Modelagem de Dados

Introdução à Modelagem e Normalização de Banco de dados

Objetivos

- Modelagem de Dados
- Siglas Importantes para Banco de Dados
    - SGDB (Sistema Gerenciador de Bando de Dados)
    - MER (Modelo Entidade Relacionamento)
    - DER (Diagrama Entidade Relacionamento)
- Entidades
- Relacionamentos
    - Relacionamento Muitos-para-muitos (n:n)
    - Relacionamento Um-para-um (1:1)
    - Relacionamento Um-para-muitos (1:n)
- Atributos
- Normalização Banco de Dados
- Criando as Tabelas
    - Criando as Chaves Primárias
    - Criando Chaves Estrangeiras
- Formas Normais
    - Primeira Forma Normal (1FN)
    - Segunda Forma Normal (2FN)
    - Terceira Forma Normal (3FN)
- Levantamento de Requisitos na Modelagem de Dados

Modelagem de Dados

A **Modelagem de Dados** é a criação de um modelo físico que explique a lógica por trás do sistema, com ele você é capaz de explicar as características de funcionamento e comportamento de um software. A **modelagem de dados** é a base de criação do banco de dados e parte essencial para a qualidade do software.

SGDB (Sistema Gerenciador de Bando de Dados)

É o software utilizado para criar e manter o banco de dados em operação. No nosso caso, estamos utilizando o **MySQL**, mas existem outras SGDBs que também podem ser utilizadas pelas empresas.

MER (Modelo Entidade Relacionamento)

O Modelo Entidade Relacionamento (também chamado Modelo ER, ou simplesmente MER), como o nome sugere, é um modelo conceitual utilizado na Engenharia de Software para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos).

DER (Diagrama Entidade Relacionamento)

Enquanto o MER é um modelo conceitual, o Diagrama Entidade Relacionamento (Diagrama ER ou ainda DER) é a sua representação gráfica e principal ferramenta. Em situações práticas, o diagrama é tido muitas vezes como sinônimo de modelo, uma vez que sem uma forma de visualizar as informações, o modelo pode ficar abstrato demais para auxiliar no desenvolvimento do sistema.
Em sua notação original, proposta por **Peter Chen** (idealizador do modelo e do diagrama), as **entidades deveriam ser representadas por retângulos, seus atributos por elipses e os relacionamentos por losangos, ligados às entidades por linhas, contendo também sua cardinalidade (1..1, 1..n ou n..n).**

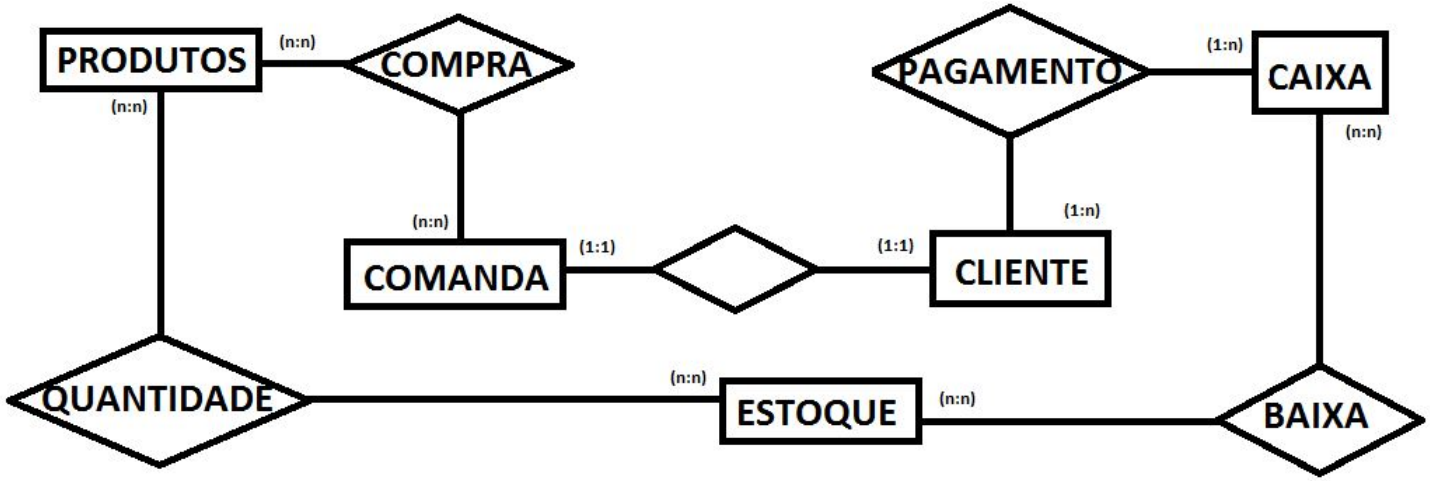
Entidades

São representações de algo no mundo físico, por exemplo no nosso caso temos as entidades produtos, comanda e cliente. Que são representados pelo símbolo abaixo.

Além das **Entidades**, também temos os **relacionamentos** entre as entidades, que nada mais é do que a ligação entre duas entidades, ou algo que faça com que essas entidades tenham algo em comum.

