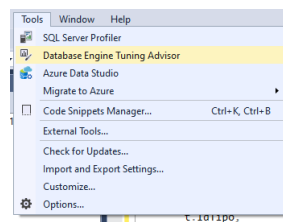


FIGURA 5.25: Abrir o Database Engine Tuning Advisor

OTIMIZAR

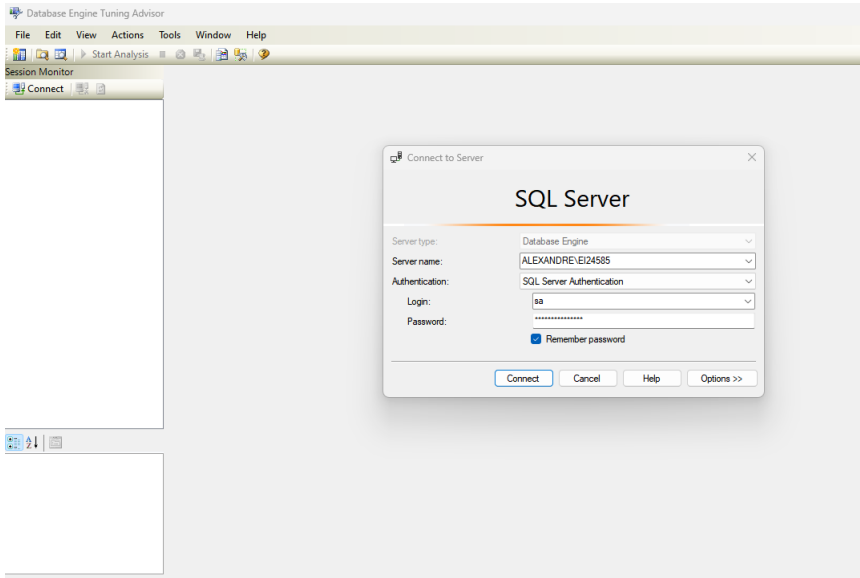


FIGURA 5.26: Conectar a instância

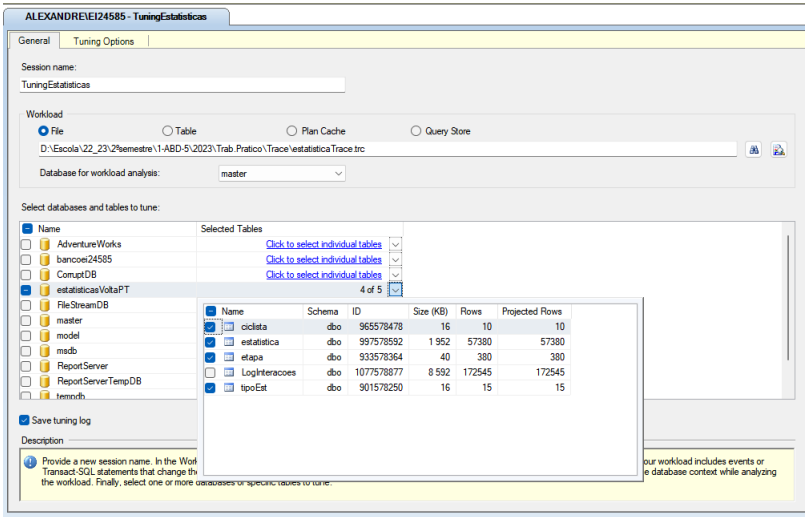


FIGURA 5.27: Selecionar as Tabelas

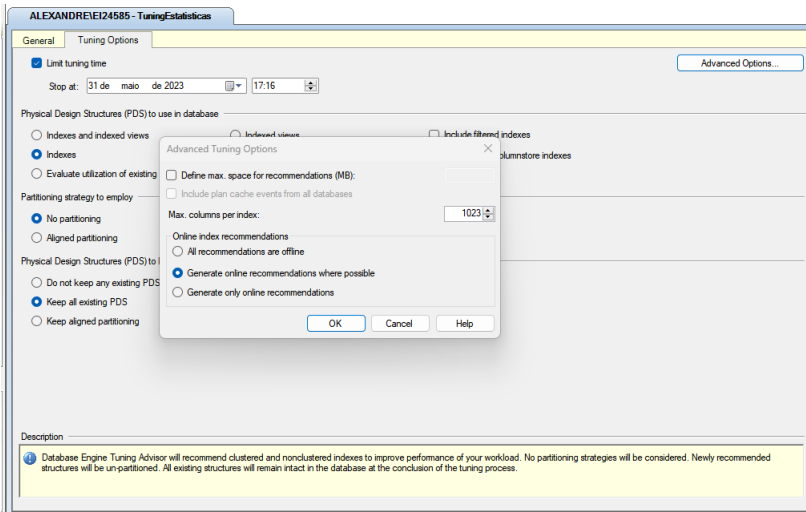


FIGURA 5.28: Gerar recomendações

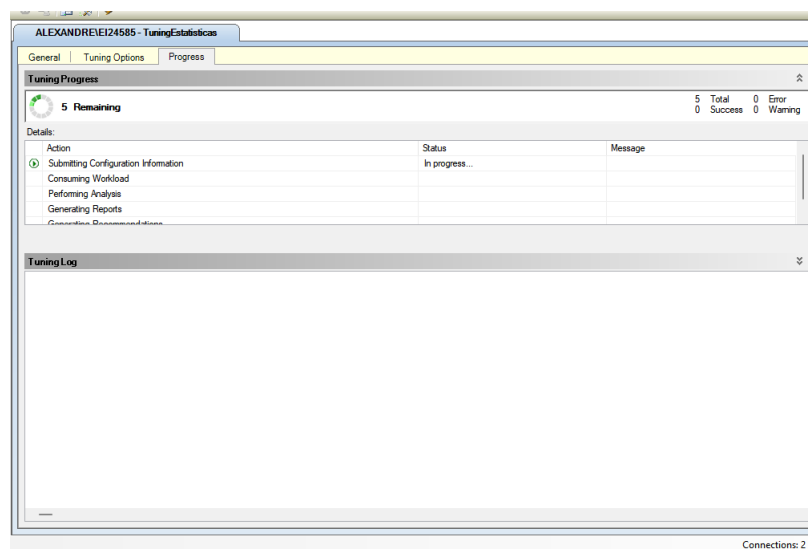


FIGURA 5.29: Análise da BD

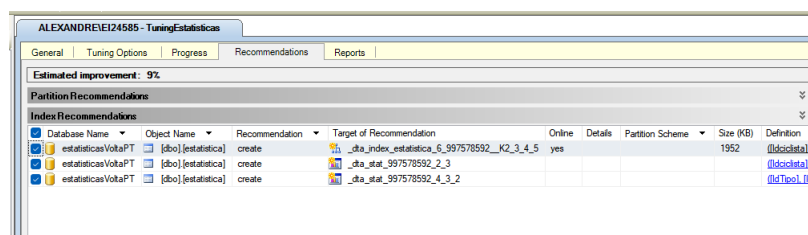


FIGURA 5.30: Resultados da análise

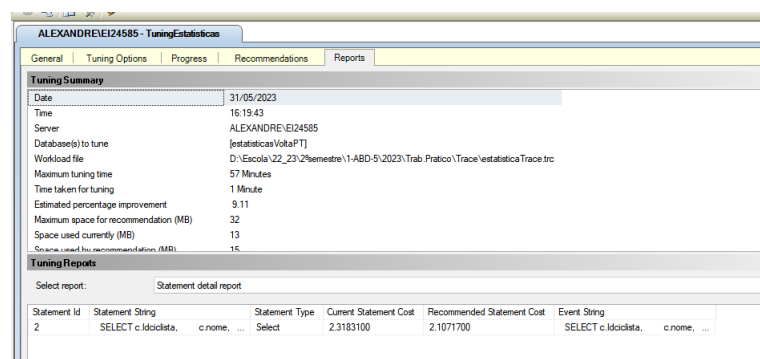


FIGURA 5.31: Report gerado

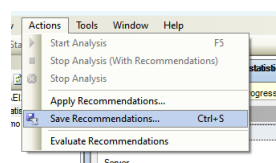


FIGURA 5.32: Guardar recomendações

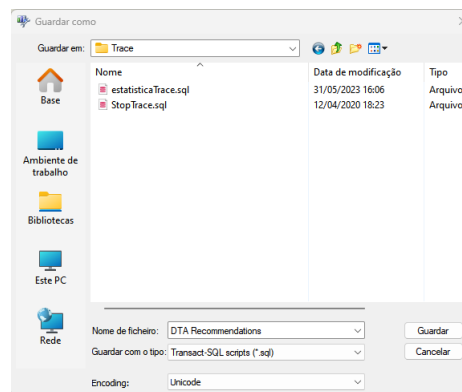


FIGURA 5.33: Localização

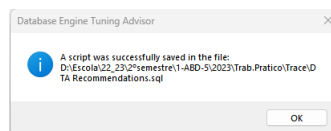


FIGURA 5.34: Sucesso

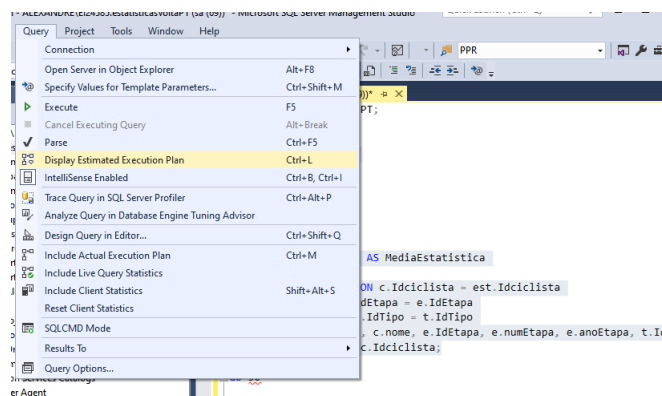


FIGURA 5.35: Estimativa do plano de execução

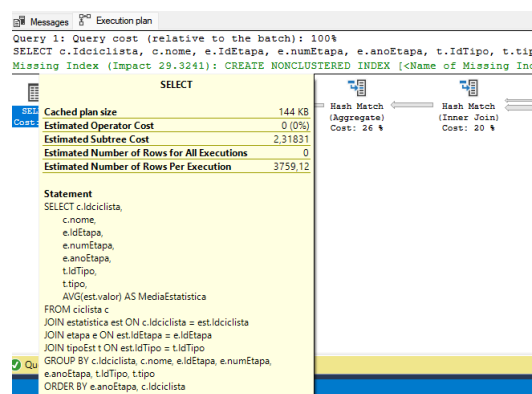


FIGURA 5.36: Subtree Cost alto

```

DIA Recommendation...VoltaPT (sa (66))  SQLQuery3.sql - ALÉ...asVoltaPT (sa (69))
use [estatisticasVoltaPT]
go

CREATE NONCLUSTERED INDEX [_dta_index_estatistica_6_997578592_K2_3_4_5] ON [dbo].[estatistica]
