**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ**

PROJETO DE BANCO DE DADOS

2ª PARTE

Gerência de Hotéis da

Companhia Elegância Beira-Mar

Thais Oliveira Amaral

Curitiba

Novembro de 2023

Sumário

[1 Domínio de Aplicação para o Banco de Dados 3](#_Toc16349)

[1.1 Identificação do Projeto 3](#_Toc16350)

[1.2 Tema do Projeto 3](#_Toc16351)

[1.3 Usuários do sistema 3](#_Toc16352)

[1.4 Funcionalidade 1 do Projeto 3](#_Toc16353)

[1.5 Funcionalidade 2 do Projeto 3](#_Toc16354)

[2 Modelo Conceitual 4](#_Toc16355)

[3 Modelo Lógico (3FN) 5](#_Toc16356)

[4 Modelo Lógico (3FN) 6](#_Toc16357)

[5 Modelo Físico (3FN) 7](#_Toc16358)

[5.1 SQL para criação de tabelas e restrições 7](#_Toc16359)

[5.2 SQL para inserção de pelo menos 15 registros para cada tabela 10](#_Toc16360)

[6 Consultas e Programação 17](#_Toc16361)

[6.1 SQL para 3 consultas com AGREGAÇÃO de recuperação de dados 17](#_Toc16362)

[6.2 SQL para 3 consultas com IR (PK + FK) de recuperação de dados 20](#_Toc16363)

[6.3 1 Stored procedure 23](#_Toc16364)

[6.4 1 Trigger 24](#_Toc16365)

# Domínio de Aplicação para o Banco de Dados

## Identificação do Projeto

Gerência de Hotéis da Companhia Elegância Beira-Mar



## Tema do Projeto

Projeto de banco de dados projetado para gerenciar hotéis, permitindo o acompanhamento de informações sobre funcionários, hotéis, hospedes, quartos e o registro de reservas para o gerenciamento da ocupação do hotel.

## Usuários do sistema

Funcionários do hotel e clientes.

## Funcionalidade 1 do Projeto

Registro de Check-in e Check-out:

Permite que os funcionários do hotel registrem o check-in e check-out de hóspedes, atualizando o status do quarto e registrando informações sobre a estadia.

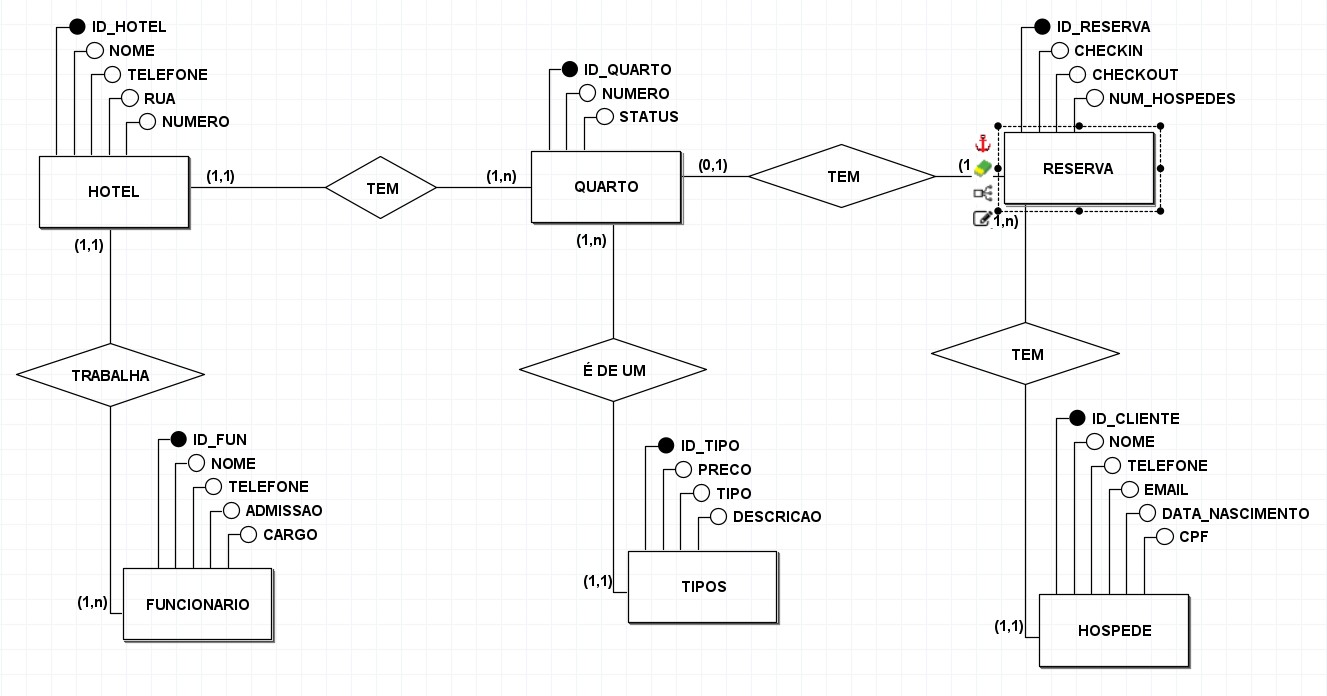
## Funcionalidade 2 do Projeto

Gestão de Funcionários:

Permite aos administradores do hotel gerenciar informações sobre os funcionários, como adição, remoção e atualização de detalhes de funcionários. Isso facilitaria a administração de recursos humanos e a manutenção de registros sobre a equipe do hotel.

