

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**

**CAIO MACEDO  
FERNANDO BULIGON ANTUNES  
JOSÉ SEBEN  
KAÍQUE MEDEIROS LIMA**

**MEMORIAL DESCRITIVO  
Laboratório de Banco de Dados**

**SANTA HELENA  
2025/2**

**CAIO MACEDO  
FERNANDO BULIGON ANTUNES  
JOSÉ SEBEN  
KAÍQUE MEDEIROS LIMA**

**MEMORIAL DESCRITIVO  
Laboratório de Banco de Dados**

**Descriptive Memorial - Database Laboratory**

Trabalho de Conclusão de Disciplina de Graduação apresentado como requisito para conclusão da disciplina de Laboratório de Banco de Dados do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Docente: Dra. Leiliane Pereira de Rezende

**SANTA HELENA  
2025/2**

## LISTA DE ALGORITMOS

## LISTA DE FIGURAS

## LISTAGEM DE CÓDIGOS FONTE

## **LISTA DE ABREVIATURAS E SIGLAS**

### **Siglas**

ACID      Atomicidade, Consistência, Isolamento e Durabilidade

## SUMÁRIO

<b>1</b>	<b>BANCO DE DADOS CONCEITUAL/LÓGICO</b>	<b>6</b>
1.1	Descrição do Banco de Dados .....	6
1.2	Modelo Entidade-Relacionamento .....	8
1.3	Estrutura do Trabalho .....	8
<b>2</b>	<b>BANCO DE DADOS FÍSICO</b>	<b>9</b>
2.1	Definição Física .....	9
2.2	Carga de Dados Iniciais .....	15
<b>3</b>	<b>CONSULTAS DE DADOS NO BD</b>	<b>20</b>
3.1	Restrições de Linha .....	20
3.2	Funções de Linha .....	22
3.3	Junções de Tabelas .....	22
3.4	Operadores de Conjunto .....	22
3.5	Sub-Consultas .....	23
3.6	Agregação e Funções de Agrupamento .....	25
3.7	SELECT nos Comandos .....	25
<b>4</b>	<b>ROTINAS DE OTIMIZAÇÃO NO BD</b>	<b>26</b>
4.1	Visões .....	26
4.2	Funções .....	26
4.3	Procedimentos .....	26
4.4	Gatilhos .....	26
4.5	Eventos .....	26
<b>5</b>	<b>CONTROLE DE ACESSO</b>	<b>27</b>
5.1	Criar Usuários .....	27
5.2	Atribuir Permissões .....	27
5.3	Revogar Permissões .....	27
<b>6</b>	<b>TRANSAÇÕES NO BD</b>	<b>28</b>
<b>7</b>	<b>FALHAS NO BD</b>	<b>29</b>
7.1	Tolerâncias à Falhas .....	29
7.2	Tratamento à Falhas .....	29
	<b>REFERÊNCIAS</b>	<b>30</b>

## **1 BANCO DE DADOS CONCEITUAL/LÓGICO**

A descrição do BD KJCF&Cia, trabalhado durante todo o documento, é apresentado na Seção 1.1. A modelagem por meio do MER é dada na Seção 1.2. A composição do restante do documento é descrita na Seção 1.3.

### **1.1 Descrição do Banco de Dados**

O BD “KJCF&Cia” corresponde à modelagem de um processo de banco de dados para uma empresa de materiais de construção, que trabalha desde produtos básicos para obras, ferramentas e utilidades em geral até móveis. Além disso, a empresa oferece serviços adicionais como entrega e montagem para seus clientes. Por se tratar de uma loja situada em uma cidade pequena, as entregas são realizadas apenas dentro da cidade e nas cidades vizinhas.

As principais informações a serem armazenadas são descritas a seguir:

- Clientes: Cadastro dos clientes que fazem compras na loja.
- Cargos: Responsável por armazenar os cargos dos funcionários.
- Funcionários: Cadastro dos funcionários da empresa com seus dados.
- Categorias: Organização utilizada para separar os produtos por categorias.
- Fornecedores: Cadastro dos fornecedores, armazenando suas informações.
- Produtos: Cadastro dos produtos da empresa, com suas informações detalhadas.
- Estoque: Controle do estoque, registrando os produtos disponíveis em loja.
- Compras: Controle das compras realizadas junto aos fornecedores.
- Vendas: Registro das vendas realizadas, vinculando clientes e funcionários.
- Cidade: Cadastro das cidades relacionadas aos endereços dos clientes.
- Bairro: Cadastro dos bairros, vinculados a uma cidade.
- Endereço: Armazena o endereço completo dos clientes (rua, número, complemento, bairro e cidade).
- Agendamento de Montagem: Controle dos agendamentos de montagens a serem realizadas em produtos vendidos.



- Entrega: Controle das entregas vinculadas às vendas, com data e status.
- Pagamento Cliente: Registro dos pagamentos efetuados pelos clientes, incluindo forma e status.
- Produto Venda: Registro dos itens vendidos em cada venda (produto, quantidade e valor).
- Receita Extra: Registro de receitas adicionais recebidas fora do processo de vendas.

Algumas regras são necessárias para que as restrições de integridade do BD sejam mantidas. As principais regras são descritas a abaixo, destacando-se que, ao modelar, algumas são inseridas devido à normalização do BD.

