INSTITUTO FEDERAL DA PARAÍBA - IFPB
CAMPUS CAMPINA GRANDE
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO
DISCIPLINA DE BANCO DE DADOS

ANA CAROLINA GALDINO PORTO
GUILHERME SARMENTO FARIAS MERGULHÃO
JOÃO VICTOR ROCHA DA COSTA
JOSUÉ FERREIRA DE SOUZA JÚNIOR
MARIA EDUARDA CORREIA SOUSA
MARIANE NUNES DE OLIVEIRA

ARQUITETURA DE SISTEMA PARA GERENCIAMENTO DE CLIENTES E RESERVAS EM RESTAURANTES

CAMPINA GRANDE - PB
2025

SUMÁRIO

1 INTRODUÇÃO

Neste projeto criaremos em detalhes um sistema de banco de dados para o gerenciamento de um aplicativo de reservas de restaurante. Serão apresentadas todas as etapas de modelagem dos dados, passando pelos níveis conceitual, lógico e físico. Por fim, será realizado o povoamento do banco de dados, bem como atualizações e consultas. Para começar o desenvolvimento do projeto, os parágrafos a seguir são dedicados à descrição do mini-mundo. Uma rede de restaurantes que opera em várias cidades brasileiras (como João Pessoa, Campina Grande e Patos) está desenvolvendo um sistema de banco de dados para gerenciar o processo completo de atendimento ao cliente, desde a reserva até a avaliação. O sistema visa integrar o controle de reservas, o gerenciamento de mesas, a administração da lista de espera e o registro de avaliações. Deseja-se que os clientes possam realizar reservas antecipadas em um restaurante da rede, informando a data da reserva, a hora da reserva e a quantidade de pessoas. É crucial que o sistema associe esta reserva a uma mesa específica que seja compatível com a capacidade. O sistema deve registrar detalhadamente o fluxo de atendimento da reserva, incluindo o horário de chegada e o horário de saída para a análise de ocupação da mesa. Para clientes que chegam sem reserva, é vital que o sistema possa registrá-los em uma lista de espera específica do restaurante, contendo a data de entrada, hora da entrada e a quantidade de pessoas. O sistema deve controlar o status da fila (Aguardando, Chamado, Desistiu). O sistema deve armazenar as informações básicas dos clientes (nome, telefone e email), garantindo que tanto o telefone quanto o e-mail sejam únicos. Ademais, após o atendimento, o sistema deve permitir que o cliente registre uma Avaliação sobre o serviço, informando uma nota decimal e um comentário, junto com a data de avaliação. O banco de dados deve controlar os relacionamentos entre todas as entidades (Cliente, Reserva, Mesa, Restaurante, Lista de Espera, Avaliação), garantindo que o funcionamento do processo de atendimento seja completo e rastreável.

2 PROJETO CONCEITUAL

O Projeto Conceitual é a primeira fase do desenvolvimento de um banco de dados. Seu objetivo é representar a estrutura lógica da informação do mundo real, de forma independente de qualquer tecnologia, linguagem SQL ou SGBD. Nesse nível, o projeto consiste apenas em modelar a realidade definindo elementos importantes que são as entidades e seus atributos e como elas vão estar relacionadas entre si.

Nesse nível de abstração há uma ferramenta que auxilia na modelagem do banco de dados que é chamada Modelo Entidade Relacionamento (MER) (PASSOS, 2025),

no projeto atual foi utilizado o software BRMODELO(SIS4, 2025)

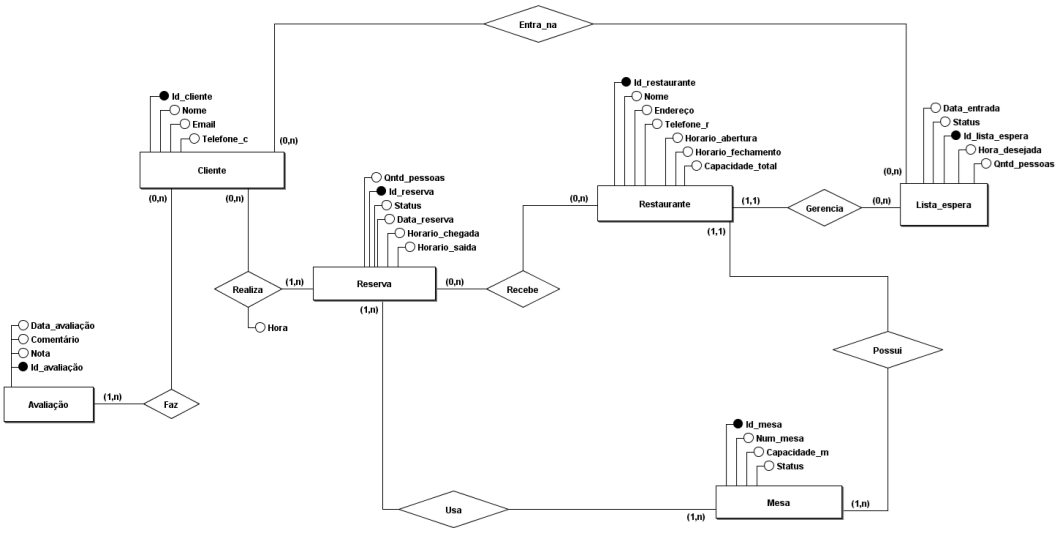

Figura 1: Modelo Entidade-Relacionamento do Projeto Conceitual

3 PROJETO LÓGICO

O projeto lógico trata-se de um nível intermediário entre o alto nível de abstração e o nível físico que é quando a implementação dos códigos é realizada. Esse nível consiste em definir melhor como as entidades vão estar relacionadas. Os conceitos de chaves primárias e estrangeiras é fundamental nessa etapa pois é a partir delas que a conexão entre as tabelas é feita e que conseguimos a partir de uma tabela obter informações de outras.