# Modelo Conceitual

Imagem de boa resolução do Modelo Conceitual (**Modelo Entidade-Relacionamento - MER**) na **3FN**



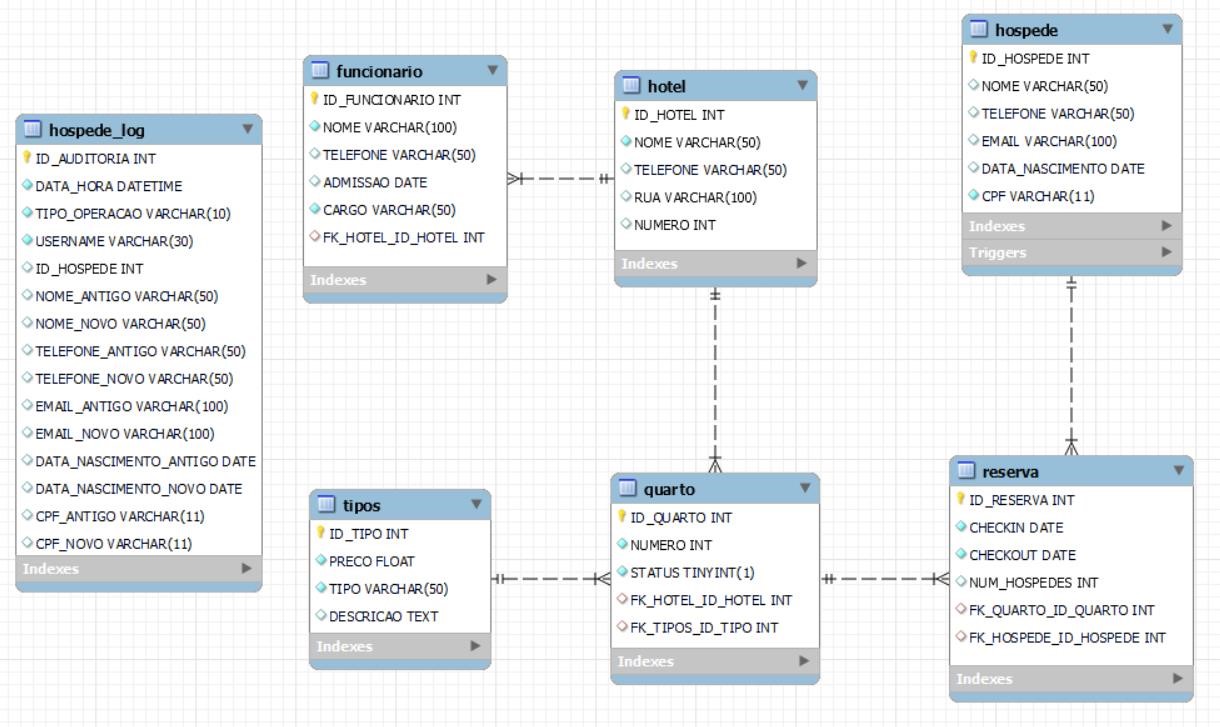
# Modelo Lógico (3FN)

Imagem de boa resolução do Modelo Lógico (**Modelo Relacional - MR**) na **3FN**, feito com o **brModelo**.



# Modelo Lógico (3FN)

Imagem de boa resolução do Modelo Lógico (**Modelo Relacional - MR**) na **3FN**, feito com o **ENGENHARIA REVERSA do MySQL Workbench**.



# Modelo Físico (3FN)

## SQL para criação de tabelas e restrições

CREATE TABLE HOTEL (

ID\_HOTEL INT PRIMARY KEY auto\_increment not null,

NOME VARCHAR(50) NOT NULL,

TELEFONE VARCHAR(50),

RUA VARCHAR(100),

NUMERO INT

);

CREATE TABLE QUARTO (

ID\_QUARTO INT PRIMARY KEY auto\_increment not null,

NUMERO INT NOT NULL,

STATUS BOOLEAN NOT NULL,

FK\_HOTEL\_ID\_HOTEL INT,

FK\_TIPOS\_ID\_TIPO INT

);

CREATE TABLE HOSPEDE (

ID\_HOSPEDE INT PRIMARY KEY auto\_increment not null,

NOME VARCHAR(50),

TELEFONE VARCHAR(50),

EMAIL VARCHAR(100),

DATA\_NASCIMENTO DATE,

CPF VARCHAR(11) NOT NULL UNIQUE

);

CREATE TABLE FUNCIONARIO (

ID\_FUNCIONARIO INT PRIMARY KEY auto\_increment not null,

NOME VARCHAR(100) NOT NULL,

TELEFONE VARCHAR(50),

ADMISSAO DATE,

CARGO VARCHAR(50) NOT NULL,

FK\_HOTEL\_ID\_HOTEL INT

);

CREATE TABLE RESERVA (

ID\_RESERVA INT PRIMARY KEY auto\_increment not null,

CHECKIN DATE NOT NULL,

CHECKOUT DATE NOT NULL,

NUM\_HOSPEDES INT,

FK\_QUARTO\_ID\_QUARTO INT,

FK\_HOSPEDE\_ID\_HOSPEDE INT

);

CREATE TABLE TIPOS (

ID\_TIPO INT PRIMARY KEY auto\_increment not null,

PRECO FLOAT NOT NULL,

TIPO VARCHAR(50) NOT NULL,

DESCRICAO TEXT

);

ALTER TABLE QUARTO ADD CONSTRAINT FK\_QUARTO\_2

FOREIGN KEY (FK\_HOTEL\_ID\_HOTEL)

REFERENCES HOTEL (ID\_HOTEL)

ON DELETE RESTRICT;

ALTER TABLE QUARTO ADD CONSTRAINT FK\_QUARTO\_3

FOREIGN KEY (FK\_TIPOS\_ID\_TIPO)

REFERENCES TIPOS (ID\_TIPO)

ON DELETE RESTRICT;

ALTER TABLE FUNCIONARIO ADD CONSTRAINT FK\_FUNCIONARIO\_2

FOREIGN KEY (FK\_HOTEL\_ID\_HOTEL)

REFERENCES HOTEL (ID\_HOTEL)

ON DELETE RESTRICT;

ALTER TABLE RESERVA ADD CONSTRAINT FK\_RESERVA\_2

FOREIGN KEY (FK\_QUARTO\_ID\_QUARTO)

REFERENCES QUARTO (ID\_QUARTO)

ON DELETE CASCADE;

