Modelagem de um Sistema de Gerenciamento de Hotel

Enzo Henrique Paulino, Guilherme Roupe

Instituto de Informática – Universidade Federal do Rio de Janeiro (UNIRIO)
Rio de Janeiro – RJ – Brasil

Resumo. *Este trabalho apresenta o desenvolvimento de uma modelagem de dados voltada para um Sistema de Gerenciamento de Hotel, abrangendo as etapas de modelo Entidade-Relacionamento (ER), modelo lógico e processo de normalização.*

O modelo ER foi elaborado com o objetivo de representar de forma estruturada os principais processos do hotel, permitindo a gestão eficiente de clientes, funcionários, quartos, serviços e pagamentos. Nesse modelo, os clientes são identificados por CPF e podem solicitar tanto quartos quanto serviços oferecidos pelo hotel. Os quartos são definidos por número e tipo, enquanto os serviços possuem tipo e valor. Os serviços solicitados pelos clientes são executados por funcionários, supervisionados por um gerente, e cada funcionário possui seu salário mensal registrado no sistema.

Na sequência, o modelo lógico foi construído a partir do modelo ER, convertendo os relacionamentos e entidades em tabelas relacionais adequadas ao banco de dados. Essa etapa incluiu a criação de tabelas associativas, como Executa e Recebe, para representar relacionamentos muitos-para-muitos, além da aplicação de chaves primárias e estrangeiras que garantem a integridade referencial do sistema.

Por fim, foi aplicado o processo de normalização, com o intuito de eliminar redundâncias, evitar inconsistências e otimizar o armazenamento e a recuperação dos dados.

1. Motivação

A crescente demanda por eficiência e organização no setor hoteleiro torna indispensável o uso de sistemas de informação robustos. A complexidade das operações — que envolvem o gerenciamento de funcionários, clientes, serviços, quartos e pagamentos — faz com que a informatização seja uma necessidade essencial para garantir o bom funcionamento do hotel.

A falta de uma modelagem de dados bem estruturada pode gerar problemas como duplicação de informações, falhas na comunicação entre setores e dificuldades na elaboração de relatórios financeiros precisos. Diante disso, este trabalho tem como objetivo desenvolver um modelo de dados claro e detalhado, que sirva como a base estrutural de um sistema de gerenciamento de hotel eficiente, escalável e confiável.

Com essa modelagem, busca-se não apenas solucionar desafios operacionais, mas também otimizar a experiência do cliente e aperfeiçoar a tomada de decisões gerenciais, assegurando a rastreabilidade e a fluidez das operações diárias do hotel.

2. Mini Mundo

O sistema proposto tem como objetivo informatizar a gestão das operações internas de um hotel, centralizando o controle de clientes, funcionários, gerentes, salários, quartos, serviços e pagamentos.

Os **funcionários** são identificados por sua matrícula e podem executar diversos serviços para os clientes. Cada funcionário está sob a supervisão de um gerente e recebe um salário, cujo valor é registrado mensalmente.

Os **clientes**, identificados por CPF, possuem informações de contato como telefone e e-mail. Eles podem solicitar tanto quartos, que possuem número e tipo (ex.: solteiro, casal, suíte), quanto serviços, que possuem tipo e valor (ex.: lavanderia, café da manhã, transporte).

Cada **pagamento** está associado a um quarto reservado por um cliente e deve conter a data, o valor total e a forma de pagamento (ex.: cartão, dinheiro, PIX). Assim, é possível manter a rastreabilidade financeira de cada transação realizada no hotel.

Dessa forma, o mini-mundo abrange os principais processos de um hotel, permitindo uma visão integrada de clientes, funcionários, serviços, quartos e pagamentos,

servindo como base para o desenvolvimento de um sistema de gerenciamento eficiente.

Além disso, o sistema de gerenciamento de hotel proposto deve ser capaz de contemplar as seguintes funcionalidades:

1) O sistema deve permitir cadastrar novos clientes no hotel.
2) O sistema deve permitir registrar reservas, vinculando cliente, quarto e datas.
3) O sistema deve permitir registrar pagamentos associados às reservas.
4) O sistema deve permitir cadastrar funcionários, gerentes e os serviços prestados no hotel.
5) O sistema deve permitir registrar a execução de serviços por funcionários e associá-los às reservas.
6) O sistema deve permitir consultar informações do cliente pelo CPF.
7) O sistema deve permitir alterar dados de clientes, reservas, serviços e pagamentos.
8) O sistema deve permitir remover registros de acordo com critérios específicos.
9) O sistema deve permitir operações que afetam mais de uma tabela ao mesmo tempo, como o cancelamento de reservas, removendo serviços e pagamentos vinculados.

