Integridade de Dados

Manutenção e garantia da consistência e precisão dos dados, sendo um aspecto crítico no design, implementação e uso de sistemas de armazenamento de dados.

A integridade é atingida por meio da aplicação de *Restrições de Integridade*

Restrições de Integridade

- Integridade Referencial
- Integridade de Domínio
- Integridade de Vazio
- Integridade de Chave
- Integridade Definida pelo Usuário

Integridade de Domínio

Valores inseridos em uma coluna devem sempre obedecer à definição dos valores que são permitidos para essa coluna - os valores do **domínio**.

Domínio tem haver com os tipos de dados que você armazena dentro de uma tabela

Ex.: em uma coluna que armazena preços de mercadorias, os valores admitidos(valores que podemos armazenar nessa coluna de preço) são do domínio numérico - ou seja, apenas número “serão aceitos nessa coluna”)

Integridade de Domínio - Fatores

- Tipo de Dado do campo
- Representação interna do tipo de dado
- Presença ou não do dado
- Intervalos de valores no domínio
- Conjuntos de valores discretos

Integridade de Domínio - Exemplo

Atributo Preço do Produto: Valor Monetário

• Valor permitido:

- 25,33

• Valores não permitidos:

- 25 Reais e 33 centavos (Escrever por extenso)
- -32,33 (valores negativos não pode ser preço de produto)

Integridade Referencial

Uma restrição de Integridade Referencial assegura que valores de uma coluna em uma tabela são válidos baseados nos valores em uma outra tabela relacionada.

Ex: Um coisa de chave primária em uma tabela está ligada com uma coluna de chave estrangeira em outra tabela

Ex.: Se um produto de ID 523 foi cadastrado em uma tabela de Vendas, então um produto com ID 523 deve existir na tabela de Produtos relacionada.

Integridade Referencial - Exemplo

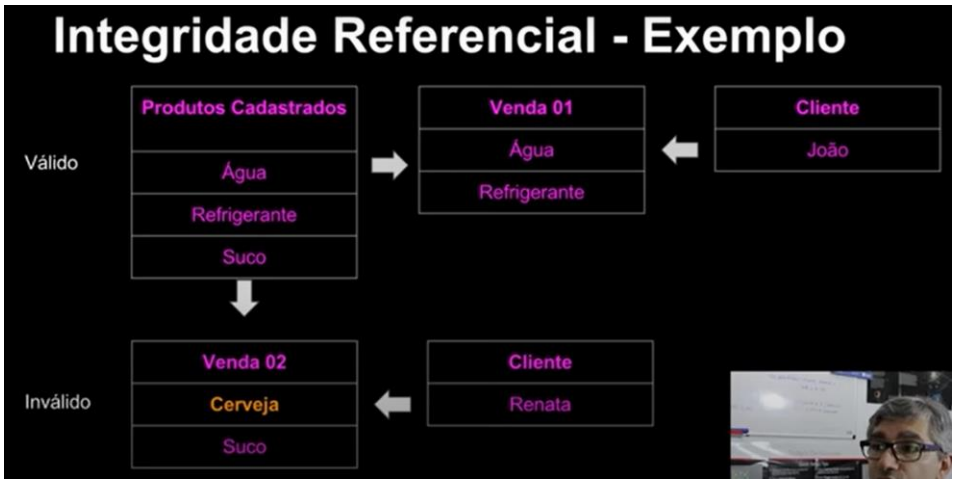
Atributo Nome_Produto: Caracteres

Valores permitidos (produtos cadastrados):

- Água
- Refrigerante
- Suco

Valores não permitidos para venda (não existentes na tabela de produtos):

- Cerveja



Integridade Referencial - Atualização e Exclusão

Se um registro for excluído em uma tabela, então os registros relacionados em outras tabelas que o referenciam talvez precisem ser excluídos.

Caso contrário ocorrerá erro.

O mesmo se dá com atualização de registros.

Integridade de Vazio

Este tipo de integridade informa se a coluna é obrigatória ou opcional - ou seja, se é possível não inserir um valor na coluna.

Uma coluna de chave primária, por exemplo,

* sempre deve ter dados inseridos, e nunca pode estar vazia, para nenhum registro

Valores Nulos (NULL)

Um valor NULL significa que não existem dados.

É diferente de zero, espaço, string vazia ou tabulação.

Os nulos podem ser problemáticos, pois indicam:

- O valor da coluna não é apropriado;
- O valor não foi inserido;
- O valor é desconhecido.

Exemplos de Valores NULL

Suponha uma tabela de cadastro de alunos.

Todo aluno deverá ter um nome cadastrado, de modo que esse campo é obrigatório (atributo não-nulo)

Nem todo aluno possui telefone, portanto esse campo não é obrigatório (atributo nulo)

Integridade de Chave

Os valores inseridos na coluna de chave primária (PK) devem ser sempre únicos, não admitindo-se repetições nesses valores.

Desta forma, as tuplas (registros) serão sempre distintas.

Os valores de chave primária também não podem ser nulos.

Integridade Definida pelo Usuário

Diz respeito a regras de negócio específicas que são definidas pelo usuário do banco de dados.

Por exemplo, pode-se definir que uma coluna somente aceitará um conjunto restrito de valores.

Diagramação

O Diagrama Entidade Relacionamento é a representação gráfica de um MER, que é um modelo conceitual.

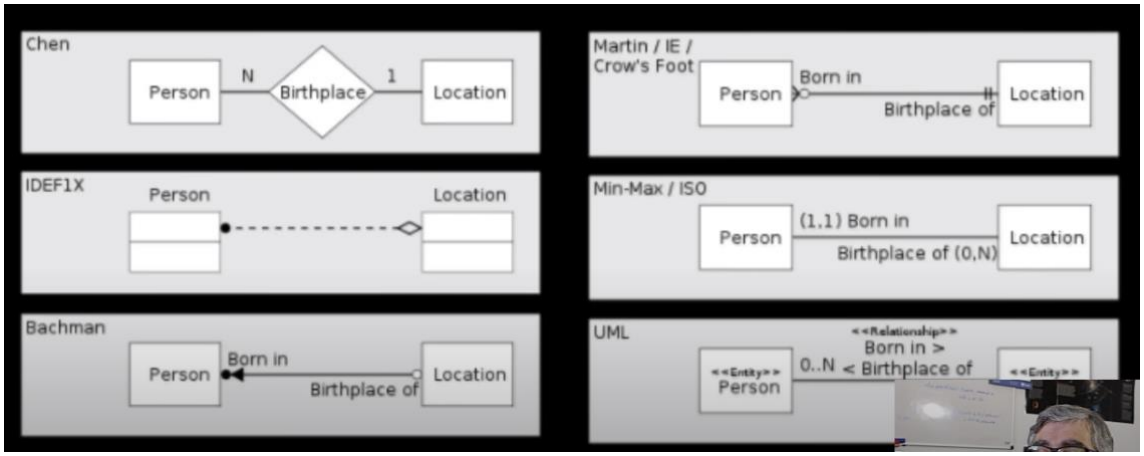
O uso de um diagrama facilita a modelagem e a comunicação entre os membros da equipe de desenvolvimento, permitindo que todos falem a mesma "língua" durante o processo.