ALTER TABLE RESERVA ADD CONSTRAINT FK\_RESERVA\_3

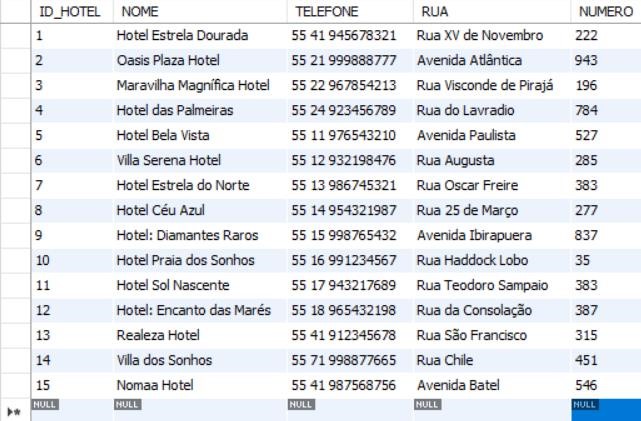
FOREIGN KEY (FK\_HOSPEDE\_ID\_HOSPEDE)

REFERENCES HOSPEDE (ID\_HOSPEDE)

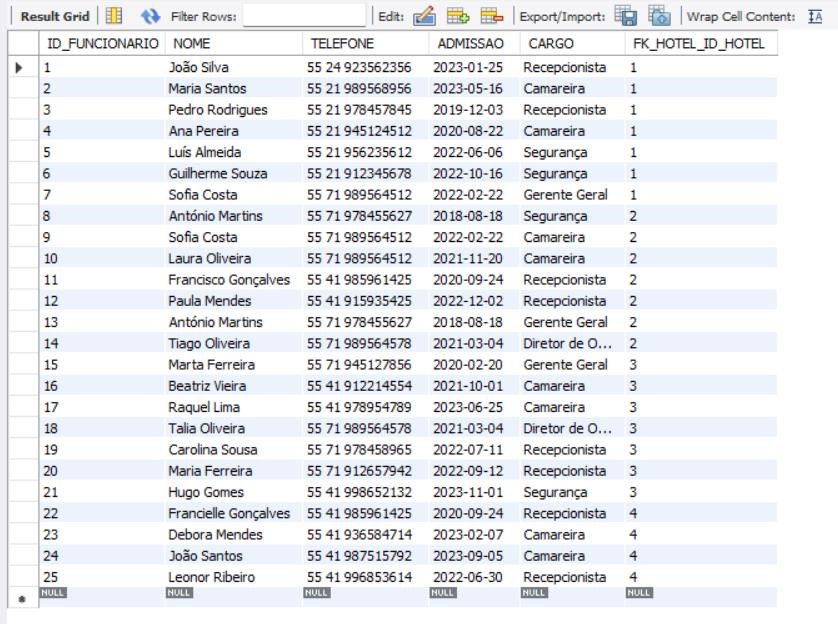
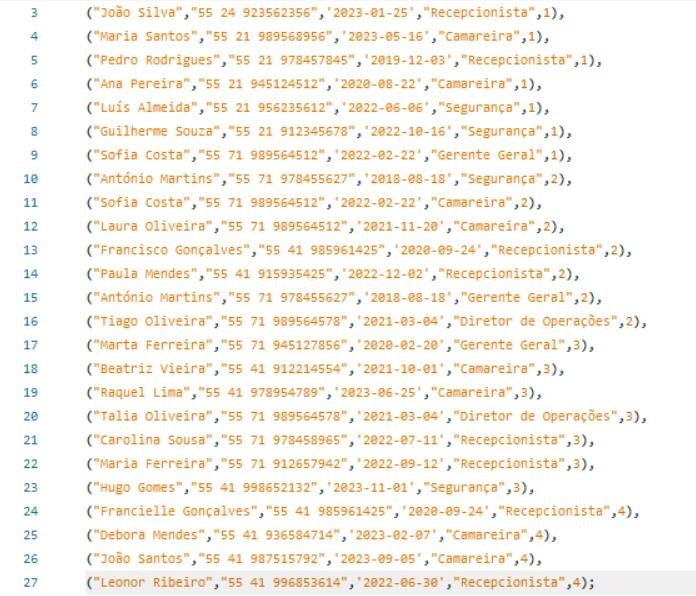
ON DELETE RESTRICT;

## SQL para inserção de pelo menos 15 registros para cada tabela

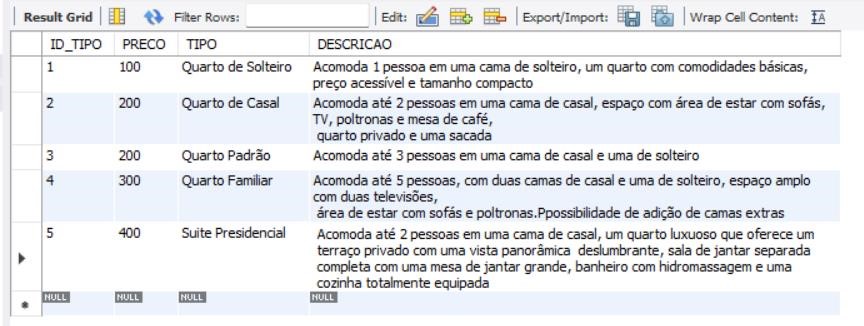
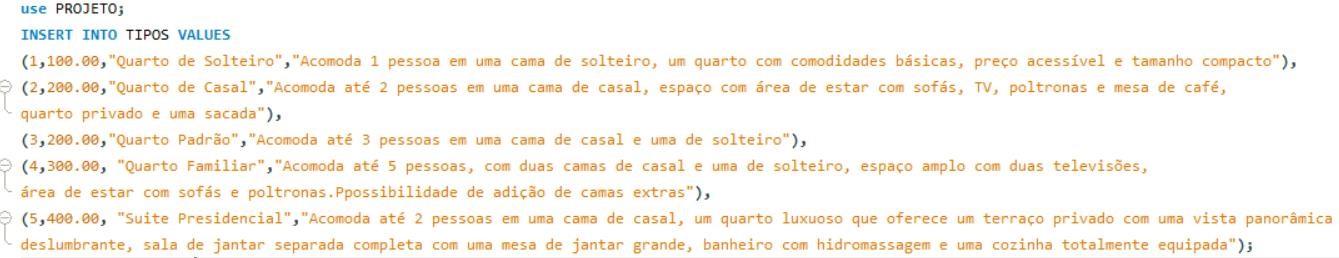
**Inserção de dados na tabela HOTEL:**



