UNIVERSIDADE SÃO FRANCISCO

Curso de Engenharia de Computação
Câmpus: Bragança Paulista
Turno : Matutino

BANCO DE DADOS

Sem Reprova

202110323 - Ricardo Fiorini Cuato
202105122 - Luis Gustavo Santos da Costa
202106958 - Gustavo Rafael Pinto

Bragança (Câmpus)
13 / junho / 2023

UNIVERSIDADE SÃO FRANCISCO
Curso de Engenharia de Computação

A plataforma "Sem Reprova" tem como objetivo principal auxiliar estudantes universitários e de cursos técnicos a encontrar conteúdos online de forma mais fácil e eficiente. O intuito é proporcionar uma experiência de estudo mais eficaz, reduzindo assim a probabilidade de reprovação e de ter que lidar com as indesejadas dependências (DPs).

Com o avanço tecnológico e o acesso à internet, o universo do conhecimento tornou-se vasto e diversificado. No entanto, muitos alunos enfrentam dificuldades em localizar materiais relevantes para os seus estudos. A plataforma "Sem Reprova" surge como uma solução para esse problema, oferecendo uma ferramenta intuitiva e eficiente de busca de conteúdos relacionados aos cursos e disciplinas em que os estudantes estão matriculados.

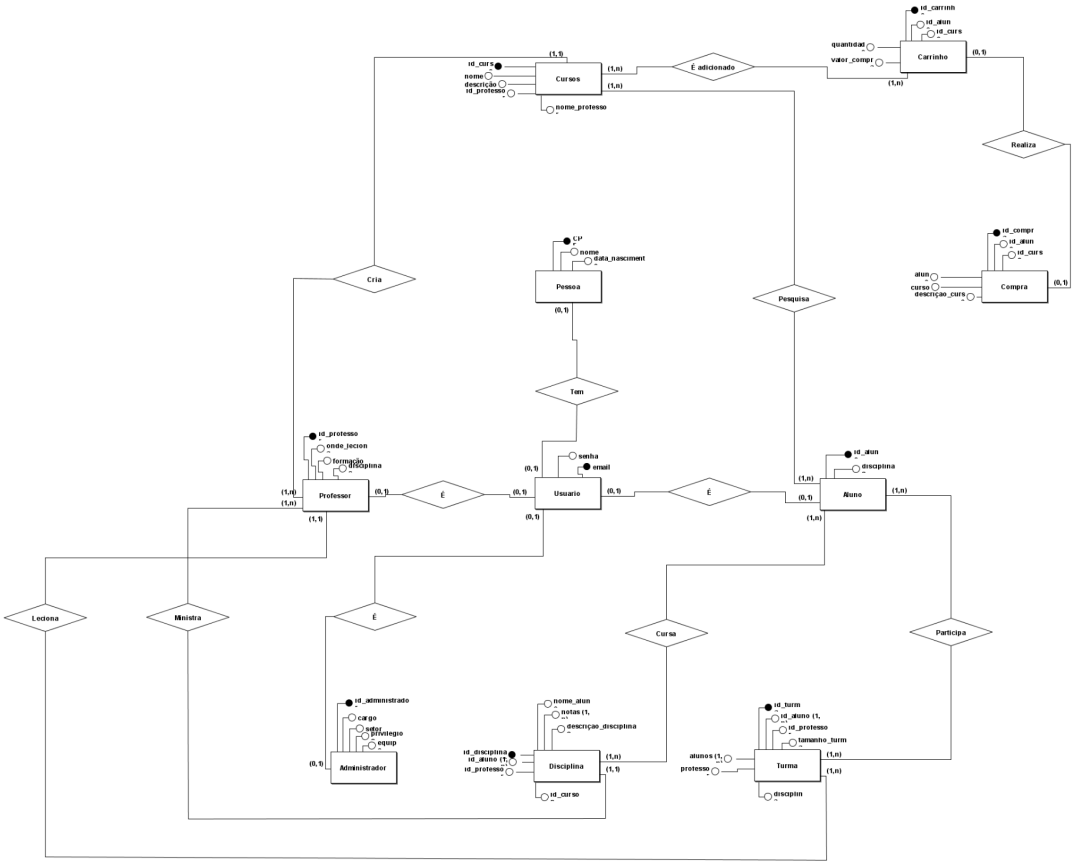
Ao utilizar a plataforma, os alunos têm acesso a uma ampla variedade de recursos, como videoaulas, artigos, tutoriais e materiais complementares. Esses recursos são organizados de forma a facilitar a busca e garantir que os estudantes encontrem exatamente o que precisam para aprofundar seus conhecimentos e se preparar para as avaliações acadêmicas.