- Regras Sobre os Produtos:
  - Deve possuir uma categoria associada e um nome;
  - Um produto representa um conjunto de produtos com aquele nome e categoria.
- Regras sobre os Agendamentos de montagem:
  - O Status de montagem deve ser atualizado com frequência
  - Montagens em atraso devem ser priorizadas
  - Apenas funcionários de certos cargos podem efetuar a montagem.
- Regras sobre a venda de produtos:
  - Sempre que um produto é vendido, sua quantidade deve ser alterada no estoque.
- Regras sobre as Entregas:
  - Entregas devem ser efetuadas na ordem de data, datas mais próximas primeiro.
  - Entrega só pode ser realizada se a venda for concluída.
- Regras sobre os Categorias:
  - As categorias devem ser generalizadas. Ex: Produtos de Jardinagem
  - Deve-se evitar categorias repetidas ou muito semelhantes.

## 1.2 Modelo Entidade-Relacionamento

O MER composto por 17 tabelas é apresentado na Figura 1.1.

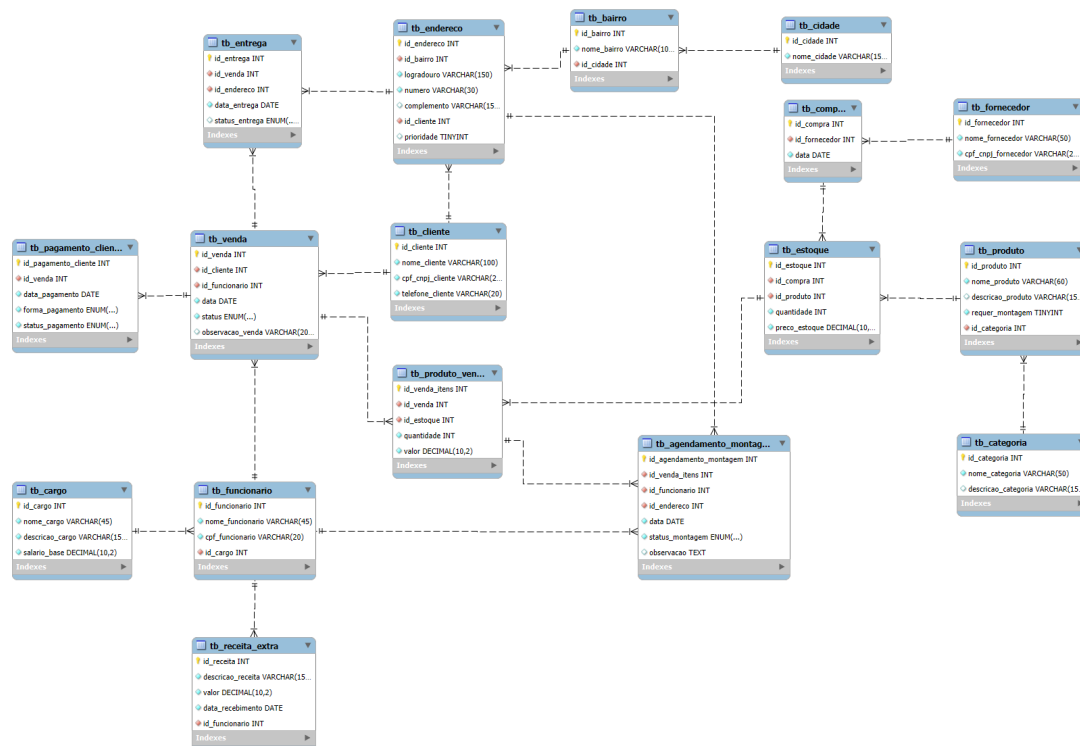


Figura 1.1: Diagrama Modelo-Entidade

### 1.3 Estrutura do Trabalho

Após a definição conceitual/lógica, o banco deve ser implementado fisicamente em MySQL. Essa implementação, juntamente com a primeira carga de dados, é descrita no Capítulo 2.

Com o banco implementado fisicamente, consultas nos dados podem ser realizadas por meio de SQL. O Capítulo apresenta inúmeras consultas, cada uma com uma característica diferente.

Algumas rotinas podem ser definidas para automatizar tarefas e, em alguns casos, melhorar a performance: visões, funções, procedimentos, gatilhos e eventos. Esses recursos são tratados no Capítulo 4.

O acesso aos dados deve ser controlado por segurança. Usuários distintos têm permissões distintas; isso é tratado no Capítulo 5.

Para garantir propriedades ACID, operações de alteração devem ser realizadas em bloco de transação (Capítulo 6).

Por fim, falhas podem ocorrer; tolerâncias e tratamentos são descritos no Capítulo 7.

## 2 BANCO DE DADOS FÍSICO

A criação do “KJCF&Cia” considerando MySQL é descrita na Seção 2.1 por meio de DDL. A carga inicial é dada na Seção por meio de DML.

### 2.1 Definição Física

Os scripts responsáveis pela criação física do schema são descritos abaixo. Todas as restrições de integridade (chaves candidatas, primárias e estrangeiras) são consideradas no momento da criação.