A notação original do DER foi proposta por Peter Chen

Existem vários métodos para representar relacionamentos entre entidades.

As notações gráficas mais utilizadas em modelagem de dados são:

- IDEF1X
- Bachman
- Min-Max
- Pé de Galinha (Crow's Foot)
- Martin
- UML
- Peter Chen

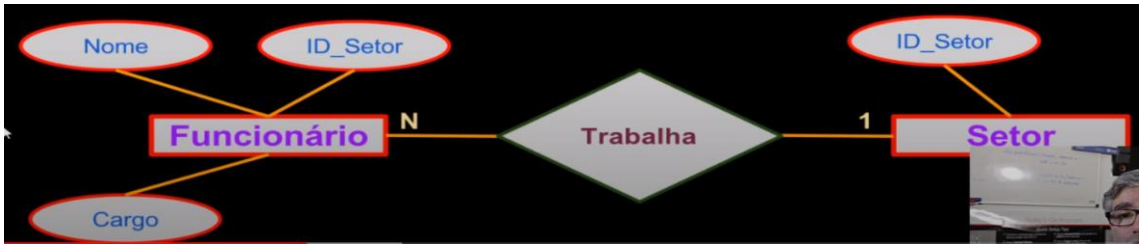


Softwares para Diagramação - Ferramentas CASE

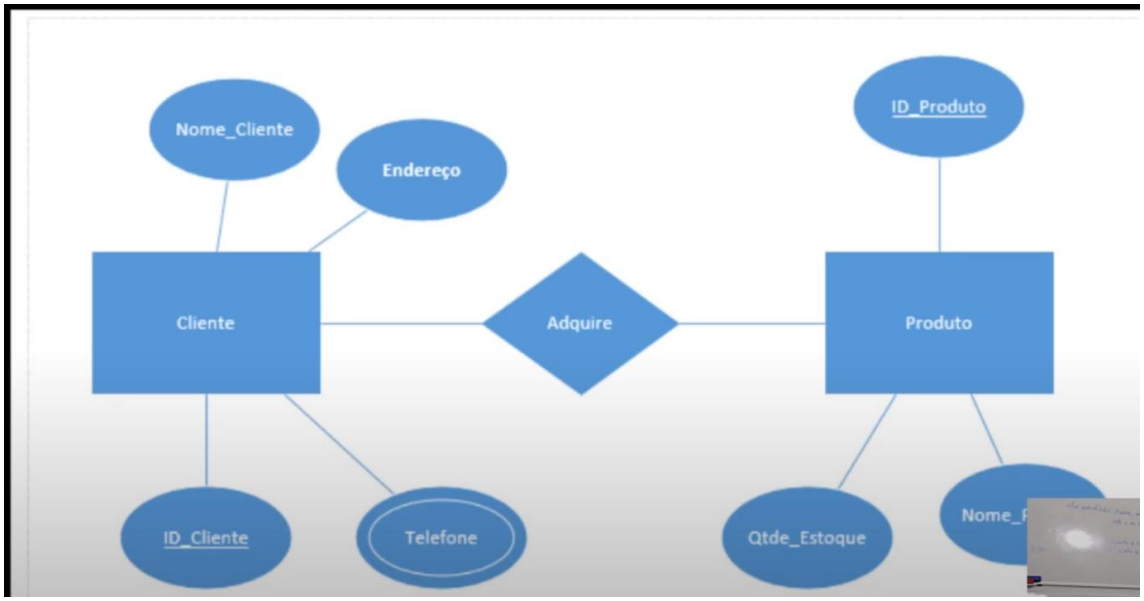
- Astah
- Lucidchart
- erwin Data Modeler
- ERDPluS
- GenMyModel
- Star UML
- Microsoft Visio
- MySQL Workbench
- Visual Paradigm

Diagramação - Peter Chen

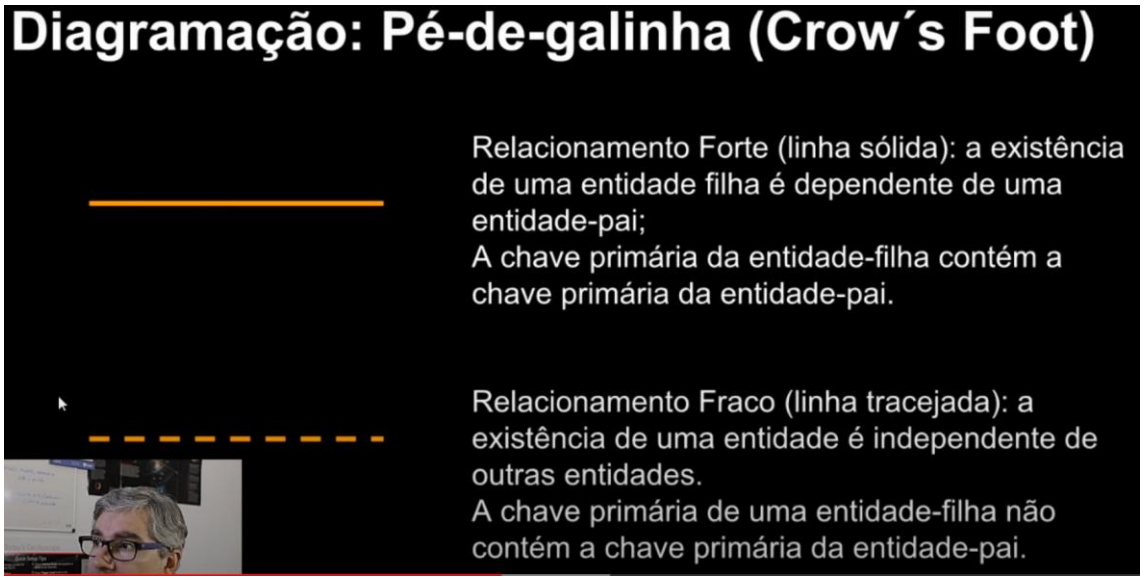
A notação de Peter Chen para DER utiliza retângulos para representar Entidades, e losangos para os Relacionamentos. Atributos são representados por elipses.



