**Documento do Implementação Técnica: DIT-2022-00001**

**Desafio Acelera Jovem SQL – Elias Gabriel**

|  |  |  |  |
| --- | --- | --- | --- |
| Histórico de Alterações | | | |
|  |  |  |  |
| Versão | **Data** | **Descrição** | **Realizado Por** |
| 1.0 | 28/01/2022 | Criação do Documento | Elias Gabriel |
| 1.1 | 11/02/2022 | Elaboração do Documento | Elias Gabriel |
| 1.2 | 22/02/2022 | Finalização do Documento | Elias Gabriel |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sumário

[Sumário 2](#_Toc96421026)

[**1.** **OBJETIVO** 3](#_Toc96421027)

[**2.** **MODELO DE NEGÓCIO** 3](#_Toc96421028)

[**3.** **FERRAMENTAS UTILIZADAS** 3](#_Toc96421029)

[**4.** **MODELO DE CONCEITUAL** 4](#_Toc96421030)

[**5.** **MODELO DE LÓGICO** 4](#_Toc96421031)

[**6.** **MODELO DE FÍSICO** 4](#_Toc96421032)

[**7.** **ESCOPO DA ENTREGA** 5](#_Toc96421033)

[**7.1** **Estrutura dos relacionamentos** 5](#_Toc96421042)

[**7.2** **SQL DDL** 5](#_Toc96421043)

[**7.2.1** **Tabela de Editoras** 5](#_Toc96421044)

[**7.2.2** **Tabela de Livros** 5](#_Toc96421045)

[**7.2.3** **Tabela de Autores** 5](#_Toc96421046)

[**7.2.4** **Tabela Associativa Livro-Autor** 5](#_Toc96421047)

[**7.2.5** **Tabela Associativa Livro-Editora** 6](#_Toc96421048)

[**7.2.6** **Tabela Histórica Alteração\_Editora** 6](#_Toc96421049)

[**7.2.7** **Detalhamento das Restrições de Domínio** 6](#_Toc96421050)

[**7.3** **SQL DML - Inserts** 6](#_Toc96421051)

[**7.3.1** **Amostra Editora** 6](#_Toc96421052)

[**7.3.2** **Amostra Livro** 6](#_Toc96421053)

[**7.3.3** **Amostra Autor** 6](#_Toc96421054)

[**7.3.4** **Amostra Livro\_Autor** 7](#_Toc96421055)

[**7.3.5** **Amostra Livro\_Editora** 7](#_Toc96421056)

[**7.4** **Queries** 7](#_Toc96421057)

[**7.4.1** **Queries Simples** 7](#_Toc96421058)

[**7.4.2** **Queries com Filtro** 7](#_Toc96421059)

[**7.4.3** **Queries com Joins** 7](#_Toc96421060)

[**7.4.4** **Queries com Agregação** 8](#_Toc96421061)

[**7.4.5** **Queries com Funções SQL** 9](#_Toc96421062)

[**7.4.6** **VIEWS** 9](#_Toc96421063)

[**7.5** **Procedures** 10](#_Toc96421064)

[**7.6** **User Functions** 14](#_Toc96421065)

[**7.7** **Trigger** 17](#_Toc96421066)

1. **OBJETIVO**

Este documento tem como finalidade descrever detalhadamente cada etapa do projeto, desde a sua estruturação até suas queries complexas. Cada etapa será explicada detalhadamente e porque foi feita de tal maneira.

1. **MODELO DE NEGÓCIO**

Uma livraria mantém o cadastro de livros disponíveis para a venda. Para cada livro são armazenados código, nome, língua e ano em que foi escrito. Para os autores é mantido igualmente um cadastro que inclui nome, data de nascimento, pais de nascimento e uma breve nota biográfica.

Cada livro pode ter vários autores e para um mesmo autor podem existir vários livros cadastrados. Um autor pode estar incluído no cadastro ainda quando não exista um livro seu para venda.

As editoras são incluídas no cadastro a partir do seu nome, endereço, telefone. Uma editora pode estar cadastrada mesmo quando não existam livros editados por ela em venda.

Para um mesmo livro podem existir várias edições realizadas por editoras diferentes ou em anos diferentes. Cada edição tem um código (ISBN), preço, ano, número de páginas e quantidade em estoque. Considere que um livro pode ser cadastrado se existe pelo menos uma edição do mesmo para venda.