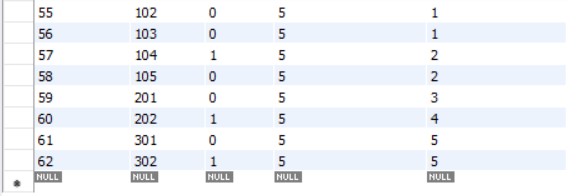
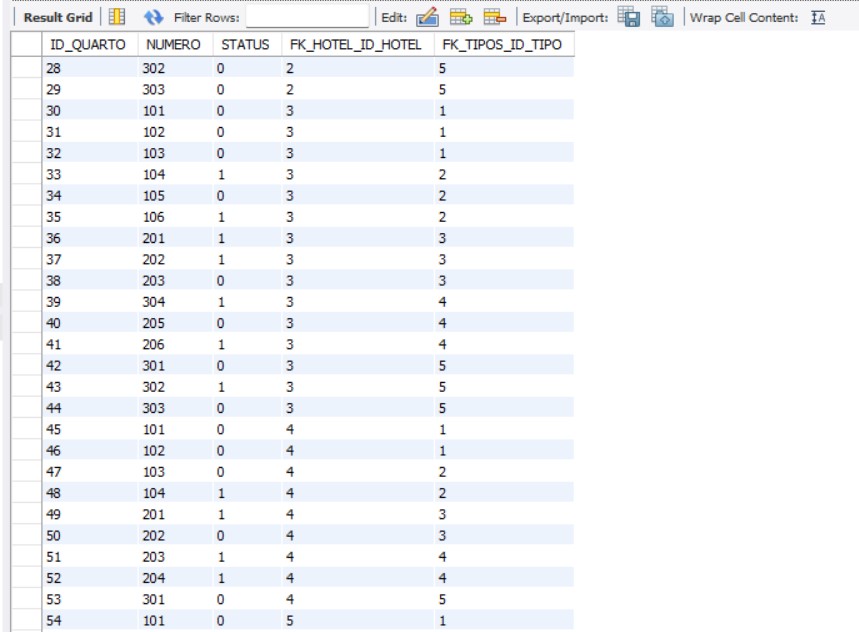
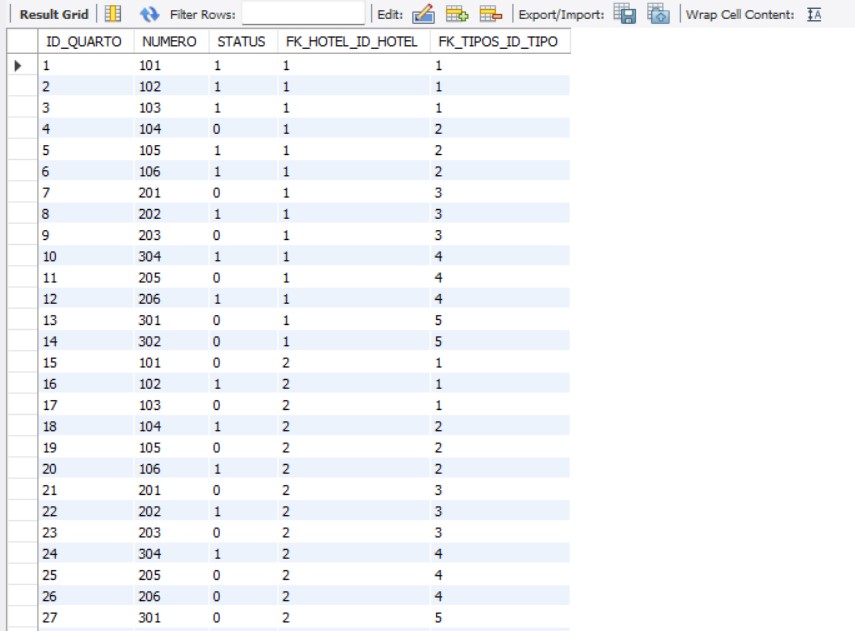
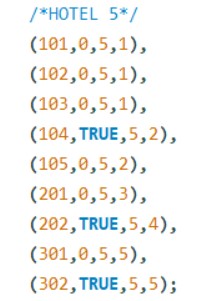
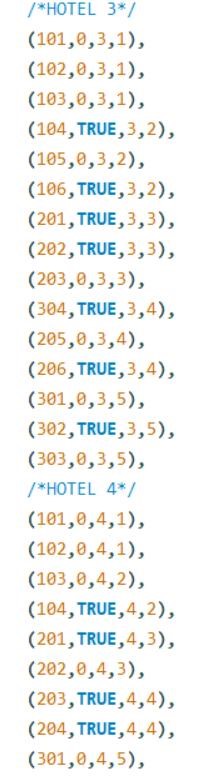
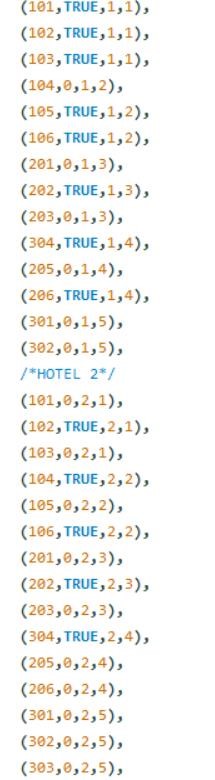
**Inserção de dados na tabela FUNCIONARIO:**