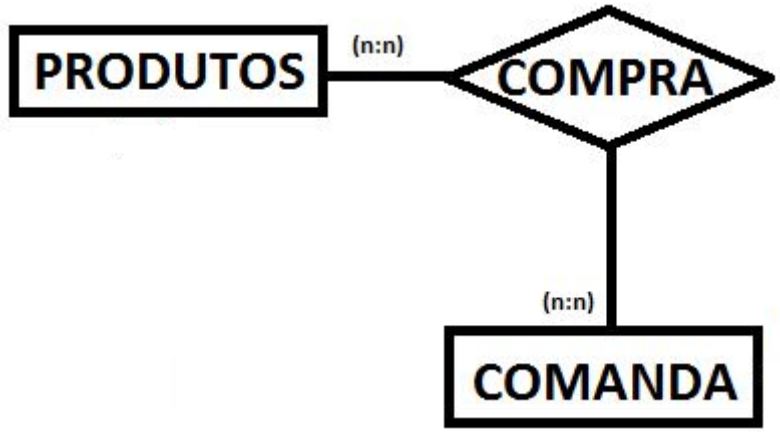
Relacionamentos

Uma vez que as entidades são identificadas, deve-se então definir como se dá o relacionamento entre elas. De acordo com a quantidade de objetos envolvidos em cada lado do relacionamento, podemos classificá-los de três formas:



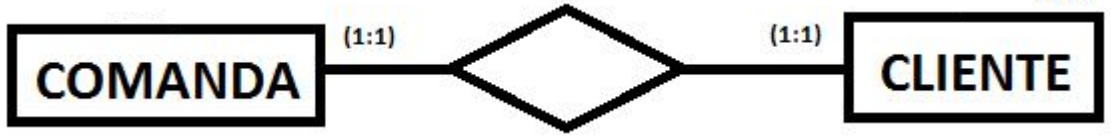
Relacionamento Muitos-para-muitos (n:n)

Na linguagem acadêmica é explicado da seguinte forma: Uma entidade em “A” está associada a qualquer número de entidades em “B” e vice-versa. Alguns autores preferem chamar esta cardinalidade de **m:n**, por considerar que podem representar valores diferentes.



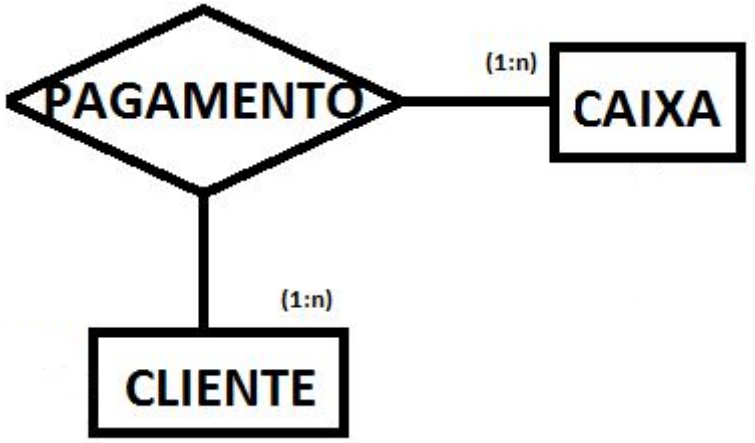
Relacionamento Um-para-um (1:1)

Na linguagem acadêmica é explicado da seguinte forma: Uma entidade em “A” está associada com no máximo uma entidade em “B”, e uma entidade em “B” está associada com no máximo uma entidade em “A”.



Relacionamento Um-para-muitos (1:n)

Na linguagem acadêmica é explicado da seguinte forma: Uma entidade em “A” está associado a qualquer número de entidades em “B”, e uma entidade em “B”, todavia, pode estar associado a no máximo uma entidade em “A”.



Atributos

Atributos são as características que descrevem cada entidade dentro do domínio. Por exemplo, um cliente possui nome, endereço e telefone. Durante o **levantamento de requisitos**, são identificados os **atributos** relevantes de cada **entidade** naquele contexto, de forma a manter o modelo o mais simples possível e consequentemente armazenar apenas as informações que serão úteis futuramente.
Os atributos podem ser classificados quanto à sua função da seguinte forma:

- **Descritivos**: representam característica intrínsecas de uma entidade, tais como nome ou cor.

- **Nominativos**: além de serem também descritivos, estes têm a função de definir e identificar um objeto. Nome, código, número são exemplos de atributos nominativos.

- **Referenciais**: representam a ligação de uma entidade com outra em um relacionamento. Por exemplo, uma venda possui o CPF do cliente, que a relaciona com a entidade cliente.

Quanto à sua estrutura, podemos ainda classificá-los como:

- **Simples**: um único atributo define uma característica da entidade. Exemplos: nome, peso.