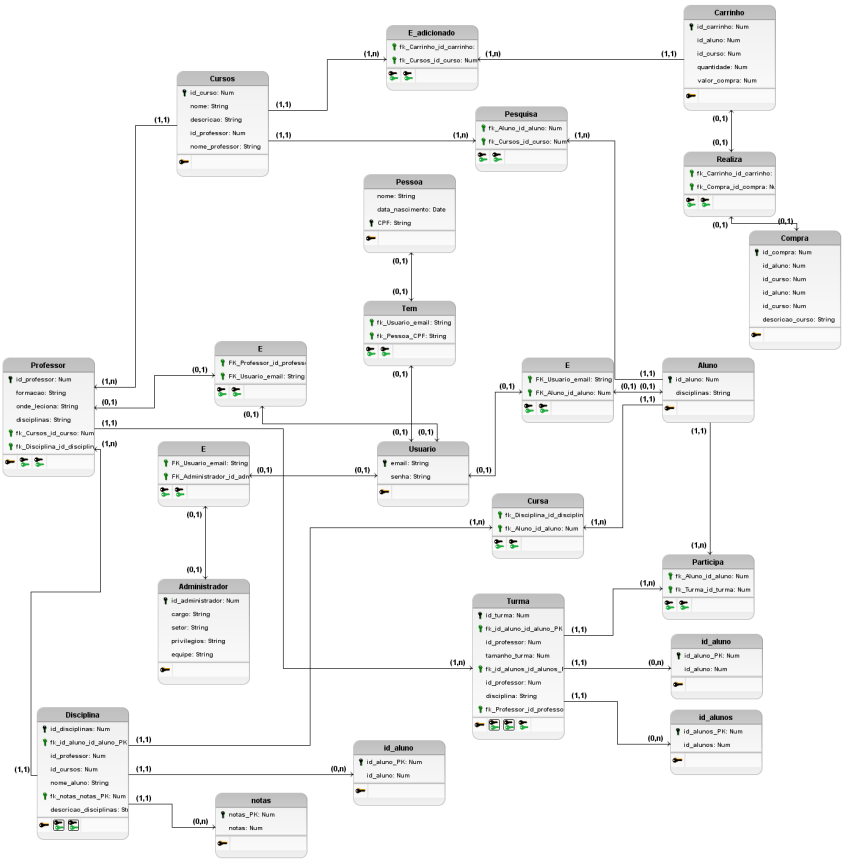
Além disso, a plataforma "Sem Reprova" conta com recursos de recomendação personalizada, baseados no perfil e nas preferências de estudo de cada aluno. Isso significa que, à medida que os estudantes utilizam a plataforma, ela aprende com suas interações e é capaz de sugerir conteúdos adicionais que possam ser relevantes para seu aprendizado.

A plataforma também oferece uma comunidade online, onde os alunos podem interagir entre si, compartilhar materiais, tirar dúvidas e trocar experiências de estudo. Essa colaboração mútua estimula o engajamento e o trabalho em equipe, contribuindo para um ambiente acadêmico mais enriquecedor.

Com a utilização da plataforma "Sem Reprova", os alunos podem otimizar seu tempo de estudo, concentrando-se nas fontes mais confiáveis e relevantes para sua formação. Isso resulta em uma maior eficiência no aprendizado, reduzindo a probabilidade de enfrentar dificuldades durante o curso e, consequentemente, minimizando o risco de reprovação e dependências.

