UNIVERSIDADE FEDERAL
DE ALAGOAS

# Relatório do Projeto - Sistema de Passágens Aéreas

**Jhony Wictor do Nascimento Santos[1], Karleandro Santos da Silva[2]**

Universidade Federal de Alagoas (UFAL), Campus Arapiraca — SEDE,
Caixa Postal: 57309-005 – Arapiraca – AL – Brasil

Bacharelado em Ciência da Computação.

[1]Teotônio Vilela — AL — Brasil,
[2]Girau do Ponciano — AL — Brasil.

jhony.santos@arapiraca.ufal.br, karleandro.silva@arapiraca.ufal.br

***RESUMO.*** *Este trabalho está sendo desenvolvido pelos discentes do curso de Ciência da Computação, turma 2024.2, na disciplina de Teoria da Computação, sob a orientação do Professor Rodolfo Carneiro. A atividade faz parte dos critérios avaliativos para a composição da nota da AB2. O objetivo deste documento é descrever um dataset em um contexto de Sistema de Passagens, apresentando suas perguntas de pesquisa e scripts DDL e DML para a manipulação do banco, através de queries SQL. Assim como, sua normalização e implementação de Tópicos Avançados.*

---

## 1. Sistema De Passagens Aéreas

Trata-se de uma plataforma digital que fornece aos usuários um Sistema de Gerenciamento de Passagens Aéreas, cujo objetivo é oferecer aos clientes uma plataforma intuitiva para a aquisição de passagens, além do gerenciamento dos voos e dos recursos envolvidos na operação. O sistema proposto possibilita ao cliente realizar a compra de passagens com informações detalhadas, como data da viagem, poltrona escolhida, classe e preço. Além disso, cada passagem estará vinculada a um voo específico, que por sua vez estará associado ao avião responsável e ao aeroporto.

## 2. Formas de Normalização

A normalização em banco de dados é um processo que visa organizar os dados de forma eficiente, eliminando redundâncias e garantindo a integridade das informações. Ela é dividida em várias etapas chamadas formas normais (ou normal forms), sendo as três primeiras as mais fundamentais: Primeira Forma Normal (1FN), Segunda Forma Normal (2FN) e Terceira Forma Normal (3FN). Cada uma tem critérios específicos que devem ser atendidos. A seguir, explicamos cada uma delas:

1. **A Primeira Forma Normal** exige que todos os atributos de uma tabela contenham valores atômicos, ou seja, sem repetições ou grupos de dados múltiplos. Isso significa que cada campo deve armazenar apenas um valor por vez.
2. **A Segunda Forma Normal** parte do princípio que a tabela já está na 1FN. Para estar na 2FN, todos os atributos não-chave devem depender completamente da chave primária, e não de uma parte dela (no caso de chaves compostas).
3. **A Terceira Forma Normal** exige que a tabela já esteja na 2FN e que não existam dependências transitivas, ou seja, um atributo não-chave não pode depender de outro atributo não-chave.

## 3. Script Data Definition Language (DDL)

A seguir, serão apresentados os scripts do tipo DDL, cuja linguagem de definição de dados é a sub-língua responsável pela definição da forma como os dados são estruturados em um banco de dados. Em SQL, isto corresponde à manipulação de tabelas através do *CREATE TABLE, ALTER TABLE, e DROP TABLE.*

```
-- tabela cliente
create table cliente (
  cpf varchar(11) primary key unique not null,
  nome varchar(100) not null,
  data_nascimento date not null,
  email varchar(70) not null,
  telefone varchar(20) not null
);

-- tabela aeroporto
create table aeroporto (
  id_aeroporto varchar(10) primary key,
  nome varchar(60) not null,
  estado varchar(2) not null, -- sigla do estado
  cidade varchar(60) not null
);

-- tabela aviões
create table avioes (
  id_aviao varchar(10) primary key,
  modelo varchar(50) not null,
  capacidade int not null,
  companhia varchar(30) not null
);

-- tabela voos
create table voos (
  id_voo varchar(10) primary key,
  horario varchar(10) not null,
  id_aviao varchar(10) references avioes(id_aviao),
```

```
  id_aeroporto varchar(10) references aeroporto(id_aeroporto)
);

-- tabela passagem
create table passagem (
  id_passagem varchar(10) primary key,
  cpf_cliente varchar(11) not null references cliente(cpf),
  id_voo varchar(10) not null references voos(id_voo),
  data date not null,
  poltrona varchar(5) not null,
  classe varchar(50) not null,
  preco decimal(10,2) not null
);
```

**Figura 1.** Representação Completa do Código do Banco de Dados.

O banco de dados é composto por cinco tabelas principais: **cliente**, **aeroporto**, **aviões**, **voos** e **passagem**.