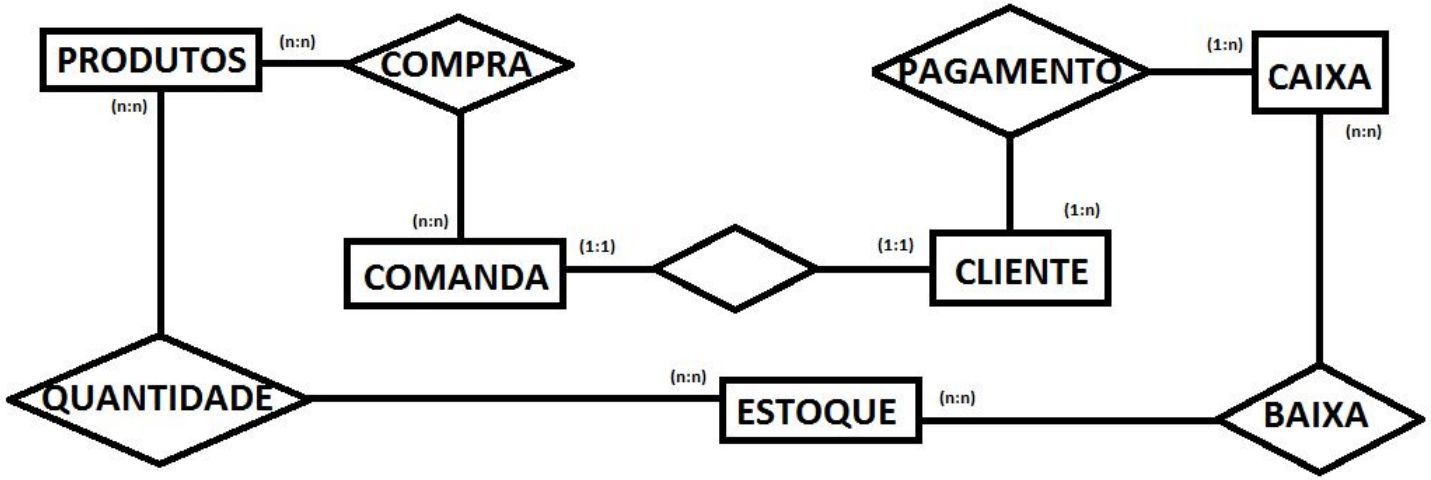
- **Compostos**: para definir uma informação da entidade, são usados vários atributos. Por exemplo, o endereço pode ser composto por rua, número, bairro, etc.

Normalização Banco de Dados

Normalização de dados é o processo formal e passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros.

A regra de ouro que devemos observar no projeto de um banco de dados baseado no Modelo Relacional de Dados é a de "não misturar assuntos em uma mesma Tabela".

Vamos converter o modelo de dados em **tabelas** e, para isso, existem algumas regras a serem seguidas.



Criando as Tabelas

- **Toda entidade vira uma tabela**
  Seguindo a regra teremos as seguintes tabelas:
  - TB_PRODUTO
  - TB_COMANDA
  - TB_ESTOQUE
  - TB_CLIENTE
  - TB_CAIXA

- **Relacionamentos que possuem atributos viram tabelas**
  Vamos criar as tabelas desses relacionamentos.
  - TB_PRODUTO_COMANDA
  - TB_PRODUTO_ESTOQUE
  - TB_CAIXA_ESTOQUE
  - TB_CLIENTE_PAGAMENTO

- **Observação**
  - Relacionamentos 1:1 podem ser mapeados numa única tabela, quando possuem a mesma chave primária); em duas tabelas, quando as chaves primárias são diferentes e um dos lados do relacionamento é obrigatório; ou em três tabelas, quando o relacionamento é opcional em ambos os lados;

  - Relacionamentos 1:n são mapeados de forma que a chave primária do lado “1” seja representada do lado “n” como chave estrangeira;

  - Relacionamentos n:n devem ser transformados em dois relacionamentos 1:n, resultando numa nova tabela.

Criando as Chaves Primárias

Toda tabela possui um ou mais campos que são os campos únicos, onde cada entidade se diferencia, por exemplo, um cliente possui um CPF único que pode ser o que diferencia todos os clientes, estes campo únicos são chamados de **chaves primárias**.

Abaixo seguem as chaves primárias de todas as tabelas criadas.

- TB_PRODUTO (ID_PRODUTO)
- TB_COMANDA (ID_COMANDA, DT_INICIO, DT_FIM)
- TB_ESTOQUE (ID_ESTOQUE)
- TB_CLIENTE (ID_CLIENTE (Que nesse caso vai ser o CPF))
- TB_CAIXA (ID_PAGAMENTO)
- TB_PRODUTO_COMANDA (ID_PRODUTO, ID_COMANDA, DT_INICIO, DT_FIM)
- TB_PRODUTO_ESTOQUE (ID_PRODUTO, ID_ESTOQUE)
- TB_CAIXA_ESTOQUE (ID_PAGAMENTO, ID_ESTOQUE)
- TB_CLIENTE_PAGAMENTO (ID_CLIENTE, ID_PAGAMENTO)

Criando Chaves Estrangeiras

Relacionamentos são representados por **chaves estrangeiras** – Foreign Key – atributo correspondente à **chave primária** de outra relação, **base para a integridade referencial**.

As tabelas que possuem chaves estrangeiras compõem os relacionamentos das tabelas do nosso banco de dados.

- TB_PRODUTO_COMANDA (ID_PRODUTO, ID_COMANDA, DT_INICIO, DT_FIM)
- TB_PRODUTO_ESTOQUE (ID_PRODUTO, ID_ESTOQUE)
- TB_CAIXA_ESTOQUE (ID_PAGAMENTO, ID_ESTOQUE)
- TB_CLIENTE_PAGAMENTO (ID_CLIENTE, ID_PAGAMENTO)

Perceberam que as **chaves estrangeiras** são os mesmos campos que formam as **chaves primárias** compostas dos relacionamentos.