3. Modelo ER

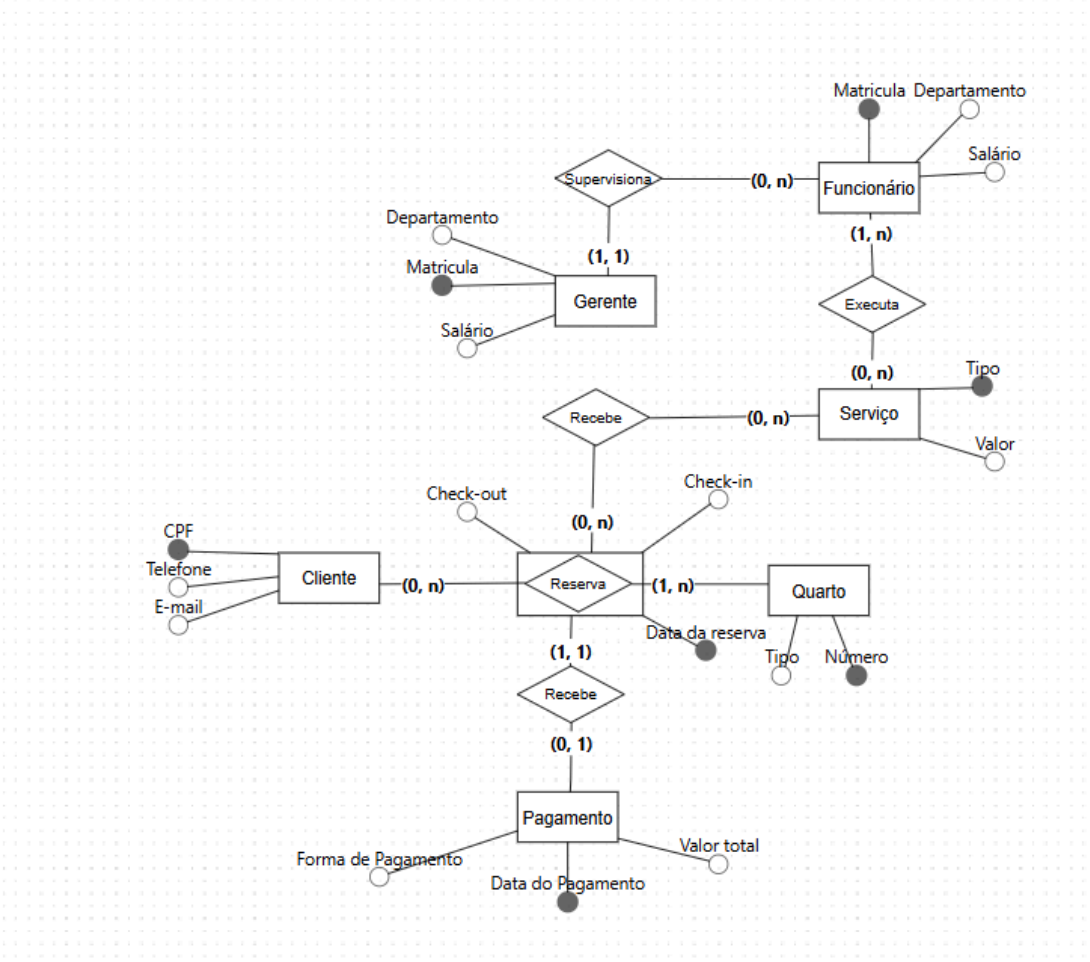


Figura 1. Modelo ER criado para o Gerenciamento de Hotel

A criação do modelo Entidade-Relacionamento (ER) foi guiada por uma análise detalhada do mini-mundo, com o objetivo de traduzir os requisitos de negócio para uma estrutura de dados formal e visualmente clara. A abordagem utilizada focou na identificação das entidades centrais, seus atributos e, crucialmente, os relacionamentos e cardinalidades que as conectam.

1. **Identificação de Entidades e Atributos:** O primeiro passo foi extrair os substantivos-chave do mini-mundo para definir as entidades principais: **Cliente**, **Funcionário**, **Gerente**, **Quarto**, **Serviço** e **Pagamento**. Para cada entidade, foram listados os atributos relevantes, como **CPF** para Cliente e **Número** para Quarto, garantindo que cada peça de informação fosse associada

ao seu "dono" correto. Chaves primárias foram designadas para garantir a unicidade de cada registro.

2. **Definição dos Relacionamentos e Cardinalidades:** A fase mais crítica envolveu o mapeamento das interações entre as entidades. As frases do mini-mundo foram convertidas em verbos de relacionamento: supervisiona, executa, recebe, solicitar, associado. As cardinalidades foram definidas com base nas regras do negócio, como:

   Um **Funcionário** é supervisionado por um Gerente.

   Um **Cliente** pode **solicitar um ou mais** Quartos.

   Um **Pagamento** está associado a **exatamente um** Quarto.

3. **Refinamento do Modelo:** A entidade Salário foi criada para representar o valor recebido por um funcionário. Em vez de ser um atributo da entidade Funcionário, optou-se por modelá-la como uma entidade separada com um relacionamento **(1, n)**. Essa abordagem permite o rastreamento do histórico de salários, o que seria impossível com um simples atributo. Da mesma forma, o relacionamento Solicitar entre Cliente e Serviço foi modelado para refletir a necessidade de um controle granular sobre os serviços consumidos.

O diagrama resultante é a representação gráfica desse processo. Ele fornece uma visão integrada e lógica das operações do hotel, servindo de base para a implementação de um banco de dados robusto e eficiente.

4. Entidades do Modelo ER

As entidades representam os elementos centrais do sistema de gestão hoteleira, correspondendo a objetos ou conceitos do mundo real que possuem informações relevantes e precisam ser armazenadas no banco de dados. Cada entidade contém atributos que descrevem suas características e se relaciona com outras entidades para garantir a integridade e a consistência das informações.

Cliente
Armazena informações sobre os hóspedes do hotel.
Atributos principais: CPF (Identificador), Telefone, E-mail.
Relacionamentos: Um cliente pode **solicitar** um ou mais quartos; cada solicitação de quarto está vinculada a um único cliente.

Funcionário
Contém dados sobre os colaboradores do hotel.
Atributos principais: Matrícula (Identificador), Departamento.
Relacionamentos: Um funcionário **executa** múltiplos serviços; cada serviço

pode ser executado por vários funcionários. Um funcionário também **recebe** um salário, e pode ser supervisionado por um gerente.

Gerente
Representa os funcionários que têm função gerencial no hotel.
Atributos principais: Matrícula (Identificador), Departamento.
Relacionamentos: Um gerente **supervisiona** vários funcionários, garantindo hierarquia e controle operacional.

Quarto
Registra informações sobre os apartamentos do hotel.
Atributos principais: Número (Identificador), Tipo.
Relacionamentos: Um quarto pode ser **solicitado** por clientes em diferentes reservas; cada quarto está **associado** a um pagamento.

Serviço
Armazena detalhes dos serviços oferecidos pelo hotel.
Atributos principais: Tipo, Valor.
Relacionamentos: Um serviço pode ser **executado** por múltiplos funcionários e **solicitado** por vários clientes; cada solicitação é registrada no sistema.

Pagamento
Registra informações sobre pagamentos realizados pelos clientes.
Atributos principais: Data do Pagamento (Identificador), Forma de Pagamento, Valor Total.
Relacionamentos: Cada pagamento está **associado** a um quarto e garante a quitação das reservas e serviços correspondentes.

Essa modelagem permite integrar de forma organizada todas as operações do hotel, evitando redundâncias e inconsistências. O Diagrama Entidade-Relacionamento resultante fornece uma visão clara das conexões entre clientes, funcionários, quartos, serviços e pagamentos, servindo como base para um sistema de gestão eficiente, confiável e escalável.

5. Modelo Lógico (Pé de Galinha)

Nesta seção, será abordado o modelo lógico correspondente ao diagrama Entidade-Relacionamento (ER) apresentado anteriormente. O modelo lógico foi construído a partir das mesmas entidades identificadas no modelo conceitual, preservando suas características principais, mas realizando adaptações necessárias para a implementação em um banco de dados.

Durante o processo de transformação, foram criadas duas novas tabelas próprias (associativas). A primeira delas, denominada “Executa”, tem o objetivo de representar o relacionamento N:N entre as entidades Funcionário e Serviço, garantindo que um funcionário possa executar diversos serviços e que um mesmo serviço possa ser

realizado por diferentes funcionários. A segunda tabela criada, chamada “Recebe”, foi desenvolvida para representar o relacionamento entre as entidades Reserva e Serviço, de forma semelhante, possibilitando o registro dos serviços associados a cada reserva efetuada.

Além disso, a entidade “Reserva”, que no modelo conceitual é uma entidade associativa binária entre Quarto e Cliente, a entidade “Reserva” foi traduzida como uma tabela própria no modelo lógico. Essa conversão possibilita o armazenamento adequado das informações referentes às reservas realizadas, incluindo suas dependências e relacionamentos com outras entidades do sistema.

Na modelagem lógica, buscamos também diminuir o uso de colunas opcionais. Nos relacionamentos 1:N, foi adotada a estratégia de adição de coluna na tabela do lado “muitos”, com o intuito de reduzir a ocorrência de campos nulos e facilitar a implementação do banco de dados em etapas posteriores. Essa escolha se mostrou a mais adequada para esse tipo de relacionamento, considerando a consistência e a simplicidade estrutural.