Listing 2.1: DDL – Tabela tb\_cliente

```
1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_cliente' (  
2   'id_cliente' INT NOT NULL AUTO_INCREMENT,  
3   'nome_cliente' VARCHAR(100) NOT NULL,  
4   'cpf_cnpj_cliente' VARCHAR(20) NOT NULL,  
5   'telefone_cliente' VARCHAR(20) NOT NULL,  
6   PRIMARY KEY ('id_cliente'),  
7   UNIQUE INDEX 'cpf_cnpj_cliente_UNIQUE' (('cpf_cnpj_cliente' ASC)  
8     VISIBLE)  
9   ENGINE = InnoDB  
   DEFAULT CHARACTER SET = utf8mb4;
```

Listing 2.2: DDL – Tabela tb\_cargo

```
1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_cargo' (  
2   'id_cargo' INT NOT NULL AUTO_INCREMENT,  
3   'nome_cargo' VARCHAR(45) NOT NULL,  
4   'descricao_cargo' VARCHAR(150) NOT NULL,  
5   'salario_base' DECIMAL(10,2) NOT NULL,  
6   PRIMARY KEY ('id_cargo'),  
7   UNIQUE INDEX 'descricao_cargo_UNIQUE' (('descricao_cargo' ASC)  
8     VISIBLE)  
9   ENGINE = InnoDB  
   DEFAULT CHARACTER SET = utf8mb4;
```

Listing 2.3: DDL – Tabela tb\_funcionario

```
1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_funcionario' (  
2   'id_funcionario' INT NOT NULL AUTO_INCREMENT,  
3   'nome_funcionario' VARCHAR(45) NOT NULL,  
4   'cpf_funcionario' VARCHAR(20) NOT NULL,
```

```

5      'id_cargo' INT NOT NULL,
6      PRIMARY KEY ('id_funcionario'),
7      UNIQUE INDEX 'cpf_funcionario_UNIQUE' ('cpf_funcionario' ASC)
      VISIBLE,
8      INDEX 'id_cargo_idx' ('id_cargo' ASC) VISIBLE,
9      CONSTRAINT 'fk_funcionario_cargo'
10     FOREIGN KEY ('id_cargo')
11     REFERENCES 'mydb`.`tb_cargo' ('id_cargo'))
12     ENGINE = InnoDB
13     DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.4: DDL – Tabela tb\_venda

```

1      CREATE TABLE IF NOT EXISTS 'mydb`.`tb_venda' (
2      'id_venda' INT NOT NULL AUTO_INCREMENT,
3      'id_cliente' INT NOT NULL,
4      'id_funcionario' INT NOT NULL,
5      'data' DATE NOT NULL,
6      'status' ENUM('pendente', 'concluida', 'cancelada') NOT NULL,
7      'observacao_venda' VARCHAR(200) NULL DEFAULT NULL,
8      PRIMARY KEY ('id_venda'),
9      INDEX 'id_cliente_idx' ('id_cliente' ASC) VISIBLE,
10     INDEX 'id_funcionario_idx' ('id_funcionario' ASC) VISIBLE,
11     CONSTRAINT 'fk_venda_cliente'
12     FOREIGN KEY ('id_cliente')
13     REFERENCES 'mydb`.`tb_cliente' ('id_cliente'),
14     CONSTRAINT 'fk_venda_funcionario'
15     FOREIGN KEY ('id_funcionario')
16     REFERENCES 'mydb`.`tb_funcionario' ('id_funcionario'))
17     ENGINE = InnoDB
18     DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.5: DDL – Tabela tb\_cidade

```

1      CREATE TABLE IF NOT EXISTS 'mydb`.`tb_cidade' (
2      'id_cidade' INT NOT NULL AUTO_INCREMENT,
3      'nome_cidade' VARCHAR(150) NOT NULL,
4      PRIMARY KEY ('id_cidade'),
5      UNIQUE INDEX 'nome_cidade_UNIQUE' ('nome_cidade' ASC) VISIBLE)
6      ENGINE = InnoDB
7      DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.6: DDL – Tabela tb\_bairro

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_bairro' (
2 'id_bairro' INT NOT NULL AUTO_INCREMENT,
3 'nome_bairro' VARCHAR(100) NOT NULL,
4 'id_cidade' INT NOT NULL,
5 PRIMARY KEY ('id_bairro'),
6 INDEX 'id_cidade_idx' ('id_cidade' ASC) VISIBLE,
7 CONSTRAINT 'fk_bairro_cidade'
8 FOREIGN KEY ('id_cidade')
9 REFERENCES 'mydb'.'tb_cidade' ('id_cidade'))
10 ENGINE = InnoDB
11 DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.7: DDL – Tabela tb\_endereco

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_endereco' (
2 'id_endereco' INT NOT NULL AUTO_INCREMENT,
3 'id_bairro' INT NOT NULL,
4 'logradouro' VARCHAR(150) NOT NULL,
5 'numero' VARCHAR(30) NOT NULL,
6 'complemento' VARCHAR(150) NULL DEFAULT NULL,
7 'id_cliente' INT NOT NULL,
8 'prioridade' TINYINT NULL DEFAULT NULL,
9 PRIMARY KEY ('id_endereco'),
10 INDEX 'id_cliente_idx' ('id_cliente' ASC) VISIBLE,
11 INDEX 'id_bairro_idx' ('id_bairro' ASC) VISIBLE,
12 CONSTRAINT 'fk_endereco_cliente'
13 FOREIGN KEY ('id_cliente')
14 REFERENCES 'mydb'.'tb_cliente' ('id_cliente'),
15 CONSTRAINT 'fk_endereco_bairro'
16 FOREIGN KEY ('id_bairro')
17 REFERENCES 'mydb'.'tb_bairro' ('id_bairro'))
18 ENGINE = InnoDB
19 DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.8: DDL – Tabela tb\_agendamento\_montagem

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_agendamento_montagem' (
2 'id_agendamento_montagem' INT NOT NULL AUTO_INCREMENT,
3 'id_venda_itens' INT NOT NULL,
4 'id_funcionario' INT NOT NULL,
5 'id_endereco' INT NOT NULL,
6 'data' DATE NOT NULL,
7 'status_montagem' ENUM('realizado', 'pendente', 'cancelado')

