Nome: Erick Daniel Teixeira Vier **RA:** 00235908

1 - O que são comandos DDL em banco de dados?

Comandos DDL em bancos de dados são usados para criar, modificar ou excluir objetos, como tabelas e índices. Eles ajudam a definir como os dados serão armazenados. Por exemplo, o comando CREATE cria um objeto, o comando ALTER modifica um objeto existente e o comando DROP exclui um objeto. Esses comandos são usados por administradores de banco de dados para organizar e manter a estrutura dos dados.

2 - O que são comandos DML em bancos dados?

Comandos DML (Data Manipulation Language) em bancos de dados são usados para manipular os dados. Eles permitem adicionar, modificar, recuperar e excluir registros em tabelas. INSERT: adiciona novos registros. UPDATE: modifica registros existentes. SELECT: recupera registros com base em critérios. DELETE: exclui registros. Esses comandos DML são usados para interagir com os dados em um banco de dados, realizando operações de manipulação nos registros.

3 - Explique o conceito de cardinalidade de relacionamentos.

A cardinalidade de relacionamentos em bancos de dados refere-se à quantidade de ocorrências (ou instâncias) de uma entidade que podem estar associadas a outra entidade por meio de um relacionamento. Ela descreve a natureza e a quantidade de correspondências entre as entidades relacionadas.

A cardinalidade de um relacionamento pode ser classificada em três tipos principais:

Um para Um (1:1): Nesse tipo de cardinalidade, cada instância de uma entidade está associada a apenas uma instância da outra entidade, e vice-versa. É como se existisse uma correspondência exata entre as entidades.

Um para Muitos (1:N): Nesse tipo de cardinalidade, cada instância de uma entidade está associada a várias instâncias da outra entidade, enquanto cada instância da outra entidade está associada a apenas uma instância da primeira entidade. É como se uma entidade "pai" tivesse vários relacionamentos com várias entidades "filhas".

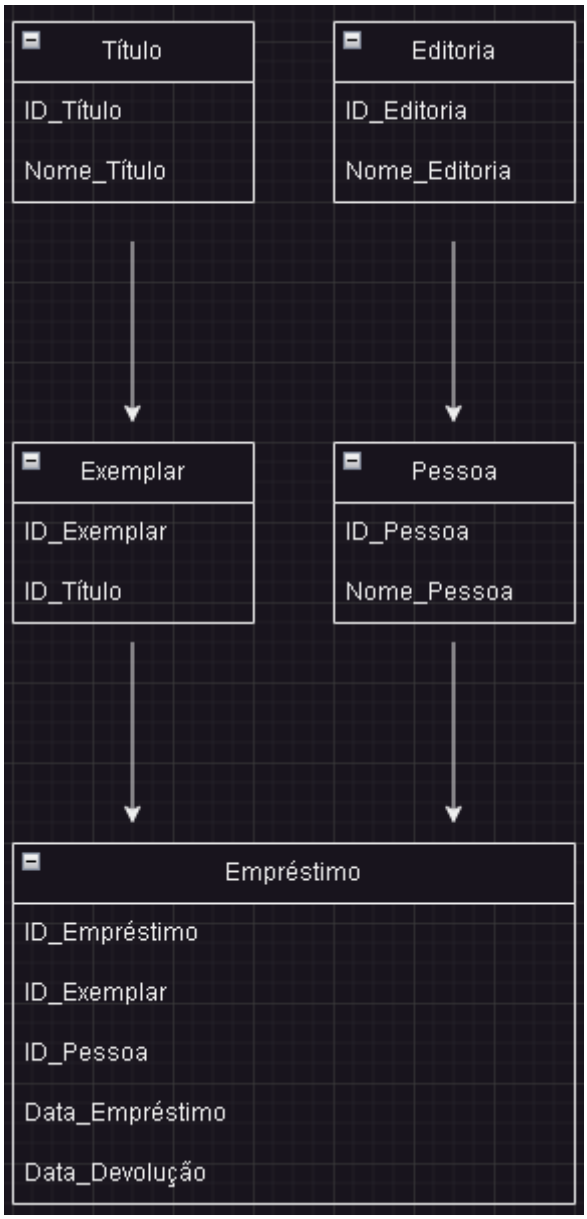
Muitos para Muitos (N:N): Nesse tipo de cardinalidade, várias instâncias de uma entidade estão associadas a várias instâncias da outra entidade. Essa cardinalidade é representada usando uma tabela de associação, que registra as correspondências entre as entidades.

A cardinalidade é uma parte importante do design de banco de dados, pois influencia a forma como as entidades são modeladas e como os relacionamentos são estabelecidos entre elas. Entender a cardinalidade de um relacionamento ajuda a definir as restrições e as regras de integridade referencial adequadas ao banco de dados.