3.1 Modelo Relacional

Assim como no modelo conceitual tinhamos o Modelo Entidade Relacionamento (MER), no nível lógico temos o Modelo Relacional (MR) ele possui regras que são utilizadas para fazer o chaveamento entre as tabelas com o que chamamos de Esquemas e Relação, os quais podem ser vistos abaixo.

- Cliente(Id_cliente, Nome, Telefone_c, E_mail)
- Avalicao(Id_avaliacao, Nota, Comentario, Data_avaliacao)
- Reserva(Id_reserva, Data_reserva, Horario_chegada, Horario_saida, Horario_reserva)

- Restaurante(Id_restaurante, Nome, Rua, Complemento, Bairro, CEP, Cidade, Horario_abertura, Horario_fechamento, Telefone_r)
- Mesa(Id_mesa, Num_mesa, Capacidade_m, Status)
- Lista_espera(Data_entrada, Status, Id_lista_espera, Hora_entrada, Qntd_pessoas)
- AVALIACAO_CLIENTE(Id_avaliação_a_fk, Id_cliente_c_fk)
- CLIENTE_RESERVA(Id_cliente_c_fk,Id_reserva_r_fk)
- CLIENTE_LISTA_ESPERA(Id_cliente_c_fk, Id_lista_espera_l_fk)
- RESERVA_RESTAURANTE(Id_restaurante_r_fk,Id_reserva_r_fk)
- RESERVA_MESA(Id_reserva_r_fk, Id_mesa_m_fk)

3.2 Dicionário de Dados

Atributo	**Tipo**	**Nulo**	**Descrição**	**Domínio**	**PK**	**FK**	**CANK**
id_cliente	inteiro	não	Identificador do cliente	> 0	X		
nome	texto(100)	não	Nome do cliente	Caracteres			
telefone	texto(15)	não	Telefone de contato	Caracteres			X
email	texto(100)	não	Email de contato	Caracteres			X

Tabela 1: Cliente

Atributo	**Tipo**	**Nulo**	**Descrição**	**Domínio**	**PK**	**FK**	**CANK**
id_avaliacao	inteiro	não	Identificador da avaliação	> 0	X		
nota	decimal(2,1)	não	Nota atribuída pelo cliente	0-5			
comentario	texto(255)	sim	Comentário sobre o restaurante	Caracteres			
data_avaliacao	data	não	Data de avaliação	AAAA-MM-DD			

Tabela 2: Avaliação

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_reserva	inteiro	não	Identificador da reserva	> 0	X		
data_reserva	data	não	Data da reserva	AAAA:MM:DD			
hora_reserva	time	não	Horário que foi reservado	HH:MM			
horario_chegada	time	não	Horário de Chegada	HH:MM			
horario_saida	time	não	Horário de Saída	HH:MM			

Tabela 3: Reserva

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_restaurante	inteiro	não	Identificador do Restaurante	> 0	X		
nome	texto(100)	não	Nome do Restaurante	Caracteres			
rua	texto(150)	sim	Logradouro	Caracteres			
bairro	texto(100)	sim	Bairro que está localizado	Caracteres			
cidade	texto(100)	sim	Cidade que está localizado	Caracteres			
telefone_r	texto(15)	sim	Telefone do Estabelecimento	Caracteres			X
horario_abertura	time	não	Horário de Abertura do Restaurante	HH:MM			
horario_fechamento	time	não	Horário de Fechamento do Restaurante	HH:MM			

Tabela 4: Restaurante

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_mesa	inteiro	não	Identificador da mesa	> 0	X		
num_mesa	inteiro	não	Numeração da Mesa	> 0			
capacidade_m	inteiro	não	Capacidade de Pessoas da Mesa	> 0			
status	texto(30)	sim	Status de Ocupação da Mesa	Caracteres			

Tabela 5: Mesa

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_lista_espera	inteiro	não	Identificador da Lista de Espera	> 0	X		
data_entrada	data	não	Data que o cliente entrou na fila	AAAA:MM:DD		X	
hora_entrada	time	não	Hora que entrou na lista	HH:MM		X	
status	texto(30)	não	Aguardando, Cancelado, Chamado	Caracteres			
qntd_pessoas	inteiro	não	Quantidade de pessoas na mesa	> 0			

Tabela 6: Lista de Espera

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_cliente_fk	inteiro	não	Referência ao Cliente	> 0	X	X	
id_avaliacao_fk	inteiro	sim	Referência à Avaliação	> 0	X	X	

Tabela 7: Avaliação Cliente

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_cliente_fk	inteiro	não	Referência ao Cliente	> 0	X	X	
id_reserva_fk	inteiro	não	Referência a Reserva	> 0	X	X	