1. **cliente** armazena os dados dos passageiros, com os seguintes atributos: **cpf** (chave primária), **nome** (obrigatório), **data_nascimento, email** e **telefone.**
2. **aeroporto** contém os dados dos aeroportos e possui os atributos: **id_aeroporto** (chave primária), **nome** (obrigatório), **estado** e **cidade**.
3. **aviões** guardam informações das aeronaves utilizadas nos voos. Seus atributos são: **id_aviao** (chave primária), **modelo** (obrigatório), **capacidade** e **companhia.**
4. **voos** representa os voos disponíveis no sistema e inclui os seguintes campos: **id_voo** (chave primária), **horario, id_aviao** (chave estrangeira que se relaciona com a tabela aviões) e **id_aeroporto** (chave estrangeira que se relaciona com a tabela aeroporto).
5. **passagem** registra as passagens compradas pelos clientes. Seus atributos são: **id_passagem** (chave primária), **cpf_cliente** (chave estrangeira que se relaciona com a tabela cliente), **id_voo** (chave estrangeira que se relaciona com a tabela voos), **data** da viagem, **poltrona, classe e preco.**

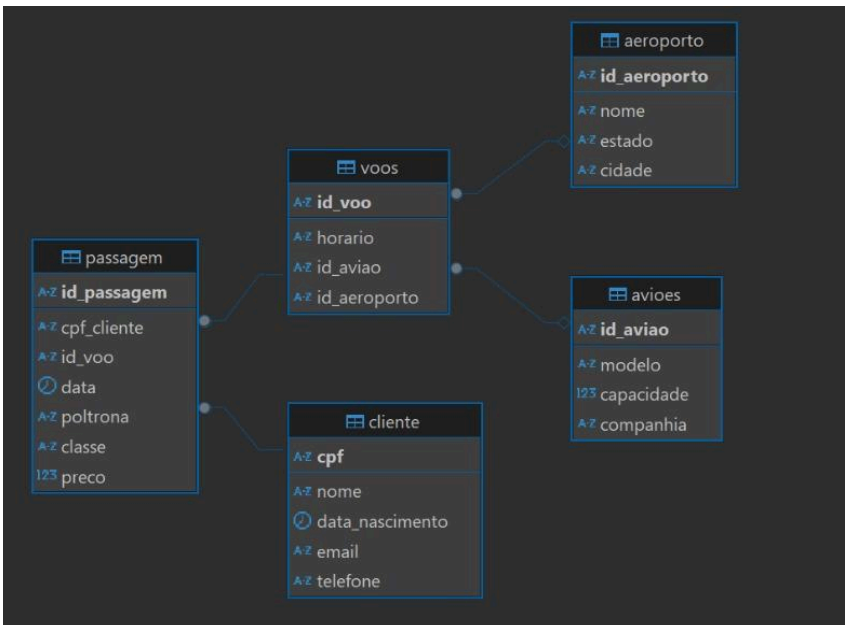


**Figura 2.** Representação Ilustrativa do Modelo Gráfico do Esquema Relacional

## 3.1. Análise das tabelas

### 1. cliente

```
create table cliente (
    cpf varchar(11) primary key unique not null,
    nome varchar(100) not null,
    data_nascimento date not null,
    email varchar(70) not null,
    telefone varchar(20) not null
);
```

1. **1FN:** Sim. Todos os atributos são atômicos (sem repetição de grupos de valores).
2. **2FN:** Sim. A chave primária é simples (**cpf**), e não há dependências parciais.
3. **3FN:** Sim. Nenhum atributo não chave depende de outro atributo não chave — todas as colunas dependem apenas do **cpf**.

### 2. aeroporto

```
create table aeroporto (
    id_aeroporto varchar(10) primary key,
    nome varchar(60) not null,
    estado varchar(2) not null,
    cidade varchar(60) not null
);
```

1. **1FN:** Sim. Todos os campos são atômicos.

2. **2FN:** Sim. A chave é simples, e todos os campos dependem completamente da chave.
3. **3FN:** Sim. Não há dependência transitiva (por exemplo, **estado** não depende de **cidade** ou vice-versa)

## 3. avioes

```
create table avioes (
    id_aviao varchar(10) primary key,
    modelo varchar(50) not null,
    capacidade int not null,
    companhia varchar(30) not null
);
```

1. **1FN:** Sim. Todos os dados são atômicos.
2. **2FN:** Sim. Chave simples, todos os atributos dependem diretamente dela.
3. **3FN:** Sim. Não há atributos derivados nem transitivos.

