BANCO DE DADOS

Trabalho – Relatório

Curso:	CST ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - DISTÂNCIA
Aluno(a):	MORONI FERNANDES DA SILVEIRA
RU:	4506455

• 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:

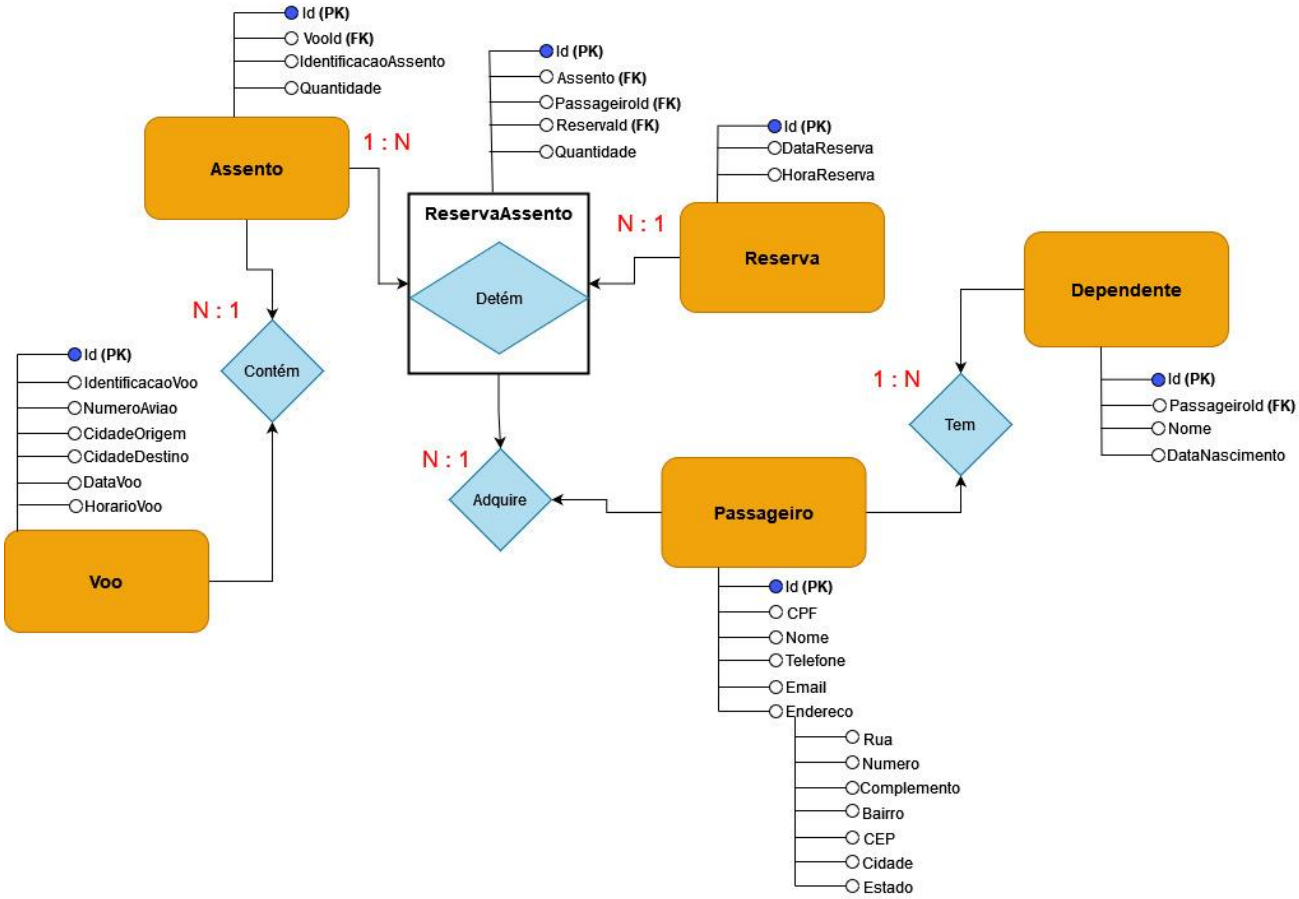
- Entidades;
- Atributos;
- Relacionamentos;
- Cardinalidades;
- Chaves primárias;
- Chaves estrangeiras.