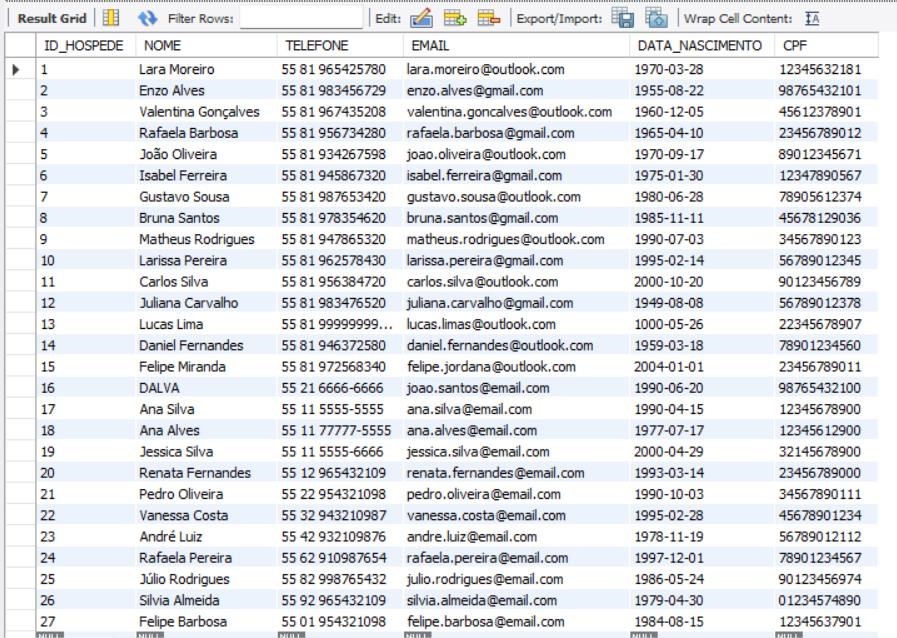
**Inserção de dados na tabela TIPOS:**



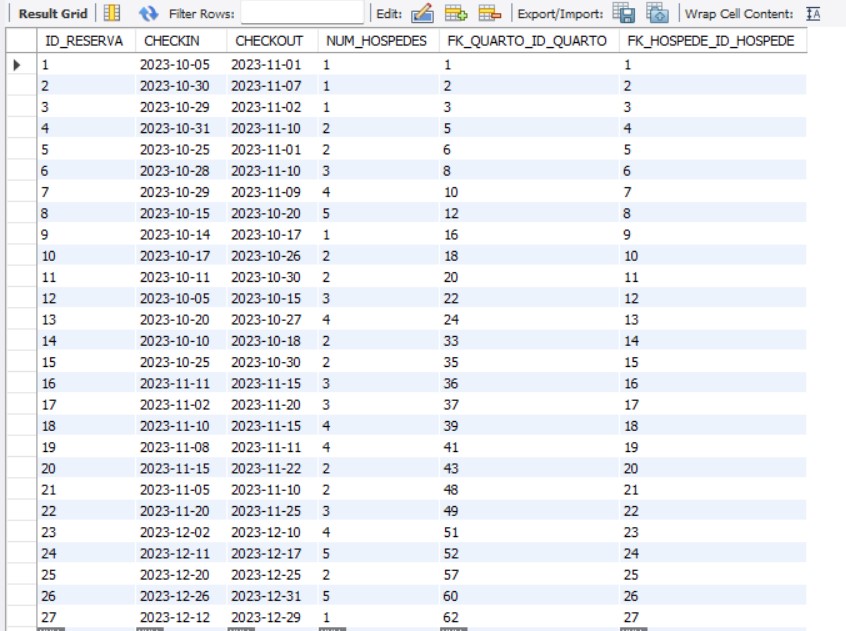
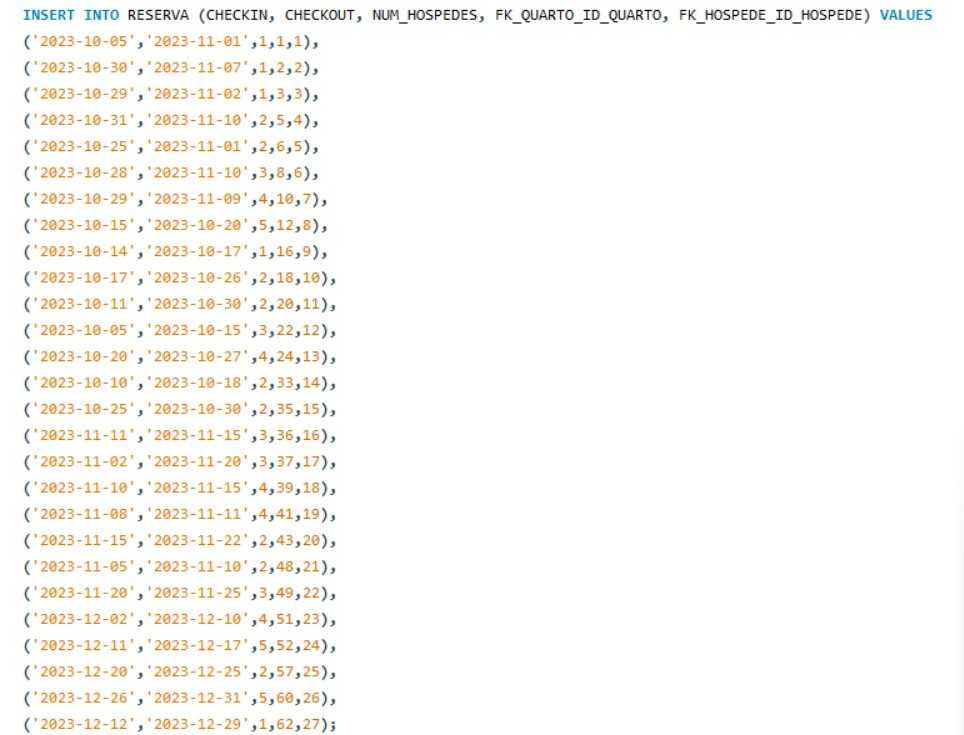
**Inserção de dados na tabela QUARTO:**



**Inserção de dados na tabela HOSPEDE:**



**Inserção de dados na tabela RESERVA:**



# Consultas e Programação

Para consultas com referências entre tabelas, usar apenas JOIN.

## SQL para 3 consultas com AGREGAÇÃO de recuperação de dados

**Consulta 1**:

1. Descreva por qual razão a consulta é importante para o usuário.

Esta consulta é importante pois oferece informações sobre os hotéis que possuem quartos disponíveis e a quantidade de quartos disponíveis.

