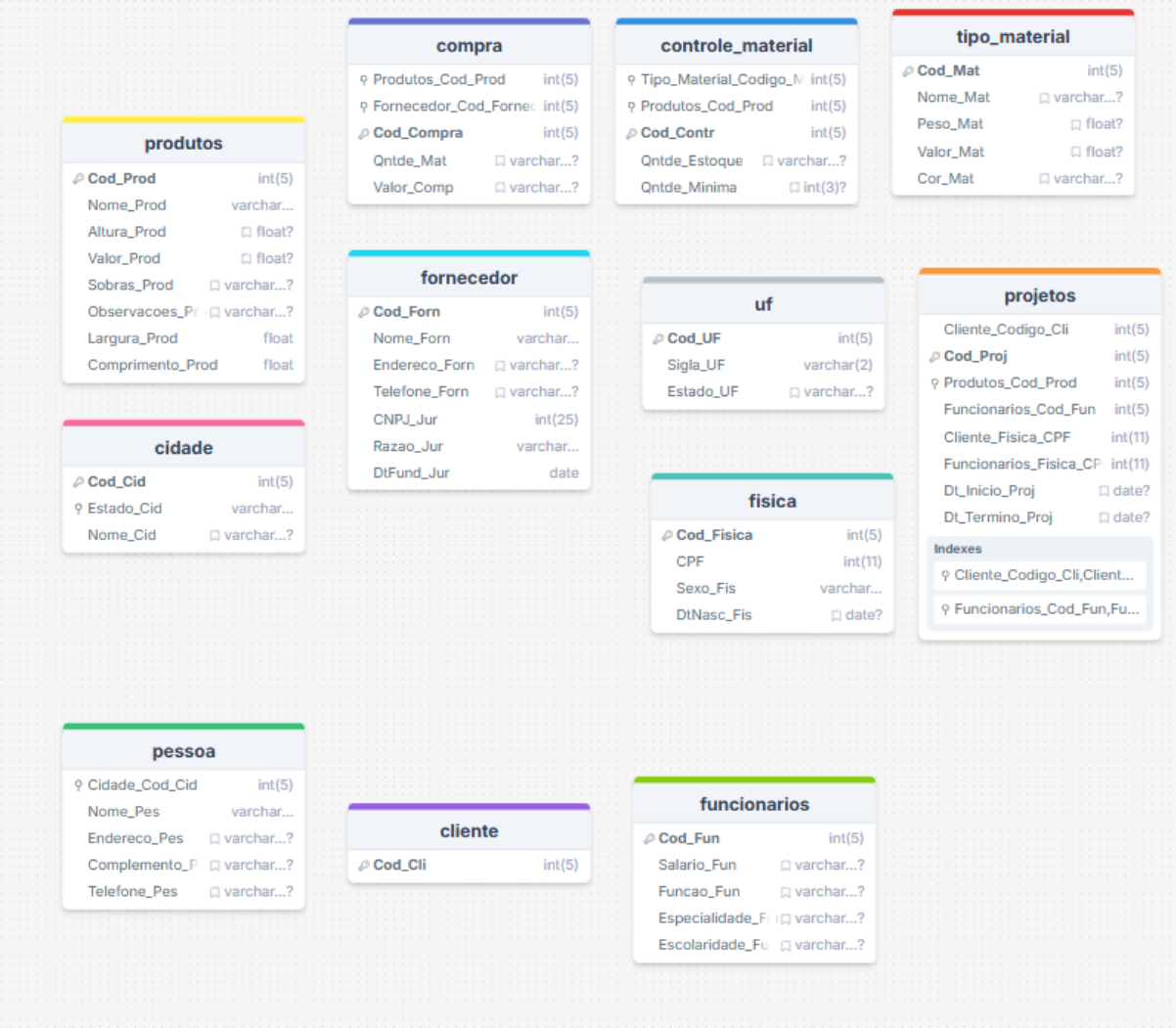
BANCO DE DADOS

FEITO POR: **MARIO GONÇALVES DE FREITAS JUNIOR.**

Introdução

A modelagem de dados é fundamental para garantir que um sistema funcione corretamente. Quando a estrutura do banco não é bem planejada, surgem problemas como informações duplicadas, dados inconsistentes e dificuldade para evoluir o sistema. Por isso é importante dedicar tempo para entender as entidades (tabelas), os atributos (colunas) e como tudo se relaciona antes de começar a programar ou montar o banco de dados, para ilustrar a situação de seu banco coloquei uma imagem abaixo para melhorar a visualização.