(
    [Idciclista] ASC
)
INCLUDE([IdEtapa],[IdTipo],[valor]) WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = ON) ON [PRIMARY]
go

CREATE STATISTICS [_dta_stat_997578592_2_3] ON [dbo].[estatistica]([Idciclista], [IdEtapa])
WITH AUTO_DROP = OFF
go

CREATE STATISTICS [_dta_stat_997578592_4_3_2] ON [dbo].[estatistica]([IdTipo], [IdEtapa], [Idciclista])
WITH AUTO_DROP = OFF
go

```

FIGURA 5.37: Execução do plano

SELECT	
Cached plan size	152 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	2,23454
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	57000
Statement	
SELECT c.Idciclista, c.nome, e.IdEtapa, e.numEtapa, e.anoEtapa, t.IdTipo, t.tipo, AVG(est.valor) AS MediaEstatistica FROM ciclista c JOIN estatistica est ON c.Idciclista = est.Idciclista JOIN etapa e ON est.IdEtapa = e.IdEtapa JOIN tipoEst t ON est.IdTipo = t.IdTipo GROUP BY c.Idciclista, c.nome, e.IdEtapa, e.numEtapa, e.anoEtapa, t.IdTipo, t.tipo ORDER BY e.anoEtapa, c.Idciclista	

FIGURA 5.38: Otimização do Subtree Cost

5.3 Database Maintenance

Para criar a manutenção da base de dados seguimos os passos descritos nas imagens seguintes:

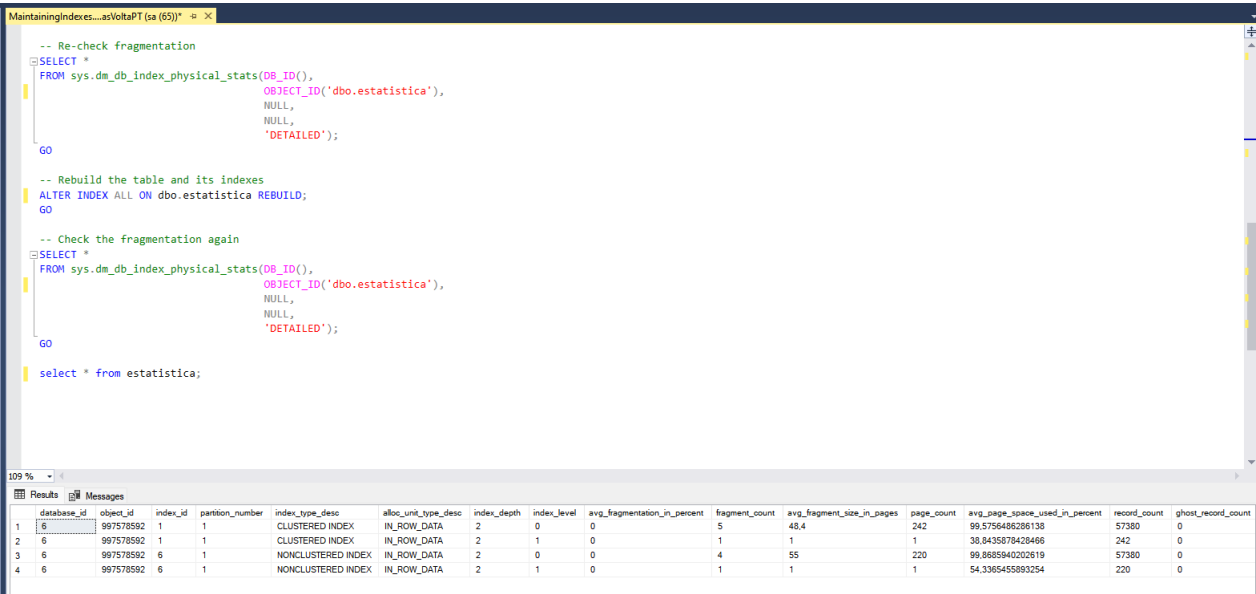


FIGURA 5.39: Fragmentação das estatísticas

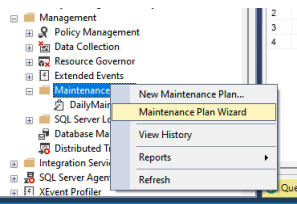


FIGURA 5.40: Abrir o Maintenance Plan Wizard

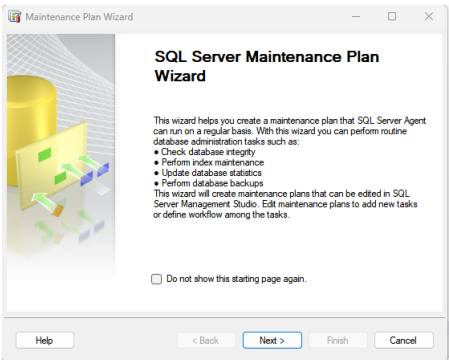


FIGURA 5.41: Maintenance Plan Wizard

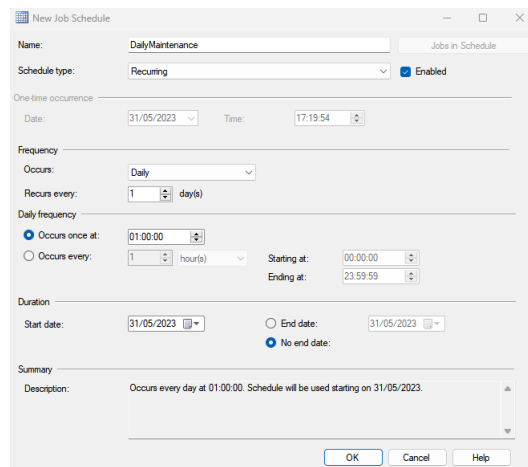


FIGURA 5.42: New Job Schedule

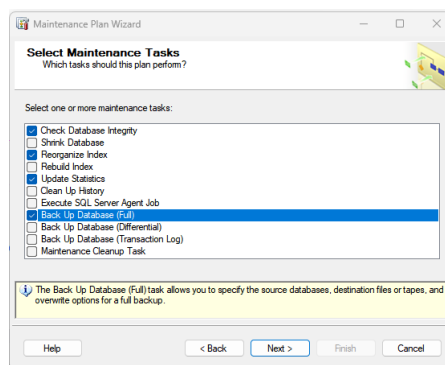


FIGURA 5.43: Tarefas de manutenção

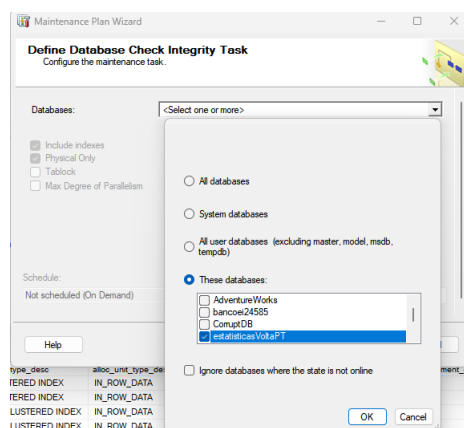


FIGURA 5.44: Tabela para verificação de integridade

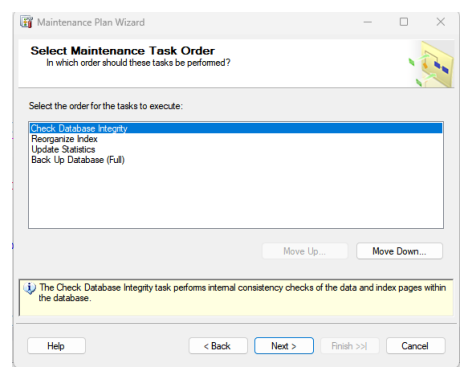


FIGURA 5.45: Ordem das tarefas