Formas Normais

O processo de normalização aplica uma série de regras sobre as tabelas de um banco de dados, para verificar se estas estão corretamente projetadas. Embora existam cinco formas normais (ou regras de normalização), na prática usamos um conjunto de três Formas Normais.

Vejamos as três primeiras formas normais do processo de normalização de dados.

- Primeira Forma Normal (1FN)
- Segunda Forma Normal (2FN)
- Terceira Forma Normal (3FN)

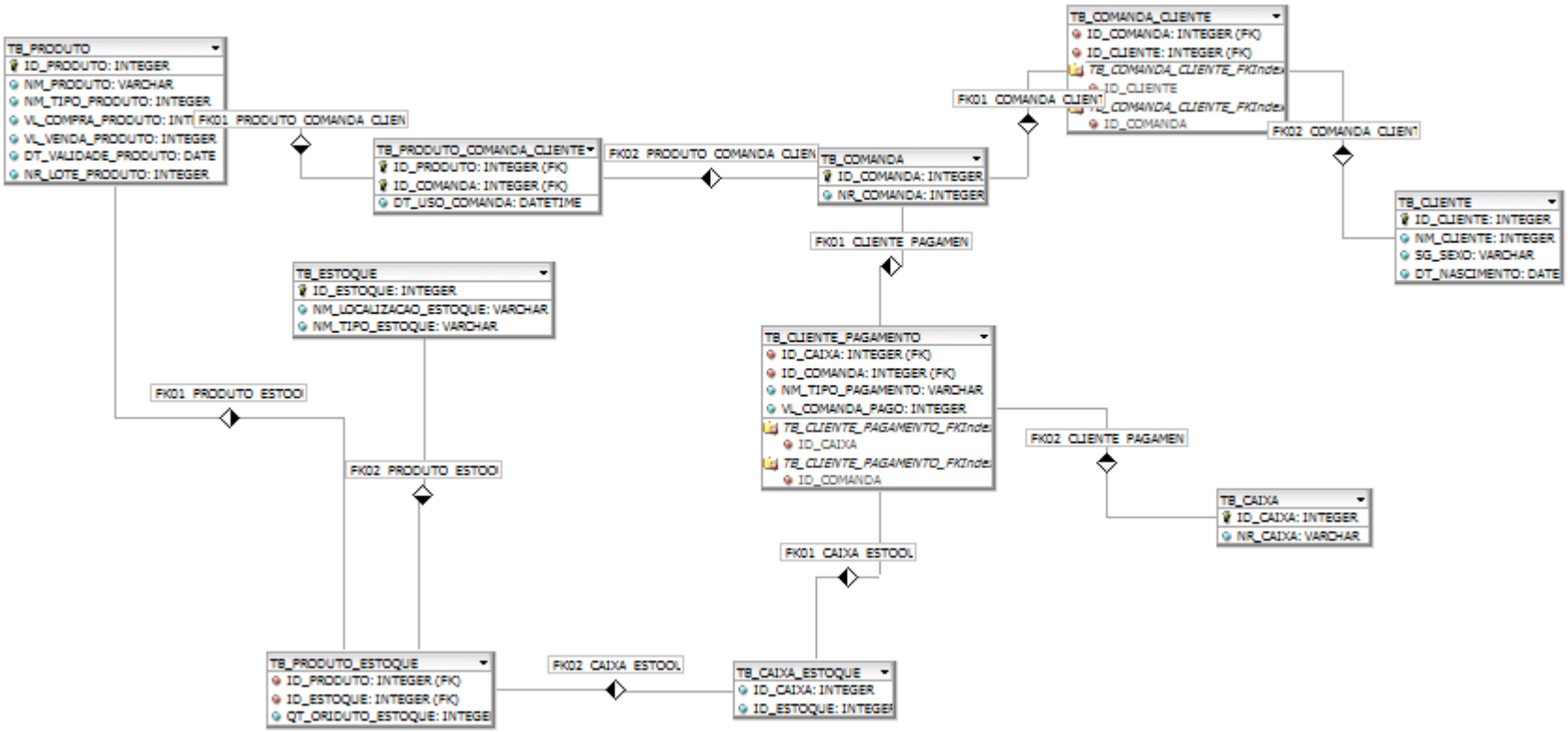
1ª Forma Normal (1FN)

Toda relação deve ter uma chave primária e deve-se garantir que todo atributo seja atômico.

Atributos compostos devem ser separados, por exemplo, um atributo Endereço deve ser subdividido em seus componentes: Logradouro, Número, Complemento, Bairro, Cidade, Estado e CEP.

Atributos multivalorados devem ser discriminados separadamente ou separados em uma outra relação, por exemplo, um atributo multivalorado Telefones poderia ser separado em Telefone Residencial, Telefone Comercial e Telefone Celular ou, ainda, ser convertido em outra relação que pudesse representar um número indeterminado de telefones.