Tabela 8: Cliente Reserva

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_cliente_fk	inteiro	não	Referência ao Cliente	> 0	X	X	
id_lista_espera_fk	inteiro	não	Referência a Lista de Espera	> 0	X	X	

Tabela 9: Cliente Lista de Espera

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_reserva_fk	inteiro	não	Referência a Reserva	> 0	X	X	
id_restaurante_fk	inteiro	não	Referência ao Restaurante	> 0	X	X	

Tabela 10: Reserva Restaurante

Atributo	Tipo	Nulo	Descrição	Domínio	PK	FK	CANK
id_reserva_fk	inteiro	não	Referência a Reserva	> 0	X	X	
id_mesa_fk	inteiro	não	Referência a Mesa	> 0	X	X	

Tabela 11: Reserva Mesa

4 PROJETO FÍSICO

A etapa de implementação física foi iniciada. A estrutura do banco de dados projetado foi codificada em linguagem SQL e efetivamente implantada no SGBD MySQL.

4.1 Scripts de Criação

O script reúne todos os comandos essenciais para estabelecer a base de dados, definir as tabelas e aplicar as regras de integridade (constraints).

4.1.1 Criação do Banco de Dados

Nesta etapa foi realizada a criação de um banco de dados com o nome `app_reserva_restaurante`, utilizando o comando CREATE DATABASE. É o local em que estará o contêiner lógico que abrigará todas as tabelas, dados e regras. Em seguida estamos habilitando usando o comando USE.

```
1 CREATE DATABASE app_reserva_restaurante;
2 USE app_reserva_restaurante;
```

4.1.2 Criação das Tabelas

Nesta etapa, estamos realizando à criação das tabelas que compõem o nosso banco de dados, seguindo a estrutura definida nos modelos conceitual e lógico. Utilizaremos o comando CREATE TABLE para este fim. Foram criadas as tabelas CLIENTE, AVALIAÇÃO, RESERVA, RESTAURANTE, MESA e LISTA DE ESPERA.

Cliente

A tabela Cliente representa os usuários que utilizam o sistema para realizar reservas em restaurantes. Armazena as informações básicas necessárias para identificação e contato, permitindo o gerenciamento completo do relacionamento com o cliente.

```
CREATE TABLE cliente (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    telefone VARCHAR(15) UNIQUE,
    email VARCHAR(150) UNIQUE
);
```

Avaliação

A tabela Avaliação armazena informações sobre avaliações de clientes, incluindo nota, comentário e data, e cada avaliação é identificada de forma única pelo id_avaliacao.

```
CREATE TABLE avaliacao (
    id_avaliacao INT AUTO_INCREMENT PRIMARY KEY,
    nota DECIMAL(2,1) NOT NULL,
    comentario VARCHAR(255),
    data_avaliacao DATE NOT NULL
);
```

Reserva

A tabela Reserva registra todas as marcações realizadas pelos clientes. É o núcleo principal do sistema, permitindo controlar horários, mesas, disponibilidade e fluxo de clientes.

```
CREATE TABLE reserva (
    id_reserva INT AUTO_INCREMENT PRIMARY KEY,
    data_reserva DATE NOT NULL,
    horario_chegada TIME NOT NULL,
    horario_saida TIME NOT NULL,
    hora_reserva TIME NOT NULL
);
```

Restaurante

A tabela Restaurante contém os dados dos estabelecimentos cadastrados no sistema que oferecem serviço de reserva e atendimento aos clientes. Essas informações permitem o gerenciamento da estrutura física e operacional de cada restaurante.

```
CREATE TABLE restaurante (
    id_restaurante INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    rua VARCHAR(150),
    bairro VARCHAR(100),
    cidade VARCHAR(100),
    horario_abertura TIME NOT NULL,
    horario_fechamento TIME NOT NULL,
    telefone_r VARCHAR(15) UNIQUE
);
```

Mesa

A tabela Mesa representa as mesas disponíveis em cada restaurante. Cada mesa possui capacidade e status, permitindo ao sistema alocar clientes de forma eficiente.

```
CREATE TABLE mesa (
    id_mesa INT AUTO_INCREMENT PRIMARY KEY,
    num_mesa INT NOT NULL,
    capacidade_m INT NOT NULL,
    status VARCHAR(30) NOT NULL
);
```

Lista de Espera

A tabela Lista_espera registra clientes que não conseguiram uma mesa ou reserva imediata e aguardam disponibilidade.

```
CREATE TABLE lista_espera (
    id_lista_espera INT AUTO_INCREMENT PRIMARY KEY,
    data_entrada DATE NOT NULL,
    hora_entrada TIME NOT NULL,
    status VARCHAR(30) NOT NULL,
    qntd_pessoas INT NOT NULL
);
```

4.1.3 Criação das Tabelas Associativas

Em seguida, focamos na implementação dos relacionamentos Muitos para Muitos (N:M), que exigem a criação de Tabelas Associativas (ou de Ligação). Essas tabelas são essenciais para conectar os dados de duas tabelas primárias.