```

```

      NOT NULL ,
8    'observacao' TEXT NULL DEFAULT NULL ,
9    PRIMARY KEY ('id_agendamento_montagem'),
10   INDEX 'id_venda_itens_idx' ('id_venda_itens' ASC) VISIBLE ,
11   INDEX 'id_funcionario_idx' ('id_funcionario' ASC) VISIBLE ,
12   INDEX 'id_endereco_idx' ('id_endereco' ASC) VISIBLE ,
13   CONSTRAINT 'fk_agendamento_produto_venda'
14   FOREIGN KEY ('id_venda_itens')
15   REFERENCES 'mydb'.'tb_produto_venda' ('id_venda_itens'),
16   CONSTRAINT 'fk_agendamento_funcionario'
17   FOREIGN KEY ('id_funcionario')
18   REFERENCES 'mydb'.'tb_funcionario' ('id_funcionario'),
19   CONSTRAINT 'fk_agendamento_endereco'
20   FOREIGN KEY ('id_endereco')
21   REFERENCES 'mydb'.'tb_endereco' ('id_endereco'))
22   ENGINE = InnoDB
23   DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.9: DDL – Tabela tb\_categoria

```

1    CREATE TABLE IF NOT EXISTS 'mydb'.'tb_categoria' (
2    'id_categoria' INT NOT NULL AUTO_INCREMENT,
3    'nome_categoria' VARCHAR(50) NOT NULL,
4    'descricao_categoria' VARCHAR(150) NULL DEFAULT NULL,
5    PRIMARY KEY ('id_categoria'),
6    UNIQUE INDEX 'nome_categoria_UNIQUE' ('nome_categoria' ASC)
      VISIBLE)
7    ENGINE = InnoDB
8    DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.10: DDL – Tabela tb\_fornecedor

```

1    CREATE TABLE IF NOT EXISTS 'mydb'.'tb_fornecedor' (
2    'id_fornecedor' INT NOT NULL AUTO_INCREMENT,
3    'nome_fornecedor' VARCHAR(50) NOT NULL,
4    'cpf_cnpj_fornecedor' VARCHAR(20) NOT NULL,
5    PRIMARY KEY ('id_fornecedor'),
6    UNIQUE INDEX 'cpf_cnpj_fornecedor_UNIQUE' ('cpf_cnpj_fornecedor'
      ' ASC) VISIBLE)
7    ENGINE = InnoDB
8    DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.11: DDL – Tabela tb\_compra

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_compra' (
2   'id_compra' INT NOT NULL AUTO_INCREMENT,
3   'id_fornecedor' INT NOT NULL,
4   'data' DATE NOT NULL,
5   PRIMARY KEY ('id_compra'),
6   INDEX 'id_fornecedor_idx' ('id_fornecedor' ASC) VISIBLE,
7   CONSTRAINT 'fk_compra_fornecedor'
8   FOREIGN KEY ('id_fornecedor')
9   REFERENCES 'mydb'.'tb_fornecedor' ('id_fornecedor')
10  ENGINE = InnoDB
11  DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.12: DDL – Tabela tb\_entrega

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_entrega' (
2   'id_entrega' INT NOT NULL AUTO_INCREMENT,
3   'id_venda' INT NOT NULL,
4   'id_endereco' INT NOT NULL,
5   'data_entrega' DATE NOT NULL,
6   'status_entrega' ENUM('pendente', 'entregue', 'cancelado') NULL
7   DEFAULT 'pendente',
8   PRIMARY KEY ('id_entrega'),
9   INDEX 'id_venda_idx' ('id_venda' ASC) VISIBLE,
10  INDEX 'id_endereco_idx' ('id_endereco' ASC) VISIBLE,
11  CONSTRAINT 'fk_entrega_venda'
12  FOREIGN KEY ('id_venda')
13  REFERENCES 'mydb'.'tb_venda' ('id_venda'),
14  CONSTRAINT 'fk_entrega_endereco'
15  FOREIGN KEY ('id_endereco')
16  REFERENCES 'mydb'.'tb_endereco' ('id_endereco')
17  ENGINE = InnoDB
  DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.13: DDL – Tabela tb\_produto

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_produto' (
2   'id_produto' INT NOT NULL AUTO_INCREMENT,
3   'nome_produto' VARCHAR(60) NOT NULL,
4   'descricao_produto' VARCHAR(150) NULL DEFAULT NULL,
5   'requer_montagem' TINYINT NOT NULL DEFAULT 0,
6   'id_categoria' INT NOT NULL,
7   PRIMARY KEY ('id_produto'),
8   UNIQUE INDEX 'nome_produto_UNIQUE' ('nome_produto' ASC) VISIBLE

```

```

,
INDEX 'id_categoria_idx' ('id_categoria' ASC) VISIBLE,
CONSTRAINT 'fk_produto_categoria'
FOREIGN KEY ('id_categoria')
REFERENCES 'mydb'.'tb_categoria' ('id_categoria'))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.14: DDL – Tabela tb\_estoque

