BANCO DE DADOS

Trabalho – Relatório

Curso:	Engenharia de Software
Aluno(a):	Yan Lucas Bezerra Silva
RU:	4351640

- ## 1ª Etapa – Modelagem

Pontuação: 25 pontos.

Dadas as regras de negócio abaixo listadas, referentes ao estudo de caso de uma companhia aérea, elabore o Modelo Entidade-Relacionamento (MER), isto é, o modelo conceitual.

O Modelo Entidade-Relacionamento (MER) deve contemplar os seguintes itens:

- Entidades;
- Atributos;
- Relacionamentos;
- Cardinalidades;
- Chaves primárias;
- Chaves estrangeiras.

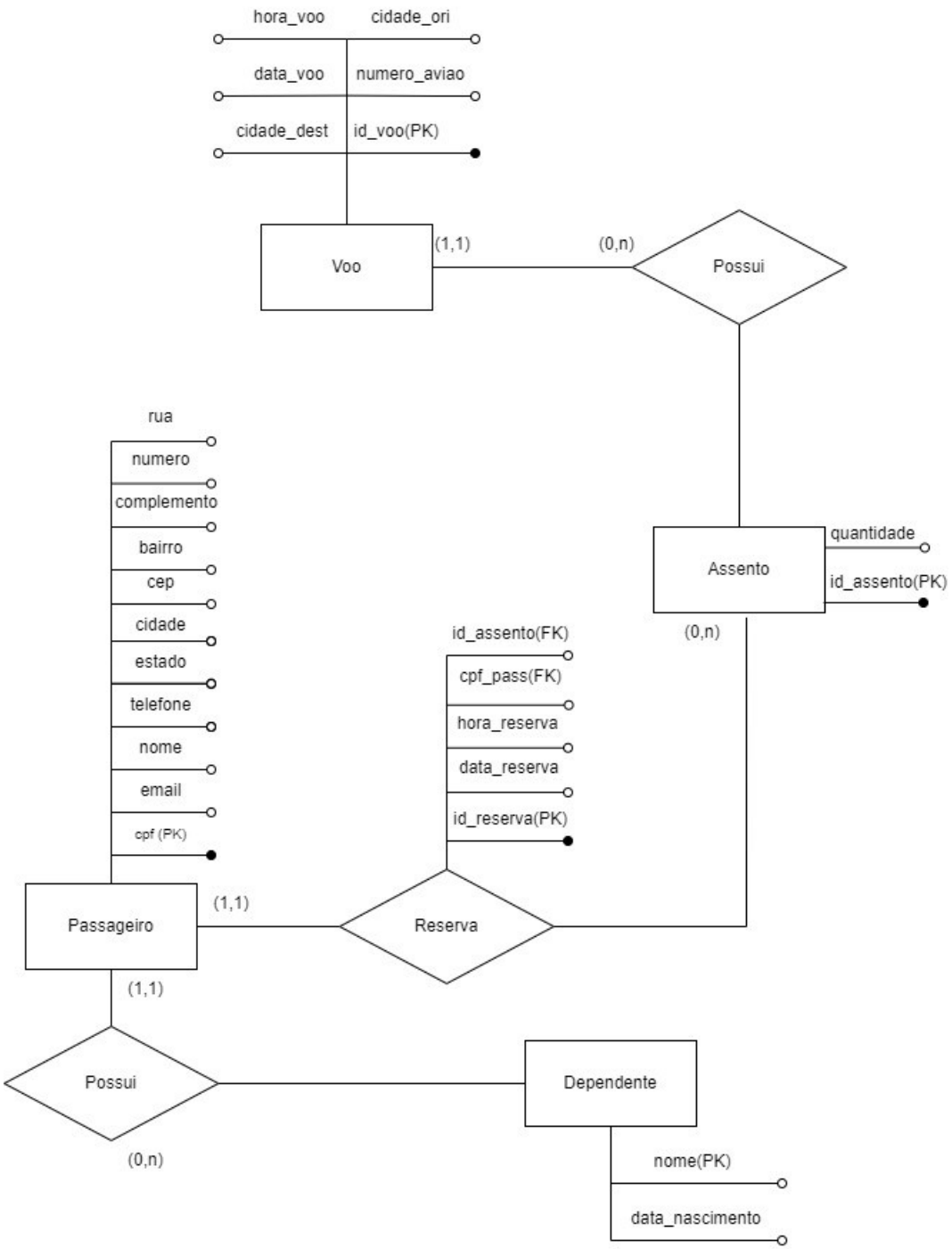
Uma companhia aérea necessita controlar os dados de seus voos. Para isso, contratou um profissional de Banco de Dados, a fim de modelar o Banco de Dados que armazenará os dados dos voos.