Em resumo, a plataforma "Sem Reprova" visa facilitar a jornada acadêmica dos estudantes, oferecendo uma solução prática e eficiente para encontrar conteúdos online. Ao utilizar essa ferramenta, os alunos podem estudar com mais facilidade e aproveitar ao máximo os recursos disponíveis, aumentando suas chances de sucesso e minimizando as preocupações com reprovações e dependências.


id_carrinh
id_alun
id_curs
quantidad
valor_compr
Carrinho
(0, 1)
(1,n)
(1, 1)
id_curs
nome
descrição
id_professo
Cursos
(1,n)
(1,n)
nome_professo
É adicionado
Realiza
id_compr
id_alun
id_curs
alun
curso
descrição_curs
Compra
(0, 1)
CP
nome
data_nasciment
Pessoa
(0, 1)
Cria
Pesquisa
Tem
id_professo
onde_lecion
formação
disciplina
Professor
(1,n)
(1,n)
(0, 1)
(1, 1)
É
senha
email
Usuario
(0, 1)
(0, 1)
(0, 1)
É
id_alun
disciplina
Aluno
(1,n)
(0, 1)
(1,n)
(1,n)
Leciona
Ministra
É
Cursa
Participa
id_administrado
cargo
salar
privilegio
equip
Administrador
(0, 1)
nome_alun
notas (1,
descriçao_disciplina
id_disciplina
id_aluno (1,
id_professo
Disciplina
(1,n)
(1, 1)
id_curso
id_turm
id_aluno (1,
id_professo
tamanho_turm
alunos (1,
professo
Turma
(1,n)
(1,n)
disciplin