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:

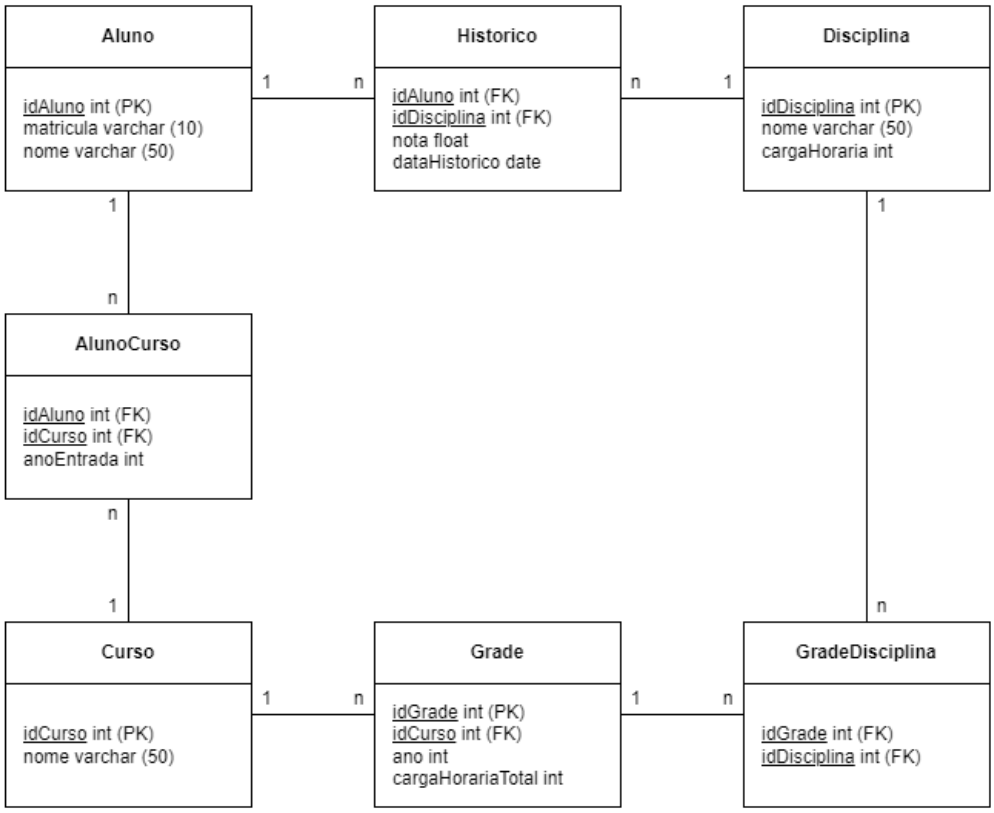
- Voo – Deverão ser armazenados os seguintes dados: identificação do voo, número do avião, cidade de origem, cidade de destino, data do voo e hora do voo;

- 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;

  show tables;

  -- criei uma tabela para os Alunos onde indiquei a id do aluno (identificação), sua matrícula e seu nome

  create table Aluno(
  idAluno int not null primary key,
  matricula varchar(10) not null,
  nome varchar(50) not null
  );

  -- criei uma tabela para as disciplinas, onde indiquei o id da disciplina, nome e sua carga horária

  create table Disciplina(
  idDisciplina int not null primary key,
  nome varchar(50) not null,
  cargaHoraria int not null);

  -- criei uma tabela para o histórico dos alunos, onde esta se relaciona com a tabela Aluno e Disciplina, de modo que indicando o id do aluno como chave estrangeira, id da disciplina como chave estrangeira, posso interligar as tabelas, além de ter inserido campos para as notas e a data do historico

  create table Historico(

idAluno int not null,
constraint fkHistoricoAluno foreign key (idAluno)
references Aluno (idAluno),
idDisciplina int not null,
constraint fkHistoricoDisciplina foreign key (idDisciplina)
references Disciplina(idDisciplina),
nota float not null,
dataHistorico date not null);

-- criei uma tabela para os cursos, indicando a id do curso e o seu nome

create table Curso(
idCurso int not null primary key,
nome varchar(50) not null);