Tabela Associativa	Relacionamento Mapeado
avaliacao_cliente	associa a AVALIAÇÃO ao CLIENTE
cliente_reserva	associa o CLIENTE a RESERVA
cliente_lista_espera	associa o CLIENTE a LISTA DE ESPERA
reserva_restaurante	associa a RESERVA ao RESTAURANTE onde foi feita
reserva_mesa	associa a RESERVA a MESA específica alocada

Avaliacao_Cliente

A tabela avaliacao_cliente relaciona clientes às suas avaliações, permitindo saber quais clientes fizeram quais avaliações, implementando um relacionamento muitos-para-muitos entre cliente e avaliação.

```
CREATE TABLE avaliacao_cliente (
    id_avaliacao_fk INT,
    id_cliente_fk INT,
    PRIMARY KEY (id_avaliacao_fk, id_cliente_fk),
    FOREIGN KEY (id_avaliacao_fk) REFERENCES avaliacao(
       id_avaliacao),
    FOREIGN KEY (id_cliente_fk) REFERENCES cliente(id_cliente
       )
);
```

Cliente_Reserva

A tabela cliente_reserva relaciona clientes às suas reservas, permitindo saber quais clientes realizaram quais reservas, implementando um relacionamento muitos-para-muitos entre cliente e reserva.

```
CREATE TABLE cliente_reserva (
    id_cliente_fk INT,
    id_reserva_fk INT,
    PRIMARY KEY (id_cliente_fk, id_reserva_fk),
    FOREIGN KEY (id_cliente_fk) REFERENCES cliente(id_cliente
       ),
    FOREIGN KEY (id_reserva_fk) REFERENCES reserva(id_reserva
       )
);
```

Cliente_Lista_Espera

A tabela cliente_lista_espera permite relacionar clientes às listas de espera em que estão registrados, implementando um relacionamento muitos-para-muitos entre cliente e lista_espera.

```
CREATE TABLE cliente_lista_espera (
    id_cliente_fk INT,
    id_lista_espera_fk INT,
    PRIMARY KEY (id_cliente_fk, id_lista_espera_fk),
    FOREIGN KEY (id_cliente_fk) REFERENCES cliente(id_cliente
        ),
    FOREIGN KEY (id_lista_espera_fk) REFERENCES lista_espera(
        id_lista_espera)
);
```

Reserva_Restaurante

A tabela reserva_restaurante permite relacionar reservas aos restaurantes correspondentes, implementando um relacionamento muitos-para-muitos entre reserva e restaurante.

```
CREATE TABLE reserva_restaurante (
    id_reserva_fk INT,
    id_restaurante_fk INT,
    PRIMARY KEY (id_reserva_fk, id_restaurante_fk),
    FOREIGN KEY (id_reserva_fk) REFERENCES reserva(id_reserva
        ),
    FOREIGN KEY (id_restaurante_fk) REFERENCES restaurante(
        id_restaurante)
);
```

Reserva_Mesa

A tabela reserva_mesa permite relacionar reservas às mesas correspondentes, implementando um relacionamento muitos-para-muitos entre reserva e mesa.

```
CREATE TABLE reserva_mesa (
    id_reserva_fk INT,
    id_mesa_fk INT,
    PRIMARY KEY (id_reserva_fk, id_mesa_fk),
    FOREIGN KEY (id_reserva_fk) REFERENCES reserva(id_reserva
        ),
    FOREIGN KEY (id_mesa_fk) REFERENCES mesa(id_mesa)
);
```

4.2 Alimentação Inicial do Banco de Dados

Nesta etapa, o foco é na inserção dos dados iniciais nas tabelas recém-criadas. Utilizamos o comando INSERT INTO da linguagem SQL para introduzir dez registros em cada uma das tabelas principais e, subsequentemente, nas tabelas associativas.

4.2.1 Alimentação das Tabelas Principais

As tabelas de entidades (CLIENTE, AVALIAÇÃO, RESERVA, RESTAURANTE, MESA e LISTA DE ESPERA) foram preenchidas com informações representativas, conforme listado nos scripts (seção 4.1.2).

Cliente

A tabela Cliente foi povoada com registros que representam os perfis de usuários do sistema. Foram inseridos clientes fictícios contendo nome, e-mail e telefone, simulando um conjunto realista de consumidores que utilizariam o serviço para realizar reservas. Esses dados permitem testar operações de cadastro, consultas e relacionamentos com reservas e avaliações.

```
INSERT INTO cliente (nome, telefone_c, email) VALUES
    ('João Silva','83988562500','joao@email.com'),
    ('Maria Oliveira','83975586604','maria@email.com'),
    ('Carlos Souza','83984026370','carlos@email.com'),
    ('Ana Pereira','83994526600','ana@email.com'),
    ('Pedro Santos','83997426529','pedro@email.com'),
    ('Lucas Lima','81997230145','lucas@email.com'),
    ('Julia Costa','84998564521','julia@email.com'),
    ('Marcos Rocha','85992564022','marcos@email.com'),
    ('Patricia Dias','83994562879','patricia@email.com'),
    ('Rafael Mota','81991620044','rafael@email.com');
```