```

CREATE TABLE IF NOT EXISTS 'mydb'.'tb_estoque' (
'id_estoque' INT NOT NULL AUTO_INCREMENT,
'id_compra' INT NOT NULL,
'id_produto' INT NOT NULL,
'quantidade' INT NOT NULL,
'preco_estoque' DECIMAL(10,2) NOT NULL,
PRIMARY KEY ('id_estoque'),
INDEX 'id_compra_idx' ('id_compra' ASC) VISIBLE,
INDEX 'id_produto_idx' ('id_produto' ASC) VISIBLE,
CONSTRAINT 'fk_estoque_compra'
FOREIGN KEY ('id_compra')
REFERENCES 'mydb'.'tb_compra' ('id_compra'),
CONSTRAINT 'fk_estoque_produto'
FOREIGN KEY ('id_produto')
REFERENCES 'mydb'.'tb_produto' ('id_produto'))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.15: DDL – Tabela tb\_pagamento\_cliente

```

CREATE TABLE IF NOT EXISTS 'mydb'.'tb_pagamento_cliente' (
'id_pagamento_cliente' INT NOT NULL AUTO_INCREMENT,
'id_venda' INT NOT NULL,
'data_pagamento' DATE NOT NULL,
'forma_pagamento' ENUM('cartao_credito', 'cartao_debito', 'pix',
, 'boleto', 'dinheiro') NOT NULL,
'status_pagamento' ENUM('pendente', 'pago', 'cancelado', '
atrasado') NOT NULL,
PRIMARY KEY ('id_pagamento_cliente'),
INDEX 'id_venda_idx' ('id_venda' ASC) VISIBLE,
CONSTRAINT 'fk_pagamento_venda'
FOREIGN KEY ('id_venda')
REFERENCES 'mydb'.'tb_venda' ('id_venda'))

```



```

12 ENGINE = InnoDB
13 DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.16: DDL – Tabela tb\_produto\_venda

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_produto_venda' (
2   'id_venda_itens' INT NOT NULL AUTO_INCREMENT,
3   'id_venda' INT NOT NULL,
4   'id_estoque' INT NOT NULL,
5   'quantidade' INT NOT NULL,
6   'valor' DECIMAL(10,2) NOT NULL,
7   PRIMARY KEY ('id_venda_itens'),
8   INDEX 'fk_venda_idx' ('id_venda' ASC) VISIBLE,
9   INDEX 'fk_estoque_idx' ('id_estoque' ASC) VISIBLE,
10  CONSTRAINT 'fk_produto_venda_to_venda'
11  FOREIGN KEY ('id_venda')
12  REFERENCES 'mydb'.'tb_venda' ('id_venda'),
13  CONSTRAINT 'fk_produto_venda_to_estoque'
14  FOREIGN KEY ('id_estoque')
15  REFERENCES 'mydb'.'tb_estoque' ('id_estoque'))
16  ENGINE = InnoDB
17  DEFAULT CHARACTER SET = utf8mb4;

```

Listing 2.17: DDL – Tabela tb\_receita\_extra

```

1 CREATE TABLE IF NOT EXISTS 'mydb'.'tb_receita_extra' (
2   'id_receita' INT NOT NULL AUTO_INCREMENT,
3   'descricao_receita' VARCHAR(150) NOT NULL,
4   'valor' DECIMAL(10,2) NOT NULL,
5   'data_recebimento' DATE NOT NULL,
6   'id_funcionario' INT NOT NULL,
7   PRIMARY KEY ('id_receita'),
8   INDEX 'id_funcionario_idx' ('id_funcionario' ASC) VISIBLE,
9   CONSTRAINT 'fk_receita_funcionario'
10  FOREIGN KEY ('id_funcionario')
11  REFERENCES 'mydb'.'tb_funcionario' ('id_funcionario'))
12  ENGINE = InnoDB
13  DEFAULT CHARACTER SET = utf8mb4;

```

## 2.2 Carga de Dados Iniciais

Após a criação do banco, uma carga inicial é inserida por meio de DML:

Listing 2.18: DML – Tabela tb\_cargo

```

1      INSERT INTO tb_cargo (id_cargo, nome_cargo, descricao_cargo,
2      salario_base) VALUES
3      (1, 'Vendedor', 'Responsável pelo atendimento e vendas em loja',
4      2000.00),
5      (2, 'Montador', 'Realiza montagem de móveis', 2200.00),
6      (3, 'Gerente', 'Gerencia a equipe e as metas da loja', 4500.00);

```

Listing 2.19: DML – Tabela tb\_funcionario

```

1      INSERT INTO tb_funcionario (id_funcionario, nome_funcionario,
2      cpf_funcionario, id_cargo) VALUES
3      (1, 'José Seben', '00000000000', 3),
4      (2, 'Caio Macedo', '11111111111', 1),
5      (3, 'Kaique Lima', '22222222222', 2),
6      (4, 'Fernando Buligon', '33333333333', 1);

```

Listing 2.20: DML – Tabela tb\_cliente

```

1      INSERT INTO tb_cliente (id_cliente, nome_cliente,
2      cpf_cnpj_cliente, telefone_cliente) VALUES
3      (1, 'João Oliveira', '00000000000', '(00) 00000-0000'),
4      (2, 'Maria Santos', '11111111111', '(11) 11111-1111'),
5      (3, 'Eduardo Costa', '22222222222', '(22) 22222-2222'),
6      (4, 'Gabriela Barros', '33333333333', '(33) 33333-3333'),
7      (5, 'Pedro Cachoeira', '44444444444', '(44) 44444-4444');

```

Listing 2.21: DML – Tabela tb\_cidade

```

1      INSERT INTO tb_cidade (id_cidade, nome_cidade) VALUES
2      (1, 'Santa Helena'),
3      (2, 'Sao Clemente'),
4      (3, 'Medianeira'),
5      (4, 'Subsede'),
6      (5, 'Entre Rios');