1. **FERRAMENTAS UTILIZADAS**

* Azure SQL
* Azure Data Studio
* Git e Github
* BrModelos

1. Diagram

   Description automatically generated**MODELO DE CONCEITUAL**
2. **MODELO DE LÓGICO**

Diagram

Description automatically generated

1. **MODELO DE FÍSICO**

O script do projeto está localizado em:

/c/Users/elias/Desktop/Dataside/Desafio Acelera/ ou no [Github](https://github.com/elias-gabriel/DesafioAceleraSQL).

1. **ESCOPO DA ENTREGA**

Neste projeto será entregue um banco de dados para cadastramento de livros, editoras e autores.

Também será entregue queries complexas com indicação de situações que podem ocorrer e que podem ser úteis ao seu negócio.

7. 2. **Estrutura dos relacionamentos**

* As tabelas Autor e Livro se relacionam, assim formando uma nova tabela, a Livro\_Autor. Assim, evitando duplicidade na hora de cadastrar livros com o mesmo autor e facilitando as consultas com JOINS.
* As tabelas Livro e Editora se relacionam, assim formando uma nova tabela, a Livro\_Editora, evitando duplicidade na hora de cadastrar vários livros para a mesma editora e facilitando as consultas com JOINS.
  1. **SQL DDL**

A seguir, será mostrado como foram criadas as tabelas dentro do SQL, mostrando suas tipagens e outros parâmetros (PRIMARY, FOREIGN KEY.)

* + 1. **Tabela de Editoras**

create table editora (

    ID\_Editora INT IDENTITY PRIMARY KEY NOT NULL,

    Nome\_Editora varchar(50) NOT NULL,

    Pais varchar(30) NOT NULL,

    Endereço varchar(120) NULL,

    Telefone varchar(16) NULL

);

* + 1. **Tabela de Livros**

create table livro (

     ID\_Livro int IDENTITY PRIMARY KEY Not Null,

     Nome\_Livro varchar(100) NOT NULL,

     Idioma varchar(30) NOT NULL,

     Ano\_Lançamento SMALLINT NOT NULL

);

* + 1. **Tabela de Autores**

create table autor (

     ID\_Autor int IDENTITY PRIMARY KEY NOT NULL,

     Nome\_Autor varchar(30) NOT NULL,

     Nacionalidade varchar(30),

     Data\_Nascimento date NULL,

     Biografia varchar(600) NULL,

     );

* + 1. **Tabela Associativa Livro-Autor**

create table livro\_autor (

     ID\_Livro int NOT NULL, FOREIGN KEY (ID\_Livro) REFERENCES livro(ID\_Livro),

     ID\_Autor INT NOT NULL, FOREIGN KEY (ID\_Autor) REFERENCES autor(ID\_Autor)

);

* + 1. **Tabela Associativa Livro-Editora**

create table livro\_editora (

     ID\_Livro int NOT NULL, FOREIGN KEY (ID\_Livro) REFERENCES livro(ID\_Livro),

     ID\_Editora INT NOT NULL, FOREIGN KEY (ID\_Editora) REFERENCES editora(ID\_Editora),

     N\_Paginas int NOT NULL,

     ISBN\_10 VARCHAR(14) UNIQUE NOT NULL,

     Valor MONEY NOT NULL,

     Estoque INT NOT NULL

     );

* + 1. **Tabela Histórica Alteração\_Editora**

    CREATE TABLE hist\_editora (

        ID INT

        ,Nome VARCHAR(50)

      ,Pais VARCHAR(30) NOT NULL

        ,Endereço VARCHAR(120) NULL

        ,Telefone VARCHAR(16) NULL

        ,Ação VARCHAR(12)

       ,Data\_Modi DATETIME

        )

* + 1. **Detalhamento das Restrições de Domínio**

Primary Key: Valores únicos da Coluna

Foreign Key: Referência de uma coluna a outra tabela que possua Primary Key em alguma de suas colunas.

Unique: Evita que os dados inseridos não estejam duplicados em uma coluna

Not Null: Garante que uma coluna não admite valores NULL. Isto significa que será abortada uma operação de INSERT ou UPDATE que coloque um valor NULL nessa coluna.