Por fim, nos relacionamentos 1:1, em que uma das entidades é obrigatória e a outra é opcional, foi utilizada a mesma estratégia de adição de coluna, inserindo a chave estrangeira na entidade opcional. Essa decisão visa manter a coerência do modelo e garantir maior eficiência na integridade referencial e na implementação prática do banco de dados.

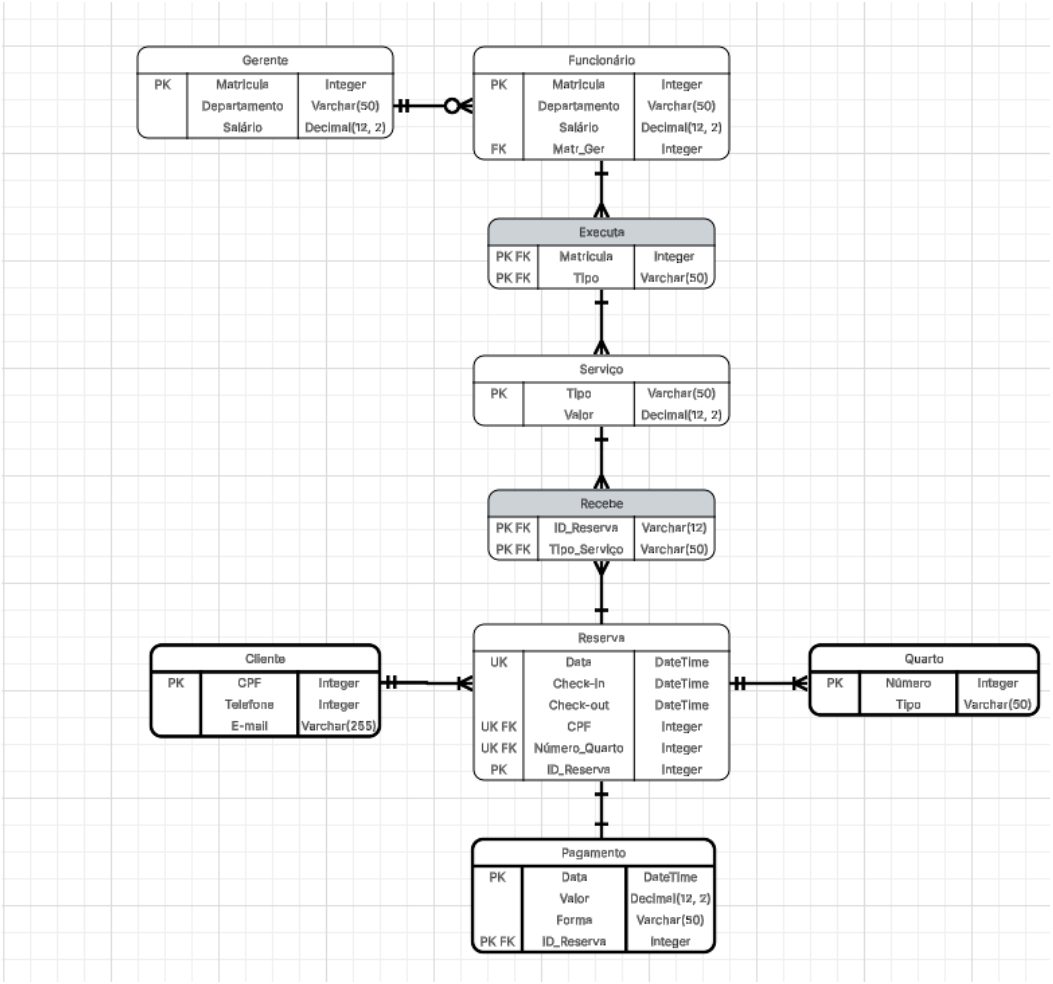


Figura 2. Modelo Lógico criado para o Gerenciamento de Hotel

6. Normalização

6.1. Forma Não Normalizada

FÑN:

```
Hotel(
   Matricula_Ger, Departamento_Ger, Salario_Ger,
      (Matricula_Func, Departamento_Func, Salario_Func,
         (Tipo_Serv, Valor_Serv,
            (ID_Reserva, Data_Reserva, Check_in, Check_out,
               CPF_Cliente, Nome_Cliente, Telefone, Email,
               Numero_Quarto, Tipo_Quarto,
               (ID_Pagamento, Data_Pagamento, Valor_Pagamento, Forma_Pagamento),
               (Tipo_Serv_Recebido)
            )
         )
      )
)
```

Figura 3. Modelo Lógico não normalizado

6.1. 1º Forma Normal

1FN:

```
Gerente (Matricula_Ger, Departamento_Ger, Salario_Ger)
Funcionario (Matricula_Func, Departamento_Func, Salario_Func, Matricula_Ger)
   Matricula_Ger referencia Gerente
Servico (Tipo_Serv, Valor_Serv)
Executa (Matricula_Func, Tipo_Serv)
   Matricula_Func referencia Funcionario
   Tipo_Serv referencia Servico
Cliente (CPF_Cliente, Nome_Cliente, Telefone, Email)
Quarto (Numero_Quarto, Tipo_Quarto)
Reserva (ID_Reserva, Data_Reserva, Check_in, Check_out, CPF_Cliente, Numero_Quarto)
   CPF_Cliente referencia Cliente
   Numero_Quarto referencia Quarto
Recebe (ID_Reserva, Tipo_Serv_Recebido)
   ID_Reserva referencia Reserva
   Tipo_Serv_Recebido referencia Servico
Pagamento (ID_Pagamento, Data_Pagamento, Valor_Pagamento, Forma_Pagamento, ID_Reserva)
   ID_Reserva referencia Reserva
```

Figura 4. Modelo Lógico na primeira forma normal

6.1. 2º Forma Normal

2FN:

Gerente (Matricula_Ger, Departamento_Ger, Salario_Ger)
Funcionario (Matricula_Func, Departamento_Func, Salario_Func, Matricula_Ger)
  Matricula_Ger referencia Gerente
Servico (Tipo_Serv, Valor_Serv)
Executa (Matricula_Func, Tipo_Serv)
  Matricula_Func referencia Funcionario
  Tipo_Serv referencia Servico
Cliente (CPF_Cliente, Nome_Cliente, Telefone, Email)
Quarto (Numero_Quarto, Tipo_Quarto)
Reserva (ID_Reserva, Data_Reserva, Check_in, Check_out, CPF_Cliente, Numero_Quarto)
  CPF_Cliente referencia Cliente
  Numero_Quarto referencia Quarto
Recebe (ID_Recebe, ID_Reserva, Tipo_Serv_Recebido)
  ID_Reserva referencia Reserva
  Tipo_Serv_Recebido referencia Servico
Pagamento (ID_Pagamento, Data_Pagamento, Valor_Pagamento, Forma_Pagamento, ID_Reserva)
  ID_Reserva referencia Reserva