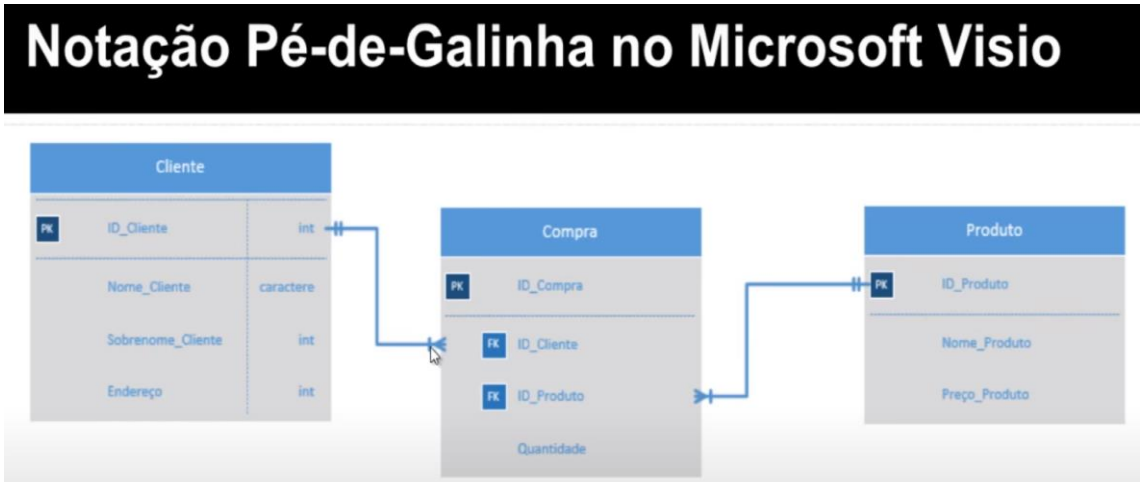
Notação Peter Chen no Microsoft Visio

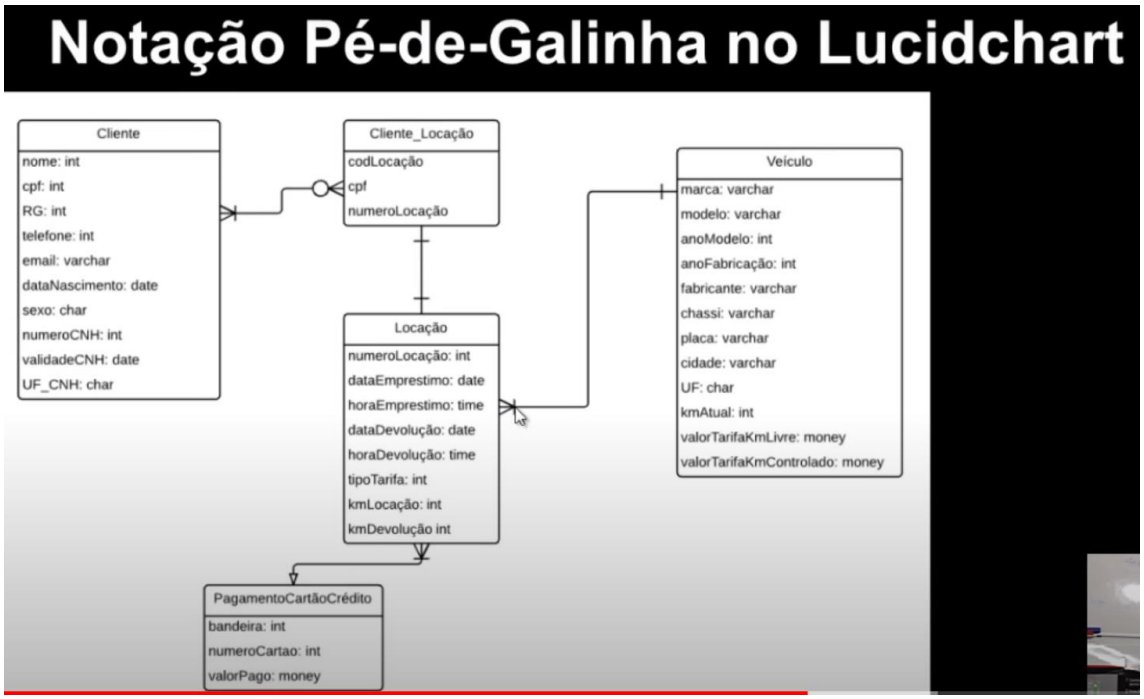


Observação: O atributo telefone que está diferenciado indicando que ele pode ocorrer mais de uma vez no banco de dados, ou seja, um atributo multivalorado



Observação:É quando uma entidade depende da outra para existir. No geral a chave primaria da entidade filha contem a chave primaria da entidade pai, ou seja, elas se repetem



