

## Abordagem de modelagem – Banco de Dados II:

O conjunto de conceitos usados na construção de um modelo é chamado de modelagem.

- Modelo conceitual: descrição do banco de dados independente de implementação de um SGBD.

- Modelo Lógico: descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD.

- Devido aos diferentes **graus de relacionamento** e **cardinalidades** o modelo Lógico pode ser interpretado de duas maneiras:

- Modelo abstrato da organização, que define suas entidades;

- Modelo abstrato do Banco de dados, que define as tabelas que farão parte do banco de dados.

As técnicas de modelagem orientada a objetos (relacional) que estamos estudando fundamentam-se nos principais conceitos da abordagem E/R: *entidade, relacionamento, atributo, generalização, especialização e entidade associativa* desenvolvida originalmente por Peter Chen.

- **Entidade:** conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações do banco de dados. Em um DER, uma entidade é representada através de um retângulo;

- **Relacionamento:** conjunto de associações entre ocorrências de entidades. Em um DER, uma associação é representada através de um losango;

- Papel de entidade em relacionamento: função que uma instância de entidade cumpre dentro de uma instância do relacionamento;

- **Cardinalidade de relacionamento:** quantas ocorrências de uma entidade podem estar associadas a uma determinada ocorrência através do relacionamento. Cardinalidade mínima e cardinalidade máxima, **1:1, 1:n, n:n**;

- **Grau do relacionamento ou classificação dos relacionamentos:** a cardinalidade mínima ou máxima pode ser utilizada para classificar relacionamentos: **unário, binário, ternário...** 1:1, 1:n, n:n;

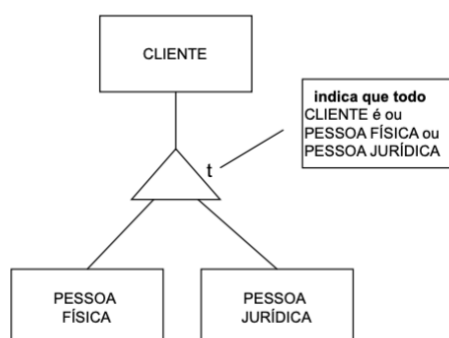
- **ATRIBUTO:** dado que é associado a cada ocorrência de uma entidade ou de um relacionamento. Normalmente os atributos não são representados graficamente, para não sobrecarregar o DER, porém, se necessário representá-

los utilize um círculo vazio. O conjunto de valores que um atributo pode assumir o chamado de **domínio do atributo**.

- **Identificadores de Entidades:** conjunto de um ou mais atributos e relacionamentos cujos valores servem para distinguir uma ocorrência da entidade das demais ocorrências da mesma entidade, esse atributo quando representado é feito através de um círculo preto cheio: chaves primárias e estrangeira.

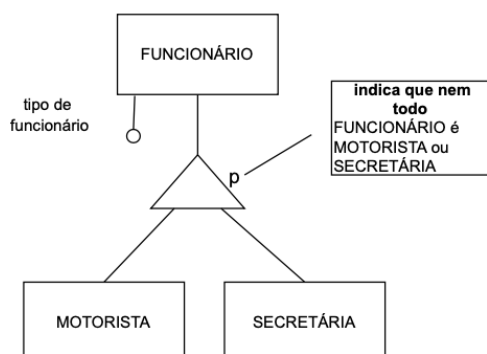
- **Generalização/especialização:** além de relacionamentos e atributos, propriedades podem ser atribuídas a entidades através desse conceito. No DER, o símbolo para representar generalização/especialização é um triângulo isósceles. A generalização/especialização está associada à ideia de herança de propriedades e pode ser classificada em dois tipos:

- **Total:** para cada ocorrência da entidade genérica existe sempre uma ocorrência de uma das entidades especializadas.



Generalização/especialização total. Fonte: Heuser (2010, páginas: 24-56)

- **Parcial:** nem toda ocorrência da entidade genérica possui uma ocorrência correspondente em uma entidade especializada.

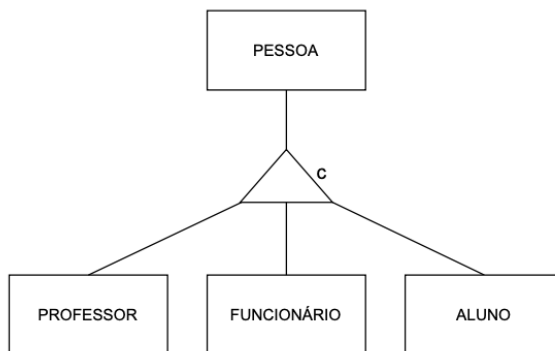


Generalização/especialização parcial. Fonte: Heuser (2010, páginas: 24-56)

- Segundo o mesmo autor, além dos tipos *total* ou *parcial*, as de generalização/especialização podem ser classificadas em compartilhada ou exclusiva.

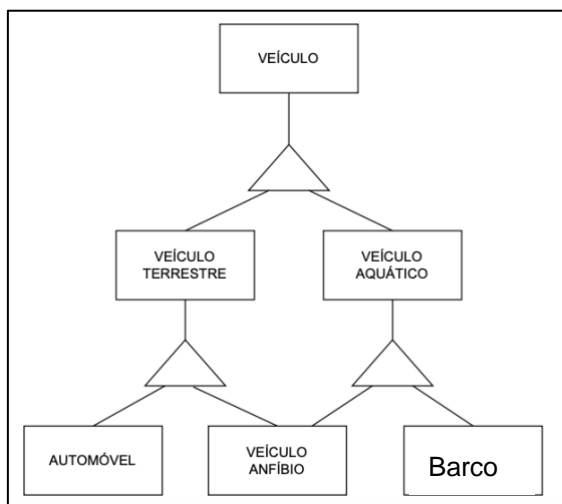
- **Generalização/especialização exclusiva:** em uma hierarquia uma ocorrência de entidade genérica é especializada no máximo uma vez. No exemplo parcial acima a instância funcionário aparece apenas somente uma vez nas entidades especializadas MOTORISTA ou SECRETÁRIA.