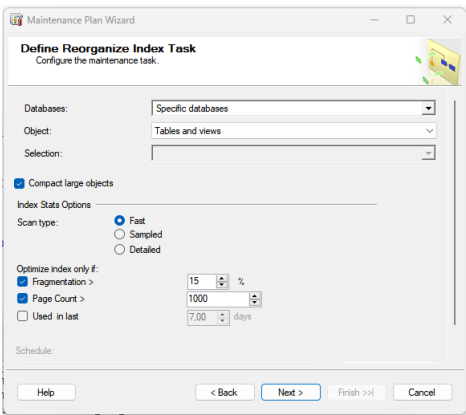


FIGURA 5.46: Reorganização de índices

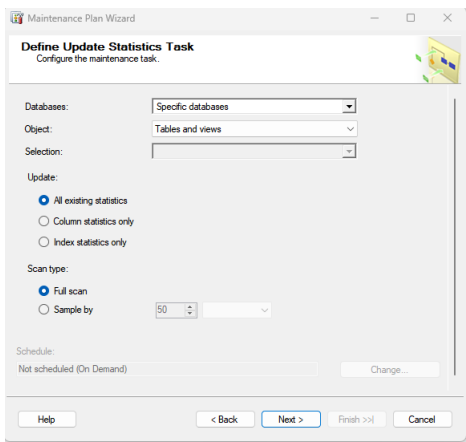


FIGURA 5.47: Atualização de estatísticas

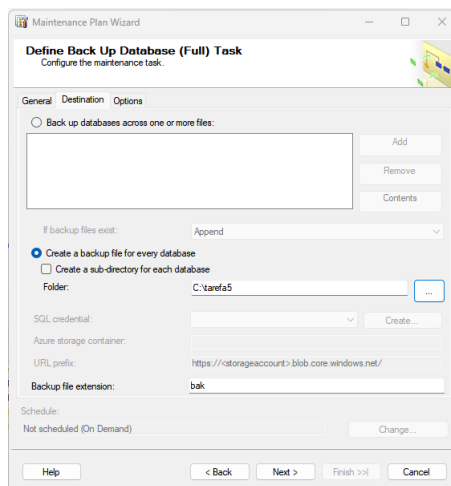


FIGURA 5.48: Definir o backup

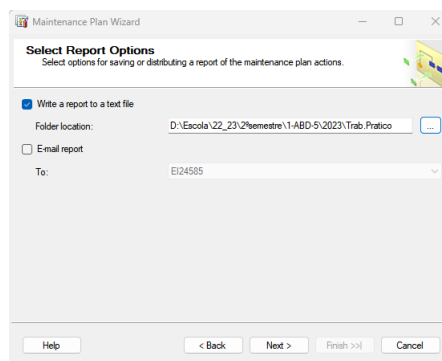


FIGURA 5.49: Opções do relatório e email

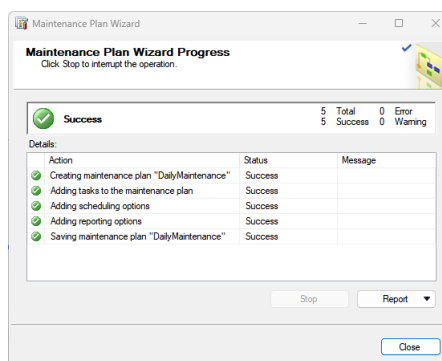


FIGURA 5.50: Resumo com sucesso de execução

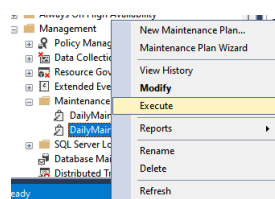


FIGURA 5.51: Executar

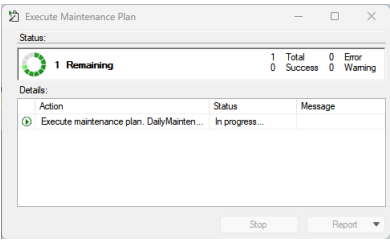


FIGURA 5.52: Execução do plano

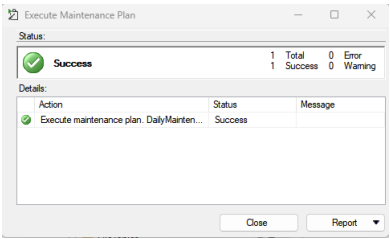


