UNIVERSIDADE DE FORTALEZA

ESPECIALIZAÇÃO EM ENGENHARIA DE DADOS

TURMA 2 - Z100

FUNDAMENTOS DE BANCO DE DADOS E MODELAGEM DE DADOS

EQUIPE:
Dante Dantas - 2518583
Evellen Silva - 2518889
Marcos Aurelio - 2519887
Rafael Tavares - 2517595

ATIVIDADE AVALIATIVA

PROFESSOR:
Marcondes Josino Alexandre

FORTALEZA - CEARÁ
MAIO / 2025

SUMÁRIO

1. INTRODUÇÃO

O presente relatório tem como objetivo apresentar a estrutura e organização do banco de dados **TransporteMaritimo**, desenvolvido para gerenciar o transporte de cargas por meio de navios entre portos, com a intermediação de agentes receptores.

2. ESTRUTURA DO BANCO DE DADOS

2.1 Tabelas

Navio

Responsável por armazenar informações sobre os navios utilizados no transporte.

- **NavioId** – Identificador único do navio (chave primária);
- **NumeroNavio** – Número de registro do navio;
- **NomeNavio** – Nome do navio;
- **CapacidadeKg** – Capacidade máxima de transporte em quilogramas.

Porto

Tabela que armazena os portos envolvidos no processo de embarque e desembarque.

- **PortoId** – Identificador único do porto (chave primária);
- **NomePorto** – Nome do porto;
- **CodigoPorto** – Código identificador do porto.

AgenteReceptor

Registra os agentes responsáveis por receber as cargas nos portos.

- **AgenteReceptorId** – Identificador do agente (chave primária);
- **NomeAgente** – Nome do agente;

- **PortoId** – Chave estrangeira que referencia o porto onde o agente atua;
- **CodigoAgente** – Código identificador do agente.

Carga

Contém os dados de cada carga transportada.

- **CargaId** – Identificador único da carga (chave primária);
- **NumeroCarga** – Número de registro da carga;
- **PesoKg** – Peso da carga em quilogramas;
- **DataMaximaDesembarque** – Prazo final para desembarque da carga;
- **AgenteReceptorId** – Chave estrangeira que identifica o agente responsável;
- **PortoDestinoId** – Chave estrangeira do porto de destino;
- **DataValidade** – Validade da carga (se aplicável);
- **TemperaturaMaxima** – Temperatura máxima permitida para transporte;
- **NavioId** – Chave estrangeira que identifica o navio que transporta a carga.

3. RELACIONAMENTOS

- A **tabela Carga** possui relacionamentos com as tabelas:
    - Navio – através da chave estrangeira NavioId;
    - AgenteReceptor – através da chave estrangeira AgenteReceptorId;
    - Porto – através da chave estrangeira PortoDestinoId.
- A **tabela AgenteReceptor** possui relacionamento com a tabela:
    - Porto – através da chave estrangeira PortoId.
- A **tabela Rota_Porto** representa um relacionamento de muitos-para-muitos entre Rota e Porto, com colunas que indicam a ordem da visita e a data

estimada de chegada:

- PortoId → Porto;
- RotaId → Rota.

- A **tabela Rota** está relacionada com a tabela:
  - Navio – através da chave estrangeira NavioId.

3.3 Função e Procedure

Função: CalcularCargaNavio

Sql:

```
CREATE FUNCTION [dbo].[CalcularCargaNavio] (@NavioId INT)
RETURNS DECIMAL(18, 2)
AS
BEGIN
    DECLARE @CargaAtual DECIMAL(18, 2);
    SELECT @CargaAtual = ISNULL(SUM(PesoKg), 0)
    FROM Carga
    WHERE NavioId = @NavioId;
    RETURN @CargaAtual;
END;
```

Procedure: AlocarCargas

Sql:

```
CREATE PROCEDURE [dbo].[AlocarCargas]
AS
BEGIN
    DECLARE @CargaId INT, @PortoDestinoId INT, @DataMaximaDesembarque DATE,
@PesoCarga DECIMAL(18, 2), @NavioId INT;

    DECLARE carga_cursor CURSOR FOR
    SELECT CargaId, PortoDestinoId, DataMaximaDesembarque, PesoKg
    FROM Carga
    WHERE NavioId IS NULL;

    OPEN carga_cursor;
    FETCH NEXT FROM carga_cursor INTO @CargaId, @PortoDestinoId,
@DataMaximaDesembarque, @PesoCarga;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SELECT TOP 1 @NavioId = r.NavioId
        FROM Rota_Porto rp
        JOIN Rota r ON rp.RotaId = r.RotaId
        JOIN Navio n ON r.NavioId = n.NavioId
        WHERE rp.PortoId = @PortoDestinoId
```

```
          AND rp.DataChegadaEstimada <= @DataMaximaDesembarque
          AND n.CapacidadeKg >= dbo.CalcularCargaNavio(r.NavioId) +
@PesoCarga
        ORDER BY rp.OrdemVisita;

        IF @NavioId IS NOT NULL
        BEGIN
            UPDATE Carga SET NavioId = @NavioId WHERE CargaId = @CargaId;
        END

        FETCH NEXT FROM carga_cursor INTO @CargaId, @PortoDestinoId,
@DataMaximaDesembarque, @PesoCarga;
    END

    CLOSE carga_cursor;
    DEALLOCATE carga_cursor;
END;
```