1. Código SQL

SELECT COUNT(q.ID\_QUARTO) AS Quartos\_Disponiveis, t.TIPO AS Tipo\_Quarto, h.NOME AS Nome\_Hotel

FROM HOTEL h

INNER JOIN QUARTO q ON h.ID\_HOTEL = q.FK\_HOTEL\_ID\_HOTEL

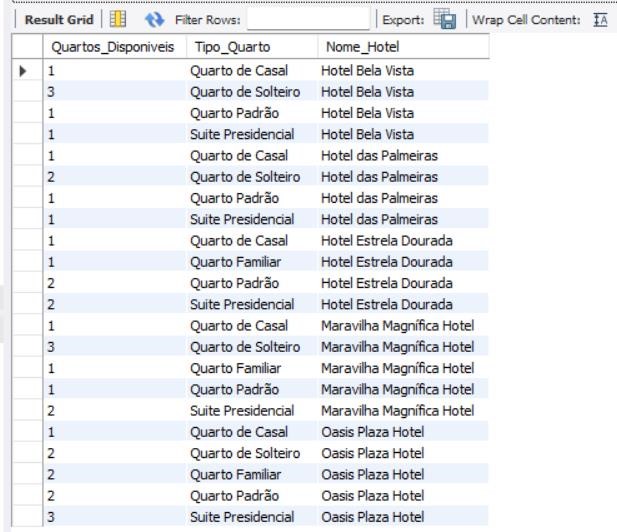
INNER JOIN TIPOS t ON q.FK\_TIPOS\_ID\_TIPO = t.ID\_TIPO

WHERE q.STATUS = FALSE

GROUP BY h.NOME, t.TIPO

ORDER BY h.NOME;

1. Resultado da consulta.



**Consulta 2**:

1. Descreva por qual razão a consulta é importante para o usuário.

Esta consulta é importante pois fornece informações sobre a distribuição dos funcionários em diferentes hotéis. Ela conta quantos funcionários de cada cargo trabalham em cada hotel.

1. Código SQL.

SELECT HOTEL.NOME, FUNCIONARIO.CARGO, COUNT(\*) AS NumeroDeFuncionarios

FROM FUNCIONARIO

JOIN HOTEL ON FUNCIONARIO.FK\_HOTEL\_ID\_HOTEL = HOTEL.ID\_HOTEL

GROUP BY HOTEL.NOME, FUNCIONARIO.CARGO;

1. Resultado da consulta.



**Consulta 3**:

1. Descreva por qual razão a consulta é importante para o usuário.

Esta consulta é importante pois executa o cálculo para análise do faturamento total de um hotel com base nas reservas de quartos. Isso pode ser útil para análises financeiras.

1. Código SQL.

SELECT h.NOME AS NOME\_HOTEL, r.ID\_RESERVA, DATE\_FORMAT(r.CHECKIN, '%d/%m/%Y') AS CHECKIN,DATE\_FORMAT(r.CHECKOUT, '%d/%m/%Y') AS CHECKOUT, DATEDIFF(r.CHECKOUT, r.CHECKIN) as DIAS\_RESERVADOS, CONCAT('R$', FORMAT((t.PRECO \* DATEDIFF(r.CHECKOUT, r.CHECKIN)), 2, 'de\_DE')) AS RECEITA\_TOTAL

FROM HOTEL h

JOIN QUARTO q ON h.ID\_HOTEL = q.FK\_HOTEL\_ID\_HOTEL

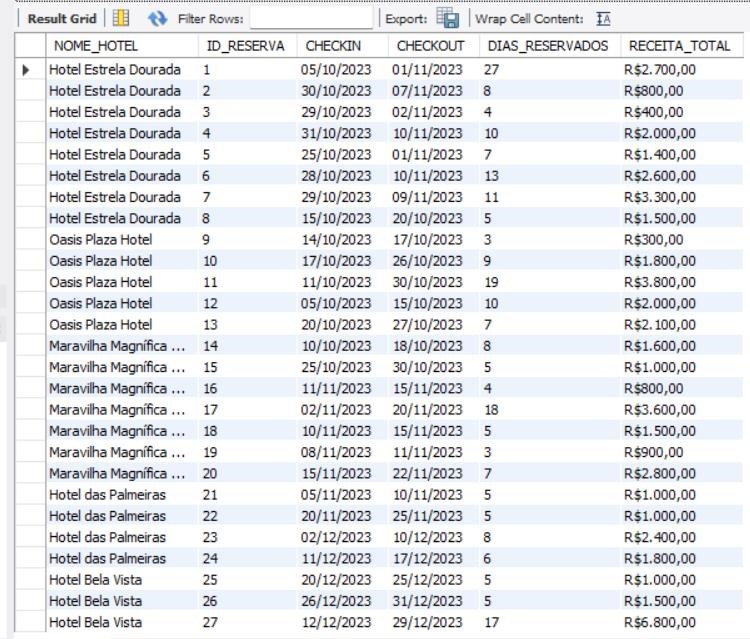
JOIN RESERVA r ON q.ID\_QUARTO = r.FK\_QUARTO\_ID\_QUARTO

JOIN TIPOS t ON q.FK\_TIPOS\_ID\_TIPO = t.ID\_TIPO

GROUP BY h.NOME, r.ID\_RESERVA

ORDER BY r.ID\_RESERVA, h.NOME;

1. Resultado da consulta.



## SQL para 3 consultas com IR (PK + FK) de recuperação de dados

**Consulta 1**:

1. Descreva por qual razão a consulta é importante para o usuário.

Esta consulta é importante para a recuperação de informações sobre as reservas feitas em um hotel, incluindo tanto detalhes sobre o quarto reservado quanto os dados do hóspede, facilitando o gerenciamento do hotel.