Carrinho
id_carrinho: Num
id_aluno: Num
id_curso: Num
quantidade: Num
valor_compra: Num
E_adicionado
fk_Carrinho_id_carrinho:
fk_Cursos_id_curso: Num
Cursos
id_curso: Num
nome: String
descricao: String
id_professor: Num
nome_professor: String
Pesquisa
fk_Aluno_id_aluno: Num
fk_Cursos_id_curso: Num
Realiza
fk_Carrinho_id_carrinho:
fk_Compra_id_compra: N
Compra
id_compra: Num
id_aluno: Num
id_curso: Num
id_aluno: Num
id_curso: Num
descricao_curso: String
Pessoa
nome: String
data_nascimento: Date
CPF: String
Tem
fk_Usuario_email: String
fk_Pessoa_CPF: String
Professor
id_professor: Num
formacao: String
onde_leciona: String
disciplinas: String
fk_Cursos_id_curso: Num
fk_Disciplina_id_disciplin
E
FK_Professor_id_profess
FK_Usuario_email: String
E
FK_Usuario_email: String
FK_Aluno_id_aluno: Num
Aluno
id_aluno: Num
disciplinas: String
E
FK_Usuario_email: String
FK_Administrador_id_adm
Usuario
email: String
senha: String
Cursa
fk_Disciplina_id_disciplin
fk_Aluno_id_aluno: Num
Participa
fk_Aluno_id_aluno: Num
fk_Turma_id_turma: Num
Administrador
id_administrador: Num
cargo: String
setor: String
privilegios: String
equipe: String
Turma
id_turma: Num
fk_id_aluno_id_aluno_PK
id_professor: Num
tamanho_turma: Num
fk_id_alunos_id_alunos_
id_professor: Num
disciplina: String
fk_Professor_id_professo
id_aluno
id_aluno_PK: Num
id_aluno: Num
id_alunos
id_alunos_PK: Num
id_alunos: Num
Disciplina
id_disciplinas: Num
fk_id_aluno_id_aluno_PK
id_professor: Num
id_cursos: Num
nome_aluno: String
fk_notas_notas_PK: Num
descricao_disciplinas: St
id_aluno
id_aluno_PK: Num
id_aluno: Num
notas
notas_PK: Num
notas: Num

```
CREATE TABLE Usuario (
    email VARCHAR(50) PRIMARY KEY,
    senha VARCHAR(50)
);

CREATE TABLE Pessoa (
    nome VARCHAR(50),
    data_nascimento DATE,
    CPF VARCHAR(50) PRIMARY KEY
);

CREATE TABLE Aluno (
    id_aluno INT PRIMARY KEY,
    disciplinas VARCHAR(50)
);

CREATE TABLE Professor (
    id_professor INT PRIMARY KEY,
    formacao VARCHAR(50),
    onde_leciona VARCHAR(50),
    disciplinas VARCHAR(50),
    fk_Cursos_id_curso INT,
    fk_Disciplina_id_disciplinas INT,
    FOREIGN KEY (fk_Cursos_id_curso) REFERENCES Cursos (id_curso),
    FOREIGN KEY (fk_Disciplina_id_disciplinas) REFERENCES Disciplina (id_disciplinas)
);

CREATE TABLE Administrador (
    id_administrador INT PRIMARY KEY,
    cargo VARCHAR(50),
    setor VARCHAR(50),
    privilegios VARCHAR(50),
    equipe VARCHAR(50)
);

CREATE TABLE Turma (
    id_turma INT PRIMARY KEY,
    fk_id_aluno_id_aluno_PK INT,
    id_professor INT,
    tamanho_turma INT,
    fk_id_alunos_id_alunos_PK INT,
    disciplina VARCHAR(50),
    fk_Professor_id_professor INT,
    fk_Turma_id_turma INT,
    FOREIGN KEY (fk_id_aluno_id_aluno_PK) REFERENCES Aluno (id_aluno),
    FOREIGN KEY (id_professor) REFERENCES Professor (id_professor),
    FOREIGN KEY (fk_id_alunos_id_alunos_PK) REFERENCES Turma (id_turma),
    FOREIGN KEY (fk_Professor_id_professor) REFERENCES Professor (id_professor),
```

```
    FOREIGN KEY (fk_Turma_id_turma) REFERENCES Turma (id_turma)
);

CREATE TABLE Cursos (
    id_curso INT PRIMARY KEY,
    nome VARCHAR(50),
    descricao VARCHAR(50),
    id_professor INT,
    nome_professor VARCHAR(50),
    FOREIGN KEY (id_professor) REFERENCES Professor (id_professor)
);

CREATE TABLE Disciplina (
    id_disciplinas INT PRIMARY KEY,
    fk_id_aluno_id_aluno_PK INT,
    id_professor INT,
    id_cursos INT,
    nome_aluno VARCHAR(50),
    fk_notas_notas_PK INT,
    descricao_disciplinas VARCHAR(50),
    FOREIGN KEY (fk_id_aluno_id_aluno_PK) REFERENCES Aluno (id_aluno),