-- criei uma tabela que relaciona a tabela Aluno com a tabela curso, nessa tabela encontra-se o id do Aluno como chave estrangeira, id do curso como chave estrangeira e o ano de entrada

create table AlunoCurso(
idAluno int not null,
constraint fkAlunoCursoAluno foreign key (idAluno) references Aluno(idAluno),
idCurso int not null,
constraint fkAlunoCursoCurso foreign key (idCurso) references Curso(idCurso),
anoEntrada int not null
);

-- criei uma tabela para a grade das matérias, onde indiquei o id da grade, o id do curso como chave estrangeira, o ano e a carga horária total

```
create table Grade(
idGrade int not null primary key,
idCurso int not null,
constraint fkGradeCurso foreign key (idCurso)
references Curso(idCurso),
ano int not null,
cargaHorariaTotal int not null
);

-- criei uma tabela para poder relacionar a tabela Grade com a tabela Disciplina, de modo que indiquei o id da grade como chave estrangeira e o id da disciplina também como chave estrangeira

create table GradeDisciplina(
idGrade int not null,
constraint fkGradeDisciplinaGrade foreign key (idGrade)
references Grade(idGrade),
idDisciplina int not null,
constraint fkGradeDisciplinaDisciplina foreign key (idDisciplina)
references Disciplina(idDisciplina));

show tables;

insert into Aluno (idAluno, matricula, nome) values (1, 'MTM001', 'Marcos Antônio'),
(2, 'FSC001', 'Vinicius Marcelino'), (3, 'BIO001', 'Lucas Dorneles'),
(4, 'ADS001', 'Matheus da Rocha'), (5, 'FSC002', 'Ruan Patrick'), (6, 'BIO002', 'Diego Bonfin'),
(7, 'MTM002', 'Kevin Junior'), (8, 'ART001', 'Amanda Fernandes'), (9, 'ADS002', 'Victor Ruiz'),
(10, 'ART002', 'Catia Snillvam');

insert into Disciplina (idDisciplina, nome, cargaHoraria) values (1, 'Cálculo I', 60),
```

(2, 'Física I', 60), (3, 'Biologia das Plantas', 40), (4, 'Lógica de Programação', 55),
(5, 'Fundamentos das Artes', 40), (6, 'Filosofia de Emmanuel kant', 60),
(7, 'Farmacologia Genérica', 80), (8, 'Geometria Analítica', 120), (9, 'Fundamentos Python',60),
(10, 'Corpo Humano', 80);

insert into Curso (idCurso, nome) values (1, 'Física'), (2, 'Matemática'), (3, 'Biologia'),
(4, 'Análise e Desenvolvimento de Sistemas'), (5, 'Artes Cênicas');

insert into Historico (idAluno, idDisciplina, nota, dataHistorico) values (1, 1, 75, '2023-06-30'),
(1, 2, 67, '2023-06-30'), (1, 8, 88, '2023-06-30'), (2, 1, 55, '2023-08-21'), (2, 2, 62, '2023-08-21'),
(2, 8, 71, '2023-08-21'), (3, 3, 84, '2023-10-15'), (3, 7, 91, '2023-10-15'), (3, 10, 74, '2023-10-15'),
(4, 4, 58, '2023-07-28'), (4, 8, 79, '2023-07-28'), (4, 9, 96, '2023-07-28'), (5, 1, 44, '2023-06-30'),
(5, 2, 69, '2023-06-30'), (5, 8, 73, '2023-06-30'), (6, 3, 61, '2023-10-15'), (6, 7, 83, '2023-10-15'),
(6, 10, 92, '2023-10-15'), (7, 1, 88, '2023-06-30'), (7, 2, 93, '2023-06-30'), (7, 8, 97, '2023-06-30'),
(8, 5, 97, '2023-09-28'), (8, 6, 96, '2023-09-28'), (8, 10, 86, '2023-09-28'), (9, 4, 64, '2023-07-28'),
(9, 8, 72, '2023-07-28'), (9, 9, 100, '2023-07-28'), (10, 5, 68, '2023-09-28'), (10, 6, 66, '2023-09-28'),
(10, 10, 100, '2023-09-28');

insert into AlunoCurso (idAluno, idCurso, anoEntrada) values (1, 2, 2023), (2, 1, 2019), (3, 3, 2019), (4, 4, 2023),
(5, 1, 2020), (6, 3, 2021), (7, 2, 2019), (8, 5, 2022), (9, 4, 2023), (10, 5, 2021);