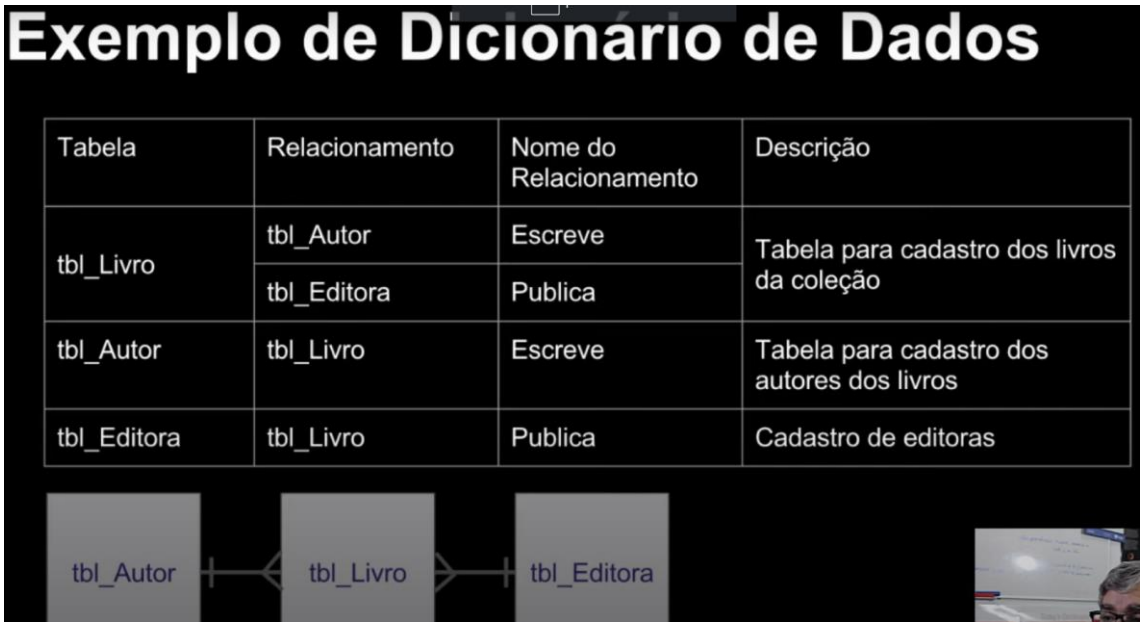


Dicionário de Dados

Um dicionário de dados é um documento usado para armazenar informações sobre o conteúdo, formato e a estrutura de um banco de dados, assim como os relacionamentos entre os seus elementos.

É importante manter um dicionário de dados para limitar erros ao criar a estrutura física do banco de dados no computador.

Também chamado de "Repositório de Metadados”



Observação: Na fotografia acima estou descrevendo as tabelas

Observação: Na fotografia acima estou descrevendo os atributos

Nesse print tem a tabela “tbl_Livro” com todas as colunas/atributos que fazem parte dessa tabela listados

Atributos: ID_Livro, Nome_Livro, ID_Autor, ID_Editora, Data_Pub

Exemplo de Dicionário de Dados

Tabela	Nome da Coluna	Tipo de Dados	Comprimento	Restrições	Valor Padrão	Descrição
tbl_Autor	ID_Autor	Inteiro	4 bytes	PK, NOT NULL	N/D	Número de identificação do autor, gerado automaticamente
	Nome_Autor	Caracteres	40 bytes	NOT NULL	N/D	Nome do autor
	Sobrenome_Autor	Caracteres	40 bytes	NOT NULL	N/D	Sobrenome do autor

Total de Dados em cada Registro: 84 bytes

Observação: Na frase abaixo estou totalizando a quantidade de dados em cada registro, para cada autor que eu cadastrar eu vou ocupar no máximo 84 bytes.

Essa informação tem a finalidade de estimar posteriormente o tamanho do banco de dados, dependendo da quantidade de registro que você vai cadastrar

Exemplo de Dicionário de Dados

Tabela	Nome da Coluna	Tipo de Dados	Comprimento	Restrições	Valor Padrão	Descrição
tbl_Editora	ID_Editora	Inteiro	4 bytes	PK, NOT NULL	N/D	Número de identificação da editora, gerado automaticamente
	Nome_Editora	Caracteres	40 bytes	NOT NULL	N/D	Nome da Editora

Total de Dados em cada Registro: 44 bytes

Exemplo de Dicionário de Dados

Relacionamento	Tabela 1 - FK	Tabela 2 - PK	Descrição
“Escreve”	tbl_Livro	tbl_Autor	Relacionamento que descreve qual autor escreve cada livro
“Publica”	tbl_Livro	tbl_Editora	Relacionamento que mostra qual editora publica cada livro

Observação: No print acima estou descrevendo o relacionamento

Dependência Funcional

Seja E uma entidade, e X e Y dois atributos quaisquer de E. Dizemos que Y é funcionalmente dependente de X se e somente se cada valor de X tiver associado a ele exatamente um valor de Y.

Simbolicamente:

X → Y

• Que lemos como "X determina funcionalmente Y".

“Y depende funcionalmente de X”

Dependência Funcional e quando um campo depende de outro campo

Dependência Funcional

Ex.: O prazo de entrega de um pedido depende do número do pedido considerado

Numero_Pedido → Prazo_Entrega_Pedido

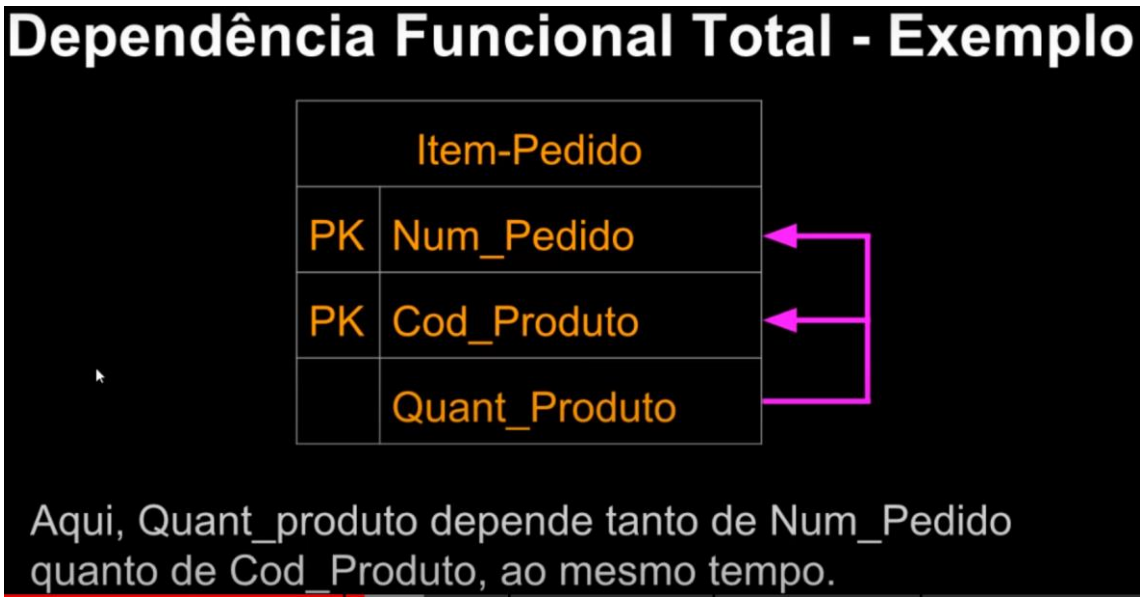
O atributo que determina o valor é chamado de Determinante.