* 1. **SQL DML - Inserts**

A seguir, será mostrado algumas amostras de Inserts com fins de enriquecimento das tabelas citadas acima.

* + 1. **Amostra Editora**

Amostra de Insert da tabela ‘editora’, seguindo a ordem de ID, Nome, País de Atuação, Endereço e Telefone.

INSERT INTO editora VALUES ('Penguin Books', 'Britânica', 'Londres, Reino Unido', '1-800-733-3000');

* + 1. **Amostra Livro**

Amostra de Insert da tabela ‘livro’, seguindo a ordem de ID, Nome, Idioma e Ano de Lançamento.

INSERT INTO livro VALUES ('Fortaleza Digital', 'Português - BR', '2005');

* + 1. **Amostra Autor**

Amostra de Insert da tabela ‘autor’, seguindo a ordem de ID, Nome, Nacionalidade, Data de Nascimento e Biografia.

INSERT INTO autor VALUES ('Machado de Assis', 'Brasileiro', '1839/06/21', 'Foi um escritor brasileiro, considerado por muitos críticos.')

* + 1. **Amostra Livro\_Autor**

Amostra de Insert da tabela ‘livro\_autor’, seguindo a ordem de ID\_Livro e ID\_Autor.

INSERT INTO livro\_autor VALUES ('1', '1')

* + 1. **Amostra Livro\_Editora**

Amostra de Insert da tabela ‘livro\_autor’, seguindo a ordem de ID\_Livro, ID\_Editora, Número de Páginas, Código ISBN, Valor e Quantidade em Estoque.

INSERT INTO livro\_editora VALUES ('1', '1', '128', '9781490517063', '19.95', '4'),

* 1. **Queries**

A seguir, será mostrado algumas queries e suas descrições, com a finalidade de possíveis relatórios.

* + 1. **Queries Simples**

-- Livros que tenham a palavra toolkit

SELECT Nome\_Livro from livro where Nome\_Livro LIKE '%toolkit%'

-- União de Autores e Livros em uma única coluna

SELECT Nome\_Autor

FROM autor

UNION

SELECT Nome\_Livro

FROM livro

* + 1. **Queries com Filtro**

-- Consultar Livros que Possuem entre 300 e 800 páginas e que sejam da editora Penguin Books ou possuam o idioma Inglês

SELECT e.Nome\_Editora

    ,l.Nome\_Livro

    ,l.Idioma

    ,le.N\_Paginas

FROM livro\_editora le

INNER JOIN livro l

    ON le.ID\_Livro = l.ID\_Livro

INNER JOIN editora e

    ON le.ID\_Editora = e.ID\_Editora

WHERE N\_Paginas BETWEEN 300

        AND 800

    AND e.Nome\_Editora = 'Penguin Books'

    OR l.Idioma LIKE '%Inglês%'

ORDER BY le.N\_Paginas

* + 1. **Queries com Joins**

-- Consultar Nome do Livro, Autor e Editora onde possuem \*todos os dados cadastrados

SELECT Nome\_Livro

    ,Nome\_Autor

    ,Nome\_Editora

FROM livro\_autor la

INNER JOIN livro\_editora le

    ON la.ID\_Livro = le.ID\_Livro

INNER JOIN livro l

    ON le.ID\_Livro = l.ID\_Livro

INNER JOIN autor a

    ON la.ID\_Autor = a.ID\_Autor

INNER JOIN editora e

    ON le.ID\_Editora = e.ID\_Editora

-- Listar livros com pouco estoque (livros com menos de 10 cópias em estoque)