```

Listing 2.22: DML – Tabela tb\_bairro

```

1      INSERT INTO tb_bairro (id_bairro, nome_bairro, id_cidade)
2      VALUES
3      (1, 'Centro', 1),
4      (2, 'Centro', 2),
5      (3, 'Centro', 3),
6      (4, 'Centro', 4),
7      (5, 'Centro', 5);

```

### Listing 2.23: DML – Tabela tb\_endereco

```
1  INSERT INTO tb_endereco (id_endereco, id_bairro, logradouro,
2      numero, complemento, id_cliente, prioridade) VALUES
3      (1,1,'Rua_Xaxim','123',NULL,1,1),
4      (2,2,'Av._Brasil','321',NULL,2,1),
5      (3,3,'Rua_Aroeira','231',NULL,3,1),
6      (4,4,'Rua_Argentina','213',NULL,4,1),
7      (5,4,'Rua_Argentina','312',NULL,5,1);
```

### Listing 2.24: DML – Tabela tb\_categoria

```
1  INSERT INTO tb_categoria (id_categoria, nome_categoria,
2      descricao_categoria) VALUES
3      (1,'Ferramentas','Makita,_martelo,_pá...'),
4      (2,'Mobilia','Guarda-roupa,_mesa,_bancada...'),
5      (3,'Materiais','Areia,_pedra,_prego...');
```

### Listing 2.25: DML – Tabela tb\_fornecedor

```
1  INSERT INTO tb_fornecedor (id_fornecedor, nome_fornecedor,
2      cpf_cnpj_fornecedor) VALUES
3      (1,'Kaique_Madeiras','22222222222222'),
4      (2,'João_das_Areias','33333333333333'),
5      (3,'Pedro_Pedradas','44444444444444'),
6      (4,'Rafael_Pregos','55555555555555'),
7      (5,'Raul_Rolamentos','66666666666666');
```

### Listing 2.26: DML – Tabela tb\_produto

```
1  INSERT INTO tb_produto (id_produto, nome_produto,
2      descricao_produto, requer_montagem, id_categoria) VALUES
3      (1,'Makita_5007N','Serra_Circular_7-1/4_1800W',0,1),
4      (2,'Madeira_Pinus','Madeira_boa',0,3),
5      (3,'Mesa_Retangular','Mesa_de_madeira',1,2),
6      (4,'Pedra_Brita','Essa_é_da_boa',0,3),
7      (5,'Areia_Média','Funciona_bem',0,3);
```

### Listing 2.27: DML – Tabela tb\_compra

```
1  INSERT INTO tb_compra (id_compra, id_fornecedor, data) VALUES
2      (1,1,'2025-08-10'),
3      (2,1,'2025-08-11'),
4      (3,1,'2025-08-12'),
5      (4,1,'2025-08-13');
```

Listing 2.28: DML – Tabela tb\_estoque

```

1  INSERT INTO tb_estoque (id_estoque, id_compra, id_produto,
    quantidade, preco_estoque) VALUES
2  (1,1,1, 50, 793.78),
3  (2,2,2,100, 120.00),
4  (3,2,3, 20, 450.00),
5  (4,3,4,200, 35.00),
6  (5,4,5,180, 30.00);

```

Listing 2.29: DML – Tabela tb\_venda

```

1  INSERT INTO tb_venda (id_venda, id_cliente, id_funcionario,
    data, status, observacao_venda) VALUES
2  (1,1,2, '2025-08-15', 'concluida', 'Cara_saiu_feliz'),
3  (2,2,2, '2025-08-16', 'pendente', 'Aguardando_pagamento'),
4  (3,3,4, '2025-08-17', 'concluida', 'Sucesso'),
5  (4,4,4, '2025-08-18', 'cancelada', 'Cartão_não_passou');

```

Listing 2.30: DML – Tabela tb\_produto\_venda

```

1  INSERT INTO tb_produto_venda (id_venda_itens, id_venda,
    id_estoque, quantidade, valor) VALUES
2  (1,1,1,189,793.78),
3  (2,2,1, 1,793.78),
4  (3,3,1, 1,793.78),
5  (4,4,1, 1,793.78);

```

Listing 2.31: DML – Tabela tb\_pagamento\_cliente

```

1  INSERT INTO tb_pagamento_cliente (id_pagamento_cliente,
    id_venda, data_pagamento, forma_pagamento, status_pagamento)
    VALUES
2  (1,1, '2025-08-15', 'pix', 'pago'),
3  (2,2, '2025-08-17', 'boleto', 'pendente'),
4  (3,3, '2025-08-17', 'boleto', 'pago'),
5  (4,4, '2025-08-18', 'cartao_credito', 'cancelado');

```

Listing 2.32: DML – Tabela tb\_entrega

```

1  INSERT INTO tb_entrega (id_entrega, id_venda, id_endereco,
    data_entrega, status_entrega) VALUES
2  (1,1,1, '2025-08-16', 'entregue'),
3  (2,2,2, '2025-08-20', 'pendente'),
4  (3,3,3, '2025-08-18', 'entregue');

```

### Listing 2.33: DML – Tabela tb\_agendamento\_montagem

```
1  INSERT INTO tb_agendamento_montagem (id_agendamento_montagem,
    id_venda_itens, id_funcionario, id_endereco, data,
    status_montagem, observacao) VALUES
2  (1,1,2,1,'2025-08-16','realizado','Cliente_chato'),
3  (2,2,2,2,'2025-08-21','pendente','No_aguardo_da_chegada_dos_
    materiais'),
4  (3,3,2,3,'2025-08-19','realizado','Sucesso');
```

### Listing 2.34: DML – Tabela tb\_receita\_extra

```
1  INSERT INTO tb_receita_extra (id_receita, descricao_receita,
    valor, data_recebimento, id_funcionario) VALUES
2  (1,'Rapaz_merece',350.00,'2025-08-19',2),
3  (2,'Bateu_o_recorde_em_montagem_de_balcão',120.00,'2025-08-19',
    3),
4  (3,'Comprou_makita',150.00,'2025-08-20',1);
```

### 3 CONSULTAS DE DADOS NO BD

A linguagem SQL permite diferentes recursos para recuperar informações. As seções abaixo listam os tipos de consultas (exemplos a completar).