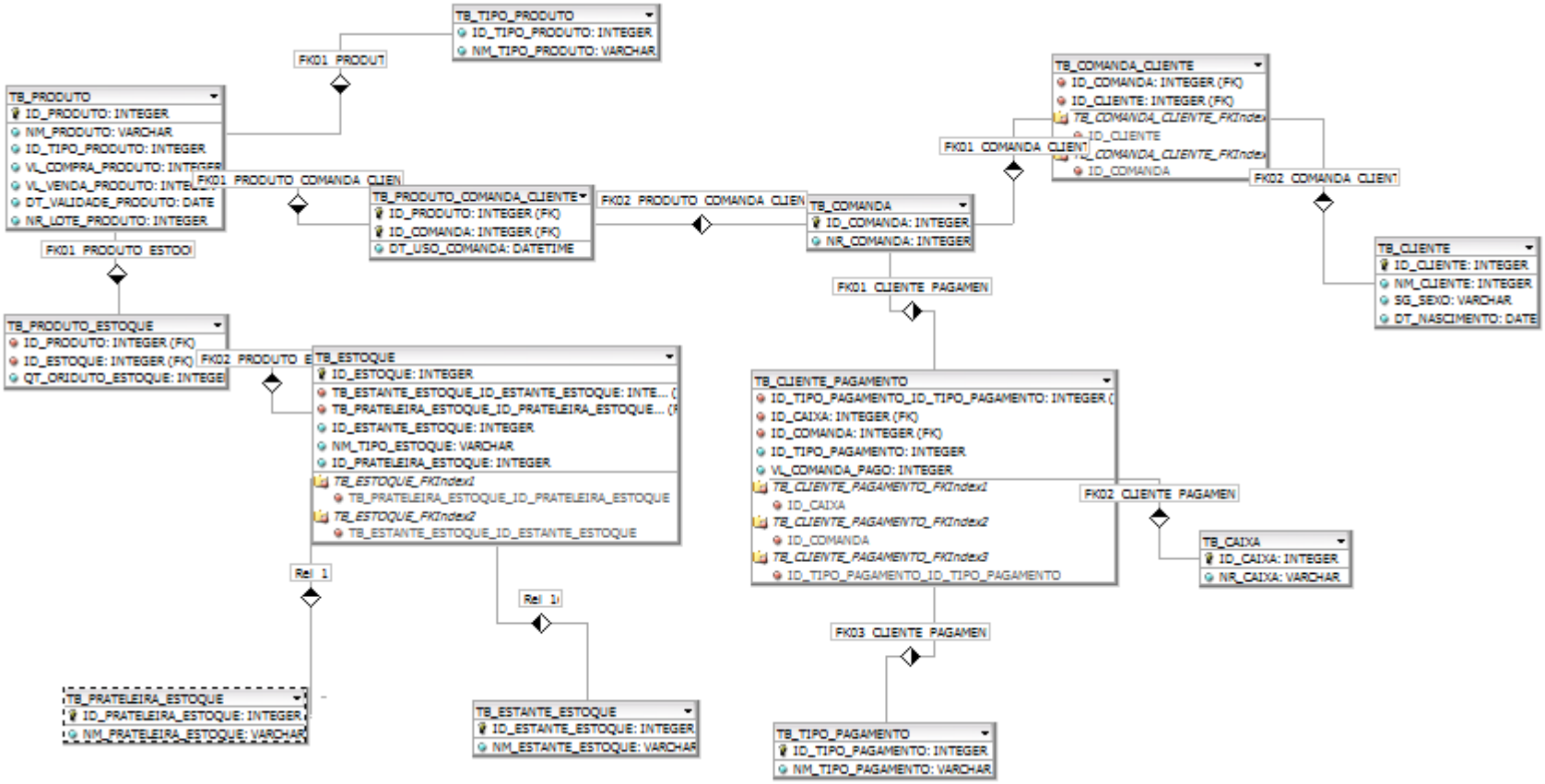
Para exemplificar vou pegar um dos casos de relacionamento, quando criamos a tabela TB_PRODUTO_COMANDA, que contém o Identificador da Comanda e o Identificador do Produto estamos transformando um relacionamento de n:n entre o produto e a comanda em um relacionamento 1:n, pois podem existir vários produtos, mas na tabela TB_PRODUTO_COMANDA vai existir apenas um ID_PRODUTO e campo quantidade para fazer os cálculos na hora da compra e o mesmo serve para a Comanda, onde podem existir várias comandas, mas apenas um identificador de comanda poderá estar atrelado a tabela TB_PRODUTO_COMANDA.