As regras de negócio são:

- Voo – Deverão ser armazenados os seguintes dados: identificação do voo, número do avião, cidade de origem, cidade de destino, data do voo e hora do voo;
- Assento – Deverão ser armazenados os seguintes dados: identificação do assento e

quantidade;

- Passageiro – Deverão ser armazenados os seguintes dados: CPF, nome, telefone, e-mail e endereço (rua, número, complemento, bairro, CEP, cidade e estado);
- Dependente – Deverão ser armazenados os seguintes dados: nome e data de nascimento;
- Um voo pode ter zero ou vários assentos, assim como zero ou vários assentos pertencem a um voo;
- Um passageiro pode ter zero ou várias reservas de assentos, assim como zero ou várias reservas de assentos pertencem a um passageiro;
- Um passageiro pode ter zero ou vários dependentes, assim como zero ou vários dependentes são de um passageiro;
- Da reserva deverão ser armazenados os seguintes dados: data da reserva e hora da reserva.



- **2ª Etapa – Implementação**

Considere o seguinte Modelo Relacional (modelo lógico), referente ao estudo de caso de uma faculdade:

Com base no Modelo Relacional dado e utilizando a *Structured Query Language* (SQL), no MySQL Workbench, implemente o que se pede.

Observação: Para testar o Banco de Dados após a implementação, utilize os comandos contidos no arquivo “Trabalho – Populando o Banco de Dados” para popular as tabelas. Tal arquivo contém todos os comandos de inserção dos dados (fictícios) necessários para a realização dos testes.

Pontuação: 25 pontos.

- Implemente um Banco de Dados chamado “Faculdade”. Após, implemente as tabelas, conforme o Modelo Relacional dado, observando as chaves primárias e as chaves estrangeiras. Todos os campos, de todas as tabelas, não podem ser nulos (*not null*).

```
CREATE DATABASE Faculdade;
USE Faculdade;

CREATE TABLE Aluno(
   idAluno INT NOT NULL,
   matricula VARCHAR(10) NOT NULL,
   nome VARCHAR(100) NOT NULL,
   PRIMARY KEY (idAluno)
);
CREATE TABLE Disciplina(
   idDisciplina INT NOT NULL,
   nome VARCHAR(100) NOT NULL,
   cargaHoraria INT NOT NULL,
   PRIMARY KEY (idDisciplina)
);
CREATE TABLE Curso(
   idCurso INT NOT NULL,
   nome VARCHAR(100) NOT NULL,
   PRIMARY KEY (idCurso)
);
CREATE TABLE Historico(
   idHistorico INT AUTO_INCREMENT PRIMARY KEY,
```

```
    idAluno INT NOT NULL,
    idDisciplina INT NOT NULL,
    nota DECIMAL(5, 2) NOT NULL,
    dataHistorico DATE NOT NULL,
    FOREIGN KEY (idAluno) REFERENCES Aluno(idAluno),
    FOREIGN KEY (idDisciplina) REFERENCES Disciplina(idDisciplina)
);
CREATE TABLE AlunoCurso(
    idAlunoCurso INT AUTO_INCREMENT PRIMARY KEY,
    idAluno INT NOT NULL,
    idCurso INT NOT NULL,
    anoEntrada INT NOT NULL,
    FOREIGN KEY (idAluno) REFERENCES Aluno(idAluno),
    FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)
);
CREATE TABLE Grade(
    idGrade INT AUTO_INCREMENT PRIMARY KEY,
    idCurso INT NOT NULL,
    ano INT NOT NULL,
    cargaHorariaTotal INT NOT NULL,
    FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)
);
CREATE TABLE GradeDisciplina(
    idGrade INT NOT NULL,
    idDisciplina INT NOT NULL,
    FOREIGN KEY (idGrade) REFERENCES Grade(idGrade),
    FOREIGN KEY (idDisciplina) REFERENCES Disciplina(idDisciplina)
);
```

Pontuação: 10 pontos.

- Implemente uma consulta para listar o quantitativo de cursos existentes.

```
SELECT count(idCurso) FROM faculdade.curso;
```

```
1 • SELECT count(idCurso) FROM faculdade.curso;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(idCurso)
9

Pontuação: 10 pontos.

- Implemente uma consulta para listar o nome das disciplinas existentes.

SELECT nome FROM faculdade.curso;