#### 3.1 Restrições de Linha

Listing 3.1: SELECT – Tabela tb\_cliente

```
1 SELECT * FROM tb_cliente
2 WHERE cpf_cnpj_cliente IN ('00000000000', '11111111111');
```

Listing 3.2: SELECT – Tabela tb\_cargo

```
1 SELECT * FROM tb_cargo
2 WHERE salario_base > 3000.00;
```

Listing 3.3: SELECT – Tabela tb\_funcionario

```
1 SELECT * FROM tb_funcionario
2 WHERE id_cargo = 1;
```

Listing 3.4: SELECT – Tabela tb\_venda

```
1 SELECT * FROM tb_venda
2 WHERE status = 'concluida'
```

Listing 3.5: SELECT – Tabela tb\_cidade

```
1 SELECT * FROM tb_cidade
2 WHERE nome_cidade IN ('Santa Helena', 'Medianeira');
```

Listing 3.6: SELECT – Tabela tb\_bairro

```
1 SELECT * FROM tb_bairro
2 WHERE nome_bairro = 'Centro'
3 AND id_cidade IN (4,5);
```

Listing 3.7: SELECT – Tabela tb\_endereco

```
1 SELECT * FROM tb_endereco
2 WHERE id_cliente IN (1,5)
3 AND complemento IS NULL;
```

Listing 3.8: SELECT – Tabela tb\_agendamento\_montagem

```
1 SELECT * FROM tb_agendamento_montagem
2 WHERE status_montagem = 'realizado'
3 AND data >= '2025-08-19';
```

Listing 3.9: SELECT – Tabela tb\_categoria

```
1 SELECT * FROM tb_categoria
2 WHERE nome_categoria <> 'Ferramentas';
```

Listing 3.10: SELECT – Tabela tb\_fornecedor

```
1 SELECT * FROM tb_fornecedor
2 WHERE cpf_cnpj_fornecedor LIKE '33%';
```

Listing 3.11: SELECT – Tabela tb\_compra

```
1 SELECT * FROM tb_compra
2 WHERE id_fornecedor = 1
3 AND data >= '2025-08-12';
```

Listing 3.12: SELECT – Tabela tb\_entrega

```
1 SELECT * FROM tb_entrega
2 WHERE status_entrega = 'pendente'
3 OR data_entrega > '2025-08-17';
```

Listing 3.13: SELECT – Tabela tb\_produto

```
1 SELECT * FROM tb_produto
2 WHERE id_categoria = 3
3 AND descricao_produto IS NOT NULL;
```

Listing 3.14: SELECT – Tabela tb\_estoque

```
1 SELECT * FROM tb_estoque
2 WHERE quantidade >= 100
3 AND preco_estoque <= 120.00;
```

Listing 3.15: SELECT – Tabela tb\_pagamento\_cliente

```
1 SELECT * FROM tb_pagamento_cliente
2 WHERE forma_pagamento IN ('pix','boleto')
3 AND status_pagamento = 'pago';
```

#### Listing 3.16: SELECT – Tabela tb\_produto\_venda

```
1      SELECT * FROM tb_produto_venda
2      WHERE id_produto = 1
3      AND quantidade >= 100;
```

#### Listing 3.17: SELECT – Tabela tb\_receita\_extra

```
1      SELECT * FROM tb_receita_extra
2      WHERE data_recebimento BETWEEN '2025-08-19' AND '2025-08-20'
3      ,
      AND valor >= 150.00;
```

### 3.2 Funções de Linha

...

### 3.3 Junções de Tabelas

...

### 3.4 Operadores de Conjunto

Consultas que combinam conjuntos de linhas podem utilizar UNION (remoção de duplicatas) e UNION ALL (mantém duplicatas). Em MySQL, operações como INTERSECT e EXCEPT podem ser obtidas por equivalentes com JOIN, EXISTS ou NOT EXISTS.

#### Listing 3.18: SELECT – UNION: nomes de clientes e fornecedores sem duplicatas

```
1      SELECT nome_cliente AS nome, 'cliente' AS origem
2      FROM tb_cliente
3      UNION
4      SELECT nome_fornecedor AS nome, 'fornecedor' AS origem
5      FROM tb_fornecedor;
```

#### Listing 3.19: SELECT – UNION ALL: contabiliza todas as ocorrências (com duplicatas)

```
1      SELECT nome_cliente AS nome
2      FROM tb_cliente
3      UNION ALL
4      SELECT nome_fornecedor AS nome
5      FROM tb_fornecedor;
6      -- Útil quando se deseja manter contagens exatas de ocorrências
```