## 4. voos

```
create table voos (
    id_voo varchar(10) primary key,
    horario varchar(10) not null,
    id_aviao varchar(10) references avioes(id_aviao),
    id_aeroporto varchar(10) references aeroporto(id_aeroporto)
);
```

1. **1FN:** Sim.
2. **2FN:** Sim. Todos os atributos são dependentes da chave primária simples **(id_voo)**.
3. **3FN:** Sim. **id_aviao** e **id_aeroporto** são chaves estrangeiras e não há dependência transitiva entre atributos.

## 5. passagem

```
create table passagem (
    id_passagem varchar(10) primary key,
    cpf_cliente varchar(11) not null references cliente(cpf),
    id_voo varchar(10) not null references voos(id_voo),
    data date not null,
    poltrona varchar(5) not null,
    classe varchar(50) not null,
    preco decimal(10,2) not null
);
```

1. **1FN:** Sim.
2. **2FN:** Sim. Todos os atributos dependem da chave simples **id_passagem**.
   **3FN:** Sim. **preco, classe, data**, etc., não dependem de outros atributos não chave.

## 3.2. Implementação de Tópicos Avançados (Views, Function e Regras de acesso)

Para otimizar a organização e funcionalidade do sistema de passagens aéreas, foram implementados recursos avançados no banco de dados, como views, funções armazenadas e regras de acesso. A principal motivação para o uso de views é simplificar o acesso a informações complexas, evitando repetição de consultas. A view **historico_passagens_cliente**, por exemplo, reúne dados de clientes, voos, aviões e passagens em uma única estrutura lógica, facilitando consultas e geração de relatórios.

Além disso, foi criada a função **total_gasto_cliente**, que retorna o total gasto por um cliente, promovendo reutilização e centralização da lógica de cálculo. Por fim, regras de acesso foram definidas para restringir operações conforme o perfil do usuário, aumentando a segurança e a integridade dos dados. Esses recursos tornam o sistema mais robusto, seguro e fácil de manter.

1. **View para consultar histórico de passagens de um cliente**

```
CREATE VIEW historico_passagens_cliente AS
SELECT
   c.nome,
   p.id_passagem,
   v.id_voo,
   a.modelo AS aviao,
   ar.nome AS aeroporto,
   p.data,
   p.poltrona,
   p.classe,
   p.preco
FROM
   cliente c
JOIN passagem p ON c.cpf = p.cpf_cliente
JOIN voos v ON p.id_voo = v.id_voo
JOIN avioes a ON v.id_aviao = a.id_aviao
JOIN aeroporto ar ON v.id_aeroporto = ar.id_aeroporto;
```

2. **Function para calcular o total gasto por um cliente**

```
CREATE FUNCTION total_gasto_cliente(cpf_input VARCHAR(11))
RETURNS DECIMAL(10,2)
BEGIN
   DECLARE total DECIMAL(10,2);
   SELECT SUM(preco) INTO total
   FROM passagem
   WHERE cpf_cliente = cpf_input;
   RETURN IFNULL(total, 0);
END;
```

3. **Regras de acesso**

```
-- Criação de role
CREATE ROLE cliente_usuario;
```

```
-- Permissões restritas
GRANT SELECT ON cliente TO cliente_usuario;
GRANT SELECT ON historico_passagens_cliente TO
cliente_usuario;

-- Revoga alterações diretas
REVOKE INSERT, UPDATE, DELETE ON cliente FROM
cliente_usuario;
REVOKE INSERT, UPDATE, DELETE ON passagem FROM
cliente_usuario;
```

# 4. Script Data Manipulation Language (DML)

A seguir, serão apresentados os scripts do tipo DML, que por sua vez, é tida como o sub idioma responsável pela adição, edição ou exclusão de dados de um banco de dados. Em SQL, isto corresponde ao *INSERT, UPDATE, e DELETE*.

## 4.1. Perguntas e Respostas

Nesta seção, serão abordadas as perguntas de pesquisa que serão respondidas por intermédio da manipulação das queries no banco de dados.

1. **Seleção Simples:**
   ***Pergunta: Quais são os nomes e e-mails de todos os clientes cadastrados?***
   ```
   SELECT nome, email FROM cliente;
   ```

2. **Filtragem com WHERE:**
   ***Pergunta: Quais voos estão marcados entre às 12h10 e às 12h20?***
   ```
   SELECT
   FROM voos
   WHERE horario >= '12:10:00' and horario <= '12:20:00';
   ```

3. **INNER JOIN:**
   ***Pergunta: Quais clientes compraram passagens para a cidade de Maceió - AL? (mostrando nome do cliente, data da passagem e id do voo)***
   ```
   SELECT
     c.nome,
     p.data,
     p.id_voo
   FROM
     cliente c
   INNER JOIN passagem p ON c.cpf = p.cpf_cliente
   INNER JOIN voos v ON p.id_voo = v.id_voo
   INNER JOIN aeroporto a ON v.id_aeroporto = a.id_aeroporto
   WHERE
     a.cidade = 'Maceió';
   ```