SELECT l.Nome\_Livro

    ,le.Estoque

FROM livro\_editora le

INNER JOIN livro l

    ON le.ID\_Livro = l.ID\_Livro

WHERE le.Estoque <= 10

ORDER BY le.Estoque

* + 1. **Queries com Agregação**

-- Total de Livros por Editora

SELECT e.Nome\_Editora

    ,count(e.Nome\_Editora) AS QTD\_Livros

FROM editora e

FULL JOIN livro\_editora le

    ON e.ID\_Editora = le.ID\_Editora

FULL JOIN livro l

    ON le.ID\_Livro = l.ID\_Livro

WHERE e.Nome\_Editora IS NOT NULL

GROUP BY e.Nome\_Editora

-- Total de Livros por Autor

SELECT a.Nome\_Autor

    ,count(a.Nome\_Autor) AS QTD\_Livros

FROM autor a

FULL JOIN livro\_autor la

    ON a.ID\_Autor = la.ID\_Autor

FULL JOIN livro l

    ON la.ID\_Livro = l.ID\_Livro

WHERE a.Nome\_Autor IS NOT NULL

GROUP BY a.Nome\_Autor

-- Estoque Total de Livros

SELECT sum(Estoque) AS Estoque

FROM livro\_editora

-- Média de Valor dos Livros

SELECT avg(Valor) AS Valor\_Médio

FROM livro\_editora

-- Quantidade de livros em estoque, agrupado por Editora

SELECT e.Nome\_Editora

    ,SUM(le.Estoque) AS Estoque

FROM livro\_editora le

INNER JOIN editora e

    ON le.ID\_Editora = e.ID\_Editora

GROUP BY e.Nome\_Editora

* + 1. **Queries com Funções SQL**

-- Idade dos Autores

DECLARE @dt DATE = CONVERT(DATE, getdate())

SELECT Nome\_Autor

    ,DATEDIFF(year, Data\_Nascimento, @dt) AS Idade

FROM autor

WHERE Data\_Nascimento IS NOT NULL

-- Converter Data de Nascimento dos Autor para o Padrão BR (dd/mm/yyy)

SELECT Nome\_Autor

    ,CONVERT(VARCHAR, Data\_Nascimento, 103) AS Data\_NascimentoBR

FROM autor

WHERE Data\_Nascimento IS NOT NULL

-- Formatar Valor do Livro para o Padrão BR

SELECT format(le.valor, 'C', 'pt-BR') AS Formato\_BR

FROM livro\_editora le

INNER JOIN livro l

    ON le.ID\_Livro = l.ID\_Livro

-- Tamanho da Biografia de cada Autor

SELECT Nome\_Autor

    ,len(Biografia) AS TM\_BIO

FROM autor

WHERE Biografia IS NOT NULL

* + 1. **VIEWS**

-- Consultar Editoras, Autores e Livros que não possuem informações completas

CREATE VIEW vw\_EAU

AS

SELECT DISTINCT e.Nome\_Editora

    ,a.Nome\_Autor

    ,l.Nome\_Livro

    ,le.ISBN\_10

FROM livro\_autor la

FULL JOIN livro\_editora le

    ON la.ID\_Livro = le.ID\_Livro

FULL JOIN autor a

    ON la.ID\_Autor = a.ID\_Autor

FULL JOIN livro l

    ON la.ID\_Livro = l.ID\_Livro

FULL JOIN editora e

    ON le.ID\_Editora = e.ID\_Editora

WHERE le.ISBN\_10 IS NULL

GO

SELECT \*

FROM vw\_EAU

GO

-- Editoras e seus Autores Vinculados

CREATE VIEW vw\_EditoraAutor

AS

SELECT DISTINCT e.Nome\_Editora

    ,a.Nome\_Autor

FROM autor a

INNER JOIN livro\_autor la

    ON a.ID\_Autor = la.ID\_Autor

INNER JOIN livro\_editora le

    ON la.ID\_Livro = le.ID\_Livro

INNER JOIN editora e

    ON le.ID\_Editora = e.ID\_Editora

GO

SELECT \*

FROM vw\_EditoraAutor

GO

-- Livros e seus Estoques Atualizados

CREATE VIEW vw\_estoque

AS

SELECT l.Nome\_Livro

    ,le.Estoque

    ,le.Valor

FROM livro\_editora le

INNER JOIN livro l

    ON le.ID\_Livro = l.ID\_Livro

ORDER BY le.Estoque

GO

SELECT \*

FROM vw\_estoque

GO

* 1. **Procedures**

- Procedure: Atualizar valor dos livros de certa editora dando um aumento a partir de uma certa porcentagem

    GO

    CREATE PROCEDURE pr\_AjusteValor (

        @idE INT

        ,@porcent DECIMAL

        )