Análise Crítica do Modelo 1

1. **Tabela “cliente” isolada**

2. A tabela cliente só tem um campo de código (Cod_Cli) e não está ligada a nada. Ao mesmo tempo, existe a tabela pessoa, mas nela faltam informações essenciais (como CPF e telefone) para identificar o cliente. Além disso, há uma tabela fisica só para dizer “pessoa física”, mas sem integrar direito com cliente ou pessoa. Isso acaba espalhando dados do mesmo cliente em várias tabelas e gera redundância.

3. **Tipos de dados inadequados**

Vários campos que guardam valores monetários ou numéricos estão como VARCHAR. Por exemplo, salário de funcionário (Salario_Fun) e valor de compra (Valor_Comp) estão em texto, quando deveriam ser DECIMAL ou FLOAT. Colunas de data (como data de fundação do fornecedor) também estão como VARCHAR, quando deveriam ser DATE. Isso dificulta qualquer operação matemática ou filtragem/ordenamento por data.

4. **Relacionamentos confusos ou ausentes**
    a. **Produto × Fornecedor**: não há chave estrangeira ligando produto a fornecedor. Não dá para saber quem fornece cada produto.
    b. **Compras × Cliente**: a tabela “compras” relaciona produto e fornecedor, mas não inclui o cliente que fez a compra. Sem esse link, não dá para saber quem comprou o quê.
    c. **Projeto × Funcionário**: não existe tabela associativa para muitos-para-muitos entre projetos e funcionários. No Modelo 1, os campos de funcionário estão dentro de “projetos” de forma desnormalizada, o que causa duplicação e dificulta atualização.
5. **Tabelas desnecessárias**

Existe “cidade” e “uf” mas elas não são usadas como chave estrangeira em nenhum lugar. Quase todo sistema acaba usando cidade e estado como texto nos cadastros, sem levar para uma tabela separada. Então essas tabelas só complicam sem necessidade.

6. **Nomenclatura confusa**

Abreviações como Cod_Cid, Cod_Fun, Cod_Form não são intuitivas. E a tabela “fisica” para indicar “pessoa física” pode confundir quem for ler depois. É melhor usar nomes mais simples, tipo “pessoa_fisica” ou incorporar esses dados diretamente na tabela de cliente.