1. Código SQL.

SELECT RESERVA.ID\_RESERVA, HOSPEDE.NOME AS NomeHospede,

DATE\_FORMAT(RESERVA.CHECKIN, '%d/%m/%Y') AS CHECKIN,

DATE\_FORMAT(RESERVA.CHECKOUT, '%d/%m/%Y') AS CHECKOUT,

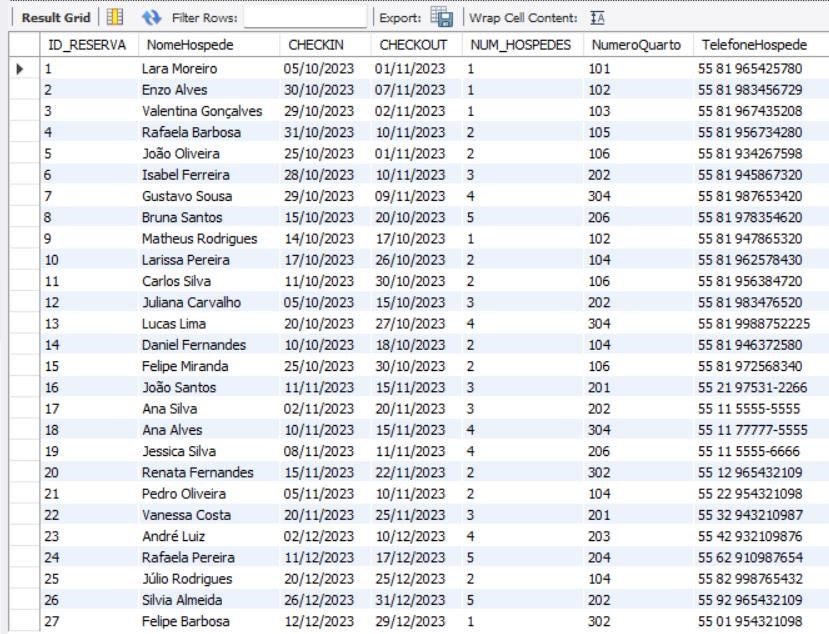
RESERVA.NUM\_HOSPEDES, QUARTO.NUMERO AS NumeroQuarto, HOSPEDE.TELEFONE AS TelefoneHospede

FROM RESERVA

JOIN QUARTO ON RESERVA.FK\_QUARTO\_ID\_QUARTO = QUARTO.ID\_QUARTO

JOIN HOSPEDE ON RESERVA.FK\_HOSPEDE\_ID\_HOSPEDE = HOSPEDE.ID\_HOSPEDE;

1. Resultado da consulta.



**Consulta 2**:

1. Descreva por qual razão a consulta é importante para o usuário.

Esta consulta é importante para recuperar informações sobre funcionários de um hotel, juntamente com seus cargos.

1. Código SQL.

SELECT F.NOME AS FUNCIONARIO, F.TELEFONE, DATE\_FORMAT(F.ADMISSAO, '%d/%m/%Y') AS

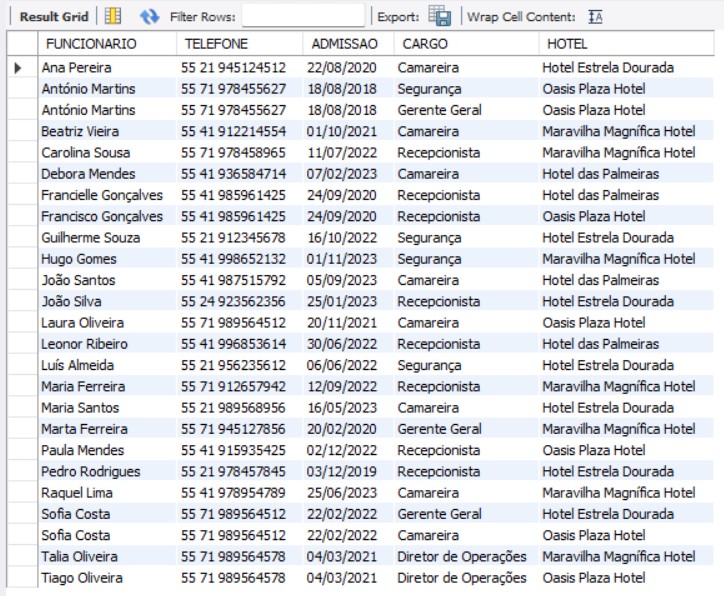
ADMISSAO, F.CARGO, H.NOME AS HOTEL

FROM FUNCIONARIO F

JOIN HOTEL H ON F.FK\_HOTEL\_ID\_HOTEL = H.ID\_HOTEL

ORDER BY F.NOME;

1. Resultado da consulta.



**Consulta 3**:

1. Descreva por qual razão a consulta é importante para o usuário.

Esta consulta é importante para recuperar informações sobre os quartos de um hotel específico, incluindo o número do quarto, o status e o tipo do quarto.

1. Código SQL.

SELECT HOTEL.NOME, QUARTO.NUMERO, TIPOS.TIPO,

CASE

WHEN QUARTO.STATUS = TRUE THEN "RESERVADO"

ELSE "DISPONÍVEL"

END AS DISPONIBILIDADE

FROM QUARTO

JOIN HOTEL ON QUARTO.FK\_HOTEL\_ID\_HOTEL = HOTEL.ID\_HOTEL

JOIN TIPOS ON QUARTO.FK\_TIPOS\_ID\_TIPO = TIPOS.ID\_TIPO

WHERE QUARTO.FK\_HOTEL\_ID\_HOTEL = 1;

1. Resultado da consulta.



## 1 Stored procedure

**SP 1**:

1. Descreva o que a *Stored Produceure* (SP) faz (valores de INPUT e de OUTPUT).
   1. Stored Procedure recebe um valor de entrada, que é o ID do cliente (ID\_CLIENTE), e retorna os detalhes da reserva associada a esse cliente. Os valores de saída serão os detalhes das reservas, como ID\_RESERVA, CHECKIN, CHECKOUT, NUM\_HOSPEDES, etc.

1. Descreva por qual razão a SP é importante para o usuário.
   1. SP é importante para o usuário pois permite que o hotel obtenha de forma mais rápida informações sobre as reservas feitas por um cliente específico, simplificando a consulta e evitando a necessidade de escrever consultas SQL toda vez que desejam obter essas informações. Nesse caso em específico é interessante a utilização da SP para a realização do checkout do hospede no hotel, podendo acessar todas as informações dele, facilitando futuramente no pagamento por exemplo. Nesse projeto não há todos os serviços que um hotel geralmente oferece, mas essa mesma ideia poderia ser utilizada para averiguar todos os serviços utilizados pelo cliente e assim poder calcular o valor final de sua estadia.