    AS

    SELECT l.Nome\_Livro

        ,e.Nome\_Editora

        ,le.Valor AS Preço\_Antigo

        ,(@porcent / 100) \* Valor + Valor AS Preço\_Novo

    FROM livro\_editora le

    INNER JOIN livro l

        ON le.ID\_Livro = l.ID\_Livro

    INNER JOIN editora e

        ON le.ID\_Editora = e.ID\_Editora

    WHERE e.ID\_Editora = @idE

    UPDATE livro\_editora

    SET Valor = (@porcent / 100) \* Valor - Valor

    WHERE ID\_Editora = @idE

    EXEC pr\_AjusteValor @porcent = 50

        ,@idE = 4

-- Procedure: Atualizar estoque de livros de uma editora a partir do seu ID

    GO

    CREATE PROCEDURE pr\_AjutaEst (

        @idE INT

        ,@est int

        )

    AS

    SELECT l.Nome\_Livro

        ,e.Nome\_Editora

        ,le.Estoque as Estoque\_Antigo

        ,@est + Estoque AS Estoque\_Novo

    FROM livro\_editora le

    INNER JOIN livro l

        ON le.ID\_Livro = l.ID\_Livro

    INNER JOIN editora e

        ON le.ID\_Editora = e.ID\_Editora

    WHERE e.ID\_Editora = @idE

    UPDATE livro\_editora

    SET Estoque = Estoque + @est

    WHERE ID\_Editora = @idE

    EXEC pr\_AjutaEst @est = 5

        ,@idE = 4

-- Procedure: Inserir Livros na tabela 'livro' a partir dos parâmetros

    GO

    CREATE PROCEDURE pr\_Livro (

        @nomel VARCHAR(100)

        ,@idiomal VARCHAR(30)

        ,@data\_lancl SMALLINT

        )

    AS

    BEGIN

        INSERT INTO livro (

            Nome\_Livro

            ,Idioma

            ,Ano\_Lançamento

            )

        VALUES (

            @nomel

            ,@idiomal

            ,@data\_lancl

            )

        select IDENT\_CURRENT('livro') AS IDLivroCadastrado

    END

    GO

    EXEC pr\_Livro @nomel = 'Código Limpo'

        ,@idiomal = 'Português - BR'

        ,@data\_lancl = 2009

    DECLARE @pr\_idL int

    set @pr\_idL = IDENT\_CURRENT('livro')

-- Procedure: Inserir Editoras na tabela 'editora' a partir dos parâmetros

        GO

    CREATE PROCEDURE pr\_Editora (

        @nomeE VARCHAR(50)

        ,@paisE VARCHAR(30)

        ,@endE VARCHAR(120)

        ,@telE VARCHAR(16)

        )

    AS

    BEGIN

        INSERT INTO editora (

            Nome\_Editora

            ,Pais

            ,Endereço

            ,Telefone

            )

        VALUES (

            @nomeE

            ,@paisE

            ,@endE

            ,@telE

            )

        SELECT IDENT\_CURRENT('editora') AS IDEditoraCadastrada

    END

    GO

    EXEC pr\_Editora @nomeE = 'Suhrkamp Verlag'

        ,@paisE = 'Alemã'

        ,@endE = 'Berlin, Berlin, 10437'

        ,@telE = '+49 30 740744-0'

-- Procedure: Inserir Autores na tabela 'autor' a partir dos parâmetros

    GO

    CREATE PROCEDURE pr\_Autor (

        @nomeA VARCHAR(50)

        ,@nacio VARCHAR(30)

        ,@data\_nas date

        ,@bio VARCHAR(600)

        )

    AS

    BEGIN

        INSERT INTO autor (

            Nome\_Autor

            ,Nacionalidade

            ,Data\_Nascimento

            ,Biografia

            )

        VALUES (

            @nomeA

            ,@nacio

            ,@data\_nas

            ,@bio

            )

        SELECT IDENT\_CURRENT('autor') AS IDAutorCadastrado

    END

    GO

    EXEC pr\_Autor @nomeA = 'Franz Kafka'

        ,@nacio = 'Alemão'

        --YYYY/MM/DD

        ,@data\_nas = '1883/07/03'

        ,@bio = null

    DECLARE @pr\_idA int

    SET @pr\_idA = IDENT\_CURRENT('autor')

-- Procedure: Inserir o ID do Livro e Autor cadastrados em sequência na tabela livro\_autor v1.0