TB_PRODUTO
ID_PRODUTO: INTEGER
NM_PRODUTO: VARCHAR
NM_TIPO_PRODUTO: INTEGER
VL_COMPRA_PRODUTO: INTE
VL_VENDA_PRODUTO: INTEGER
DT_VALIDADE_PRODUTO: DATE
NR_LOTE_PRODUTO: INTEGER
FK01_PRODUTO_COMANDA_CLIEN
TB_PRODUTO_COMANDA_CLIENTE
ID_PRODUTO: INTEGER (FK)
ID_COMANDA: INTEGER (FK)
DT_USO_COMANDA: DATETIME
FK02_PRODUTO_COMANDA_CLIEN
TB_COMANDA
ID_COMANDA: INTEGER
NR_COMANDA: INTEGER
TB_COMANDA_CLIENTE
ID_COMANDA: INTEGER (FK)
ID_CLIENTE: INTEGER (FK)
TB_COMANDA_CLIENTE_FKIndex
ID_CLIENTE
ID_COMANDA
FK01_COMANDA_CLIENT
FK02_COMANDA_CLIENT
TB_CLIENTE
ID_CLIENTE: INTEGER
NM_CLIENTE: INTEGER
SG_SEXO: VARCHAR
DT_NASCIMENTO: DATE
FK01_CLIENTE_PAGAMEN
TB_ESTOQUE
ID_ESTOQUE: INTEGER
NM_LOCALIZACAO_ESTOQUE: VARCHAR
NM_TIPO_ESTOQUE: VARCHAR
TB_CLIENTE_PAGAMENTO
ID_CAIXA: INTEGER (FK)
ID_COMANDA: INTEGER (FK)
NM_TIPO_PAGAMENTO: VARCHAR
VL_COMANDA_PAGO: INTEGER
TB_CLIENTE_PAGAMENTO_FKInde
ID_CAIXA
TB_CLIENTE_PAGAMENTO_FKInde
ID_COMANDA
FK02_CLIENTE_PAGAMEN
TB_CAIXA
ID_CAIXA: INTEGER
NR_CAIXA: VARCHAR
FK01_PRODUTO_ESTOQ
FK02_PRODUTO_ESTOQ
FK01_CAIXA_ESTOQU
TB_PRODUTO_ESTOQUE
ID_PRODUTO: INTEGER (FK)
ID_ESTOQUE: INTEGER (FK)
QT_ORIDUTO_ESTOQUE: INTEGE
FK02_CAIXA_ESTOQU
TB_CAIXA_ESTOQUE
ID_CAIXA: INTEGER
ID_ESTOQUE: INTEGE

2ª Forma Normal (2FN)

Toda relação deve estar na 1FN e devem-se eliminar dependências funcionais parciais, ou seja, todo atributo não chave deve ser totalmente dependente da chave primária.

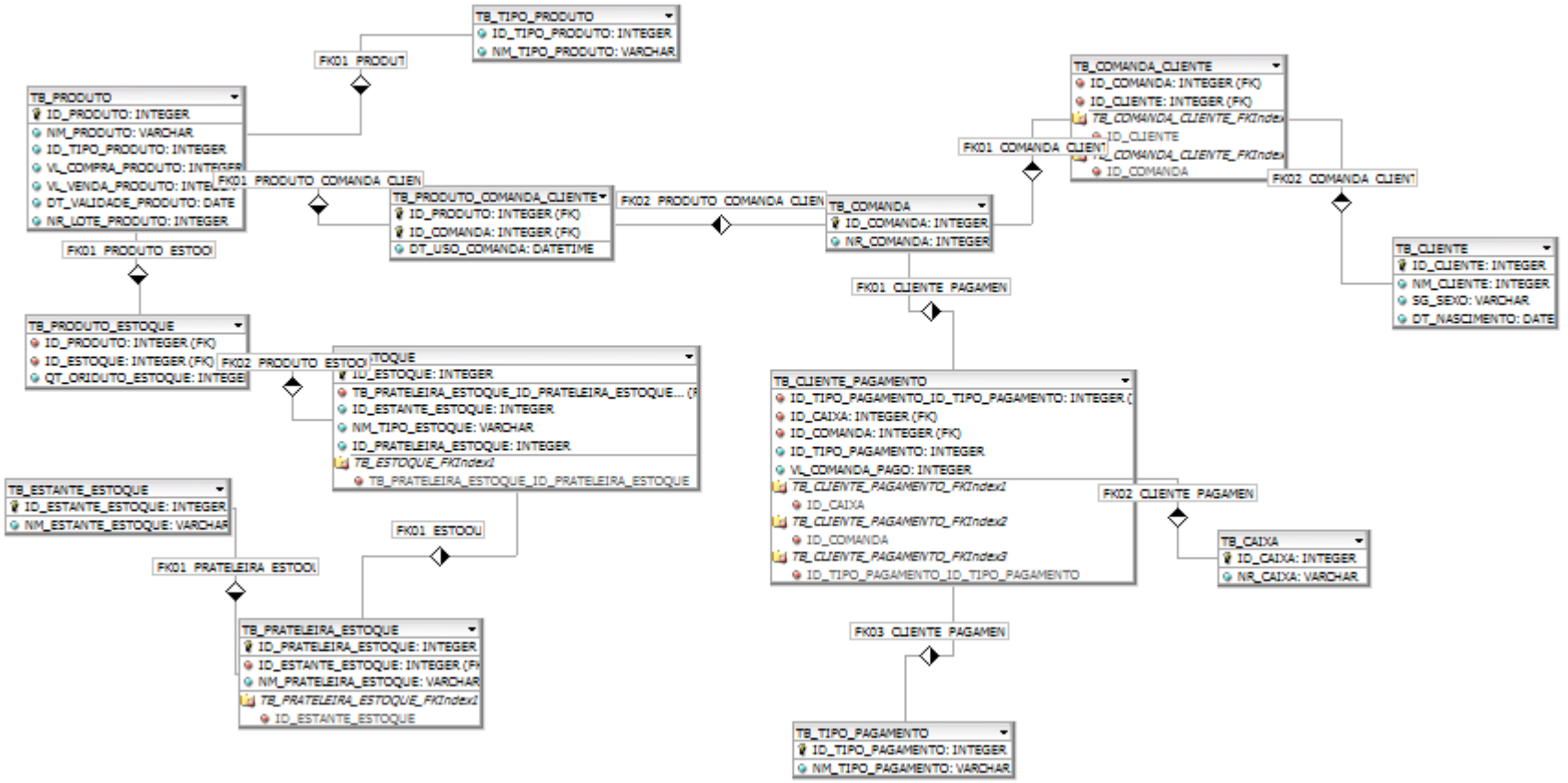
Essa regra é bem interessante, porque eliminamos informações duplicados e conseguimos conservar a integridade das informações, por exemplo, na tabela TB_PRODUTO colocamos o nome do tipo do produto.