#### Listing 3.20: SELECT – INTERSECT (equivalente): nomes presentes em cliente e fornecedor



```

1      SELECT c.nome_cliente AS nome
2      FROM tb_cliente c
3      WHERE EXISTS (
4          SELECT 1
5          FROM tb_fornecedor f
6          WHERE f.nome_fornecedor = c.nome_cliente
7      );
8      -- Equivalente ao INTERSECT dos nomes (quando houver
        coincidência literal)

```

Listing 3.21: SELECT – EXCEPT (equivalente): clientes que nunca compraram

```

1      SELECT c.id_cliente, c.nome_cliente
2      FROM tb_cliente c
3      WHERE NOT EXISTS (
4          SELECT 1
5          FROM tb_venda v
6          WHERE v.id_cliente = c.id_cliente
7      );
8      -- Equivalente ao "clientes" EXCEPT "clientes com venda"

```

Listing 3.22: SELECT – UNION: cidades e bairros (rótulo por tipo)

```

1      SELECT nome_cidade AS nome, 'cidade' AS tipo
2      FROM tb_cidade
3      UNION
4      SELECT nome_bairro AS nome, 'bairro' AS tipo
5      FROM tb_bairro;

```

### 3.5 Sub-Consultas

Subconsultas podem ser escalares (retornam um único valor), de conjunto (IN), correlacionadas (EXISTS/NOT EXISTS) ou em FROM (tabelas derivadas). Abaixo, exemplos aplicados ao schema definido.

Listing 3.23: SELECT – Subconsulta escalar: maior salário-base de cargo

```

1      SELECT nome_cargo, salario_base
2      FROM tb_cargo
3      WHERE salario_base = (
4          SELECT MAX(salario_base) FROM tb_cargo
5      );

```

Listing 3.24: SELECT – IN: clientes com ao menos uma venda concluída

```

1      SELECT c.id_cliente, c.nome_cliente
2      FROM tb_cliente c
3      WHERE c.id_cliente IN (
4          SELECT v.id_cliente
5          FROM tb_venda v
6          WHERE v.status = 'concluida'
7      );

```

Listing 3.25: SELECT – EXISTS (correlata): fornecedores que já tiveram compras

```

1      SELECT f.id_fornecedor, f.nome_fornecedor
2      FROM tb_fornecedor f
3      WHERE EXISTS (
4          SELECT 1
5          FROM tb_compra co
6          WHERE co.id_fornecedor = f.id_fornecedor
7      );

```

Listing 3.26: SELECT – Subconsulta correlata com agregação: vendas com total > 1000

```

1      SELECT v.id_venda, v.data, v.status
2      FROM tb_venda v
3      WHERE (
4          SELECT SUM(pv.valor * pv.quantidade)
5          FROM tb_produto_venda pv
6          WHERE pv.id_venda = v.id_venda
7      ) > 1000.00;

```

Listing 3.27: SELECT – Tabela derivada (FROM): top categorias por itens vendidos

```

1      SELECT d.id_categoria, d.nome_categoria, d.qtd_itens
2      FROM (
3          SELECT pr.id_categoria,
4                 ca.nome_categoria,
5                 SUM(pv.quantidade) AS qtd_itens
6          FROM tb_produto_venda pv
7          JOIN tb_estoque es  ON es.id_estoque = pv.id_estoque
8          JOIN tb_produto pr  ON pr.id_produto = es.id_produto
9          JOIN tb_categoria ca ON ca.id_categoria = pr.
10                 id_categoria
11          GROUP BY pr.id_categoria, ca.nome_categoria
12      ) AS d
13      ORDER BY d.qtd_itens DESC

```

13

`LIMIT 5;`

Listing 3.28: SELECT – NOT EXISTS (correlata): produtos que requerem montagem e nunca foram vendidos

```
1      SELECT p.id_produto, p.nome_produto
2      FROM tb_produto p
3      WHERE p.requer_montagem = 1
4             AND NOT EXISTS (
5                 SELECT 1
6                 FROM tb_estoque e
7                 JOIN tb_produto_venda pv ON pv.id_estoque = e.
8                     id_estoque
9                     WHERE e.id_produto = p.id_produto
10             );
```

Listing 3.29: SELECT – Subconsulta escalar em projeção: total (R\$) por venda

```
1      SELECT v.id_venda,
2             v.status,
3             (
4                 SELECT SUM(pv.valor * pv.quantidade)
5                 FROM tb_produto_venda pv
6                 WHERE pv.id_venda = v.id_venda
7             ) AS total_venda
8      FROM tb_venda v;
```

### 3.6 Agregação e Funções de Agrupamento

...

### 3.7 SELECT nos Comandos

...

## **4 ROTINAS DE OTIMIZAÇÃO NO BD**

### **4.1 Visões**

...

### **4.2 Funções**

...

### **4.3 Procedimentos**

...

### **4.4 Gatilhos**

...

### **4.5 Eventos**

...

## **5 CONTROLE DE ACESSO**

O controle de acesso é dado por GRANT e REVOKE. As subseções a seguir seguem o PDF.

### **5.1 Criar Usuários**

...

### **5.2 Atribuir Permissões**

...

### **5.3 Revogar Permissões**

...

## **6 TRANSAÇÕES NO BD**

Defina blocos transacionais atômicos para preservar ACID. . . .

## **7 FALHAS NO BD**

Durante a execução de qualquer software, falhas podem ocorrer; tolerâncias e tratamentos devem ser estabelecidos.

### **7.1 Tolerâncias à Falhas**

...

### **7.2 Tratamento à Falhas**

...

## REFERÊNCIAS

- ...