    GO

    CREATE PROCEDURE pr\_LivroAutor

    AS

    BEGIN

        EXEC pr\_Livro @nomel = 'Código Limpo'

            ,@idiomal = 'Português - BR'

            ,@data\_lancl = 2009

        DECLARE @pr\_idL INT

        SET @pr\_idL = IDENT\_CURRENT('livro')

        SELECT @pr\_idL

        EXEC pr\_Autor @nomeA = 'Franz Kafka'

            ,@nacio = 'Alemão'

            --YYYY/MM/DD

            ,@data\_nas = '1883/07/03'

            ,@bio = NULL

        DECLARE @pr\_idA INT

        SET @pr\_idA = IDENT\_CURRENT('autor')

        SELECT @pr\_idA

        INSERT INTO livro\_autor (

            ID\_Livro

            ,ID\_Autor

            )

        VALUES (

            @pr\_idL

            ,@pr\_idA

            )

    END

    GO

* 1. **User Functions**

-- Função Table Valued que retorna as principais informações das tabelas

    CREATE FUNCTION fn\_Multi\_tb ()

    RETURNS @valores TABLE (

        Nome\_Livro VARCHAR(100)

        ,Idioma VARCHAR(30)

        ,AnoLançamento SMALLINT

        ,NumeroPaginas INT

        ,Autor VARCHAR(30)

        ,Editora VARCHAR(50)

        ,Valor CHAR(10)

        ,ISBN10 VARCHAR(16) UNIQUE NOT NULL

        ,Estoque INT

        )

    AS

    BEGIN

        INSERT @valores (

            Nome\_Livro

            ,Idioma

            ,AnoLançamento

            ,NumeroPaginas

            ,Autor

            ,Editora

            ,Valor

            ,ISBN10

            ,Estoque

            )

        SELECT l.Nome\_Livro

            ,l.Idioma

            ,l.Ano\_Lançamento

            ,le.N\_Paginas

            ,a.Nome\_Autor

            ,e.Nome\_Editora

            ,format(le.Valor, 'c', 'pt-br')

            ,le.ISBN\_10

            ,le.Estoque

        FROM livro\_editora le

        INNER JOIN livro\_autor la

            ON le.ID\_Livro = la.ID\_Livro

        INNER JOIN editora e

            ON le.ID\_Livro = e.ID\_Editora

        INNER JOIN autor a

            ON la.ID\_Autor = a.ID\_Autor

        INNER JOIN livro l

            ON la.ID\_Livro = l.ID\_Livro

        RETURN

    END

    GO

    SELECT \*

    FROM fn\_Multi\_tb()

--

-- Função Scalar que retorna a soma dos valores de todos os livros de um certo autor

    CREATE FUNCTION fn\_ValorTotal (@nome VARCHAR(30))

    RETURNS INT

    AS

    BEGIN

        DECLARE @cons REAL

        SELECT @cons = sum(Valor)

        FROM livro\_editora le

        INNER JOIN livro\_autor la

            ON le.ID\_Livro = la.ID\_Livro

        INNER JOIN autor a

            ON la.ID\_Autor = a.ID\_Autor

        INNER JOIN livro l

            ON la.ID\_Livro = l.ID\_Livro

        WHERE a.Nome\_Autor = @nome

        RETURN @cons

    END

    GO

    SELECT dbo.fn\_ValorTotal('Stephen Hawking')

--

-- Função Table Valued que retorna todos os livros que foram escritos depois de um certo ano

    CREATE FUNCTION fn\_Ano (@ano SMALLINT)

    RETURNS TABLE

    AS

    RETURN (

            SELECT \*

            FROM livro

            WHERE Ano\_Lançamento > @ano)

    GO

    SELECT \* FROM fn\_Ano(2020)

-- Função Table Valued que retorna todas as informações de um livro a partir do seu id

    GO

    CREATE FUNCTION fn\_InfoLivro (@id SMALLINT)