FIGURA 5.53: Resultado da execução

5.4 Relatório da Base de Dados

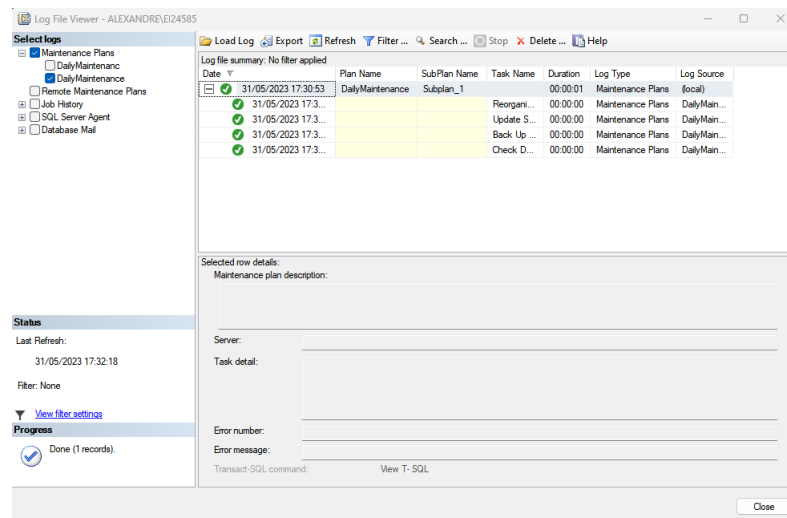


FIGURA 5.54: Finalização

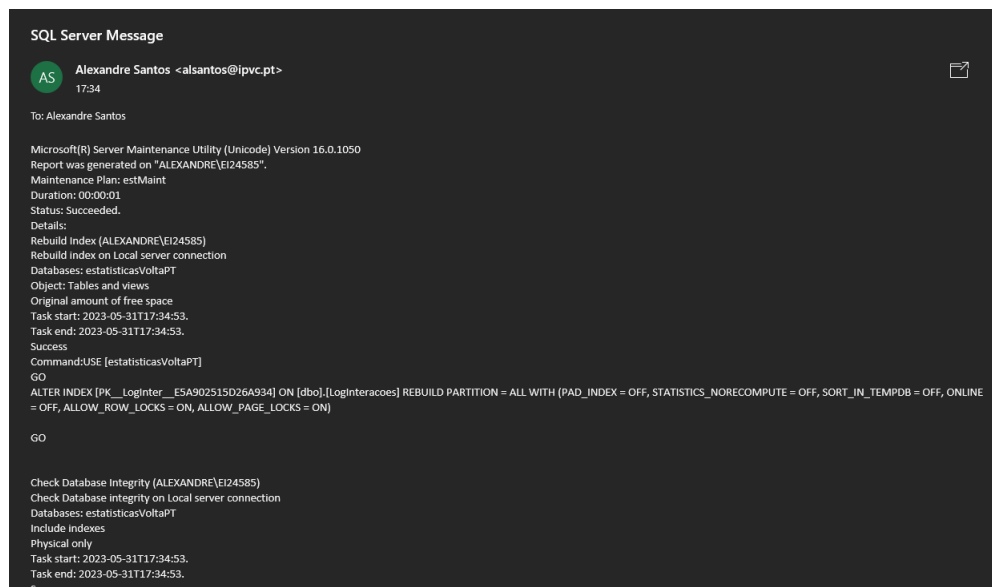


FIGURA 5.55: Email enviado para o administrador

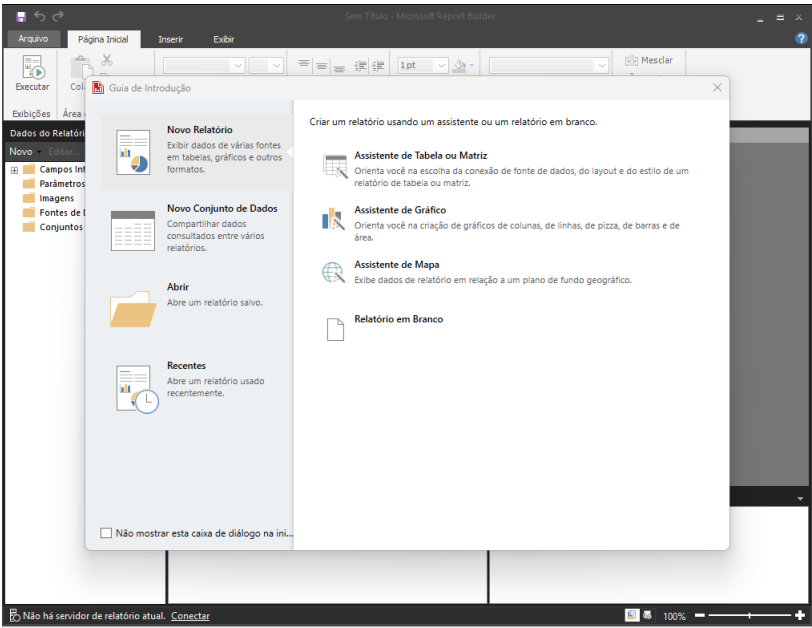


FIGURA 5.56: Criação de um relatório novo

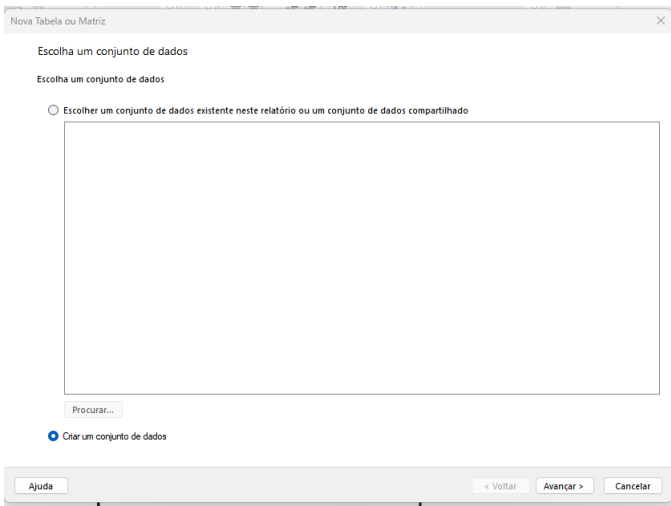


FIGURA 5.57: Criar novo conjunto de dados

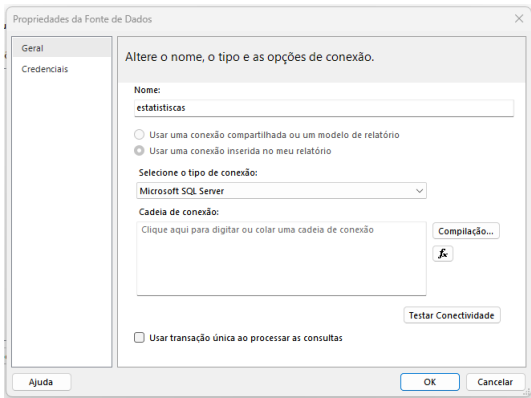


FIGURA 5.58: Janela de propriedades da fonte de dados

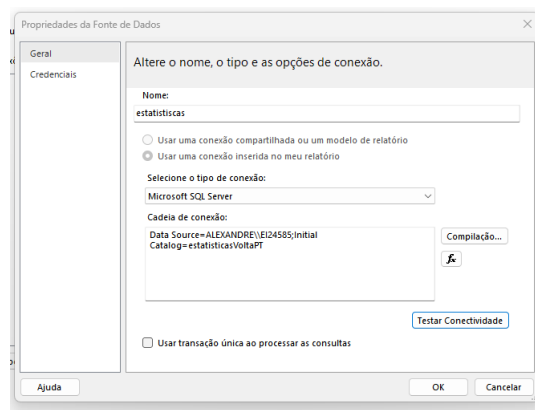


FIGURA 5.59: Compilação da instância e base de dados