O outro atributo é chamado de Dependente.

Uma chave primária em uma relação determina funcionalmente todos os outros atributos não-chave na linha.

Dependência Funcional Total

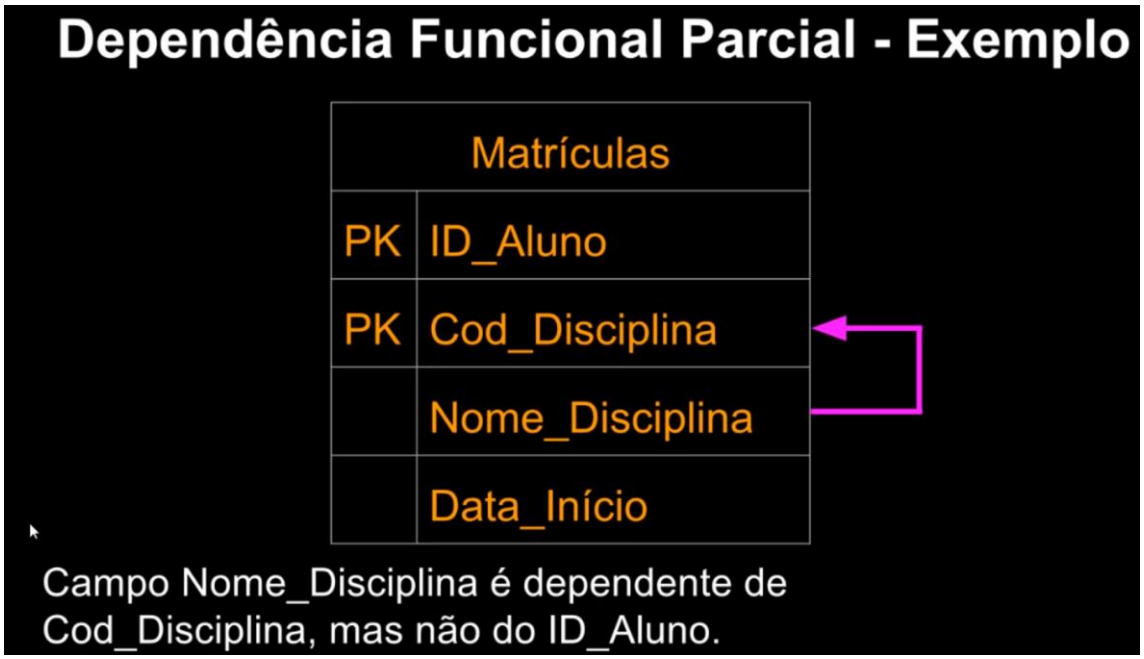
Em uma relação com uma PK composta, um atributo não-chave que dependa dessa PK como um todo, e não somente de parte dela, é dito como possuindo Dependência Funcional Total.



Dependência Funcional Parcial

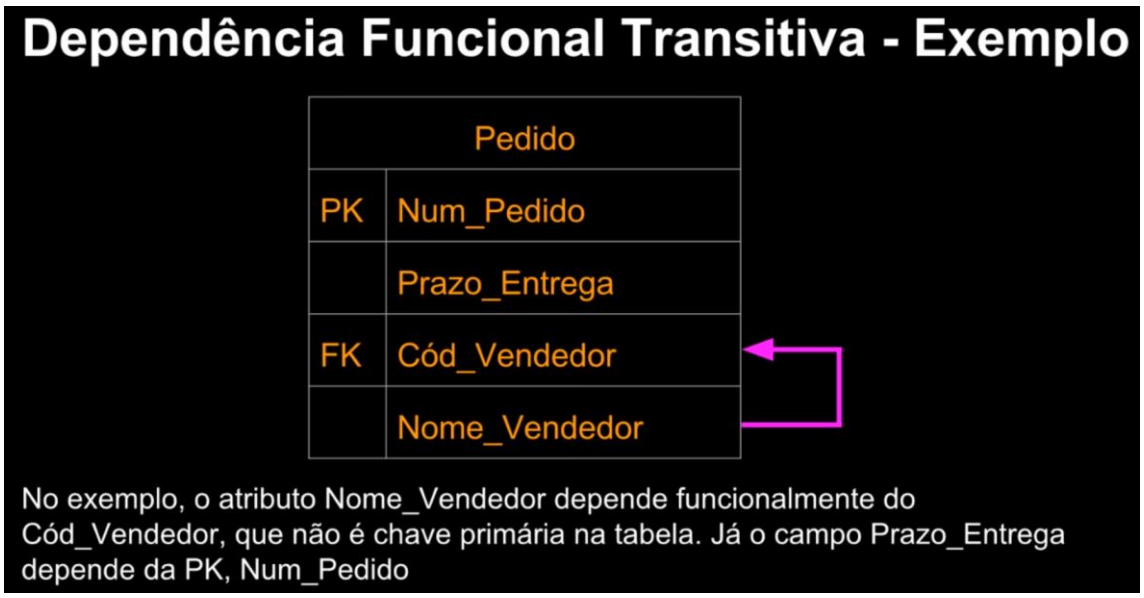
Uma dependência funcional é parcial quando os atributos não-chave não dependem funcionalmente de toda a PK quando esta for composta.

Ou seja, existe uma dependência funcional, mas somente de uma parte da chave primária.



Dependência Funcional Transitiva

Ocorre quando um campo não depende diretamente da chave primária da tabela (nem mesmo parcialmente), mas depende de um outro campo não-chave.



O campo “Nome_Vendedor” não devia está nessa tabela, devia está em outra tabela

Dependência Multivalorada

Ocorre quando, para cada valor de um atributo A, existe um conjunto de valores para outros atributos B e C que estão associados a ele, mas são independentes entre si.

Representamos a dependência multivalorada assim:

A ->> B

Onde B é a coluna que depende de A.