    RETURNS TABLE

    AS

    RETURN (

            SELECT l.Nome\_Livro

                ,l.Idioma

                ,l.Ano\_Lançamento

                ,le.N\_Paginas

                ,le.ISBN\_10

                ,format(le.Valor, 'c', 'pt-br') AS Preço\_BR

                ,a.Nome\_Autor

                ,a.Nacionalidade

                ,a.Data\_Nascimento

                ,e.Nome\_Editora

                ,e.Pais AS Pais\_Editora

                ,e.Endereço

                ,e.Telefone

            FROM livro l

            INNER JOIN livro\_editora le

                ON l.ID\_Livro = le.ID\_Livro

            INNER JOIN livro\_autor la

                ON le.ID\_Livro = la.ID\_Livro

            INNER JOIN autor a

                ON la.ID\_Autor = a.ID\_Autor

            INNER JOIN editora e

                ON le.ID\_Editora = e.ID\_Editora

            WHERE l.ID\_Livro = @id

            )

    GO

    SELECT \*

    FROM fn\_InfoLivro(14)

--

-- Função Table Valued que retorna todos os livros que sejam abaixo de um determinado preço

    GO

    CREATE FUNCTION fn\_Valor (@valor MONEY)

    RETURNS TABLE

    AS

    RETURN (

            SELECT l.Nome\_Livro AS Livro

                ,format(le.Valor, 'c', 'pt-br') AS Preço

            FROM livro l

            INNER JOIN livro\_editora le

                ON l.ID\_Livro = le.ID\_Livro

            WHERE le.Valor < @valor

            )

    GO

    SELECT \* FROM fn\_Valor(22)

-- Função Table Valued que retona o preço total por quantidade de livros desejadar a partir do ID do Livro

    GO

    CREATE FUNCTION fn\_ValorQuantiaLivro (

        @idL MONEY

        ,@Quantidade SMALLINT

        )

    RETURNS TABLE

    AS

    RETURN (

            SELECT l.Nome\_Livro AS Livro

                ,format(le.Valor, 'c', 'pt-br') AS Valor\_Unitário

                ,le.Valor \* @Quantidade AS PreçoTotal

            FROM livro l

            INNER JOIN livro\_editora le

                ON l.ID\_Livro = le.ID\_Livro

            WHERE l.ID\_Livro = @idL

            )

    GO

    -- Parametros: ID Livro, Quantia desejada

    SELECT \*

    FROM fn\_ValorQuantiaLivro(1, 7)

--

-- Função Scalar que retorna o peso de um livro a partir do número de paginas. Métrica usada: A6

    GO

    CREATE FUNCTION fn\_Peso (@nomeL VARCHAR(30))

    RETURNS DECIMAL

    AS

    BEGIN

        DECLARE @cons DECIMAL

        SELECT @cons = N\_Paginas \* 1.90

        FROM livro\_editora le

        INNER JOIN livro\_autor la

            ON le.ID\_Livro = la.ID\_Livro

        INNER JOIN autor a

            ON la.ID\_Autor = a.ID\_Autor

        INNER JOIN livro l

            ON la.ID\_Livro = l.ID\_Livro

        WHERE l.Nome\_Livro = @nomeL

        RETURN @cons

    END

    GO

    SELECT dbo.fn\_Peso('Hamlet')

* 1. **Trigger**

- Trigger com finalidade de registrar o que ocorre na tabela 'editora', seja uma inserção, atualização ou deleção. Esse histórico fica registrado na tabela 'hist\_editora'.

CREATE TRIGGER Alteração\_Editora ON editora

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

    INSERT INTO hist\_editora (

        ID

        ,Nome

        ,Pais

        ,Endereço

        ,Telefone

        ,Ação

        ,Data\_Modi )

    SELECT ID\_Editora

        ,Nome\_Editora

        ,Pais

        ,Endereço

        ,Telefone

        ,'INSERT'

        ,GETDATE()

    FROM inserted

    INSERT INTO hist\_editora (

        ID

        ,Nome

        ,Pais

        ,Endereço

        ,Telefone

        ,Ação

        ,Data\_Modi

        )

    SELECT ID\_Editora

        ,Nome\_Editora

        ,Pais

        ,Endereço

        ,Telefone

        ,'UPDATE'

        ,GETDATE()

    FROM deleted

        INSERT INTO hist\_editora (

        ID

        ,Nome

        ,Pais

        ,Endereço

        ,Telefone

        ,Ação

        ,Data\_Modi

        )

    SELECT ID\_Editora

        ,Nome\_Editora

        ,Pais

        ,Endereço

        ,Telefone

        ,'DELETED'

        ,GETDATE()

    FROM deleted

END