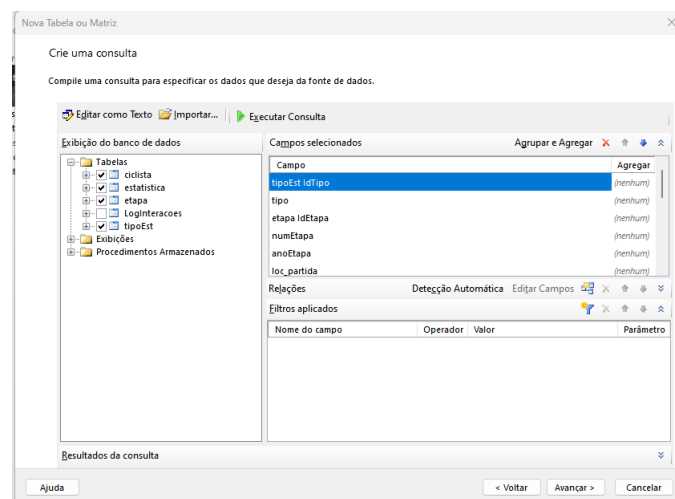


FIGURA 5.60: Seleção das tabelas para o gráfico

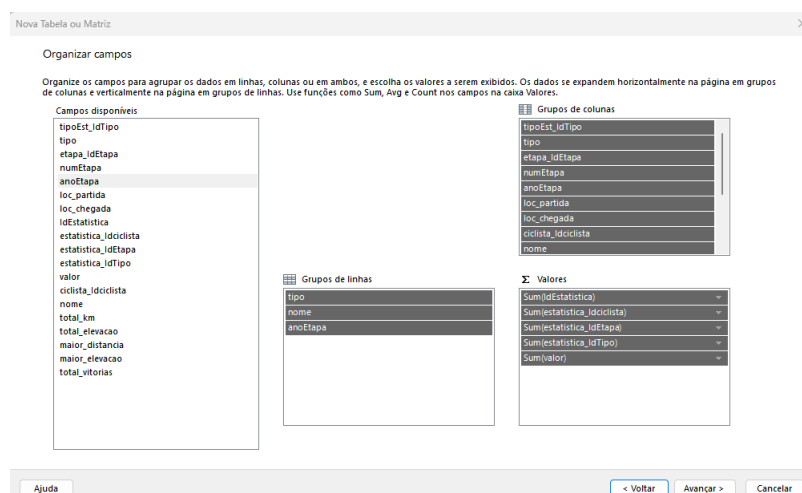


FIGURA 5.61: Seleção entre colunas, linhas, valores e tipo de função

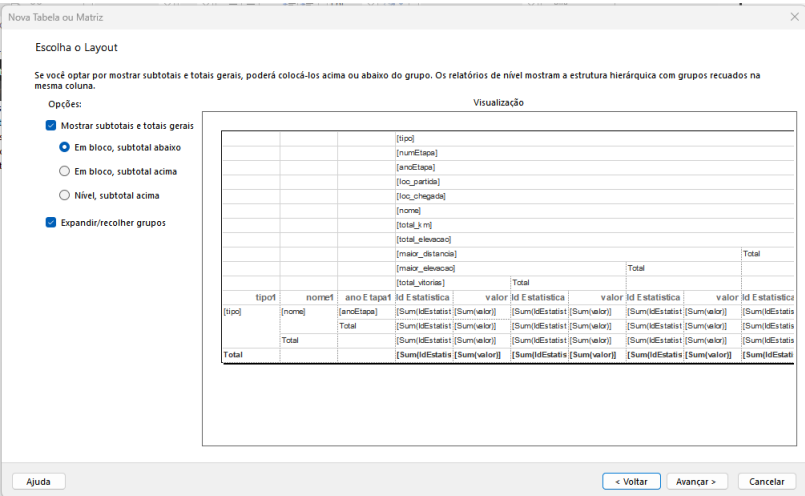


FIGURA 5.62: Definir layout do relatório

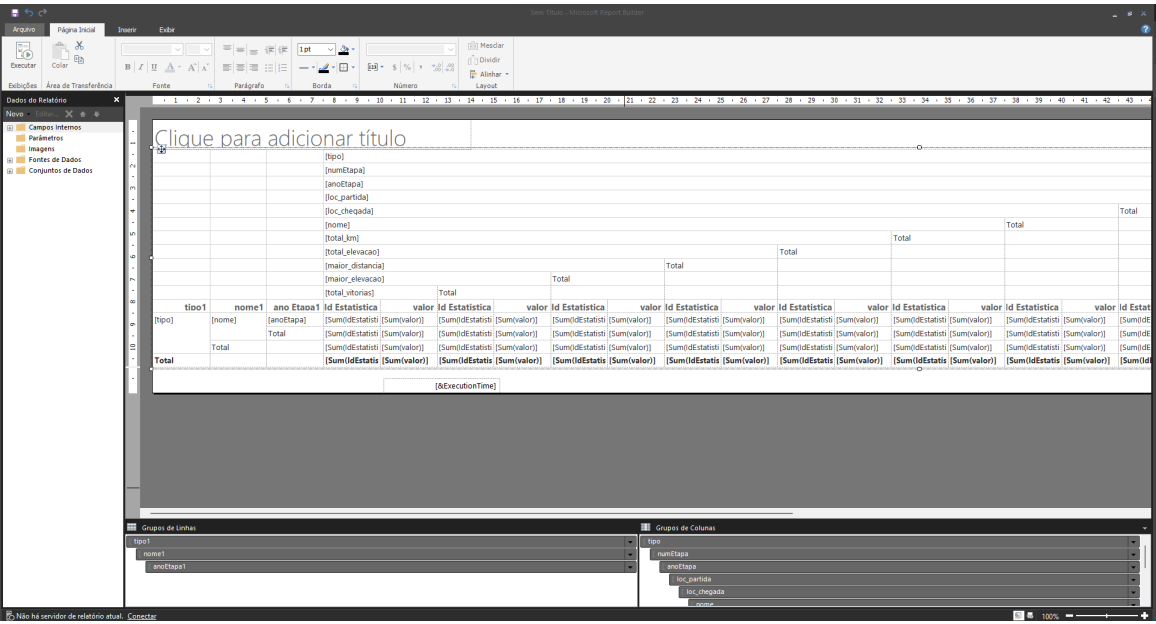


FIGURA 5.63: Resumo do relatório



FIGURA 5.64: Tentativa de criação de uma tabela com erro

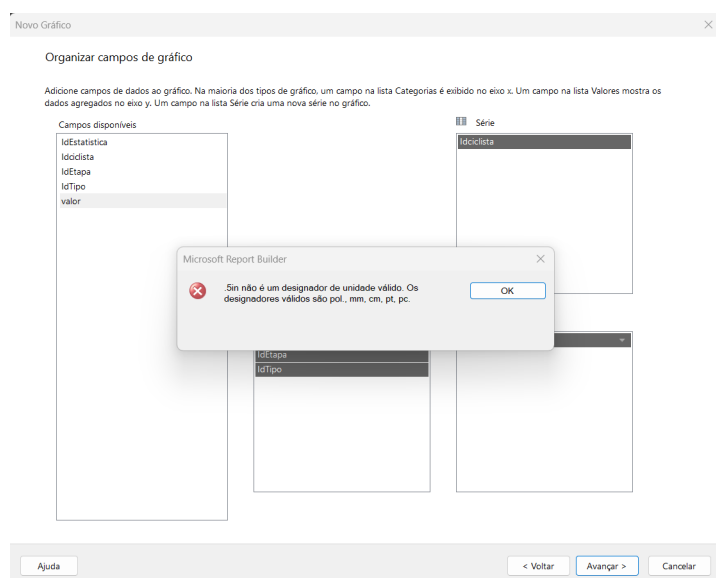


FIGURA 5.65: Tentativa de criação de um gráfico com erro

Capítulo 6

Manutenção da Base de Dados (Pontos Opcionais)