Figura 5. Modelo Lógico na segunda forma normal

6.1. 3º Forma Normal

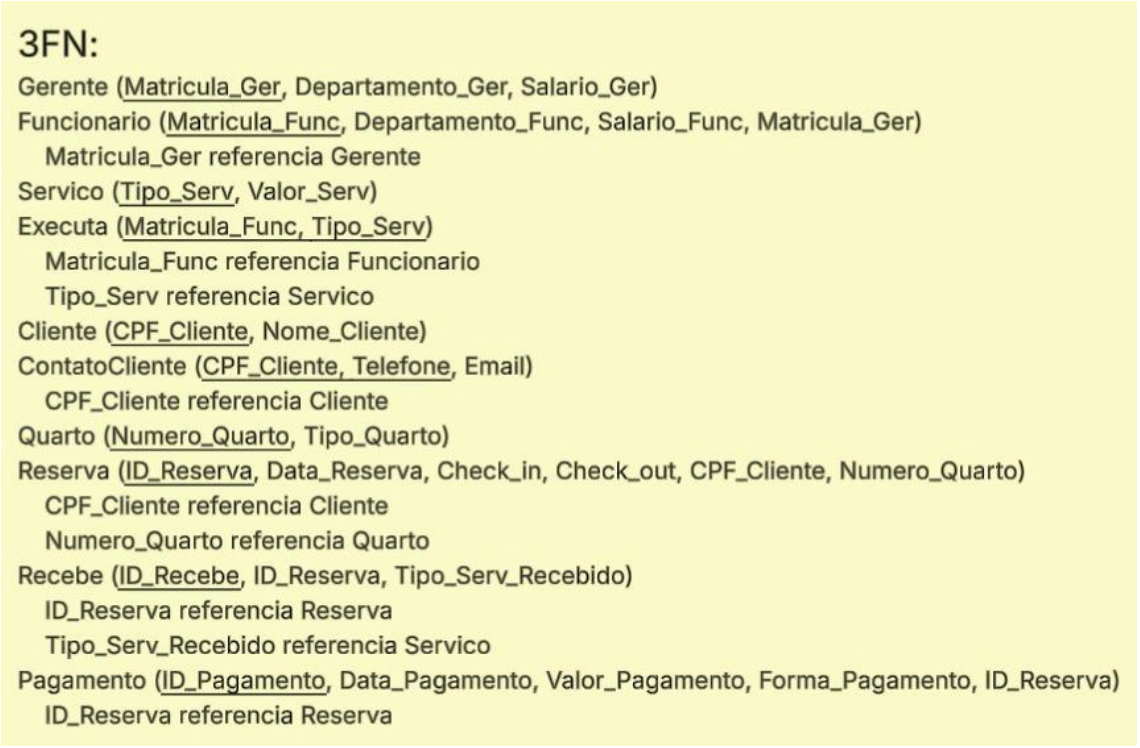


Figura 4. Modelo Lógico na terceira forma normal

A Primeira Forma Normal (1FN) estabelece que todos os valores armazenados nas colunas de uma tabela devem ser atômicos, isto é, indivisíveis, e que cada registro deve ser único.

No modelo inicial, as tabelas Reserva e Pagamento apresentavam o campo *Data* como chave primária. Essa configuração era inadequada, pois, em um ambiente de hotelaria, diversas reservas e pagamentos podem ocorrer na mesma data, impossibilitando a unicidade dos registros.

Para resolver essa limitação, foram criadas chaves primárias artificiais (auto-incremento) em ambas as tabelas:

a tabela Reserva recebeu o campo ID_Reserva, e a tabela Pagamento recebeu o campo ID_Pagamento.
Com isso, cada registro passou a possuir um identificador único, atendendo plenamente às exigências da 1FN.

A Segunda Forma Normal (2FN) exige que, além de o modelo estar em 1FN, nenhum atributo não-chave dependa apenas de parte da chave primária, o que se aplica especialmente às tabelas com chaves compostas.
No caso analisado, a tabela Recebe, responsável por registrar os serviços prestados em cada reserva, apresentava uma chave primária composta por *Data* e *Tipo*, o que gerava dependências parciais.
Determinados atributos dependiam apenas de um dos componentes da chave, e não da combinação completa, o que contrariava os princípios da 2FN.
Para corrigir essa situação, foram feitas duas adequações principais:
primeiro, as tabelas de relacionamento, como Executa (que liga Funcionário e Serviço) e Recebe (que liga Serviço e Reserva), passaram a depender de toda a chave composta;
segundo, optou-se pela criação de identificadores artificiais — como o ID_Recebe — para simplificar as relações e eliminar possíveis dependências parciais.
Essas modificações garantiram que todas as tabelas estivessem de acordo com os requisitos da 2FN.

A Terceira Forma Normal (3FN) determina que, além do cumprimento da 2FN, nenhum atributo não-chave dependa de outro atributo não-chave, evitando assim as chamadas dependências transitivas.
Após a aplicação das correções anteriores, verificou-se que o modelo já se encontrava adequado a essa forma normal.
Tabelas como Gerente, Funcionário, Serviço, Entidade e Quarto não apresentavam esse tipo de dependência.
Por exemplo, na tabela Funcionário, os campos *Salário* e *Departamento* dependem exclusivamente da *Matrícula* — a chave primária — e não entre si, o que confirma o atendimento às exigências da 3FN.

Com essas etapas concluídas, o modelo passou a possuir uma estrutura coerente, livre de redundâncias e com integridade referencial assegurada, garantindo que cada tabela represente de maneira clara uma única entidade ou relacionamento dentro do sistema de reservas.

7. Códigos DDL

Nesta seção do nosso trabalho, iremos mostrar os scripts SQL que realizam a DDL (Data Definition Language), que é fundamental para a criação da estrutura do nosso sistema de gerenciamento de hotel. Isso inclui a criação do esquema, das tabelas, a definição de chaves (primárias e estrangeiras) e a aplicação de diversas restrições.