    FOREIGN KEY (fk_notas_notas_PK) REFERENCES notas (notas_PK)
);

CREATE TABLE Carrinho (
    id_carrinho INT PRIMARY KEY,
    id_aluno INT,
    id_curso INT,
    quantidade INT,
    valor_compra INT
);

CREATE TABLE Compra (
    id_compra INT PRIMARY KEY,
    id_aluno INT,
    id_curso INT,
    descricao_curso VARCHAR(50)
);

-- Inserir dados na tabela Usuario
INSERT INTO Usuario (email, senha)
VALUES ('usuario1@example.com', 'senha123');

INSERT INTO Usuario (email, senha)
VALUES ('usuario2@example.com', 'senha456');

-- Inserir dados na tabela Pessoa
INSERT INTO Pessoa (nome, data_nascimento, CPF)
```

```
VALUES ('João Silva', '1990-05-15', '12345678901');

INSERT INTO Pessoa (nome, data_nascimento, CPF)
VALUES ('Maria Souza', '1985-09-10', '98765432109');

-- Inserir dados na tabela Aluno
INSERT INTO Aluno (id_aluno, disciplinas)
VALUES (1, 'Matemática, Ciências');

INSERT INTO Aluno (id_aluno, disciplinas)
VALUES (2, 'História, Geografia');

-- Inserir dados na tabela Professor
INSERT INTO Professor (id_professor, formacao, onde_leciona, disciplinas,
fk_Cursos_id_curso, fk_Disciplina_id_disciplinas)
VALUES (1, 'Doutorado em Física', 'Universidade XYZ', 'Física, Matemática', 1, 1);

INSERT INTO Professor (id_professor, formacao, onde_leciona, disciplinas,
fk_Cursos_id_curso, fk_Disciplina_id_disciplinas)
VALUES (2, 'Mestrado em História', 'Instituto ABC', 'História, Sociologia', 2, 2);

-- Inserir dados na tabela Administrador
INSERT INTO Administrador (id_administrador, cargo, setor, privilegios, equipe)
VALUES (1, 'Gerente', 'Departamento de Finanças', 'Acesso total', 'Equipe de 10 pessoas');

INSERT INTO Administrador (id_administrador, cargo, setor, privilegios, equipe)
VALUES (2, 'Coordenador', 'Departamento de Recursos Humanos', 'Acesso restrito', 'Equipe
de 5 pessoas');

-- Inserir dados na tabela Turma
INSERT INTO Turma (id_turma, fk_id_aluno_id_aluno_PK, id_professor, tamanho_turma,
fk_id_alunos_id_alunos_PK, disciplina, fk_Professor_id_professor, fk_Turma_id_turma)
VALUES (1, 1, 1, 30, 1, 'Física', 1, 1);

INSERT INTO Turma (id_turma, fk_id_aluno_id_aluno_PK, id_professor, tamanho_turma,
fk_id_alunos_id_alunos_PK, disciplina, fk_Professor_id_professor, fk_Turma_id_turma)
VALUES (2, 2, 2, 25, 2, 'História', 2, 2);

-- Inserir dados na tabela Cursos
INSERT INTO Cursos (id_curso, nome, descricao, id_professor, nome_professor)
VALUES (1, 'Física Avançada', 'Curso avançado de Física', 1, 'Prof. Silva');

INSERT INTO Cursos (id_curso, nome, descricao, id_professor, nome_professor)
VALUES (2, 'História do Brasil', 'Curso de História do Brasil', 2, 'Prof. Souza');

-- Inserir dados na tabela Disciplina
INSERT INTO Disciplina (id_disciplinas, fk_id_aluno_id_aluno_PK, id_professor, id_cursos,
nome_aluno, fk_notas_notas_PK, descricao_disciplinas)
```

```
VALUES (1, 1, 1, 1, 'João Silva', 1, 'Disciplina de Física');

INSERT INTO Disciplina (id_disciplinas, fk_id_aluno_id_aluno_PK, id_professor, id_cursos,
nome_aluno, fk_notas_notas_PK, descricao_disciplinas)
VALUES (2, 2, 2, 2, 'Maria Souza', 2, 'Disciplina de História');

-- Inserir dados na tabela Carrinho
INSERT INTO Carrinho (id_carrinho, id_aluno, id_curso, quantidade, valor_compra)
VALUES (1, 1, 1, 2, 100);

INSERT INTO Carrinho (id_carrinho, id_aluno, id_curso, quantidade, valor_compra)
VALUES (2, 2, 2, 1, 50);

-- Inserir dados na tabela Compra
INSERT INTO Compra (id_compra, id_aluno, id_curso, descricao_curso)
VALUES (1, 1, 1, 'Física Avançada');

INSERT INTO Compra (id_compra, id_aluno, id_curso, descricao_curso)
VALUES (2, 2, 2, 'História do Brasil');

-- consultas de teste