6.1 Procedimento de inserção de dados em formato JSON

Para a inserção de dados em ficheiros JSON, realizamos um procedimento de teste com a tabela *Etapa*, onde usamos a função **JSON VALUE** para converter os dados de json para texto que seja armazenado na base de dados.

```
CREATE PROCEDURE InsertEtapaFromJSON
    @jsonData NVARCHAR(MAX)
AS
BEGIN
    INSERT INTO etapa (numEtapa, anoEtapa, loc_partida, loc_chegada)
    SELECT
        JSON_VALUE(@jsonData, '$.numEtapa'),
        JSON_VALUE(@jsonData, '$.anoEtapa'),
        JSON_VALUE(@jsonData, '$.loc_partida'),
        JSON_VALUE(@jsonData, '$.loc_chegada');
END;

DECLARE @jsonData NVARCHAR(MAX) = '
{
    "numEtapa": 0,
    "anoEtapa": 2024,
    "loc_partida": "TESTE A",
    "loc_chegada": "TESTE B"
}';

EXEC InsertEtapaFromJSON @jsonData;
```

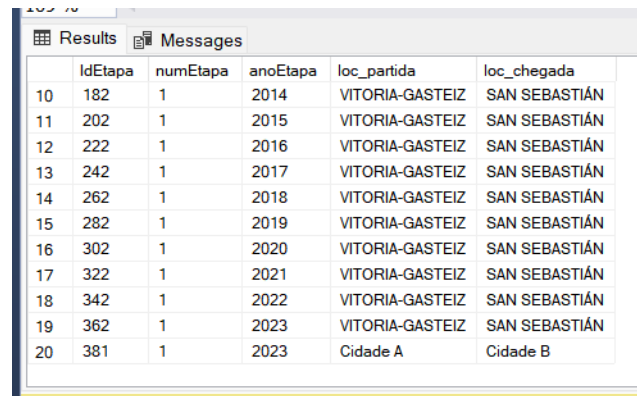
Para vermos como a inserção foi realizada com sucesso, realizamos um select que irá mostrar os dados existentes e o que inserimos de teste:

```
SELECT * FROM etapa where numEtapa = 1;
```

6.2 Procedimento que guarde dados em formato JSON

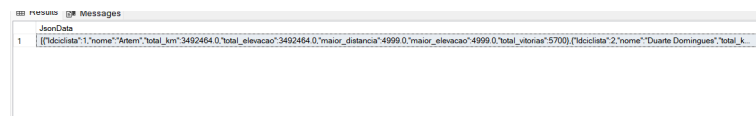
Para passar dados da base de dados para um formato JSON, criámos uma tabela temporária que armazena o output dos dados em JSON.

```
CREATE PROCEDURE GetCiclistaDataAsJSON
AS
```



	IdEtapa	numEtapa	anoEtapa	loc_partida	loc_chegada
10	182	1	2014	VITORIA-GASTEIZ	SAN SEBASTIÁN
11	202	1	2015	VITORIA-GASTEIZ	SAN SEBASTIÁN
12	222	1	2016	VITORIA-GASTEIZ	SAN SEBASTIÁN
13	242	1	2017	VITORIA-GASTEIZ	SAN SEBASTIÁN
14	262	1	2018	VITORIA-GASTEIZ	SAN SEBASTIÁN
15	282	1	2019	VITORIA-GASTEIZ	SAN SEBASTIÁN
16	302	1	2020	VITORIA-GASTEIZ	SAN SEBASTIÁN
17	322	1	2021	VITORIA-GASTEIZ	SAN SEBASTIÁN
18	342	1	2022	VITORIA-GASTEIZ	SAN SEBASTIÁN
19	362	1	2023	VITORIA-GASTEIZ	SAN SEBASTIÁN
20	381	1	2023	Cidade A	Cidade B

FIGURA 6.1: Conversão JSON para SQL



JsonData
1 [{"idciclista":1,"nome":"Adem","total_km":3492464.0,"total_elevacao":3492464.0,"maior_distancia":4999.0,"maior_elevacao":4999.0,"total_vitorias":5700}], [{"idciclista":2,"nome":"Duarte Domingues","total_k...

FIGURA 6.2: Conversão SQL para JSON

```

BEGIN
    SET NOCOUNT ON;

    DECLARE @jsonOutput NVARCHAR(MAX);

    SELECT @jsonOutput = (
        SELECT *
        FROM ciclista
        FOR JSON AUTO
    );

    SELECT @jsonOutput AS JsonData;
END;

EXEC GetCiclistaDataAsJSON;

```

Para poder inserir os dados json num ficheiro, seria necessário desenvolver um script que extraísse os dados da base de dados dado uso ao procedimento que desenvolvemos.