4 - Crie uma estrutura (diagrama) de um banco de dados para controle de biblioteca, se atentar para controlar títulos, editoras, quantidade de exemplares, pessoas e datas.

Link do diagrama:

https://drive.google.com/file/d/1smWu9nyXMC1tYrchYd-1A-nQzWq9UW-C/view?usp=sharing



5 - Crie o banco de dados referente ao diagrama do exercício anterior.

```sql
CREATE TABLE Titulo (
    ID_Titulo INT PRIMARY KEY,
    Nome_Titulo VARCHAR(255)
);

CREATE TABLE Editora (
    ID_Editora INT PRIMARY KEY,
    Nome_Editora VARCHAR(255)
);

CREATE TABLE Exemplar (
    ID_Exemplar INT PRIMARY KEY,
    ID_Titulo INT,
    FOREIGN KEY (ID_Titulo) REFERENCES Titulo(ID_Titulo)
);

CREATE TABLE Pessoa (
    ID_Pessoa INT PRIMARY KEY,
    Nome_Pessoa VARCHAR(255)
);

CREATE TABLE Emprestimo (
    ID_Empréstimo INT PRIMARY KEY,
    ID_Exemplar INT,
    ID_Pessoa INT,
    Data_Emprestimo DATE,
    Data_Devolucao DATE,
    FOREIGN KEY (ID_Exemplar) REFERENCES Exemplar(ID_Exemplar),
    FOREIGN KEY (ID_Pessoa) REFERENCES Pessoa(ID_Pessoa)
);
```

Link do código escrito:

https://github.com/ErickDaniel7/trabalho-bd/blob/main/create-table

6 - Insira dados de 15 livros.

```
INSERT INTO Titulo (ID_Titulo, Nome_Titulo) VALUES
    (1, 'Livro A'),
    (2, 'Livro B'),
    (3, 'Livro C'),
    (4, 'Livro D'),
    (5, 'Livro E'),
    (6, 'Livro F'),
    (7, 'Livro G'),
    (8, 'Livro H'),
    (9, 'Livro I'),
    (10, 'Livro J'),
    (11, 'Livro K'),
    (12, 'Livro L'),
    (13, 'Livro M'),
    (14, 'Livro N'),
    (15, 'Livro O');
```

Link do código escrito:

https://github.com/ErickDaniel7/trabalho-bd/blob/main/insercao-dados

7 - Insira 10 pessoas.

```
INSERT INTO Pessoa (ID_Pessoa, Nome_Pessoa) VALUES
    (1, 'Rafaela'),
    (2, 'Sofia'),
    (3, 'Pedro'),
    (4, 'Marcus'),
    (5, 'Matheus'),
    (6, 'Maria'),
    (7, 'Guilherme'),
    (8, 'Bruno'),
    (9, 'Milena'),
    (10, 'Camila');
```

Link do código escrito:

https://github.com/ErickDaniel7/trabalho-bd/blob/main/insercao-pessoa

8 - Crie uma trigger que quando um livro for emprestado, um registro com data de saída e previsão de devolução seja inserido na tabela Log_emprestimo.

```
CREATE TABLE IF NOT EXISTS Log_emprestimo (
    ID_Empréstimo INT,
    Data_Saída DATE,
    Previsão_Devolução DATE
);

CREATE TRIGGER inserir_log_emprestimo AFTER INSERT ON Empréstimo
FOR EACH ROW
BEGIN
    INSERT INTO Log_emprestimo (ID_Empréstimo, Data_Saída, Previsão_Devolução)
    VALUES (NEW.ID_Empréstimo, NEW.Data_Emprestimo, NEW.Data_Devolucao);
END;
```

Link do código escrito:

https://github.com/ErickDaniel7/trabalho-bd/blob/main/tirgger-log_emprestimo

9 - Crie uma trigger que quando o livro for devolvido, a tabela Log_emprestimo seja alterada com uma marcação para devolvido.

```
CREATE TRIGGER marcar_devolucao AFTER UPDATE ON Empréstimo
FOR EACH ROW
BEGIN
    IF NEW.Data_Devolucao IS NOT NULL THEN
        UPDATE Log_emprestimo
        SET Devolvido = 1
        WHERE ID_Empréstimo = NEW.ID_Empréstimo;
    END IF;
END;
```

Link do código escrito:

https://github.com/ErickDaniel7/trabalho-bd/blob/main/trigger-marcar_devolucao

10 - Apresente 5 consultas (sql) com diferentes usos de inner join e cláusulas where.