-- Consulta para obter informações sobre um aluno específico, incluindo suas disciplinas e
os cursos associados a essas disciplinas:
SELECT A.id_aluno, A.disciplinas, D.id_cursos, C.nome AS nome_curso
FROM Aluno A
JOIN Disciplina D ON A.id_aluno = D.fk_id_aluno_id_aluno_PK
JOIN Cursos C ON D.id_cursos = C.id_curso
WHERE A.id_aluno = 1;

--Consulta para obter todos os professores que lecionam uma determinada disciplina:
SELECT P.id_professor, P.formacao, P.onde_leciona
FROM Professor P
JOIN Disciplina D ON P.id_professor = D.id_professor
WHERE D.descricao_disciplinas = 'Física';

--Consulta para listar todas as compras feitas por um determinado aluno, incluindo
informações sobre o curso comprado:
SELECT C.id_compra, C.descricao_curso, CO.nome AS nome_aluno, CU.nome AS
nome_curso
FROM Compra C
JOIN Aluno CO ON C.id_aluno = CO.id_aluno
JOIN Cursos CU ON C.id_curso = CU.id_curso
WHERE CO.nome = 'João Silva';

--Consulta para obter o número total de alunos matriculados em cada turma:
SELECT T.id_turma, COUNT(T.fk_id_aluno_id_aluno_PK) AS total_alunos
FROM Turma T
JOIN Aluno A ON T.fk_id_aluno_id_aluno_PK = A.id_aluno
```

```
GROUP BY T.id_turma;

-- Consulta para listar todos os administradores e os cursos relacionados à equipe que eles supervisionam:
SELECT ADM.id_administrador, ADM.cargo, ADM.setor, ADM.equipe, C.nome AS nome_curso
FROM Administrador ADM
JOIN Cursos C ON ADM.equipe = C.nome_professor;

--Views
--View para listar todos os alunos e suas respectivas disciplinas:
CREATE VIEW vw_alunos_disciplinas AS
SELECT A.id_aluno, A.disciplinas, D.nome_aluno, D.descricao_disciplinas
FROM Aluno A
JOIN Disciplina D ON A.id_aluno = D.fk_id_aluno_id_aluno_PK;


--View para mostrar todas as compras feitas por cada aluno:
CREATE VIEW vw_compras_por_aluno AS
SELECT C.id_compra, C.id_aluno, C.id_curso, A.nome AS nome_aluno, CO.nome AS nome_curso
FROM Compra C
JOIN Aluno A ON C.id_aluno = A.id_aluno
JOIN Cursos CO ON C.id_curso = CO.id_curso;

--View para exibir todas as turmas com o número total de alunos matriculados:
CREATE VIEW vw_turmas_total_alunos AS
SELECT T.id_turma, T.tamanho_turma, COUNT(A.id_aluno) AS total_alunos
FROM Turma T
LEFT JOIN Aluno A ON T.fk_id_aluno_id_aluno_PK = A.id_aluno
GROUP BY T.id_turma;


--View para listar todos os cursos e seus respectivos professores:
CREATE VIEW vw_cursos_professores AS
SELECT C.id_curso, C.nome AS nome_curso, P.id_professor, P.nome AS nome_professor
FROM Cursos C
JOIN Professor P ON C.id_professor = P.id_professor;

--View para mostrar todas as disciplinas com suas respectivas notas:
CREATE VIEW vw_disciplinas_notas AS
SELECT D.id_disciplinas, D.nome_aluno, N.nota
FROM Disciplina D
JOIN Notas N ON D.fk_notas_notas_PK = N.id_notas;

--Triggers
--Trigger para atualizar a quantidade de alunos matriculados em uma turma sempre que um novo aluno for inserido:
```

```
CREATE TRIGGER tr_atualizar_quantidade_alunos
AFTER INSERT ON Turma
FOR EACH ROW
BEGIN
   UPDATE Turma
   SET tamanho_turma = (SELECT COUNT(*) FROM Aluno WHERE id_aluno =
NEW.fk_id_aluno_id_aluno_PK)
   WHERE id_turma = NEW.id_turma;
END;

--Trigger para garantir que não haja alunos duplicados na tabela Aluno:
CREATE TRIGGER tr_verificar_aluno_duplicado
BEFORE INSERT ON Aluno
FOR EACH ROW
BEGIN
   IF (SELECT COUNT(*) FROM Aluno WHERE id_aluno = NEW.id_aluno) > 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Já existe um aluno com esse
ID.';
   END IF;
END;

--Trigger para registrar um log sempre que uma nova compra for inserida na tabela Compra:
CREATE TRIGGER tr_registrar_log_compra
AFTER INSERT ON Compra
FOR EACH ROW
BEGIN