6.3 Trigger para notificação por email

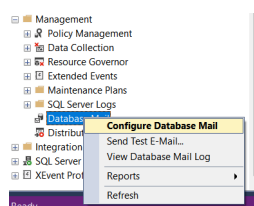


FIGURA 6.3: Abrir Database Mail Configuration

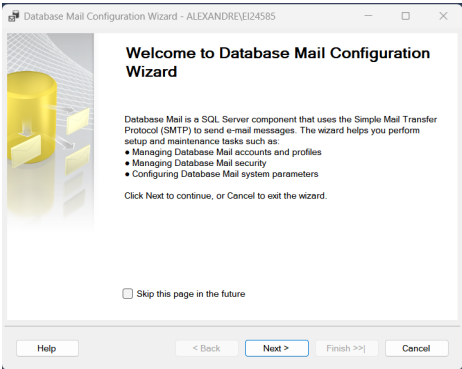


FIGURA 6.4: Database Mail Confiduration

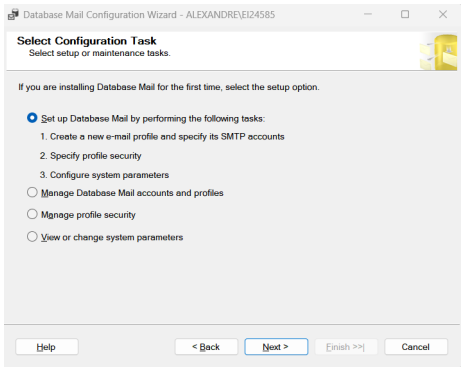


FIGURA 6.5: Configuração de tarefas

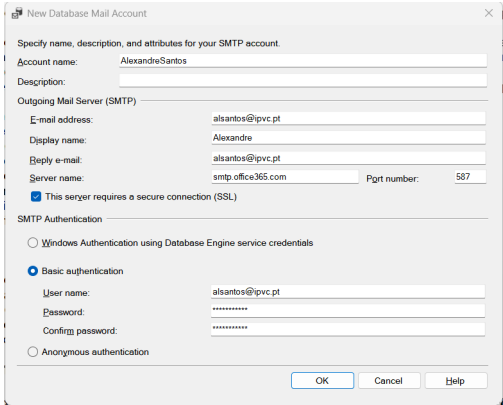


FIGURA 6.6: Perfis

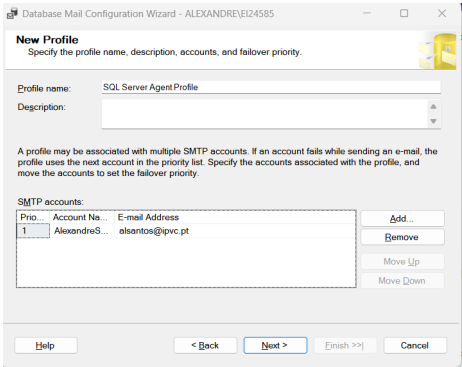


FIGURA 6.7: Criar perfil

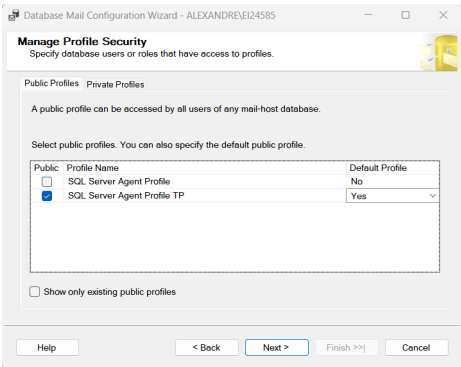


FIGURA 6.8: Segurança do perfil

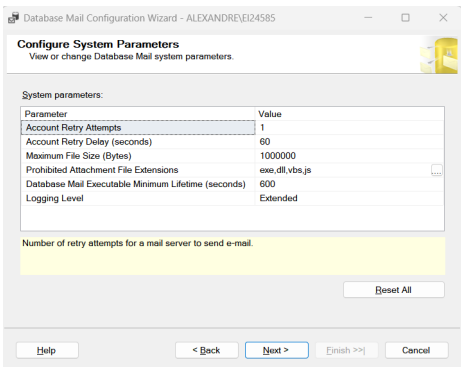


FIGURA 6.9: Parâmetros do sistema

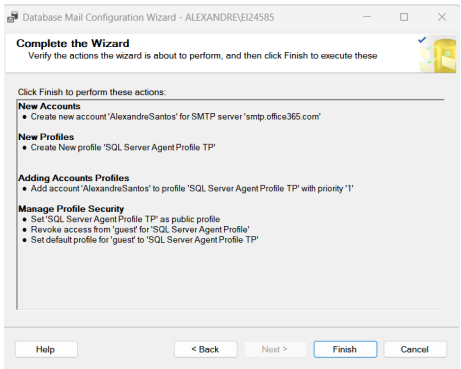


FIGURA 6.10: Finalizar

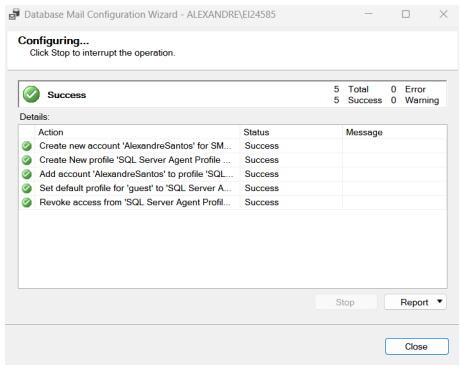


FIGURA 6.11: Configuração do email completa

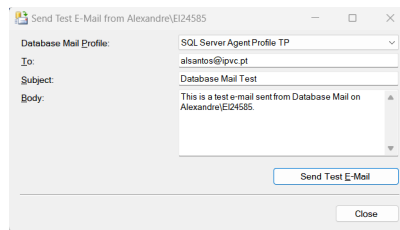


FIGURA 6.12: Envio de e-mail de teste

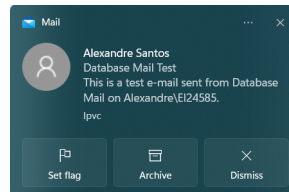


FIGURA 6.13: E-mail recebido

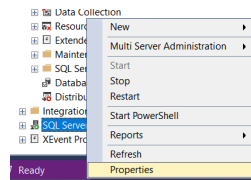


FIGURA 6.14: Propriedades do agente

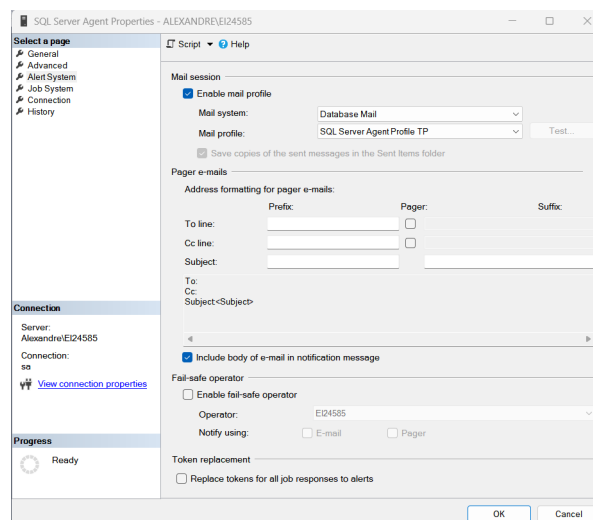


FIGURA 6.15: Configuração do perfil

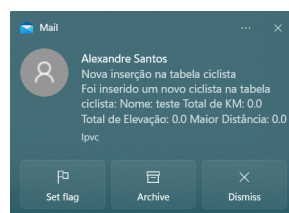


FIGURA 6.16: Email enviado através do trigger

Capítulo 7

Conclusões

Nesta fase final, resta-nos concluir que, a administração de uma base de dados não é tarefa simples. Do nosso ponto de vista, as tarefas mais difíceis foram, como gerar os dados em grandes quantidades, o desenvolvimento de algumas das queries, como procedimentos e cursores, e por fim a implementação do filestream que teve as suas dificuldades iniciais. De resto, podemos dizer que gostamos da oportunidade de desenvolver este trabalho prático e que conseguimos aprimorar os nossos conhecimentos sobre bases de dados e a sua administração.