TB_TIPO_PRODUTO
ID_TIPO_PRODUTO: INTEGER
NM_TIPO_PRODUTO: VARCHAR
FK01_PRODUT
TB_PRODUTO
ID_PRODUTO: INTEGER
NM_PRODUTO: VARCHAR
ID_TIPO_PRODUTO: INTEGER
VL_COMPRA_PRODUTO: INTEGER
VL_VENDA_PRODUTO: INTE
DT_VALIDADE_PRODUTO: DATE
NR_LOTE_PRODUTO: INTEGER
FK01_PRODUTO_COMANDA_CLIEN
TB_PRODUTO_COMANDA_CLIENTE
ID_PRODUTO: INTEGER (FK)
ID_COMANDA: INTEGER (FK)
DT_USO_COMANDA: DATETIME
FK02_PRODUTO_COMANDA_CLIEN
TB_COMANDA
ID_COMANDA: INTEGER
NR_COMANDA: INTEGER
TB_COMANDA_CLIENTE
ID_COMANDA: INTEGER (FK)
ID_CLIENTE: INTEGER (FK)
TB_COMANDA_CLIENTE_FKIndex
ID_CLIENTE
TB_COMANDA_CLIENTE_FKIndex
ID_COMANDA
FK01_COMANDA_CLIENT
FK02_COMANDA_CLIENT
TB_CLIENTE
ID_CLIENTE: INTEGER
NM_CLIENTE: INTEGER
SG_SEXO: VARCHAR
DT_NASCIMENTO: DATE
FK01_PRODUTO_ESTOQ
TB_PRODUTO_ESTOQUE
ID_PRODUTO: INTEGER (FK)
ID_ESTOQUE: INTEGER (FK)
QT_ORIDUTO_ESTOQUE: INTEGE
FK02_PRODUTO_E
TB_ESTOQUE
ID_ESTOQUE: INTEGER
TB_ESTANTE_ESTOQUE_ID_ESTANTE_ESTOQUE: INTE...
TB_PRATELEIRA_ESTOQUE_ID_PRATELEIRA_ESTOQUE...
ID_ESTANTE_ESTOQUE: INTEGER
NM_TIPO_ESTOQUE: VARCHAR
ID_PRATELEIRA_ESTOQUE: INTEGER
TB_ESTOQUE_FKIndex1
TB_PRATELEIRA_ESTOQUE_ID_PRATELEIRA_ESTOQUE
TB_ESTOQUE_FKIndex2
TB_ESTANTE_ESTOQUE_ID_ESTANTE_ESTOQUE
Rel_1
Rel_1
FK01_CLIENTE_PAGAMEN
TB_CLIENTE_PAGAMENTO
ID_TIPO_PAGAMENTO_ID_TIPO_PAGAMENTO: INTEGER
ID_CAIXA: INTEGER (FK)
ID_COMANDA: INTEGER (FK)
ID_TIPO_PAGAMENTO: INTEGER
VL_COMANDA_PAGO: INTEGER
TB_CLIENTE_PAGAMENTO_FKIndex1
ID_CAIXA
TB_CLIENTE_PAGAMENTO_FKIndex2
ID_COMANDA
TB_CLIENTE_PAGAMENTO_FKIndex3
ID_TIPO_PAGAMENTO_ID_TIPO_PAGAMENTO
FK02_CLIENTE_PAGAMEN
TB_CAIXA
ID_CAIXA: INTEGER
NR_CAIXA: VARCHAR
FK03_CLIENTE_PAGAMEN
TB_PRATELEIRA_ESTOQUE
ID_PRATELEIRA_ESTOQUE: INTEGER
NM_PRATELEIRA_ESTOQUE: VARCHAR
TB_ESTANTE_ESTOQUE
ID_ESTANTE_ESTOQUE: INTEGER
NM_ESTANTE_ESTOQUE: VARCHAR
TB_TIPO_PAGAMENTO
ID_TIPO_PAGAMENTO: INTEGER
NM_TIPO_PAGAMENTO: VARCHAR

3ª Forma Normal (3FN)

Toda relação deve estar na 2FN e devem-se eliminar dependências funcionais transitivas, ou seja, todo atributo não chave deve ser mutuamente independente.

Esta forma normal também ajuda a diminuir redundâncias e aumentar a independência das relações.