Avaliações

A tabela Avaliação foi preenchida com comentários e notas dadas por clientes que já utilizaram o serviço de reserva. Foram incluídas avaliações positivas e negativas, com diferentes datas, permitindo testar relatórios de satisfação, ranking de restaurantes e funcionalidades analíticas.

```
INSERT INTO avaliacao (nota, comentario, data_avaliacao)
   VALUES
    (5.0,'Excelente!','2023-10-01'),
    (4.5,'Muito bom','2023-10-02'),
    (3.0,'Razoável','2023-10-03'),
    (2.0,'Atendimento lento','2023-10-04'),
    (5.0,'Comida perfeita','2023-10-05'),
    (4.0,'Ambiente agradável','2023-10-06'),
    (1.0,'Péssima experiência','2023-10-07'),
    (3.5,'Preço ok','2023-10-08'),
    (5.0,'Voltarei sempre','2023-10-09'),
    (4.5,'Sobremesa incrível','2023-10-10');
```

Reservas

A tabela Reserva foi povoada com registros representando agendamentos realizados por clientes. As reservas incluem datas futuras e passadas, horários distintos de chegada e saída. O objetivo foi testar a lógica do sistema, incluindo disponibilidade, ocupação e histórico de uso.

```
1  INSERT INTO reserva (data_reserva, horario_chegada,
      horario_saida, hora_reserva) VALUES
2     ('2023-12-01','19:00','21:00','19:00'),
3     ('2023-12-01','20:00','22:00','20:00'),
4     ('2023-12-02','12:00','14:00','12:00'),
5     ('2023-12-02','13:00','15:00','13:00'),
6     ('2023-12-03','19:30','21:30','19:30'),
7     ('2023-12-03','20:30','22:30','20:30'),
8     ('2023-12-04','18:00','20:00','18:00'),
9     ('2023-12-04','21:00','23:00','21:00'),
10    ('2023-12-05','12:30','14:30','12:30'),
11    ('2023-12-05','13:30','15:30','13:30');
```

Restaurantes

A tabela Restaurante foi preenchida com estabelecimentos fictícios com características distintas, incluindo horários de funcionamento, localização e dados de contato. O objetivo foi simular diferentes cenários operacionais, testando reservas, mesas e listas de espera em diferentes restaurantes e localidades.

```
1  INSERT INTO restaurante (nome, rua, bairro, cidade,
       horario_abertura, horario_fechamento, telefone_r) VALUES
2    ('Cantina Italiana','Rua das Trincheiras','Centro','João
         Pessoa','11:00','23:00','83999990001'),
3    ('Burger King King','Av. Epitácio Pessoa','Tambauzinho','
         João Pessoa','10:00','00:00','83999990002'),
4    ('Sushi House','Rua Bancário Sérgio Guerra','Bancários','
         João Pessoa','18:00','23:00','83999990003'),
5    ('Churrascaria Boi','Av. Floriano Peixoto','Centro','
         Campina Grande','11:00','16:00','83999990004'),
6    ('Vegan Life','Rua João Domingos','Manaíra','João Pessoa'
         ,'10:00','20:00','83999990005'),
7    ('Taco Mexicano','Rua Aprígio Veloso','Universitário','
         Campina Grande','17:00','01:00','83999990006'),
8    ('Pizza Express','Rua Coremas','Jaguaribe','João Pessoa',
         '18:00','00:00','83999990007'),
9    ('Bistro Frances','Av. Cabo Branco','Cabo Branco','João
         Pessoa','12:00','22:00','83999990008'),
10   ('Cafe Colonial','Rua Sólon de Lucena','Centro','João
         Pessoa','08:00','18:00','83999990009'),
11   ('Boteco do Ze','Rua Epitácio Pessoa','Centro','Patos','
         16:00','02:00','83999990010');
```

Mesas

A tabela Mesa foi preenchida com mesas distribuídas entre os restaurantes cadastrados. Cada mesa recebeu um número identificador, uma capacidade e um status inicial (como disponível ou ocupada). Os dados simulam a estrutura física dos restaurantes, permitindo testar alocação automática e o relacionamento entre reservações e mesas.

```
INSERT INTO mesa (num_mesa, capacidade_m, status) VALUES
    (1, 2, 'Livre'),
    (2, 2, 'Ocupada'),
    (3, 4, 'Reservada'),
    (4, 4, 'Livre'),
    (5, 4, 'Livre'),
    (6, 6, 'Ocupada'),
    (7, 6, 'Reservada'),
    (8, 8, 'Livre'),
    (9, 2, 'Manutencao'),
    (10, 10, 'Livre');
```

Lista de Espera

A tabela Lista_espera foi povoada com registros de clientes aguardando disponibilidade em diferentes restaurantes. Foram incluídos horários de entrada variados e status como “Aguardando” ou “Desistiu”, simulando situações de pico de movimento e necessidade de fila. Esses dados permitem validar a lógica de atendimento e liberação de mesas.

```
INSERT INTO lista_espera (data_entrada, hora_entrada, status,
    qntd_pessoas) VALUES
    ('2023-12-01','19:00','Aguardando', 2),
    ('2023-12-01','19:10','Chamado', 4),
    ('2023-12-01','19:20','Desistiu', 2),
    ('2023-12-01','19:30','Aguardando', 6),
    ('2023-12-02','20:00','Chamado', 2),
    ('2023-12-02','20:15','Desistiu', 3),
    ('2023-12-02','20:30','Aguardando', 4),
    ('2023-12-03','21:00','Chamado', 2),
    ('2023-12-03','21:10','Aguardando', 5),
    ('2023-12-03','21:20','Desistiu', 2);
```

4.2.2 Alimentação das Tabelas Associativas

Para validar os relacionamentos Muitos para Muitos (N:M), foi realizada a alimentação das tabelas associativas (seção 4.1.3). A inserção de dados nessas tabelas é crítica, pois ela depende da existência prévia das chaves primárias nas tabelas principais. Cada registro inserido (por exemplo, em avaliacao_cliente) cria um vínculo lógico entre as chaves primárias das tabelas envolvidas, testando efetivamente a integridade das Chaves Estrangeiras (FOREIGN KEY).