Link do código escrito:

https://github.com/ErickDaniel7/trabalho-bd/blob/main/consultas

```
SELECT Empréstimo.*
FROM Empréstimo
INNER JOIN Exemplar ON Empréstimo.ID_Exemplar = Exemplar.ID_Exemplar
INNER JOIN Título ON Exemplar.ID_Título = Título.ID_Título
WHERE Título.Nome_Título = 'Nome do Título';
```

```
SELECT Empréstimo.*
FROM Empréstimo
INNER JOIN Pessoa ON Empréstimo.ID_Pessoa = Pessoa.ID_Pessoa
WHERE Pessoa.Nome_Pessoa = 'Nome da Pessoa';
```

```
SELECT Título.Nome_Título, Empréstimo.Data_Devolucao
FROM Empréstimo
INNER JOIN Exemplar ON Empréstimo.ID_Exemplar = Exemplar.ID_Exemplar
INNER JOIN Título ON Exemplar.ID_Título = Título.ID_Título
INNER JOIN Pessoa ON Empréstimo.ID_Pessoa = Pessoa.ID_Pessoa
WHERE Pessoa.Nome_Pessoa = 'Nome da Pessoa';
```

```
SELECT Empréstimo.*
FROM Empréstimo
WHERE Empréstimo.Data_Devolucao < CURDATE();
```

```
SELECT Título.Nome_Título
FROM Título
INNER JOIN Exemplar ON Título.ID_Título = Exemplar.ID_Título
LEFT JOIN Empréstimo ON Exemplar.ID_Exemplar = Empréstimo.ID_Exemplar
WHERE Empréstimo.ID_Exemplar IS NULL;
```

Codigo Completo:

https://github.com/ErickDaniel7/trabalho-bd/blob/main/codigo-completo

```
CREATE TABLE Titulo (
    ID_Titulo INT PRIMARY KEY,
    Nome_Titulo VARCHAR(255)
);

CREATE TABLE Editora (
    ID_Editora INT PRIMARY KEY,
    Nome_Editora VARCHAR(255)
);

CREATE TABLE Exemplar (
    ID_Exemplar INT PRIMARY KEY,
    ID_Titulo INT,
    FOREIGN KEY (ID_Titulo) REFERENCES Titulo(ID_Titulo)
);

CREATE TABLE Pessoa (
    ID_Pessoa INT PRIMARY KEY,
    Nome_Pessoa VARCHAR(255)
);

CREATE TABLE Emprestimo (
    ID_Empréstimo INT PRIMARY KEY,
    ID_Exemplar INT,
    ID_Pessoa INT,
    Data_Emprestimo DATE,
    Data_Devolucao DATE,
    FOREIGN KEY (ID_Exemplar) REFERENCES Exemplar(ID_Exemplar),
    FOREIGN KEY (ID_Pessoa) REFERENCES Pessoa(ID_Pessoa)
);

INSERT INTO Titulo (ID_Titulo, Nome_Titulo) VALUES
    (1, 'Livro A'),
    (2, 'Livro B'),
    (3, 'Livro C'),
    (4, 'Livro D'),
    (5, 'Livro E'),
    (6, 'Livro F'),
    (7, 'Livro G'),
    (8, 'Livro H'),
    (9, 'Livro I'),
    (10, 'Livro J'),
    (11, 'Livro K'),
    (12, 'Livro L'),
    (13, 'Livro M'),
    (14, 'Livro N'),
    (15, 'Livro O');
```

```
INSERT INTO Pessoa (ID_Pessoa, Nome_Pessoa) VALUES
    (1, 'Rafaela'),
    (2, 'Sofia'),
    (3, 'Pedro'),
    (4, 'Marcus'),
    (5, 'Matheus'),
    (6, 'Maria'),
    (7, 'Guilherme'),
    (8, 'Bruno'),
    (9, 'Milena'),
    (10, 'Camila');

CREATE TABLE IF NOT EXISTS Log_emprestimo (
    ID_Empréstimo INT,
    Data_Saída DATE,
    Previsão_Devolução DATE
);

CREATE TRIGGER inserir_log_emprestimo AFTER INSERT ON Emprestimo
FOR EACH ROW
BEGIN
    INSERT INTO Log_emprestimo (ID_Empréstimo, Data_Saída, Previsão_Devolução)
    VALUES (NEW.ID_Empréstimo, NEW.Data_Emprestimo, NEW.Data_Devolucao);
END;

CREATE TRIGGER marcar_devolucao AFTER UPDATE ON Emprestimo
FOR EACH ROW
BEGIN
    IF NEW.Data_Devolucao IS NOT NULL THEN
        UPDATE Log_emprestimo
        SET Devolvido = 1
        WHERE ID_Empréstimo = NEW.ID_Empréstimo;
    END IF;
END;
```