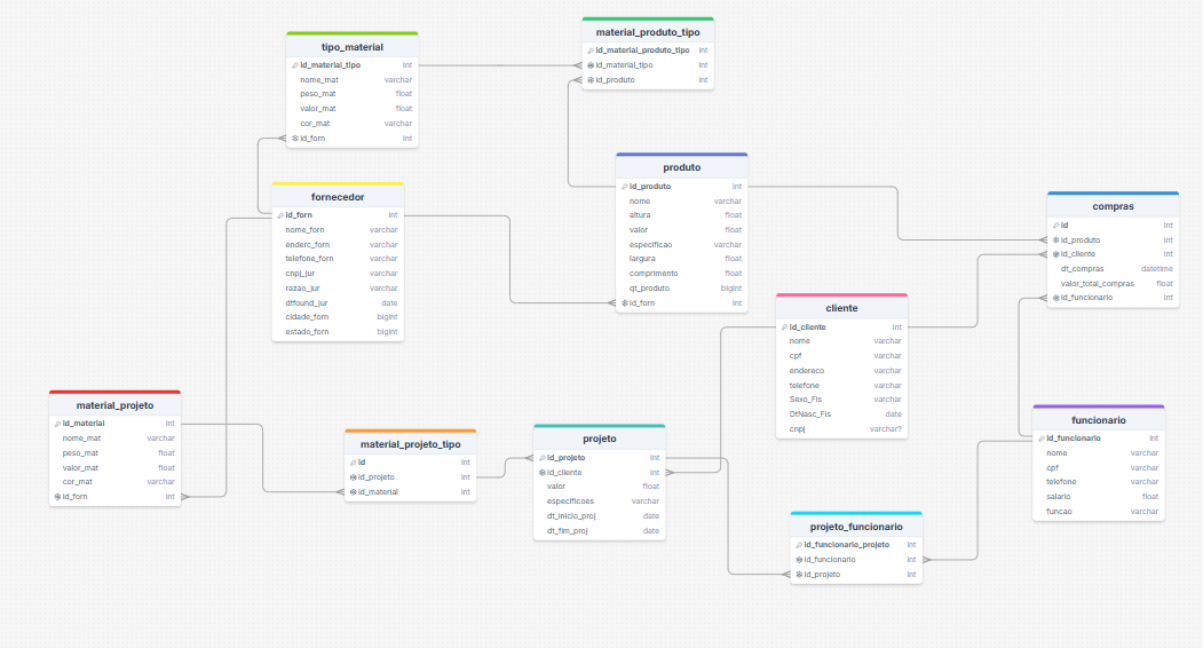
Propostas de Melhoria

1. **Unificar entidade Cliente**
    a. Antes: tinha “cliente”, “pessoa” e “fisica” separadas, sem ligação clara e sem CPF.
    b. Agora: crio tabela cliente com campos:
        i. id_cliente INT (chave primária)
        ii. nome VARCHAR(100)
        iii. cpf CHAR(11) (único e NOT NULL)
        iv. data_nascimento DATE
        v. telefone VARCHAR(20)
        vi. endereco VARCHAR(200)
    c. **Por que:** assim a informação fica toda num lugar só, sem redundância nem fragmentação. O CPF garante que cada pessoa seja única.
2. **Tabelas Funcionário e Fornecedor simplificadas**
    a. **Funcionário:** crio funcionario com id_funcionario INT, nome, cpf CHAR(11), telefone, salario DECIMAL(10,2) e funcao VARCHAR(50).
    b. **Fornecedor:** crio fornecedor com id_forn INT, nome, cnpj CHAR(14), razao_social, data_fundacao DATE, telefone, endereco, cidade VARCHAR(50) e estado CHAR(2).
    c. **Por que:** removo tabelas “cidade” e “uf” e coloco essas informações direto no fornecedor. Ajusto salário e valores para DECIMAL, datas para DATE.
3. **Tabela Produto com relação a Fornecedor**
    a. Crio produto com id_produto INT, nome VARCHAR(100), especificacao VARCHAR(200), altura DECIMAL(6,2), largura DECIMAL(6,2), comprimento DECIMAL(6,2), quantidade INT, valor DECIMAL(10,2) e id_forn INT (chave estrangeira para fornecedor).
    b. **Por que:** assim fica claro quem fornece cada produto e todos os tipos de dados ficam adequados.
4. **Tabela Compras ligando Cliente e Produto**
    a. Crio compras com id_compra INT, id_produto INT, id_cliente INT, data_compra DATETIME e valor_total DECIMAL(12,2).
    b. **Por que:** agora sei quem comprou, o que comprou e quanto pagou, tudo num esquema normalizado.
5. **Projetos e associações muitos-para-muitos**
    a. Tabela projeto com id_projeto INT, id_cliente INT, descricao VARCHAR(200), valor DECIMAL(12,2), data_inicio DATE e data_fim DATE.
    b. Tabela projeto_funcionario com id_projeto_funcionario INT, id_projeto INT e id_funcionario INT (com UNIQUE em (id_projeto, id_funcionario)).

c. Tabela material_projeto_tipo com id_material_projeto_tipo INT, id_projeto INT, id_material_tipo INT e quantidade BIGINT.
d. **Por que:** resolvo o relacionamento muitos-para-muitos entre projetos e funcionários, e controlo uso de materiais por projeto de forma clara e normalizada.

6. **Padronização de nomes e tipos**
   a. Uso sempre id_<tabela> como chave primária (INT AUTO_INCREMENT).
   b. Chaves estrangeiras nomeadas como id_<tabela_referenciada>.
   c. Valores monetários em DECIMAL, datas em DATE/DATETIME, CPF/CNPJ em CHAR fixo.
   d. Removo tabelas desnecessárias: “cidade”, “uf”, “controle_material”, “pessoa” e “fisica”.

A IMAGEM ABAIXO MOSTRA A VERSÃO FINAL DO BANCO DE DADOS.



Conclusão e Recomendações Finais

Com essas mudanças, o banco de dados fica mais consistente:

- **Integridade referencial** garantida (todas as chaves estrangeiras estão no lugar certo).
- **Ausência de redundância** (cada dado aparece uma vez em apenas uma tabela).
- **Tipos de dados adequados** facilitam cálculos, filtros e ordenações.
- **Nomes claros** ajudam quem for manter ou evoluir o sistema.

Dessa forma, o banco de dados fica pronto para crescer de forma organizada, atender às necessidades do negócio e facilitar a manutenção no dia a dia.

Atenciosamente,

Mario Gonçalves de Freitas Júnior

02/06/2025.