```
1 • SELECT nome FROM faculdade.curso;
```

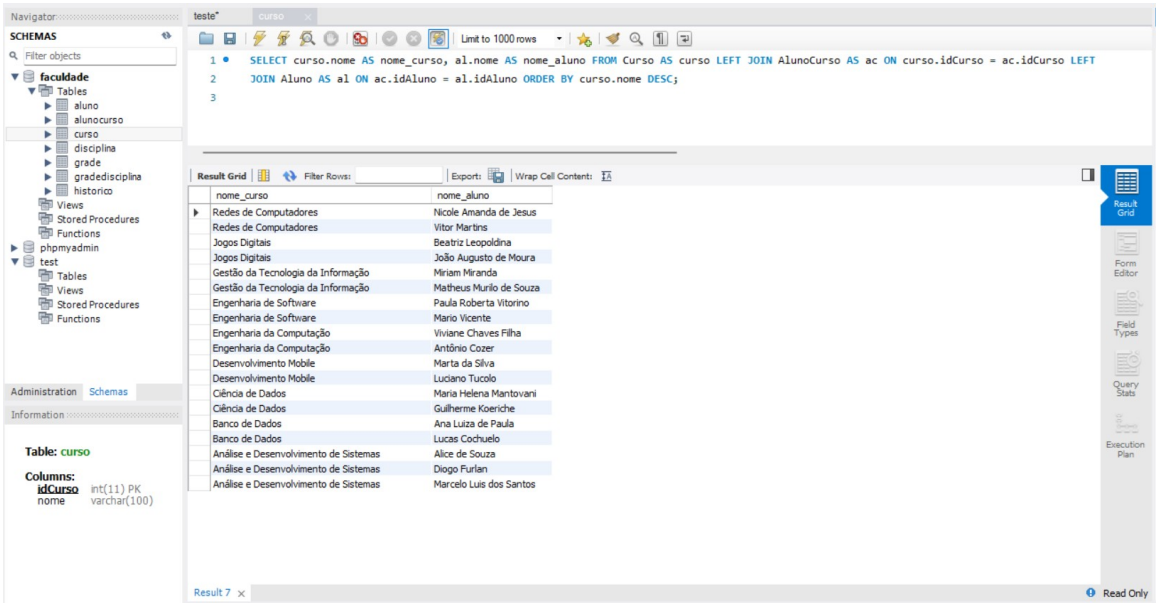
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

nome
Análise e Desenvolvimento de Sistemas
Banco de Dados
Ciência de Dados
Desenvolvimento Mobile
Engenharia da Computação
Engenharia de Software
Gestão da Tecnologia da Informação
Jogos Digitais
Redes de Computadores

Pontuação: 10 pontos.

- Implemente uma consulta para listar o nome de todos os cursos e o nome de seus respectivos alunos. A listagem deve ser mostrada em ordem decrescente pelo nome dos cursos.

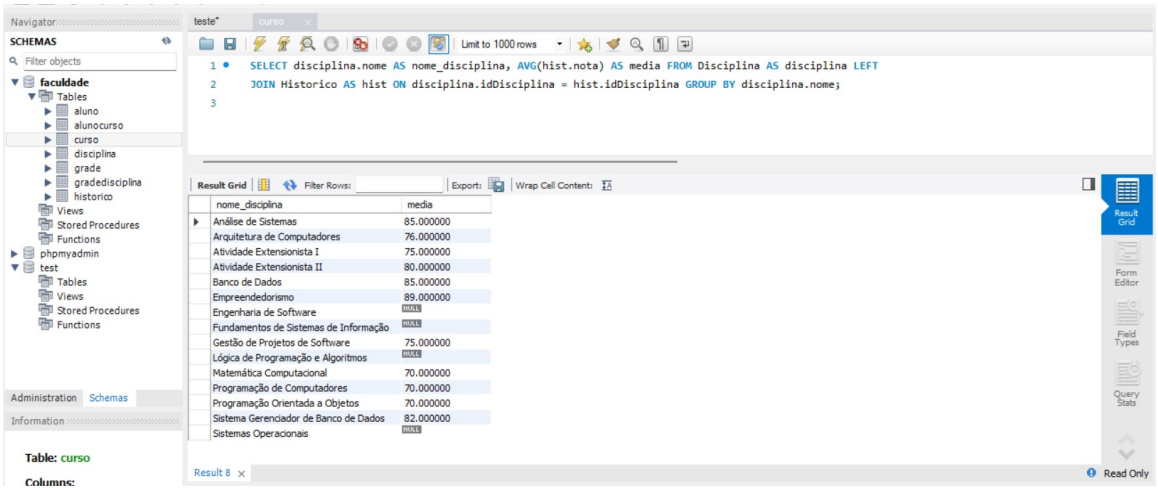
```
SELECT curso.nome AS nome_curso, al.nome AS nome_aluno FROM Curso AS curso LEFT JOIN AlunoCurso AS ac ON curso.idCurso = ac.idCurso LEFT
JOIN Aluno AS al ON ac.idAluno = al.idAluno ORDER BY curso.nome DESC;
```



Pontuação: 10 pontos.

- Implemente uma consulta para listar o nome das disciplinas e a média das notas das disciplinas em todos os cursos. Para isso, utilize o comando *group by*.

```
SELECT disciplina.nome AS nome_disciplina, AVG(hist.nota) AS media FROM Disciplina AS disciplina LEFT
JOIN Historico AS hist ON disciplina.idDisciplina = hist.idDisciplina GROUP BY disciplina.nome;
```



Pontuação: 10 pontos.

- Implemente uma consulta para listar o nome de todos os cursos e a quantidade de alunos em cada curso. Para isso, utilize os comandos *join* e *group by*.

```
SELECT curso.nome AS nome_curso, COUNT(alunoc.idAluno) AS total_alunos
FROM Curso AS curso LEFT
JOIN AlunoCurso AS alunoc ON curso.idCurso = alunoc.idCurso GROUP BY
curso.nome;
```