**Normalização**:

O assunto normalização foi explicado em detalhes na aula 2.10, mas para facilitar adiciono esse material complementar para ajudar na fixação.

Normalização é o processo de organizar os dados em um banco de dados. Isso inclui regras para criação de tabelas e seus relacionamentos, a fim de proteger os dados e tornar o banco de dados mais flexível, eliminando a redundância e a dependência inconsistente.

Existem 6 regras para normalização mais difundidas, mas as principais são 3. Nós a chamamos de formas normais (FN).

1°Forma Normal (1FN)  
2° Forma Normal (2FN)  
3° Forma Normal (3FN)

4° Forma Normal (4FN)

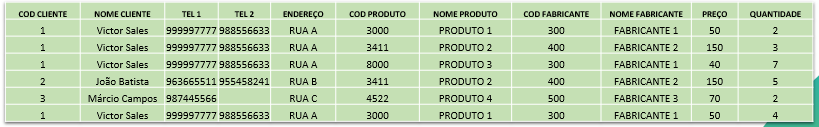
5° Forma Normal (5FN)

6° Forma Normal (6FN)

A partir da 3° forma normal diz-se que o banco de dados já se encontra normalizado. As formas normais 4FN, 5FN e 6FN são usadas para refinar ainda mais o banco. No entanto a maioria dos projetos decidem parar na 3FN pois as outras formas, dependendo da situação, podem exigir um mais processamento.

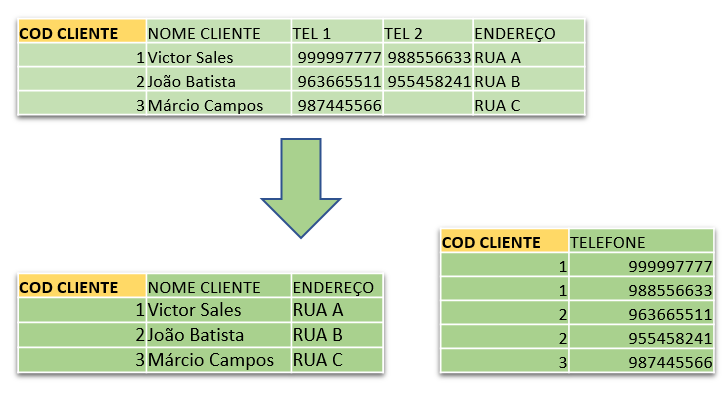
Vale ressaltar que as formas normais são acumulativas, portanto para dizermos que um banco está na 2° forma normal, por exemplo, a primeira deve ter sido também aplicada, assim como para a 3°, a 1° e a 2° devem ter sido aplicadas.

Vamos utilizar a tabela abaixo para exemplificar o processo de normalizar tabelas no ambiente relacional até a 3° forma normal.



* 1FN: Atomicidade: Todos os atributos de uma tabela devem ser atômicos, ou seja, a tabela não pode conter grupos ou atributos repetidos com mais de um valor. Em resumo, a primeira forma normal visa eliminar atributos multivalorados, “quebrando-os” em uma segunda tabela e relacionando ambas as tabelas através das chaves.

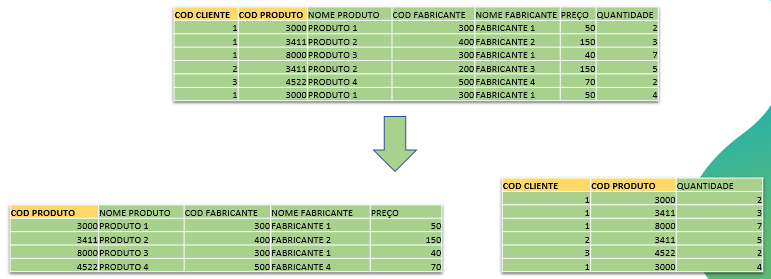
Na imagem abaixo, portanto, tínhamos o atributo multivalorado Telefone (Telefone 1 e 2) e caso no futuro precisássemos de um terceiro ou quarto telefone teríamos que alterar a estrutura da tabela, o que pode ser bem problemático. Dessa maneira, ao aplicarmos a primeira forma normal criamos a tabela telefone e levamos a chave CodCliente como chave estrangeira, e agora um cliente pode ter quantos telefones precisar. Após aplicarmos essa alteração dizemos que esta tabela está na primeira forma normal.



* 2FN: Para aplicar a 2FN, é necessário aplicar a 1FN. Todos os atributos não chaves da tabela devem depender unicamente da chave primária (não podendo depender apenas de parte dela).

Então no exemplo abaixo, para garantir unicidade da primeira tabela precisamos criar a PK composta (com mais de uma coluna) Cod\_Cliente e Cod\_Produto. Isso porque se apenas Cod\_Cliente fosse PK teríamos erro ao tentar adicionar outra compra para esse mesmo cliente. O mesmo se dá se escolhêssemos apenas Cod\_Produto como PK, pois para a próxima compra deste mesmo produto, teríamos erro.

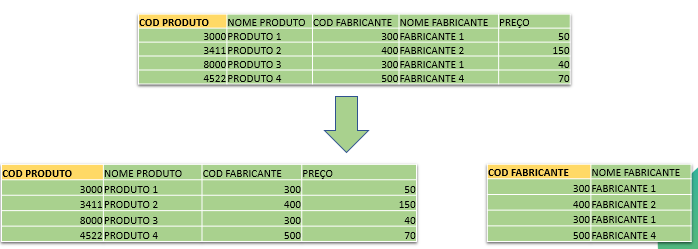
Dessa maneira, seguindo a 2° forma normal, os atributos que pertencem apenas a Produto e não a Cliente, ou seja, se ligam a apenas parte da chave, não podem estar na mesma tabela. Por isso na segunda imagem, “quebramos” em duas tabelas, a de Produto com seus atributos e outra com o que pertence às duas chaves “Cod\_Cliente e Cod\_Produto”, como quantidade. Se Cliente tivesse atributos somente dele, teríamos que criar uma terceira tabela para Cliente com seus atributos.



Após aplicar essa regra dizemos que o banco está na Segunda Forma Normal.

* 3FN: Para fazer a 3FN, é necessário aplicar a 2FN e 1FN: os atributos não chave de uma tabela devem ser mutuamente independentes e dependentes unicamente e exclusivamente da chave primária.

No exemplo abaixo a chave é apenas Cod\_Produto. E Nome\_Fabricante se liga a Cod\_Fabricante, que não é chave. Portanto essa primeira tabela não está seguindo a terceira forma normal. Para aplicar a terceira forma normal basta “quebrar” a primeira tabela em duas, sendo que Cod\_Fabricante se torna chave da segunda e leva consigo os seus atributos.



Feita essa alteração podemos dizer que o banco está na 3° Forma normal ou 3FN.