Anomalias de Atualização

Anomalias são problemas que ocorrem em bancos de dados mal planejados e não-normalizados, geralmente ocorrendo por excesso de dados armazenados em uma mesma tabela.

São causadas pelas dependências parciais e transitivas.

As anomalias de atualização são classificadas em anomalias de inserção, de exclusão e de modificação.

Anomalia de Inclusão

Anomalia de Inclusão: Não deve ser possível adicionar um dado a não ser que outro dado esteja disponível.

Por exemplo, não deve ser permitido cadastrar um novo livro sem que um autor já esteja cadastrado.

Anomalia de Modificação

Anomalia de Modificação: ao alterar um dado em uma tabela, dados em outras tabelas precisam ser alterados.

Por exemplo, se o código de um autor for modificado, esse código deve ser modificado na tabela de autores e na de livros também, para manter o relacionamento entre livros e seus autores corretos.

Eliminar anomalias

Projetar as/os (estrutura de uma tabela) esquemas de relações (tabelas) no banco de dados de modo que nenhuma anomalia de inserção, exclusão ou modificação esteja presente nas relações.

Para isso, usamos o processo de Normalização

Normalização

Normalização consiste em um processo de análise de uma relação(na tabela) para assegurar que seja bem formada. “Bem formada” significa que ela tenha os campos, atributos coerentes e corretos para o tipo de dado, para o tipo de informação que ela se presta a armazenar

Normalização consiste em decompor relações com anomalias para produzir relações menores e bem-estruturadas.

Decompor significa pegar uma tabela que tem vários campos e extrair os campos que não são relevantes a essa tabela e formar outras tabelas com esses campos, não se trata de jogar dados no lixo, se trata de organizar esses dados no lugar correto que muitas das vezes podem ser em outras tabelas

Ou seja, em uma relação normalizada podemos inserir, excluir ou modificar registros sem criar anomalias.

Normalização

O processo de normalização, proposto por Codd em 1972, aplica a um esquema de relação(tabela) uma série de testes para certificar que ele satisfaça uma **Forma Normal** (FN)

Codd propôs originalmente 3 formas normais:1ª, 2ª e 3ª FNs.

Posteriormente, a 3FN foi revisada e uma definição mais robusta foi proposta por Boyce e Codd, denominada Forma Normal de Boyce-Codd (FNBC)

Objetivos da Normalização

Analisar esquemas de relação (tabelas) com base em suas dependências funcionais e chaves primárias para:

1. Minimizar redundâncias (minimizar dados repetidos usando a normalização)

2. Minimizar anomalias de inserção, exclusão e modificação

As relações são decompostas em esquemas de relação menores que atendem aos testes de forma normal.

Objetivos da Normalização

O ideal é que o projeto do banco de dados relacional alcance a FNBC ou a 3FN para cada tabela.

Não é adequado normalizar apenas até a 1FN ou à 2FN, pois na verdade essas formas normais são usadas para se chegar à 3FN ou FNBC.

Primeira Forma Normal

Definida historicamente para reprovar atributos multivalorados, compostos e suas combinações.

Se sua tabela possuir atributos multivalorados ao aplicar o processo da “Primeira Forma Normal” iremos eliminar esses atributos, ou seja, esses dados irão compor outras tabelas(relações)

O domínio de um atributo deve incluir apenas valores atômicos (indivisíveis), e o valor de qualquer atributo em uma tupla deve ser único valor do domínio desse atributo.

Observação:

Valor atômico e um valor indivisível

Tupla/Registro

Primeira Forma Normal

Uma tabela está na 1ª forma normal quando:

- Somente possui valores atômicos (valores indivisíveis, ou seja, não tem atributos multivalorados)

- Não há grupos de atributos repetidos (há apenas um dado por coluna nas linhas)

- Existe uma chave primária

- Relação não possui atributos multivalorados ou relações aninhadas

Observação: Relação aninhada significa uma tabela dentro de outra tabela. Quando você tem na sua tabela um conjunto de atributos(colunas) que em si formam outra tabela com dados, informações que não precisariam está dentro daquela tabela “Mãe”

Primeira Forma Normal

Uma tabela está na 1ª forma normal se somente houverem valores atômicos no domínio de seus atributos.

Um valor atômico é um valor indivisível.

Como exemplo, um campo de Endereço possui subdomínios Rua, Número e CEP. Esses itens devem ser separados no processo e normalização.

Cada informação deve ser colocada em um campo diferente e eventualmente para uma tabela diferente.

Dados Atômicos

Elementos de dados que representam o nível mais baixo de detalhamento.

Então, campos não-atômicos são aqueles que podem ser subdivididos em mais de um campo, pois eles escondem detalhes, como por exemplo o nome de uma pessoa, que contém o primeiro nome e o sobrenome.

Cod_Cliente e uma chave primaria

Nome_Cliente e um campo simples

Tel_Cliente e um campo multivalorado

Endereço_Cliente e um campo composto

Para normalizar a tabela “tbl_Cliente” até chegar a 1ª FN:

- Iremos montar uma tabela no papel ou software especifico ou usando um programa de planilha ficando a seu critério o local de montar a tabela.
- Vai colocar cada um dos campos que você tem na sua relação(tabela) como uma coluna dessa tabela e vai preencher com dados fictícios, pode ser 4 registros para permitir enxergar o que está acontecendo na tabela

No campo “Tel_Cliente” tem cadastrado alguns clientes com dois telefones. Isso e um problema devido a 1ª forma Normal dizer que os valores devem ser atômicos, mas nesse caso temos valores multivalorados.

Tenho 2 informações diferentes (dois números de telefones) no mesmo campo “Tel_Cliente” da minha tabela “tbl_Cliente”.Isso está errado e não pode acontecer!!!

No campo “Endereço_Cliente” temos problema já que esses endereços são atributos compostos (Tem nome de rua, número de apartamento, número de rua e bairro, ou seja, tem quatro informações)

Tabela Normalizada - 1FN

tbl_Cliente

Cod_Cliente	Nome_Cliente	Rua	Bairro
2532	José	Rua das Giestas, 234	Vila Bela
2536	Marcos	Av. Carlos de Almeida, 459	Vila das Rosas
2453	Ana	Rua Min. Alberto Jorge, 1492	Vila Primavera

tbl_Telefone

Cod_Cliente	Tel_Cliente
2532	99653-2145
2532	2865-3212
2536	2643-5321
2453	4213-6532
2453	97563-5632

Essa tabela acima já se encontra normalizada na 1ª FN, mantivemos os campos “Cod_Cliente” e “Nome_Cliente”. O campo Endereco_cliente foi desmembrado em duas partes “Rua” e “Bairro”

O campo telefone foi colocado em outra tabela “tbl_Telefone”.Tiramos o campo telefone da tabela “tbl_Cliente” devido ser um campo multivalorado e criamos uma tabela especifica só para colocar o telefone.