AVALIACAO_CLIENTE (ASSOCIAÇÃO ENTRE AVALIAÇÃO E CLIENTE)

A inserção nesta tabela envolveu a associação dos IDs dos clientes com os IDs das avaliações, estabelecendo o vínculo entre cada cliente e as avaliações que ele fez.

```
INSERT INTO avaliacao_cliente VALUES
    (1,3),
    (2,1),
    (3,5),
    (4,2),
    (5,1),
    (6,4),
    (7,6),
    (8,3),
    (9,2),
    (10,1);
```

CLIENTE_RESERVA (ASSOCIAÇÃO ENTRE CLIENTE E RESERVA)

A inserção nesta tabela envolveu a associação dos IDs dos clientes com os IDs das reservas, estabelecendo o vínculo entre cada cliente e as reservas que ele fez.

```
INSERT INTO cliente_reserva VALUES
    (1,5),
    (1,6),
    (2,2),
    (3,1),
    (4,3),
    (5,4),
    (6,7),
    (7,8),
    (8,9),
    (9,10);
```

CLIENTE_LISTA_ESPERA (ASSOCIAÇÃO ENTRE CLIENTE E LISTA DE ESPERA)

Nesta tabela são adicionados registros que associam cada cliente à sua respectiva entrada na lista de espera. Cada linha conterá o ID do cliente e o ID da lista de espera, formando uma combinação única que indica que aquele cliente está aguardando naquela lista específica.

```
INSERT INTO cliente_lista_espera VALUE
    (1,1),
    (2,1),
    (3,2),
    (4,3),
    (5,4),
    (6,5),
    (7,6),
    (8,7),
    (9,8),
    (10,9);
```

RESERVA_RESTAURANTE (ASSOCIAÇÃO ENTRE RESERVA E RESTAURANTE)

Na tabela reserva_restaurante, foram adicionados registros que vinculam cada reserva ao restaurante correspondente. Cada linha contém o ID da reserva e o ID do restaurante, formando uma combinação única que indica que aquela reserva foi feita para aquele restaurante específico.

```
INSERT INTO reserva_restaurante VALUES
    (1,1),
    (2,1),
    (3,2),
    (4,3),
    (5,4),
    (6,5),
    (7,6),
    (8,7),
    (9,8),
    (10,9);
```

RESERVA_MESA (ASSOCIAÇÃO N:M ENTRE RESERVA E MESA)

Na tabela reserva_mesa, são adicionados registros que associam cada reserva à(s) mesa(s) utilizada(s). Cada linha contém o ID da reserva e o ID da mesa, formando uma combinação única que indica qual mesa foi destinada a qual reserva.

```
INSERT INTO reserva_mesa VALUES
    (1,3),
    (2,5),
    (3,1),
    (4,2),
    (5,4),
    (6,6),
    (7,8),
    (8,9),
    (9,7),
    (10,10);
```

4.3 Atualização do Banco de Dados

Esta etapa, nomeada Atualização do Banco de Dados, demonstra a Manipulação de Dados (DML - Data Manipulation Language) através das três operações: Inserção, Remoção e Atualização.

4.3.1 Inserção

Esta seção utiliza o comando INSERT INTO para adicionar um novo registro em uma tabela existente.

```
INSERT INTO cliente (nome, telefone_c, email) VALUES
    ('Beatriz Almeida','83988887777','beatriz@email.com');
```

4.3.2 Remoções

Esta seção utiliza o comando DELETE FROM para remover dados específicos do banco. É crucial notar a ordem das remoções para manter a integridade referencial (regras de FOREIGN KEY).

```
DELETE FROM avaliacao WHERE id_avaliacao = 7;
```

Uma avaliação específica (ID 7) foi removida.

```
DELETE FROM reserva_mesa WHERE id_reserva_fk = 5;
DELETE FROM reserva_restaurante WHERE id_reserva_fk = 5;
DELETE FROM reserva WHERE id_reserva = 5;
```

Remoção completa de uma reserva, exigindo a exclusão dos registros nas tabelas associativas primeiro, para não violar as chaves estrangeiras.

```
DELETE FROM avaliacao_cliente WHERE id_cliente_fk = 10;
DELETE FROM cliente_reserva WHERE id_cliente_fk = 10;
DELETE FROM cliente_lista_espera WHERE id_cliente_fk = 10;
DELETE FROM cliente WHERE id_cliente = 10;
```

Remoção completa de um cliente, exigindo a remoção dos seus dados em todas as tabelas associativas primeiro.