   INSERT INTO Log (descricao)
   VALUES ('Nova compra registrada - ID da compra: ' + CAST(NEW.id_compra AS
VARCHAR));
END;

--Trigger para atualizar a quantidade disponível de um curso no carrinho sempre que um
curso for adicionado ou removido:
CREATE TRIGGER tr_atualizar_quantidade_curso_carrinho
AFTER INSERT, DELETE ON Carrinho
FOR EACH ROW
BEGIN
   IF EXISTS (SELECT * FROM Carrinho WHERE id_curso = NEW.id_curso) THEN
      UPDATE Carrinho
      SET quantidade = (SELECT COUNT(*) FROM Carrinho WHERE id_curso =
NEW.id_curso)
      WHERE id_curso = NEW.id_curso;
   ELSE
      UPDATE Carrinho
      SET quantidade = 0
      WHERE id_curso = NEW.id_curso;
   END IF;
END;
```

```
--Trigger para atualizar a descrição da disciplina sempre que a nota relacionada for alterada:
CREATE TRIGGER tr_atualizar_descricao_disciplina
AFTER UPDATE ON Notas
FOR EACH ROW
BEGIN
    UPDATE Disciplina
    SET descricao_disciplinas = 'Nota atualizada'
    WHERE id_disciplinas = (SELECT fk_id_aluno_id_aluno_PK FROM Disciplina WHERE
fk_notas_notas_PK = NEW.id_notas);
END;


--Store procedures
--Procedimento armazenado para inserir um novo aluno:
CREATE PROCEDURE sp_inserir_aluno (
    IN aluno_id INT,
    IN aluno_disciplinas VARCHAR(50)
)
BEGIN
    INSERT INTO Aluno (id_aluno, disciplinas)
    VALUES (aluno_id, aluno_disciplinas);
END;

--Procedimento armazenado para atualizar a formação de um professor:
CREATE PROCEDURE sp_atualizar_formacao_professor (
    IN professor_id INT,
    IN nova_formacao VARCHAR(50)
)
BEGIN
    UPDATE Professor
    SET formacao = nova_formacao
    WHERE id_professor = professor_id;
END;

--Procedimento armazenado para calcular a média das notas de um aluno em uma
disciplina:
CREATE PROCEDURE sp_calcular_media_notas (
    IN aluno_id INT,
    IN disciplina_id INT,
    OUT media FLOAT
)
BEGIN
    SELECT AVG(nota) INTO media
    FROM Notas
    WHERE fk_id_aluno_id_aluno_PK = aluno_id AND fk_disciplina_id_disciplinas =
disciplina_id;
END;
```

```
--Procedimento armazenado para obter o total de cursos em um determinado setor administrativo:
CREATE PROCEDURE sp_obter_total_cursos_por_setor (
    IN setor_administrativo VARCHAR(50),
    OUT total_cursos INT
)
BEGIN
    SELECT COUNT(*) INTO total_cursos
    FROM Cursos
    WHERE setor = setor_administrativo;
END;

--Procedimento armazenado para excluir um aluno e todas as suas referências em outras tabelas:
CREATE PROCEDURE sp_excluir_aluno (
    IN aluno_id INT
)
BEGIN
    DELETE FROM Aluno WHERE id_aluno = aluno_id;
    DELETE FROM Disciplina WHERE fk_id_aluno_id_aluno_PK = aluno_id;
    DELETE FROM Carrinho WHERE id_aluno = aluno_id;
    DELETE FROM Compra WHERE id_aluno = aluno_id;
END;

--Indices
--Índice na tabela Aluno para melhorar o desempenho em consultas que buscam alunos por ID:
CREATE INDEX idx_aluno_id_aluno ON Aluno (id_aluno);

--Índice na tabela Turma para melhorar o desempenho em consultas que envolvem o ID do aluno:
CREATE INDEX idx_turma_fk_id_aluno_id_aluno_PK ON Turma (fk_id_aluno_id_aluno_PK);

--Índice na tabela Compra para consultas que envolvem o ID do aluno ou do curso:
CREATE INDEX idx_compra_id_aluno ON Compra (id_aluno);
CREATE INDEX idx_compra_id_curso ON Compra (id_curso);

--Índice na tabela Disciplina para consultas que envolvem o ID do aluno ou do professor:
CREATE INDEX idx_disciplina_fk_id_aluno_id_aluno_PK ON Disciplina (fk_id_aluno_id_aluno_PK);
CREATE INDEX idx_disciplina_id_professor ON Disciplina (id_professor);

--Índice na tabela Cursos para consultas que envolvem o ID do professor:
CREATE INDEX idx_cursos_id_professor ON Cursos (id_professor);
```