Na tabela “tbl_Telefone” o “Cod_Cliente” não e chave primaria porque se repete o número do código

Aqui foi realizado a separação dos dados multivalorados da tabela e expandindo os dados compostos

Segunda Forma Normal

• Baseada no conceito de Dependência Funcional Total.

Um esquema de relação(Tabela) R está na 2FN se cada atributo não-chave de R for total e funcionalmente dependente da PK de R. Cada atributo que não for chave tem de ser total e funcionalmente dependentes da chave primaria, si o atributo não for dependente totalmente da chave primaria a tabela não está na 2ªFN e precisa ser normalizada

Para testar a 2FN, testamos as dependências funcionais cujos atributos fazem parte da chave primária.

• Caso a PK tenha um único atributo, esse teste não precisa ser aplicado.

Segunda Forma Normal

Uma tabela está na 2ª FN se:

- Está na 1ªFN. Para a tabela está na 2ªFN ela precisa já está normalizada na 1ªFN, não e possível pular etapas.

- Todos os atributos não-chave são funcionalmente dependentes de todas as partes da chave primária. Si a chave primaria for composta, si tiver duas ou três partes, os atributos tem de depender de todas as partes da chave

- Não existem dependências parciais.

- Caso contrário, deve-se gerar uma nova tabela com os dados.

Um atributo-chave é um atributo que é uma PK ou parte de uma PK composta.

O atributo-não-chave são os demais atributos

Segunda Forma Normal

Deve-se criar uma nova relação para cada chave PK ou combinação de atributos que forem determinantes em uma dependência funcional.

Esse atributo será a PK na nova tabela.

•Mova os atributos não-chave dependentes desta PK para a nova tabela.

Na tabela “tbl_Peça” temos uma chave primaria composto que e composta pelo campo “Cod_Peça” e “Cod_Fornec” juntos

Agora iremos expandir a tabela/relação e vai preencher com dados fictícios para tentar enxergar o que está acontecendo de forma simples e em seguida aplicar a normalização

Regra 2º FN:

Os atributos tem que ser totalmente dependentes da chave primaria inteira, ou seja, não pode ser só dependente de uma parte chave da primaria, tem que ser dependente da chave primaria toda

O campo “Local_Fornecedor” depende do “Cod_Fornec”, a partir do “Cod_Fornec” eu descubro o local do fornecedor.

O “Local_Fornecedor” não depende do “Cod_Peça”, porque eventualmente uma peça pode ser fornecida por mais de um fornecedor, tendo o código da peça em si não vai indicar qual e o local do fornecedor

A “Qtde_Estoque” depende do “Cod_Peça”

O “Tel_Fornecedor” depende do “Cod_Fornec”, mas não depende do “Cod_Peça” , não e a peça que determina o telefone do fornecedor e o código dele

A “Qtde_Caixas” depende do “Cod_Peça” e depende do fornecedor que está fornecendo aquele produto, a “Qtde_Caixas” depende da “Qtde_Estoque”

As colunas “Local_Fornecedor” e “Tel_Fornecedor” que não dependem da peça, ou seja, não dependem da chave primaria composta e colocamos esses 2 atributos em outras tabelas

A coluna “Cod_Fornec” e chave primaria da tabela “tbl_Fornecedor”

Terceira Forma Normal

Baseada no conceito de Dependência Transitiva.

A relação(tabela) não deve ter um atributo não-chave determinado funcionalmente por outro atributo não-chave (ou conjunto).Os atributos na nossa tabela devem depender completa e totalmente apenas da chave primaria, não pode depender de outro atributo que não seja chave primaria ou parte da chave primaria

Não deve haver dependência transitiva de um atributo não-chave sobre a PK.

Deve-se decompor e montar uma nova relação que inclua os atributos não-chave que determinam funcionalmente outros atributos não-chave.

Terceira Forma Normal

Uma tabela está na 3FN se:

- Estiver na 2FN

- Não existirem **dependências transitivas. Dependencia transitiva e uma dependência funcional entre dois ou mais atributos que não sejam chaves**

- Uma tabela está na Terceira Forma Normal se ela estiver na segunda forma normal e se nenhuma coluna não-chave depender de outra coluna não-chave.

Uma dependência transitiva em uma tabela é uma dependência funcional entre dois ou mais atributos não-chave.

Terceira Forma Normal

Para cada atributo (ou grupo) não-chave que for um determinante na relação, crie uma nova tabela.

Esse atributo será a PK na nova relação. Se o atributo A depende do atributo B e nenhum deles e chave a gente cria uma nova relação e esse atributo B passa a ser chave primaria dessa nova relação

Mova então todos os atributos que são dependentes funcionalmente do atributo chave para a nova tabela.

O atributo (PK na nova relação) fica também na tabela original, e servirá como uma chave estrangeira para , associar as duas relações.

Essa tabela acima tem problemas, ou seja, ela não está na 3FN. Tem atributos que não dependem única e exclusivamente da chave primaria que o campo “Nota_Fiscal”