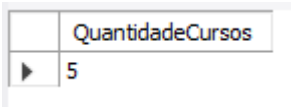
insert into Grade (idGrade, idCurso, ano, cargaHorariaTotal) values (1, 1, 2019, 3840),
(2, 2, 2019, 3294), (3, 3, 2019, 3600), (4, 4, 2020, 1600), (5, 5, 2020, 1875), (6, 1, 2023, 3840),
(7, 2, 2023, 3600), (8, 3, 2023, 3600), (9, 4, 2023, 1600), (10, 5, 2023,1875);

insert into GradeDisciplina (idGrade, idDisciplina) values (1,1), (1,2), (1,8), (2,1), (2,2), (2,8),
(3,3), (3,7), (3,10), (4,4), (4,8),(4,9), (5,5), (5,6), (5,10), (6,1), (6,2), (6,8), (7,1), (7,2),
(7,8), (8,3), (8,7), (8,10), (9,4), (9,8), (9,9), (10,5), (10,6), (10,10);

Pontuação: 10 pontos.

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

select count(*) as QuantidadeCursos from Curso;

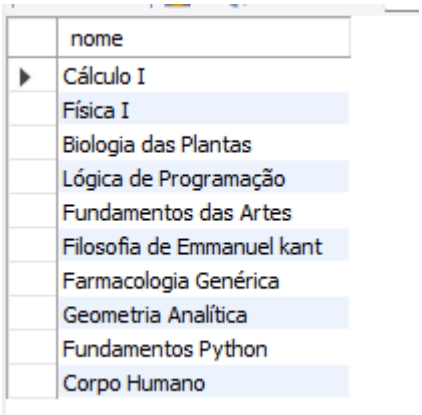


	QuantidadeCursos
▶	5

Pontuação: 10 pontos.

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

select nome from Disciplina;



	nome
▶	Cálculo I
	Física I
	Biologia das Plantas
	Lógica de Programação
	Fundamentos das Artes
	Filosofia de Emmanuel kant
	Farmacologia Genérica
	Geometria Analítica
	Fundamentos Python
	Corpo Humano

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 C.nome as Curso, A.nome as Aluno
from Curso C
left join AlunoCurso AC on C.idCurso = AC.idCurso
left join Aluno A on AC.idAluno = A.idAluno
order by C.nome desc;
```

Curso	Aluno
Matemática	Marcos Antônio
Matemática	Kevin Junior
Física	Vinicius Marcelino
Física	Ruan Patrick
Biologia	Diego Bonfin
Biologia	Lucas Dorneles
Artes Cênicas	Catia Snillvam
Artes Cênicas	Amanda Fernandes
Análise e Desenvolvimento de Sistemas	Matheus da Rocha
Análise e Desenvolvimento de Sistemas	Victor Ruiz

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 D.nome as Disciplina, avg (H.nota) as MediaNotas
from Disciplina D
left join GradeDisciplina GD on D.idDisciplina = GD.idDisciplina
left join Historico H on GD.idGrade = H.idDisciplina
group by D.nome;
```

	Disciplina	MediaNotas
▶	Biologia das Plantas	77
	Cálculo I	72.1
	Corpo Humano	80.66666666666667
	Farmacologia Genérica	77
	Filosofia de Emmanuel kant	85.25
	Física I	72.1
	Fundamentos das Artes	85.25
	Fundamentos Python	73.33333333333333
	Geometria Analítica	72.38461538461539
	Lógica de Programação	73.33333333333333

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 C.nome as Curso, COUNT(AC.idAluno) as QuantidadeDeAlunos
from Curso C
left join AlunoCurso AC on C.idCurso = AC.idCurso
group by C.nome;
```

	Curso	QuantidadeDeAlunos
▶	Análise e Desenvolvimento de Sistemas	2
	Artes Cênicas	2
	Biologia	2
	Física	2
	Matemática	2