Abaixo está o script DDL que foi utilizado para estabelecer o banco de dados:

```
/*Iniciando os scripts de DDL */

CREATE SCHEMA trabalhomdi;

USE trabalhomdi;


DROP TABLE IF EXISTS PAGAMENTO;

DROP TABLE IF EXISTS RECEBE;

DROP TABLE IF EXISTS EXECUTA;

DROP TABLE IF EXISTS RESERVA;

DROP TABLE IF EXISTS SERVICO;

DROP TABLE IF EXISTS QUARTO;

DROP TABLE IF EXISTS FUNCIONARIO;

DROP TABLE IF EXISTS GERENTE;

DROP TABLE IF EXISTS CLIENTE;


CREATE TABLE GERENTE (

    MATRICULA INTEGER PRIMARY KEY,

    DEPARTAMENTO VARCHAR(50),

    SALARIO DECIMAL(12,2)

);


CREATE TABLE FUNCIONARIO (
```

```
    MATRICULA INTEGER PRIMARY KEY,
    DEPARTAMENTO VARCHAR(50),
    SALARIO DECIMAL(12,2),
    MATR_GER INTEGER,
    FOREIGN KEY (MATR_GER) REFERENCES GERENTE(MATRICULA)
);

CREATE TABLE CLIENTE (
    CPF VARCHAR(11) PRIMARY KEY,
    TELEFONE VARCHAR(11),
    EMAIL VARCHAR(255)
);

CREATE TABLE QUARTO (
    NUMERO INTEGER PRIMARY KEY,
    TIPO VARCHAR(50)
);

CREATE TABLE SERVICO (
    TIPO VARCHAR(50) PRIMARY KEY,
    VALOR DECIMAL(12,2)
);

CREATE TABLE RESERVA (
    ID_RESERVA INTEGER PRIMARY KEY,
    DATA DATETIME,
```

```
    CHECKIN DATETIME,
    CHECKOUT DATETIME,
    CPF VARCHAR(11),
    NUMERO_QUARTO INTEGER,
    FOREIGN KEY (CPF) REFERENCES CLIENTE(CPF),
    FOREIGN KEY (NUMERO_QUARTO) REFERENCES QUARTO(NUMERO),
    UNIQUE(DATA, NUMERO_QUARTO, CPF)
);

CREATE TABLE EXECUTA (
    MATRICULA INTEGER,
    TIPO_SERVICO VARCHAR(50),
    PRIMARY KEY (MATRICULA, TIPO_SERVICO),
    FOREIGN KEY (MATRICULA) REFERENCES FUNCIONARIO(MATRICULA),
    FOREIGN KEY (TIPO_SERVICO) REFERENCES SERVICO(TIPO)
);

CREATE TABLE RECEBE (
    ID_RESERVA INTEGER,
    TIPO_SERVICO VARCHAR(50),
    PRIMARY KEY (ID_RESERVA, TIPO_SERVICO),
    FOREIGN KEY (ID_RESERVA) REFERENCES RESERVA(ID_RESERVA),
    FOREIGN KEY (TIPO_SERVICO) REFERENCES SERVICO(TIPO)
);

CREATE TABLE PAGAMENTO (
```

DATA DATETIME PRIMARY KEY,

VALOR DECIMAL(12,2),

FORMA VARCHAR(50),

ID_RESERVA INTEGER,

FOREIGN KEY (ID_RESERVA) REFERENCES RESERVA(ID_RESERVA)

);

```sql
/*Iniciando os scripts de DDL */
USE trabalhomdi;

DROP TABLE IF EXISTS PAGAMENTO;
DROP TABLE IF EXISTS RECEBE;
DROP TABLE IF EXISTS EXECUTA;
DROP TABLE IF EXISTS RESERVA;
DROP TABLE IF EXISTS SERVICO;
DROP TABLE IF EXISTS QUARTO;
DROP TABLE IF EXISTS FUNCIONARIO;
DROP TABLE IF EXISTS GERENTE;
DROP TABLE IF EXISTS CLIENTE;

CREATE TABLE GERENTE (
    MATRICULA INTEGER PRIMARY KEY,
    DEPARTAMENTO VARCHAR(50),
    SALARIO DECIMAL(12,2)
);

CREATE TABLE FUNCIONARIO (
    MATRICULA INTEGER PRIMARY KEY,
    DEPARTAMENTO VARCHAR(50),
    SALARIO DECIMAL(12,2),
    MATR_GER INTEGER,
    FOREIGN KEY (MATR_GER) REFERENCES GERENTE(MATRICULA)
);
```

Figura 5. Script do código que realiza DDL

```sql

CREATE TABLE CLIENTE (
    CPF VARCHAR(11) PRIMARY KEY,
    TELEFONE VARCHAR(11),
    EMAIL VARCHAR(255)
);

CREATE TABLE QUARTO (
    NUMERO INTEGER PRIMARY KEY,
    TIPO VARCHAR(50)
);

CREATE TABLE SERVICO (
    TIPO VARCHAR(50) PRIMARY KEY,
    VALOR DECIMAL(12,2)
);

CREATE TABLE RESERVA (
    ID_RESERVA INTEGER PRIMARY KEY,
    DATA DATETIME,
    CHECKIN DATETIME,
    CHECKOUT DATETIME,
    CPF VARCHAR(11),
    NUMERO_QUARTO INTEGER,
    FOREIGN KEY (CPF) REFERENCES CLIENTE(CPF),
    FOREIGN KEY (NUMERO_QUARTO) REFERENCES QUARTO(NUMERO),
    UNIQUE(DATA, NUMERO_QUARTO, CPF)
);
```

Figura 6. Script do código que realiza DDL

```sql

CREATE TABLE EXECUTA (
    MATRICULA INTEGER,
    TIPO_SERVICO VARCHAR(50),
    PRIMARY KEY (MATRICULA, TIPO_SERVICO),
    FOREIGN KEY (MATRICULA) REFERENCES FUNCIONARIO(MATRICULA),
    FOREIGN KEY (TIPO_SERVICO) REFERENCES SERVICO(TIPO)
);

CREATE TABLE RECEBE (
    ID_RESERVA INTEGER,
    TIPO_SERVICO VARCHAR(50),
    PRIMARY KEY (ID_RESERVA, TIPO_SERVICO),
    FOREIGN KEY (ID_RESERVA) REFERENCES RESERVA(ID_RESERVA),
    FOREIGN KEY (TIPO_SERVICO) REFERENCES SERVICO(TIPO)
);

CREATE TABLE PAGAMENTO (
    DATA DATETIME PRIMARY KEY,
    VALOR DECIMAL(12,2),
    FORMA VARCHAR(50),
    ID_RESERVA INTEGER,
    FOREIGN KEY (ID_RESERVA) REFERENCES RESERVA(ID_RESERVA)
);

```

Figura 7. Script do código que realiza DDL

8. Códigos DML

Nesta etapa do projeto, apresentamos os scripts SQL referentes à DML (Data Manipulation Language), que é essencial para a manipulação e o gerenciamento dos dados dentro do nosso sistema de gerenciamento de hotel. Isso inclui a inserção de

registros para popular as tabelas (povoamento de dados), bem como operações de atualização e exclusão necessárias para o funcionamento dinâmico do sistema.