1. Código SQL

DELIMITER //

CREATE PROCEDURE ObterReservaPorID(IN client\_id INT)

BEGIN

SELECT R.ID\_RESERVA,

DATE\_FORMAT(R.CHECKIN, '%d/%m/%Y') AS CHECKIN,

DATE\_FORMAT(R.CHECKOUT, '%d/%m/%Y') AS CHECKOUT,

R.NUM\_HOSPEDES,

Q.NUMERO AS Quarto\_Numero,

CONCAT('R$', FORMAT(T.preco, 2, 'de\_DE')) AS Preco,

T.TIPO AS Tipo\_Quarto,

H.NOME AS Hotel\_Nome

FROM RESERVA R

JOIN QUARTO Q ON R.FK\_QUARTO\_ID\_QUARTO = Q.ID\_QUARTO

JOIN HOTEL H ON Q.FK\_HOTEL\_ID\_HOTEL = H.ID\_HOTEL

JOIN TIPOS T ON Q.FK\_TIPOS\_ID\_TIPO = T.ID\_TIPO

WHERE R.FK\_HOSPEDE\_ID\_HOSPEDE = client\_id;

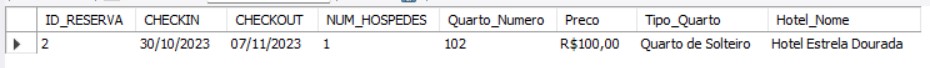
END;

//

DELIMITER ;

CALL ObterReservaPorID(2);

d) Resultado da execução da SP



## 1 Trigger

**Trigger 1**:

1. Descreva o que o *Trigger* faz para as operações de INSERT / UPDATE / DELETE.

Este trigger será acionado após qualquer operação de INSERT, UPDATE ou DELETE na tabela HOSPEDE, registrando as informações sobre a operação em uma tabela de histórico chamada HOSPEDE\_LOG e capturando os detalhes da operação, incluindo o tipo de operação, data e hora da operação e os dados relevantes da linha afetada na tabela HOSPEDE.

1. Descreva por qual razão o *Trigger* é importante para o usuário.

O trigger é importante para os usuários por várias razões:

* + Rastreamento de mudanças: Permite o rastreamento de todas as alterações feitas na tabela HOSPEDE, o que pode ser útil para fins de auditoria e conformidade.
  + Identificação de responsabilidade: Ajuda a identificar quem fez as alterações e quando as alterações foram feitas.
  + Recuperação de dados: Permite recuperar informações sobre as versões anteriores dos dados, no caso de atualizações incorretas ou exclusões por acidente.

1. Código SQL

CREATE TABLE HOSPEDE\_LOG (

ID\_AUDITORIA INT PRIMARY KEY auto\_increment not null,

DATA\_HORA DATETIME NOT NULL,

TIPO\_OPERACAO VARCHAR(10) NOT NULL,

USERNAME VARCHAR(30) NOT NULL,

ID\_HOSPEDE INT,

NOME\_ANTIGO VARCHAR(50),

NOME\_NOVO VARCHAR(50),

TELEFONE\_ANTIGO VARCHAR(50),

TELEFONE\_NOVO VARCHAR(50),

EMAIL\_ANTIGO VARCHAR(100),

EMAIL\_NOVO VARCHAR(100),

DATA\_NASCIMENTO\_ANTIGO DATE,

DATA\_NASCIMENTO\_NOVO DATE,

CPF\_ANTIGO VARCHAR(11),

CPF\_NOVO VARCHAR(11)

);

DELIMITER $

CREATE TRIGGER HOSPEDE\_LOG\_Insert

AFTER INSERT ON HOSPEDE

FOR EACH ROW

BEGIN

INSERT INTO HOSPEDE\_LOG (DATA\_HORA, TIPO\_OPERACAO, USERNAME, NOME\_NOVO, TELEFONE\_NOVO, EMAIL\_NOVO, DATA\_NASCIMENTO\_NOVO, CPF\_NOVO, ID\_HOSPEDE)

SELECT NOW(), 'INSERT', CURRENT\_USER(), NEW.NOME, NEW.TELEFONE, NEW.EMAIL, NEW.DATA\_NASCIMENTO, NEW.CPF, NEW.ID\_HOSPEDE;

END$

CREATE TRIGGER HOSPEDE\_LOG\_Update

AFTER UPDATE ON HOSPEDE

FOR EACH ROW

BEGIN

INSERT INTO HOSPEDE\_LOG (DATA\_HORA, TIPO\_OPERACAO, USERNAME, ID\_HOSPEDE, NOME\_ANTIGO, NOME\_NOVO, TELEFONE\_ANTIGO, TELEFONE\_NOVO, EMAIL\_ANTIGO, EMAIL\_NOVO, DATA\_NASCIMENTO\_ANTIGO, DATA\_NASCIMENTO\_NOVO, CPF\_ANTIGO, CPF\_NOVO)

SELECT NOW(), 'UPDATE', CURRENT\_USER(), OLD.ID\_HOSPEDE, OLD.NOME, NEW.NOME, OLD.TELEFONE, NEW.TELEFONE, OLD.EMAIL, NEW.EMAIL, OLD.DATA\_NASCIMENTO, NEW.DATA\_NASCIMENTO, OLD.CPF, NEW.CPF;

END$

CREATE TRIGGER HOSPEDE\_LOG\_Delete

AFTER DELETE ON HOSPEDE

FOR EACH ROW

BEGIN

INSERT INTO HOSPEDE\_LOG (DATA\_HORA, TIPO\_OPERACAO, USERNAME, NOME\_ANTIGO, TELEFONE\_ANTIGO, EMAIL\_ANTIGO, DATA\_NASCIMENTO\_ANTIGO, CPF\_ANTIGO, ID\_HOSPEDE)

SELECT NOW(), 'DELETE', CURRENT\_USER(), OLD.NOME, OLD.TELEFONE, OLD.EMAIL,

OLD.DATA\_NASCIMENTO, OLD.CPF, OLD.ID\_HOSPEDE;

END$

DELIMITER ;

h) Resultado da execução do *Trigger’*

