1. Triggers de Validação de Datas

```
DELIMITER //

CREATE TRIGGER ValidarDatasAulaInsert
BEFORE INSERT ON Aula
FOR EACH ROW
BEGIN
    IF NEW.Data_Hora_inicio >= NEW.Data_Hora_fim THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: Data/hora de início deve ser anterior à data/hora de fim.';
    END IF;
END //

DELIMITER ;

-- Trigger para validação em UPDATE
DELIMITER //

CREATE TRIGGER ValidarDatasAulaUpdate
BEFORE UPDATE ON Aula
FOR EACH ROW
BEGIN
    IF NEW.Data_Hora_inicio >= NEW.Data_Hora_fim THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: Data/hora de início deve ser anterior à data/hora de fim.';
    END IF;
END //

DELIMITER ;
```

Exemplo de uso:

```
INSERT INTO Aula (Id_turma, Id_disciplina, Id_professor, Data_Hora_inicio, Data_Hora_fim)
VALUES (1, 1, 1, '2023-06-01 10:00:00', '2023-06-01 08:00:00');
```

```
46  21:22:33  INSERT INTO Aula (Id_turma, Id_disciplina, Id_professor, Data_Hora_inicio, Data_Hora_fim) VALUES (1, 1, 1, '2023-06-01 10:00:00', '2...  Error Code: 1644. Erro: Data/hora de início deve ser anterior à data/hora de fim.
```

As triggers ValidarDatasAulaInsert e ValidarDatasAulaUpdate atuam como "guardas" do banco de dados, garantindo que nunca seja possível cadastrar uma aula com data/hora de início posterior à data/hora de término. Elas são acionadas automaticamente antes de operações de INSERT e UPDATE na tabela Aula.

Benefícios:

Integridade dos dados: Garante consistência temporal nas marcações de aula

Prevenção de erros: Elimina a possibilidade de aulas com tempo negativo

Automático: Não requer intervenção manual do usuário

2. Stored Procedure AdicionarNovaAula

```
DELIMITER //

CREATE PROCEDURE AdicionarNovaAula(
    IN n_turma INT,
    IN n_disciplina INT,
    IN n_professor INT,
    IN n_data_hora_inicio DATETIME,
    IN n_data_hora_fim DATETIME
)
BEGIN
    DECLARE turma_exists INT;
    DECLARE disciplina_exists INT;
    DECLARE professor_exists INT;
    DECLARE overlap_exists INT;

    -- Verificar se a turma existe
    SELECT COUNT(*) INTO turma_exists FROM Turma WHERE Id_turma = n_turma;
    IF turma_exists = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Turma não encontrada.';
    END IF;

    -- Verificar se a disciplina existe
    SELECT COUNT(*) INTO disciplina_exists FROM Disciplina WHERE Id_disciplina = n_disciplina;
    IF disciplina_exists = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Disciplina não encontrada.';
    END IF;

    -- Verificar se o professor existe
    SELECT COUNT(*) INTO professor_exists FROM Professor WHERE Id_professor = n_professor;
    IF professor_exists = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Professor não encontrado.';
    END IF;

    -- Verificar se a data/hora de início é menor que a data/hora de fim
    IF n_data_hora_inicio >= n_data_hora_fim THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Data/hora de início deve ser anterior à data/hora de fim.';
    END IF;
```

```
    -- Verificar se há sobreposição de horários para a mesma turma
    SELECT COUNT(*) INTO overlap_exists FROM Aula
    WHERE Id_turma = n_turma
    AND (
        (n_data_hora_inicio BETWEEN Data_Hora_inicio AND Data_Hora_fim) OR
        (n_data_hora_fim BETWEEN Data_Hora_inicio AND Data_Hora_fim) OR
        (Data_Hora_inicio BETWEEN n_data_hora_inicio AND n_data_hora_fim) OR
        (Data_Hora_fim BETWEEN n_data_hora_inicio AND n_data_hora_fim)
    );

    IF overlap_exists > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Conflito de horário para esta turma.';
    END IF;

    -- Verificar se há sobreposição de horários para o mesmo professor
    SELECT COUNT(*) INTO overlap_exists FROM Aula
    WHERE Id_professor = n_professor
    AND (
        (n_data_hora_inicio BETWEEN Data_Hora_inicio AND Data_Hora_fim) OR
        (n_data_hora_fim BETWEEN Data_Hora_inicio AND Data_Hora_fim) OR
        (Data_Hora_inicio BETWEEN n_data_hora_inicio AND n_data_hora_fim) OR
        (Data_Hora_fim BETWEEN n_data_hora_inicio AND n_data_hora_fim)
    );

    IF overlap_exists > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Professor já possui aula neste horário.';
    END IF;

    -- Inserir a nova aula se todas as verificações passarem
    INSERT INTO Aula (Id_turma, Id_disciplina, Id_professor, Data_Hora_inicio, Data_Hora_fim)
    VALUES (n_turma, n_disciplina, n_professor, n_data_hora_inicio, n_data_hora_fim);

    SELECT 'Aula adicionada com sucesso.' AS Resultado;
END //

DELIMITER ;

CALL AdicionarNovaAula(1, 1, 1, '2023-06-01 08:00:00', '2023-06-01 10:00:00');
```

A procedure AdicionarNovaAula é um processo completo e seguro para agendamento de aulas, realizando múltiplas validações antes de inserir um novo registro. Ela verifica a existência de registros relacionados, valida o intervalo temporal e previne conflitos de horário.

1 - Verifica se a turma, disciplina e professor existem no sistema.

2 - Garante que o início seja anterior ao término da aula.

3 - Evita sobreposição de horários para a mesma turma

4 - Impede que um professor seja alocado em duas aulas simultâneas

Vantagens:

Segurança: Todas as validações necessárias em um único lugar

Consistência: Elimina duplicidade de código de verificação

Usabilidade: Interface simplificada para agendamento de aulas

Controle de erros: Mensagens claras e específicas para cada tipo de falha

Exemplo de uso:

```
CALL AdicionarNovaAula(1, 1, 1, '2023-06-01 08:00:00', '2023-06-01 10:00:00');
```

Resultado
Aula adicionada com sucesso.