Abaixo estão os scripts DML utilizados para alimentar e interagir com o banco de dados:

```
-- Inserindo Gerente primeiro para referenciar em FUNCIONARIO

INSERT INTO GERENTE VALUES (100, 'Administracao', 5000.00);



/* Inserindo 10x na tabela CLIENTE */


INSERT INTO CLIENTE VALUES ('11111111111', '11988887777',
'ana.silva@email.com');

INSERT INTO CLIENTE VALUES ('22222222222', '21998887766',
'bruno.costa@email.com');

INSERT INTO CLIENTE VALUES ('33333333333', '21997776655',
'carlos.dias@email.com');

INSERT INTO CLIENTE VALUES ('44444444444', '31977665544',
'daniela.souza@email.com');

INSERT INTO CLIENTE VALUES ('55555555555', '21966554433',
'eduardo.lima@email.com');

INSERT INTO CLIENTE VALUES ('66666666666', '11955443322',
'fernanda.alves@email.com');

INSERT INTO CLIENTE VALUES ('77777777777', '21944332211',
'gabriel.torres@email.com');

INSERT INTO CLIENTE VALUES ('88888888888', '21933221100',
'helena.martins@email.com');
```

```
INSERT INTO CLIENTE VALUES ('99999999999', '21922110099',
'igor.ferreira@email.com');

INSERT INTO CLIENTE VALUES ('10101010101', '21911009988',
'julia.rocha@email.com');



/* Inserindo 10x na tabela FUNCIONARIO */


INSERT INTO FUNCIONARIO VALUES (1, 'Recepcao', 2500.00, 100);

INSERT INTO FUNCIONARIO VALUES (2, 'Cozinha', 2200.00, 100);

INSERT INTO FUNCIONARIO VALUES (3, 'Servicos', 2800.00, 100);

INSERT INTO FUNCIONARIO VALUES (4, 'Servicos', 2100.00, 100);

INSERT INTO FUNCIONARIO VALUES (5, 'Recepcao', 2600.00, 100);

INSERT INTO FUNCIONARIO VALUES (6, 'Servicos', 2000.00, 100);

INSERT INTO FUNCIONARIO VALUES (7, 'Seguranca', 3000.00, 100);

INSERT INTO FUNCIONARIO VALUES (8, 'Cozinha', 2200.00, 100);

INSERT INTO FUNCIONARIO VALUES (9, 'Servicos', 2000.00, 100);

INSERT INTO FUNCIONARIO VALUES (10, 'Recepcao', 2500.00, 100);



/* Inserindo 10x na tabela QUARTO */


INSERT INTO QUARTO VALUES (101, 'Padrão');

INSERT INTO QUARTO VALUES (102, 'Padrão');

INSERT INTO QUARTO VALUES (103, 'Luxo');

INSERT INTO QUARTO VALUES (104, 'Luxo');
```

```
INSERT INTO QUARTO VALUES (105, 'Master');

INSERT INTO QUARTO VALUES (201, 'Padrão');

INSERT INTO QUARTO VALUES (202, 'Padrão');

INSERT INTO QUARTO VALUES (203, 'Luxo');

INSERT INTO QUARTO VALUES (204, 'Master');

INSERT INTO QUARTO VALUES (301, 'Presidencial');


/* Inserindo 10x na tabela SERVICO */


INSERT INTO SERVICO VALUES ('Lavanderia', 50.00);

INSERT INTO SERVICO VALUES ('Café da Manhã', 30.00);

INSERT INTO SERVICO VALUES ('Almoço', 60.00);

INSERT INTO SERVICO VALUES ('Jantar', 70.00);

INSERT INTO SERVICO VALUES ('Spa', 150.00);

INSERT INTO SERVICO VALUES ('Piscina Aquecida', 40.00);

INSERT INTO SERVICO VALUES ('Academia', 20.00);

INSERT INTO SERVICO VALUES ('Translado', 80.00);

INSERT INTO SERVICO VALUES ('Frigobar', 25.00);

INSERT INTO SERVICO VALUES ('Bar da Piscina', 35.00);


/* Inserindo 10x na tabela RESERVA */


INSERT INTO RESERVA VALUES (1, '2025-01-10 00:00:00', '2025-01-15 10:00:00',
'2025-01-15 10:00:00', '11111111111', 101);

INSERT INTO RESERVA VALUES (2, '2025-02-02 00:00:00', '2025-02-05 12:00:00',
'2025-02-05 12:00:00', '22222222222', 103);
```

```
INSERT INTO RESERVA VALUES (3, '2025-03-01 00:00:00', '2025-03-10 09:30:00', '2025-03-10 09:30:00', '33333333333', 105);

INSERT INTO RESERVA VALUES (4, '2025-03-20 00:00:00', '2025-03-22 16:20:00', '2025-03-22 16:20:00', '44444444444', 102);

INSERT INTO RESERVA VALUES (5, '2025-04-01 00:00:00', '2025-04-07 14:00:00', '2025-04-07 14:00:00', '55555555555', 104);

INSERT INTO RESERVA VALUES (6, '2025-04-15 00:00:00', '2025-04-17 11:15:00', '2025-04-17 11:15:00', '66666666666', 201);

INSERT INTO RESERVA VALUES (7, '2025-05-02 00:00:00', '2025-05-05 10:50:00', '2025-05-05 10:50:00', '77777777777', 203);

INSERT INTO RESERVA VALUES (8, '2025-05-15 00:00:00', '2025-05-20 18:30:00', '2025-05-20 18:30:00', '88888888888', 204);

INSERT INTO RESERVA VALUES (9, '2025-06-01 00:00:00', '2025-06-03 15:00:00', '2025-06-03 15:00:00', '99999999999', 301);

INSERT INTO RESERVA VALUES (10, '2025-06-10 00:00:00', '2025-06-14 13:00:00', '2025-06-14 13:00:00', '10101010101', 202);


/* Inserindo nas tabelas EXECUTA e RECEBE */


INSERT INTO EXECUTA VALUES (1, 'Lavanderia');

INSERT INTO EXECUTA VALUES (1, 'Café da Manhã');

INSERT INTO EXECUTA VALUES (2, 'Jantar');

INSERT INTO EXECUTA VALUES (3, 'Spa');

INSERT INTO EXECUTA VALUES (4, 'Frigobar');

INSERT INTO EXECUTA VALUES (5, 'Academia');

INSERT INTO EXECUTA VALUES (6, 'Piscina Aquecida');

INSERT INTO EXECUTA VALUES (7, 'Translado');

INSERT INTO EXECUTA VALUES (8, 'Almoço');

INSERT INTO EXECUTA VALUES (9, 'Bar da Piscina');
```

```
INSERT INTO RECEBE VALUES (1, 'Lavanderia');

INSERT INTO RECEBE VALUES (1, 'Café da Manhã');

INSERT INTO RECEBE VALUES (2, 'Jantar');

INSERT INTO RECEBE VALUES (3, 'Spa');

INSERT INTO RECEBE VALUES (4, 'Almoço');

INSERT INTO RECEBE VALUES (5, 'Café da Manhã');

INSERT INTO RECEBE VALUES (6, 'Translado');

INSERT INTO RECEBE VALUES (7, 'Piscina Aquecida');

INSERT INTO RECEBE VALUES (8, 'Frigobar');

INSERT INTO RECEBE VALUES (9, 'Bar da Piscina');


/* Inserindo 10x na tabela PAGAMENTO */


INSERT INTO PAGAMENTO VALUES ('2025-01-15 10:00:00', 1000.00, 'Cartão',
1);

INSERT INTO PAGAMENTO VALUES ('2025-02-05 12:00:00', 800.00, 'Pix', 2);

INSERT INTO PAGAMENTO VALUES ('2025-03-10 09:30:00', 2300.00, 'Cartão',
3);

INSERT INTO PAGAMENTO VALUES ('2025-03-22 16:20:00', 450.00, 'Dinheiro',
4);

INSERT INTO PAGAMENTO VALUES ('2025-04-07 14:00:00', 1500.00, 'Pix', 5);

INSERT INTO PAGAMENTO VALUES ('2025-04-17 11:15:00', 600.00, 'Cartão', 6);

INSERT INTO PAGAMENTO VALUES ('2025-05-05 10:50:00', 900.00, 'Dinheiro',
7);

INSERT INTO PAGAMENTO VALUES ('2025-05-20 18:30:00', 2100.00, 'Cartão',
8);

INSERT INTO PAGAMENTO VALUES ('2025-06-03 15:00:00', 3200.00, 'Pix', 9);
```

```
INSERT INTO PAGAMENTO VALUES ('2025-06-14 13:00:00', 780.00, 'Dinheiro',
10);


-- Comando para desativar o Safe Update Mode

SET SQL_SAFE_UPDATES = 0;


-- 1. Atualização: Mudar o TELEFONE de um cliente específico (CPF 11111111111)

UPDATE CLIENTE

SET TELEFONE = '11999998888'

WHERE CPF = '11111111111';


-- 2. Atualização: Aumentar o SALARIO em 10% para todos os funcionários do
departamento 'Recepcao'

UPDATE FUNCIONARIO

SET SALARIO = SALARIO * 1.10

WHERE DEPARTAMENTO = 'Recepcao';


-- 3. Atualização: Corrigir o VALOR de um pagamento que foi feito com 'Dinheiro' na
reserva 4

UPDATE PAGAMENTO

SET VALOR = 500.00

WHERE FORMA = 'Dinheiro' AND ID_RESERVA = 4;


-- 1. Remoção: Excluir serviços cujo VALOR seja menor que R$ 30,00


-- A. Deletar referências em tabelas filhas

DELETE FROM EXECUTA
```

```
WHERE TIPO_SERVICO IN (SELECT TIPO FROM SERVICO WHERE VALOR < 30.00);


DELETE FROM RECEBE

WHERE TIPO_SERVICO IN (SELECT TIPO FROM SERVICO WHERE VALOR < 30.00);


-- B. Deletar o serviço na tabela pai

DELETE FROM SERVICO

WHERE VALOR < 30.00;


-- 2. Remoção: Excluir o quarto com NUMERO 301


-- A. Deletar as referências nas tabelas filhas (PAGAMENTO e RECEBE)

DELETE FROM PAGAMENTO

WHERE ID_RESERVA IN (SELECT ID_RESERVA FROM RESERVA WHERE NUMERO_QUARTO = 301);


DELETE FROM RECEBE

WHERE ID_RESERVA IN (SELECT ID_RESERVA FROM RESERVA WHERE NUMERO_QUARTO = 301);


-- B. Deletar a RESERVA (tabela pai para pagamento/recebe)

DELETE FROM RESERVA

WHERE NUMERO_QUARTO = 301;


-- C. Deletar o QUARTO (após remover todas as referências)
```

```
DELETE FROM QUARTO
WHERE NUMERO = 301;

-- Mudar o TIPO de quarto para 'Premium' para todos os quartos
-- que foram incluídos em reservas feitas depois de 2025-05-01.

UPDATE QUARTO AS Q
JOIN RESERVA AS R ON Q.NUMERO = R.NUMERO_QUARTO
SET Q.TIPO = 'Premium'
WHERE R.DATA > '2025-05-01 00:00:00';

-- Reativar o Safe Update Mode
SET SQL_SAFE_UPDATES = 1;
```