4.3.3 Atualizações

Esta seção utiliza o comando UPDATE para modificar valores em registros já existentes.

```
1 UPDATE cliente SET telefone_c = '83999999999' WHERE
    id_cliente = 1;
```

Atualizou-se o número de telefone do Cliente de ID 1 para um novo valor.

```
1 UPDATE mesa SET status = 'Ocupada' WHERE id_mesa = 1;
```

Alterou-se o status da Mesa de ID 1 para "Ocupada".

```
1 UPDATE cliente SET telefone_c = (SELECT telefone_r FROM
    restaurante WHERE id_restaurante = 1) WHERE id_cliente =
    2;
```

O telefone do Cliente de ID 2 foi atualizado para ser igual ao telefone do Restaurante de ID 1.

4.4 Consultas

Com o banco de dados devidamente criado e populado, realizaram-se consultas destinadas à extração e análise de informações essenciais ao suporte à tomada de decisões. A seguir, apresenta-se a relação dos comandos empregados no conjunto de dados.

Relatório Completo de Reservas Ordenados por Dia

Como etapa inicial, procedeu-se à análise das reservas com base na data em que foram registradas, destacando informações como o horário de chegada e o nome do cliente responsável pela reserva e o telefone de contato. Além disso, também foram obtidas informações relacionadas ao restaurante, como o nome e o número da mesa alocada. Para a realizaão dessa consulta, utilizou-se como ferramentas JOINs em cascada, bem como a cláusula ORDER BY que auxilia na ordenação dos dados selecionados.

```sql
SELECT r.data_reserva, r.horario_chegada, c.nome AS
   nome_cliente,
c.telefone_c, rest.nome AS nome_restaurante, m.num_mesa
FROM reserva r JOIN cliente_reserva cr ON r.id_reserva = cr.
   id_reserva_fk
JOIN cliente c ON cr.id_cliente_fk = c.id_cliente
JOIN reserva_restaurante rr ON r.id_reserva = rr.
   id_reserva_fk
JOIN restaurante rest ON rr.id_restaurante_fk = rest.
   id_restaurante
JOIN reserva_mesa rm ON r.id_reserva = rm.id_reserva_fk
JOIN mesa m ON rm.id_mesa_fk = m.id_mesa
ORDER BY r.data_reserva, r.horario_chegada;
```

Clientes na Lista de Espera

A seguir, ainda usando JOINs entre tabelas foi feita a consulta para identificar clientes que ainda estavam aguardando na lista de espera. Nesse caso, além do uso da cláusula ORDER BY para ordenação, também foi utilizada a cláusula WHERE que serviu para selecionar apenas clientes com o status de "Aguardando".

```sql
SELECT le.data_entrada, le.hora_entrada, c.nome AS cliente,
   le.status
FROM lista_espera le
JOIN cliente_lista_espera cle ON le.id_lista_espera = cle.
   id_lista_espera_fk
JOIN cliente c ON cle.id_cliente_fk = c.id_cliente
WHERE le.status = 'Aguardando'
ORDER BY le.hora_entrada;
```

Quantas Mesas Estão Livres e Ocupadas no Momento

A terceira consulta teve como objetivo identificar a disponibilidade das mesas. Para isso, realizou-se um agrupamento por status utilizando a cláusula GROUP BY, permitindo analisar a quantidade total de lugares em cada categoria. Essa informação foi obtida por meio da função de agregação SUM() aplicada à capacidade das mesas.

```sql
SELECT status as status_mesa, SUM(capacidade_m) AS
   Total_Lugares
FROM mesa GROUP BY status;
```

Horários de maior fluxo nos restaurantes

Nesta consulta, buscou-se identificar os horários com maior concentração de reservas. Para isso, os registros foram agrupados pelo atributo hora_reserva, que representa o horário em que a reserva foi realizada. Em seguida, utilizou-se a função de agregação COUNT() para contabilizar o número de reservas em cada horário.

```
SELECT hora_reserva, COUNT(*) AS Volume_Reservas
FROM reserva
GROUP BY hora_reserva
ORDER BY volume_reservas DESC;
```

Média das Notas

Com base nos registros da tabela avaliacao, foram calculadas a média geral das notas, a menor avaliação registrada e a maior avaliação atribuída. Para isso, utilizaram-se as funções de agregação AVG, MIN e MAX, que permitem resumir estatisticamente os valores presentes na coluna nota.

```
SELECT AVG(nota) AS Media_Geral, MIN(nota) as NotaMaisBaixa,
   MAX(nota) NotaMaisAlta
FROM avaliacao;
```

Notas e Comentários

Esta consulta teve como objetivo identificar todas as avaliações realizadas pelos clientes, exibindo a nota atribuída, o comentário associado, o nome do cliente responsável pela avaliação e a data em que ela foi registrada.

```
SELECT c.nome, a.nota, a.comentario, a.data_avaliacao
FROM avaliacao a
JOIN avaliacao_cliente ac ON a.id_avaliacao = ac.
   id_avaliacao_fk
JOIN cliente c ON ac.id_cliente_fk = c.id_cliente
ORDER BY a.nota;
```

Quantidade de Restaurantes por Cidade

Esta consulta teve como objetivo identificar a distribuição de restaurantes por cidade. Para isso, os registros da tabela restaurante foram agrupados pelo atributo cidade utilizando a cláusula GROUP BY, e em seguida foi realizada a contagem

de restaurantes em cada cidade com a função de agregação COUNT(). A cláusula HAVING foi aplicada para filtrar apenas as cidades com pelo menos um restaurante registrado, permitindo visualizar de forma clara quais cidades possuem mais ou menos estabelecimentos cadastrados.

```
SELECT r.cidade AS Cidade, COUNT(*) AS total_restaurantes
FROM restaurante r GROUP BY r.cidade HAVING COUNT(*) > 0;
```