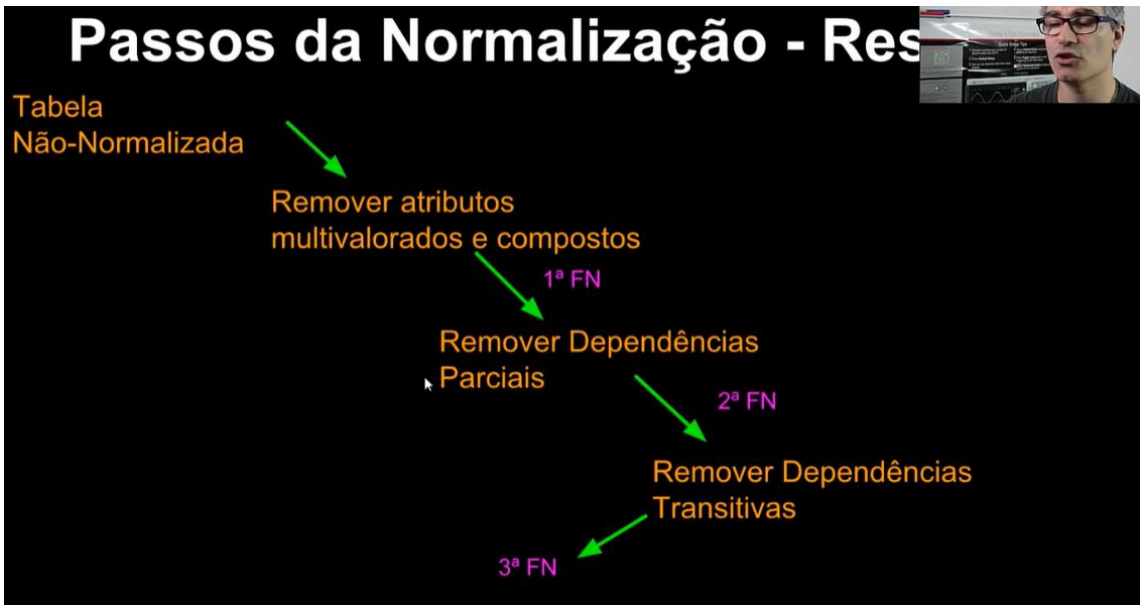
O “Cod_Vendedor” depende da chave primaria “Nota_Fiscal”

O “Nome_Vendedor” não depende da chave primaria “Nota_Fiscal”, depende do “Cod_Vendedor”.Para saber o “Nome_Vendedor” eu preciso saber o “Cod_Vendedor”

Para saber o código do vendedor que fez a venda e só olhar na nota fiscal, porque na nota fiscal vai está registrado o código do vendedor

O “Cod_Produto” e a “Qtde_vendida” depende da chave primaria nota fiscal

A 3FN tem como finalidade eliminar os atributos não chaves que dependam de outros atributos que são “não chaves”



Forma Normal de Boyce-Codd

A definição original da 3FN de Codd não lidava adequadamente com uma relação que:

- Tivesse duas ou mais chaves candidatas.
- Essas chaves candidatas fossem compostas
- Elas tivessem superposição (atributos em comum).

Caso a combinação das condições acima não ocorra em uma tabela, basta aplicar a 3FN.

Uma relação está em FNBC se e somente se os únicos determinantes são chaves candidatas.

Forma Normal de Boyce-Codd

- Cada relação na FNBC também está na 3FN, mas uma relação na 3FN não está necessariamente na FNBC (a maioria está).
- Quando uma tabela possui mais de uma chave candidata, podem ocorrer anomalias.
- Na FNBC as chaves candidatas não possuem dependências parciais por outros atributos
- Uma relação R está na FNBC sempre que uma dependência funcional não-trivial **X** → **A** se mantiver em R, assim X é uma superchave de R.

FNBC - Pontos a considerar

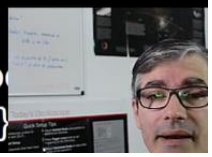
Determinante: "lado esquerdo" de uma DF, como o X em **X → Y** (Y é "dependente"); X *determina funcionalmente* Y

Dependência Funcional Trivial: dependência que não pode deixar de ser satisfeita. Uma DF é trivial se o lado direito da expressão é um subconjunto da lado esquerdo.

A → B é uma DF trivial se B for um subconjunto de A.

Exemplo:

{ID_Func, Nome_Func} → ID_Func é uma DF trivial p
ID_Func é um subconjunto de {ID_Func, Nome_Func}



Forma Normal de Boyce-Codd

Definindo: Seja a relação FORN que contém os atributos F#, F_Nome, P#, Qtde:

FORN { F#, F_Nome, P#, Qtde }

Temos como chaves candidatas as combinações {F#,P#} e {F_Nome,P#}, como mostra a tabela ao lado.

FORN

F#	F_Nome	P#	Qtde
F1	Acme	P1	600
F1	Acme	P2	300
F1	Acme	P3	250
F1	Acme	P4	280
F2	Umbrella	P1	350
...	...		

FNBC - Normalizando

Para normalizar uma tabela até a FNBC devemos decompor a tabela com os passos a seguir:

- Encontrar uma dependência funcional não-trivial X → Y que viole a condição de FNBC. X não deve ser uma superchave.
- Dividir a tabela em duas: Uma com os atributos XY, ou seja, todos os atributos da dependência
- Outra com os atributos X juntamente com os atributos restantes da tabela original.

FNBC - Resolvendo o exemplo

Forn

F#	F_Nome
F1	Acme
F2	Umbrella
...	...

+

Forn_Prod

F#	P#	Qtde
F1	P1	600
F1	P2	300
F1	P3	250
F1	P4	280
F2	P1	350
...	...	...

ou

Forn_Prod

F_Nome	P#	Qtde
Acme	P1	600
Acme	P2	300
Acme	P3	250
Acme	P4	280
Umbrella	P1	350
...		

Pode haver mais de uma decomposição válida na FNBC.

FNBC - Exemplo 2 (não normalizado)

tbl_ADP		
Aluno	**Disciplina**	**Professor**
500	Matemática	Fábio
501	Física	Jorge
501	História	Ana Maria
503	Matemática	Sandra
503	História	Nunes

Restrições:
Cada estudante aprende uma disciplina lecionada por um professor.
Cada professor leciona apenas uma disciplina, e uma disciplina pode ser lecionada por vários professores.

Problemas:

- Se um aluno for excluído da tabela, as informações do orientador somem também.
- Se um novo aluno ou orientador for adicionado, obrigatoriamente devemos adicionar as informações relacionadas, as quais podem não existir.
- Caso as informações de u atualizadas, deveremos at informações sobre o orient

Disciplina é parte de uma chave candidata composta e é determinado por um atributo não-chave da tabela.

FNBC - Normalizado

tbl_AP	
Aluno	Orientador
500	Fábio
501	Jorge
502	Ana Maria
503	Sandra
503	Nunes
504	Nunes

tbl_PD	
Professor	Disciplina
Fábio	Matemática
Jorge	Física
Ana Maria	História
Sandra	Matemática
Nunes	História