9. Códigos DQL

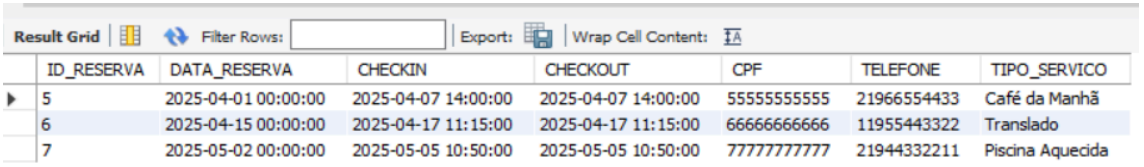
Nesta etapa do projeto, apresentamos os scripts SQL referentes à DQL (Data Query Language), que é a linguagem essencial para a recuperação e análise de dados dentro do nosso sistema de gerenciamento de hotel.

As consultas apresentadas a seguir demonstram como extraímos e transformamos os dados inseridos, obtendo visões operacionais e estratégicas sobre o funcionamento do hotel:

```
-- Relatório de serviços recebidos num período

SELECT R.ID_RESERVA, R.DATA AS DATA_RESERVA, R.CHECKIN,
R.CHECKOUT, C.CPF, C.TELEFONE, REC.TIPO_SERVICO
FROM RECEBE REC
```

```
JOIN RESERVA R ON REC.ID_RESERVA = R.ID_RESERVA

JOIN CLIENTE C ON R.CPF = C.CPF

WHERE R.DATA BETWEEN '2025-04-01' AND '2025-05-31'

ORDER BY R.DATA;


-- Relatório de receita por mês


SELECT substr(P.DATA,1,7) AS ANO_MES, COUNT(*) AS
NUM_PAGAMENTOS, SUM(P.VALOR) AS TOTAL_RECEITA

FROM PAGAMENTO P

GROUP BY ANO_MES

ORDER BY ANO_MES;


-- Consulta individual (cpf do cliente)


SELECT * FROM CLIENTE WHERE CPF = '11111111111';


-- Consulta individual (detalhes da reserva id da reserva=3)


SELECT R.*, C.TELEFONE, Q.TIPO AS TIPO_QUARTO

FROM RESERVA R

JOIN CLIENTE C ON R.CPF = C.CPF

JOIN QUARTO Q ON R.NUMERO_QUARTO = Q.NUMERO

WHERE R.ID_RESERVA = 3;


-- Consulta de serviços executados por funcionário (matricula=1)
```

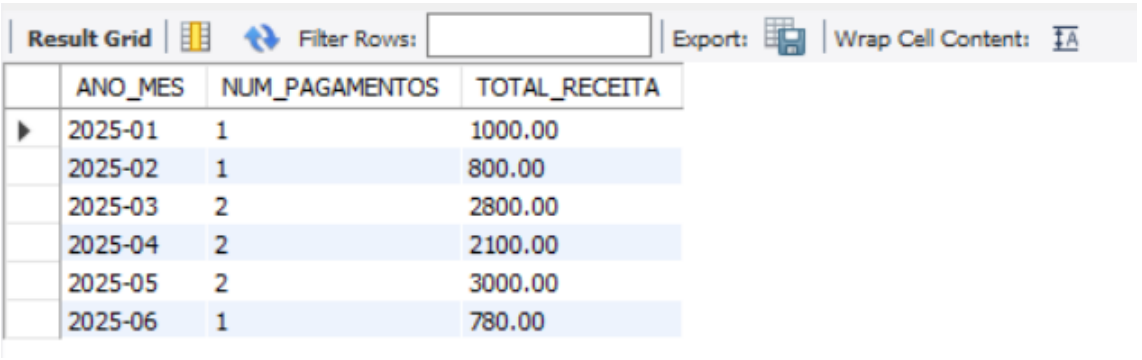
```
SELECT F.MATRICULA, F.DEPARTAMENTO, E.TIPO_SERVICO, S.VALOR

FROM EXECUTA E

JOIN FUNCIONARIO F ON E.MATRICULA = F.MATRICULA

JOIN SERVICO S ON E.TIPO_SERVICO = S.TIPO

WHERE F.MATRICULA = 1;


-- Consulta de pagamento com info do cliente e do quarto


SELECT P.DATA AS DATA_PAGAMENTO, P.VALOR, P.FORMA,
R.ID_RESERVA, R.DATA AS DATA_RESERVA, C.CPF, C.TELEFONE,
R.NUMERO_QUARTO

FROM PAGAMENTO P

JOIN RESERVA R ON P.ID_RESERVA = R.ID_RESERVA

JOIN CLIENTE C ON R.CPF = C.CPF

ORDER BY P.DATA;
```