4. **INNER JOIN:**

   ***Pergunta: Liste todos os voos do dia 2025-04-23?***

```
SELECT
  v.id_voo,
  v.horario,
  p.data
FROM
  voos v
INNER JOIN passagem p ON v.id_voo = p.id_voo
WHERE
  p.data = '2025-04-23';
```

5. **LEFT JOIN:**

   ***Pergunta: Quais voos existem, mesmo que não tenham passagens vendidas?***

```
SELECT v.id_voo, p.id_passagem
FROM voos v
LEFT JOIN passagem p ON v.id_voo = p.id_voo;
```

6. **RIGHT JOIN:**

   ***Pergunta: Quais voos foram agendados, mesmo que o avião não tenha mais dados de modelo, nem companhia?***

```
SELECT v.id_voo, a.modelo, a.companhia
FROM avioes a
RIGHT JOIN voos v ON a.id_aviao = v.id_aviao;
```

7. **FULL OUTER JOIN:**

   ***Pergunta: Liste todos os voos e todas as passagens, inclusive os voos sem passagens e as passagens com voos não registrados? (inconsistência)***

```
SELECT
  v.id_voo AS voo_id,
  v.horario,
  p.id_passagem AS passagem_id,
  p.data AS data_passagem
FROM voos v
FULL OUTER JOIN passagem p ON v.id_voo = p.id_voo;
```

8. **JOIN com múltiplas tabelas:**

   ***Pergunta: Quais são os nomes dos clientes, suas poltronas, companhia, o tipo da classe de voo comprada e o nome do aeroporto?***

```
SELECT
  c.nome AS nome_cliente,
  p.poltrona AS numero_poltrona,
  a.companhia AS nome_companhia,
  p.classe AS tipo_classe,
  ap.nome AS nome_aeroporto
FROM cliente c
JOIN passagem p ON c.cpf = p.cpf_cliente
```

```
JOIN voos v ON p.id_voo = v.id_voo
JOIN avioes a ON v.id_aviao = a.id_aviao
JOIN aeroporto ap ON v.id_aeroporto = ap.id_aeroporto;
```

9. **Filtragem com várias tabelas:**

   ***Pergunta: Quais são os voos operados pela companhia "LATAM" que têm capacidade maior que 150 lugares?***

```
SELECT
  v.id_voo,
  v.horario,
  a.modelo,
  a.capacidade,
  a.companhia
FROM voos v
JOIN avioes a ON v.id_aviao = a.id_aviao
WHERE a.companhia = 'LATAM'
 AND a.capacidade > 150;
```

## 5. Considerações Finais

O desenvolvimento do sistema de passagens aéreas permitiu aplicar de forma prática os conceitos de modelagem relacional, normalização e uso de recursos avançados em bancos de dados. A análise das tabelas demonstrou conformidade com as formas normais, garantindo consistência, integridade e ausência de redundâncias nos dados. Além disso, a implementação de *views*, *funções armazenadas* e *regras de acesso* contribuiu para melhorar a organização do sistema, facilitar consultas recorrentes e reforçar a segurança das informações.

Esses recursos tornam o banco mais eficiente, reutilizável e preparado para futuras integrações com sistemas externos ou interfaces gráficas. Conclui-se, portanto, que a aplicação dos princípios de banco de dados relacionais, aliada ao uso de funcionalidades mais avançadas, resulta em soluções mais robustas, seguras e aderentes às boas práticas de desenvolvimento de sistemas.

## 6. Referências Bibliográficas


BRITO, P. H.S; CAVALCANTE, R. C. Aula 01.1 - Introdução

BRITO, P. H.S; CAVALCANTE, R. C. Aula 01.2 - Introdução

BRITO, P. H.S; CAVALCANTE, R. C. Aula 02.1 - Funcionalidades do SGBD

BRITO, P. H.S; CAVALCANTE, R. C. Aula 02.2 - SGBD Postgres

BRITO, P. H.S; CAVALCANTE, R. C. Aula 03 - MER

BRITO, P. H.S; CAVALCANTE, R. C. Aula 03e04 - MER

BRITO, P. H.S; CAVALCANTE, R. C. Aula 03e04c - MER, Exercícios

BRITO, P. H.S; CAVALCANTE, R. C. Aula 06-Sql-parte1.pdf

BRITO, P. H.S; CAVALCANTE, R. C. 1.

BRITO, P. H.S; CAVALCANTE, R. C. Aula 08-Sql-parte2.pdf

BRITO, P. H.S; CAVALCANTE, R. C. Aula 08-Sql-parte3.pdf

WIKIPEDIA, Sistema de Gerenciamento de Dados.