4. RELATÓRIOS, VIEWS E TRIGGER

4.1 Relatórios

Relatório 1: Cargas por Porto
Consulta que retorna o total de cargas e o peso total transportado para cada porto.

Sql:

```
SELECT
    'Cargas por Porto' AS Relacionamento,
    p.NomePorto,
    COUNT(c.CargaId) AS Total_Cargas,
    SUM(c.PesoKg) AS Peso_Total
FROM Porto p
LEFT JOIN Carga c ON p.PortoId = c.PortoDestinoId
GROUP BY p.NomePorto;
```

Relatório 2: Cargas por Navio
Consulta que mostra a quantidade de cargas e o peso total transportado por cada navio.

Sql:

```
SELECT
    'Cargas por Navio' AS Relacionamento,
    n.NomeNavio,
    COUNT(c.CargaId) AS Total_Cargas,
    SUM(c.PesoKg) AS Peso_Total
FROM Navio n
LEFT JOIN Carga c ON n.NavioId = c.NavioId
GROUP BY n.NomeNavio;
```

Relatório 3: Cargas por Navio e Porto com filtros
Apresenta a quantidade, peso total e peso médio das cargas transportadas por navio com destino a cada porto, considerando apenas cargas cujo peso total excede 1000 kg.

Sql:

```
SELECT
    n.NomeNavio,
    p.NomePorto AS PortoDestino,
    COUNT(c.CargaId) AS TotalCargas,
    SUM(c.PesoKg) AS PesoTotalCargas,
    AVG(c.PesoKg) AS PesoMedioCargas
FROM Navio n
JOIN Carga c ON n.NavioId = c.NavioId
JOIN Porto p ON c.PortoDestinoId = p.PortoId
GROUP BY n.NomeNavio, p.NomePorto
HAVING SUM(c.PesoKg) > 1000
ORDER BY PesoTotalCargas DESC;
```

4.2 Views

View 1: `vw_navios`
Retorna os nomes dos navios e os pesos das cargas associadas a eles.

Sql:

```
CREATE VIEW vw_navios AS
SELECT
    n.NomeNavio,
    c.PesoKg
FROM Navio n
LEFT JOIN Carga c ON n.NavioId = c.NavioId;
```

View 2: `vw_DetalhesCargasNavios`
Apresenta informações detalhadas das cargas transportadas, incluindo o navio, porto de destino e agente receptor.

Sql:

```
CREATE VIEW [dbo].[vw_DetalhesCargasNavios] AS
SELECT
    c.CargaId,
    c.NumeroCarga,
    c.PesoKg,
    c.DataMaximaDesembarque,
    n.NomeNavio,
    n.NumeroNavio,
    p.NomePorto AS PortoDestino,
    ar.NomeAgente AS AgenteReceptor
FROM Carga c
```

```
LEFT JOIN Navio n ON c.NavioId = n.NavioId
LEFT JOIN Porto p ON c.PortoDestinoId = p.PortoId
LEFT JOIN AgenteReceptor ar ON c.AgenteReceptorId =
ar.AgenteReceptorId;
```

4.3 Trigger

Trigger: Atualizar carga atual dos navios
Essa trigger é executada após inserções, atualizações ou exclusões na tabela `Carga`, e atualiza o campo `CapacidadeKg` do navio com a soma atual de cargas atribuídas a ele (nota: você pode querer criar um novo campo como `CargaAtualKg` em vez de sobrescrever `CapacidadeKg`).

Sql:

```
CREATE TRIGGER [dbo].[trg_AtualizarCargaAtualNavio]
ON [dbo].[Carga]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @NaviosAfetados TABLE (NavioId INT);

    INSERT INTO @NaviosAfetados (NavioId)
    SELECT DISTINCT NavioId FROM inserted WHERE NavioId IS NOT
NULL
    UNION
    SELECT DISTINCT NavioId FROM deleted WHERE NavioId IS NOT
NULL;

    UPDATE n
    SET n.CapacidadeKg = ISNULL((
        SELECT SUM(c.PesoKg)
        FROM Carga c
        WHERE c.NavioId = n.NavioId
    ), 0)
    FROM Navio n
    INNER JOIN @NaviosAfetados na ON na.NavioId = n.NavioId;
END;
```