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

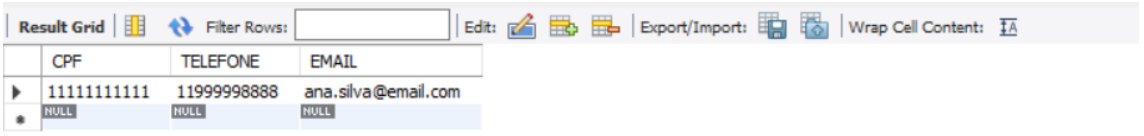
ID_RESERVA	DATA_RESERVA	CHECKIN	CHECKOUT	CPF	TELEFONE	TIPO_SERVICO
5	2025-04-01 00:00:00	2025-04-07 14:00:00	2025-04-07 14:00:00	55555555555	21966554433	Café da Manhã
6	2025-04-15 00:00:00	2025-04-17 11:15:00	2025-04-17 11:15:00	66666666666	11955443322	Translado
7	2025-05-02 00:00:00	2025-05-05 10:50:00	2025-05-05 10:50:00	77777777777	21944332211	Piscina Aquecida

Figura 8. Resultado das consultas



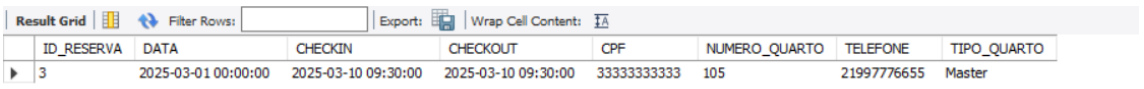
	ANO_MES	NUM_PAGAMENTOS	TOTAL_RECEITA
▶	2025-01	1	1000.00
	2025-02	1	800.00
	2025-03	2	2800.00
	2025-04	2	2100.00
	2025-05	2	3000.00
	2025-06	1	780.00

Figura 9. Resultado das consultas



	CPF	TELEFONE	EMAIL
▶	11111111111	11999998888	ana.silva@email.com
*	NULL	NULL	NULL

Figura 10. Resultado das consultas



	ID_RESERVA	DATA	CHECKIN	CHECKOUT	CPF	NUMERO_QUARTO	TELEFONE	TIPO_QUARTO
▶	3	2025-03-01 00:00:00	2025-03-10 09:30:00	2025-03-10 09:30:00	33333333333	105	21997776655	Master

Figura 11. Resultado das consultas

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

MATRICULA	DEPARTAMENTO	TIPO_SERVICO	VALOR
1	Recepcao	Café da Manhã	30.00
1	Recepcao	Lavanderia	50.00

Figura 12. Resultado das consultas

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

DATA_PAGAMENTO	VALOR	FORMA	ID_RESERVA	DATA_RESERVA	CPF	TELEFONE	NUMERO_QUARTO
2025-01-15 10:00:00	1000.00	Cartão	1	2025-01-10 00:00:00	11111111111	11999998888	101
2025-02-05 12:00:00	800.00	Pix	2	2025-02-02 00:00:00	22222222222	21998887766	103
2025-03-10 09:30:00	2300.00	Cartão	3	2025-03-01 00:00:00	33333333333	21997776655	105
2025-03-22 16:20:00	500.00	Dinheiro	4	2025-03-20 00:00:00	44444444444	31977665544	102
2025-04-07 14:00:00	1500.00	Pix	5	2025-04-01 00:00:00	55555555555	21966554433	104
2025-04-17 11:15:00	600.00	Cartão	6	2025-04-15 00:00:00	66666666666	11955443322	201
2025-05-05 10:50:00	900.00	Dinheiro	7	2025-05-02 00:00:00	77777777777	21944332211	203
2025-05-20 18:30:00	2100.00	Cartão	8	2025-05-15 00:00:00	88888888888	21933221100	204
2025-06-14 13:00:00	780.00	Dinheiro	10	2025-06-10 00:00:00	10101010101	21911009988	202

Figura 13. Resultado das consultas

10. Conclusão

O desenvolvimento do modelo Entidade-Relacionamento (ER) para o Sistema de Gerenciamento de Hotel representou o ponto de partida para a estruturação de um

banco de dados coerente e eficiente. A partir da análise dos requisitos do mini-mundo, o modelo ER permitiu compreender e representar as principais entidades do sistema além de seus relacionamentos, garantindo uma visão clara das operações e interações existentes no ambiente hoteleiro.

Em seguida, o modelo lógico foi construído com base no modelo ER, traduzindo os conceitos conceituais em uma estrutura relacional compatível com os Sistemas Gerenciadores de Banco de Dados (SGBDs). Nessa etapa, foram criadas tabelas associativas, como Executa e Recebe, para representar relacionamentos muitos-para-muitos, além da definição de chaves primárias e estrangeiras que asseguram a integridade referencial e a consistência das informações. O modelo lógico também buscou otimizar a estrutura das tabelas por meio da adição de colunas em relacionamentos 1:N e 1:1, reduzindo campos opcionais e facilitando a implementação prática do sistema.

Por fim, o processo de normalização foi aplicado com o objetivo de eliminar redundâncias, evitar anomalias de inserção, atualização e exclusão, e garantir maior eficiência no armazenamento e recuperação dos dados. A aplicação das formas normais contribuiu para um modelo mais limpo, organizado e aderente às boas práticas de modelagem relacional. Concluindo, o modelo pé de galinha normalizado foi transposto para o ambiente SQL, onde foram desenvolvidos e testados os scripts DDL (Linguagem de Definição de Dados) para a criação da estrutura do banco de dados, os scripts DML (Linguagem de Manipulação de Dados) para a inserção de dados de teste, e os scripts DQL (Linguagem de Consulta de Dados) para a recuperação e análise das informações.

Referência

Heuser, C. A. (2009) *Projeto de Banco de Dados*. Editora Bookman, Porto Alegre.

Dataedo (2021) “Best Practices for Entity Relationship Diagrams (ERD)”, https://docs.dataedo.com/data-catalog/database-diagrams/erd-best-practices/, March.

Gautam, T. and Gaurav, S. (2022) “A Research on Hotel Management System”, *International Journal of Research in Engineering, Science and Management (IJRESM)*, Vol. 5, No. 7, July, p. 122–124.

GeeksforGeeks (2020) “Best Practices for Documenting Database Design”, https://www.geeksforgeeks.org/best-practices-for-documenting-database-design/, October.