Quantidade de Restaurantes por Bairro em João Pessoa

Esta consulta identifica a quantidade de restaurantes por bairro em João Pessoa, agrupando os registros pelo bairro e contando os estabelecimentos com a função COUNT(). A cláusula HAVING filtra bairros que tem mais de um restaurante.

```
SELECT bairro, COUNT(*) AS total_restaurantes
FROM restaurante WHERE cidade = 'João Pessoa'
GROUP BY bairro HAVING COUNT(*) > 2;
```

Dias com mais reservas (acima de 3)

Essa consulta identifica os dias em que houve mais de 3 reservas. Ela agrupa os registros pelo atributo data_reserva, conta o número de reservas em cada dia com COUNT(*) e exibe apenas os dias com mais de 3 reservas, ordenando o resultado do dia com maior número de reservas para o menor.

```
SELECT r.data_reserva, COUNT(*) AS total
FROM reserva r GROUP BY r.data_reserva
HAVING COUNT(*) > 3 ORDER BY total DESC;
```

Quantidade de entradas na lista de espera por dia

SELECT data_entrada, COUNT(*) AS total_entradas FROM lista_espera GROUP BY data_entrada;

```
SELECT data_entrada, COUNT(*) AS total_entradas FROM
   lista_espera GROUP BY data_entrada;
```

Media de notas por cliente

Ainda utilizando funções de agregação, foi possível calcular a média das avaliações de cada cliente, exibindo também informações complementares como o código e o nome do cliente.

```sql
SELECT c.id_cliente, c.nome, AVG(a.nota) AS media_nota
FROM cliente c
JOIN avaliacao_cliente ac ON c.id_cliente = ac.id_cliente_fk
JOIN avaliacao a ON ac.id_avaliacao_fk = a.id_avaliacao
GROUP BY c.id_cliente, c.nome
ORDER BY media_nota ASC;
```

Lista de Clientes em Ordem Alfabética

Esse SELECT retorna todos os registros da tabela cliente e ordena os resultados em ordem alfabética crescente com base no nome do cliente.

```sql
SELECT * FROM cliente ORDER BY nome ASC;
```

Restaurantes com mais de três horas aberto

Esta consulta teve como objetivo identificar os restaurantes que permanecem abertos por mais de três horas. Para isso, foi calculada a diferença entre o horário de fechamento e o horário de abertura usando a função TIMEDIFF(). Em seguida, os resultados foram filtrados com a cláusula HAVING e ordenados em ordem decrescente de tempo de funcionamento, permitindo visualizar quais estabelecimentos ficam mais tempo disponíveis para atendimento.

```sql
SELECT r.nome, TIMEDIFF(r.horario_fechamento, r.
   horario_abertura) AS horas_aberto
FROM restaurante r GROUP BY r.nome, r.horario_fechamento, r.
   horario_abertura HAVING TIMEDIFF(r.horario_fechamento, r.
   horario_abertura) > '03:00:00'
ORDER BY horas_aberto DESC;
```

Comentários e Notas de Clientes para o Restaurante de ID 6

Esta consulta teve como objetivo listar todas as avaliações relacionadas a um restaurante específico (no caso, com id_restaurante = 6). Para cada avaliação, são exibidos o nome do restaurante, o nome do cliente, a nota, o comentário e a data em que a avaliação foi realizada. O resultado é ordenado de forma decrescente pela data da avaliação, mostrando primeiro as mais recentes.

```sql
SELECT r.nome AS Restaurante, c.nome AS Cliente, a.nota, a.
   comentario, a.data_avaliacao
FROM restaurante r
JOIN reserva_restaurante rr ON r.id_restaurante = rr.
   id_restaurante_fk
JOIN reserva res ON rr.id_reserva_fk = res.id_reserva
JOIN cliente_reserva cr ON res.id_reserva = cr.id_reserva_fk
JOIN cliente c ON cr.id_cliente_fk = c.id_cliente
JOIN avaliacao_cliente ac ON c.id_cliente = ac.id_cliente_fk
JOIN avaliacao a ON ac.id_avaliacao_fk = a.id_avaliacao
WHERE r.id_restaurante = 6
ORDER BY a.data_avaliacao DESC;
```

Clientes que deram notas baixas

O comando lista todos os clientes que deram notas baixas (< 3), mostrando seus dados de contato, comentários e datas das avaliações, do mais recente para o mais antigo.

```sql
SELECT
    c.nome AS Cliente,
    c.telefone_c,
    a.nota,
    a.comentario,
    a.data_avaliacao
FROM cliente c
JOIN avaliacao_cliente ac ON c.id_cliente = ac.id_cliente_fk
JOIN avaliacao a ON ac.id_avaliacao_fk = a.id_avaliacao
WHERE a.nota < 3.0
ORDER BY a.data_avaliacao DESC;
```

REFERÊNCIAS

PASSOS, I. D. C. F. *Notas de aula da disciplina Banco de Dados*. 2025.

SIS4. *BRMODELO*. 2025. Disponível em: <https://www.sis4.com/>.