TB_TIPO_PRODUTO
ID_TIPO_PRODUTO: INTEGER
NM_TIPO_PRODUTO: VARCHAR
FK01_PRODUT
TB_PRODUTO
ID_PRODUTO: INTEGER
NM_PRODUTO: VARCHAR
ID_TIPO_PRODUTO: INTEGER
VL_COMPRA_PRODUTO: INTEGER
VL_VENDA_PRODUTO: INTEGER
DT_VALIDADE_PRODUTO: DATE
NR_LOTE_PRODUTO: INTEGER
FK01_PRODUTO_COMANDA_CLIEN
TB_PRODUTO_COMANDA_CLIENTE
ID_PRODUTO: INTEGER (FK)
ID_COMANDA: INTEGER (FK)
DT_USO_COMANDA: DATETIME
FK02_PRODUTO_COMANDA_CLIEN
TB_COMANDA
ID_COMANDA: INTEGER
NR_COMANDA: INTEGER
TB_COMANDA_CLIENTE
ID_COMANDA: INTEGER (FK)
ID_CLIENTE: INTEGER (FK)
TB_COMANDA_CLIENTE_FKIndex
ID_CLIENTE
TB_COMANDA_CLIENTE_FKIndex
ID_COMANDA
FK01_COMANDA_CLIENT
FK02_COMANDA_CLIENT
TB_CLIENTE
ID_CLIENTE: INTEGER
NM_CLIENTE: INTEGER
SG_SEXO: VARCHAR
DT_NASCIMENTO: DATE
FK01_PRODUTO_ESTOQ
TB_PRODUTO_ESTOQUE
ID_PRODUTO: INTEGER (FK)
ID_ESTOQUE: INTEGER (FK)
QT_ORIDUTO_ESTOQUE: INTEGE
FK02_PRODUTO_ESTOQ
TOQUE
ID_ESTOQUE: INTEGER
TB_PRATELEIRA_ESTOQUE_ID_PRATELEIRA_ESTOQUE...
ID_ESTANTE_ESTOQUE: INTEGER
NM_TIPO_ESTOQUE: VARCHAR
ID_PRATELEIRA_ESTOQUE: INTEGER
TB_ESTOQUE_FKIndex1
TB_PRATELEIRA_ESTOQUE_ID_PRATELEIRA_ESTOQUE
FK01_CLIENTE_PAGAMEN
TB_CLIENTE_PAGAMENTO
ID_TIPO_PAGAMENTO_ID_TIPO_PAGAMENTO: INTEGER
ID_CAIXA: INTEGER (FK)
ID_COMANDA: INTEGER (FK)
ID_TIPO_PAGAMENTO: INTEGER
VL_COMANDA_PAGO: INTEGER
TB_CLIENTE_PAGAMENTO_FKIndex1
ID_CAIXA
TB_CLIENTE_PAGAMENTO_FKIndex2
ID_COMANDA
TB_CLIENTE_PAGAMENTO_FKIndex3
ID_TIPO_PAGAMENTO_ID_TIPO_PAGAMENTO
FK02_CLIENTE_PAGAMEN
TB_CAIXA
ID_CAIXA: INTEGER
NR_CAIXA: VARCHAR
TB_ESTANTE_ESTOQUE
ID_ESTANTE_ESTOQUE: INTEGER
NM_ESTANTE_ESTOQUE: VARCHAR
FK01_ESTOQU
FK01_PRATELEIRA_ESTOQ
TB_PRATELEIRA_ESTOQUE
ID_PRATELEIRA_ESTOQUE: INTEGER
ID_ESTANTE_ESTOQUE: INTEGER (FK
NM_PRATELEIRA_ESTOQUE: VARCHAR
TB_PRATELEIRA_ESTOQUE_FKIndex1
ID_ESTANTE_ESTOQUE
FK03_CLIENTE_PAGAMEN
TB_TIPO_PAGAMENTO
ID_TIPO_PAGAMENTO: INTEGER
NM_TIPO_PAGAMENTO: VARCHAR

Levantamento de Requisitos

Como saber quando criar uma tabela ou não. Antes do processo de **Modelagem de Dados** existe um outro processo que se chama **Levantamento de Requisitos.**
No levantamento de requisitos o analista de sistemas deve entrar em contato com o solicitante e verificar todas as solicitações. O processo deve mapear:

- **Coisas Tangíveis**: elementos que têm existência concreta, que ocupam lugar no espaço. Ex: Meio de Transporte (avião, carro, apólices de seguros, etc);

- **Funções**: percepção dos objetos através da função por eles exercida (papel, atribuição, classificação, capacitação, etc). Ex: Organização (órgãos funcionais – venda, suporte, despacho de mercadorias, etc), especialistas (médicos, engenheiros, etc), cliente (pessoa atendida), atendente (pessoa que atende), etc;

- **Eventos ou Ocorrências**: alguns objetos só conseguem ser individualizados ou percebidos enquanto uma certa ação se desenrola (identifica-se características que tornam determinado fato materializável). Ex: voo comercial, acidente de trânsito, jogo de futebol, etc.

- **Interações**: resultantes das associações entre objetos em função de um processo executado – cada objeto participante da interação preserva suas características não sendo impactados pela materialização da interação. Ex: compra de um imóvel, adoção de uma criança, venda de um produto;

- **Especificações**: são elementos que definem características de outros objetos. Ex: modelos de carro (cor, dimensões, etc), espécies animais (mamíferos, carnívoros, etc.).

Este mapeamento, geralmente, não é feito pelo desenvolvedor de softwares, na maior parte dos casos ele é feito por um **Analista de Sistemas (Funcional ou Requisitos)**, que passa as especificações do sistema para que os desenvolvedores.