- **Generalização/especialização compartilhada:** em uma hierarquia uma ocorrência de entidade genérica pode aparecer em várias entidades de generalização/especialização:



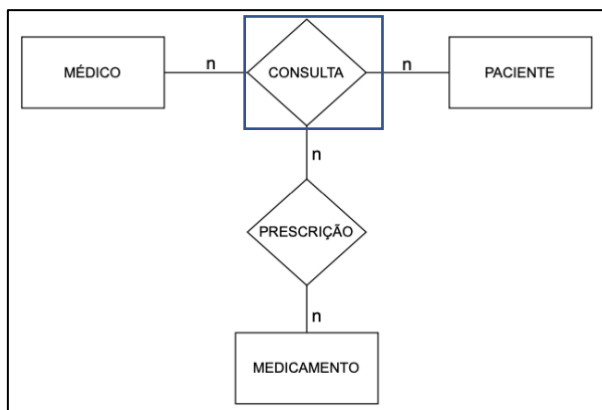
Generalização/especialização compartilhada. Fonte: Heuser (2010, pg: 24-58)

- **Generalização/especialização múltiplos níveis:** uma entidade pode ser genérica e em um outra generalização/especialização. É admissível, inclusive, que uma mesma entidade seja especialização de diversas entidades genérica (a chamada **herança múltipla**).



Generalização/especialização em múltiplos níveis e com herança múltipla. Fonte: Heuser (2010, pg: 24-58)

- **Entidade associativa ou agregação:** um relacionamento é uma associação/agregação entre entidades. Segundo Heuser (2010, p. 60) “Na modelagem ER não foi prevista a possibilidade de associar uma entidade com um relacionamento ou então associar dois relacionamentos entre si”.

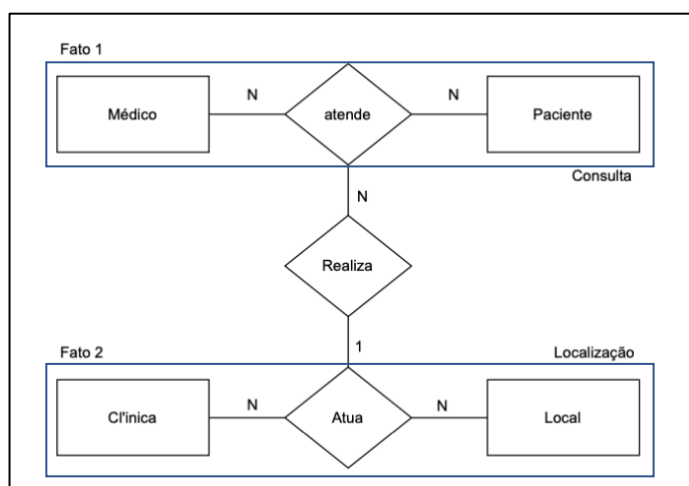


Agregação por implementação de entidade associativa. Fonte: Heuser (2010, 61)

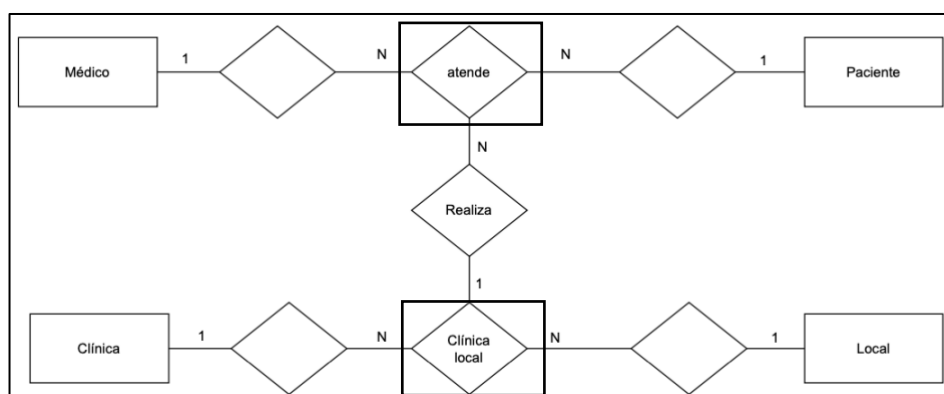
Se não usar o conceito de entidade associativa/agregação, seria necessário transformar o relacionamento CONSULTA em uma entidade, que então poderia ser relacionada a MEDICAMENTO.

Só podemos utilizar entidade associativa/agregação quando temos um relacionamento de Muitos-para-Muitos, que representa um fato.

### - Relacionamento entre blocos do Modelo




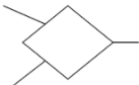



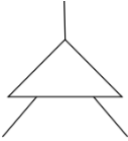
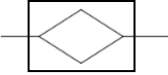
Relacionamento entre dois blocos de agregação. Fonte: Machado (2012, p. 127)



Relacionamento entre dois blocos de agregação. Fonte: Machado (2012, p. 127)

Segundo o autor, um detalhe importante é a fidelidade semântica dos dados e relacionamentos, procure sempre nomear entidades e relacionamentos que expressam um sentido, evitando todas as formas de siglas, tanto para entidades quanto para relacionamentos, que impeçam a leitura simples e natural de um diagrama de Entidade e Relacionamentos.

### - Esquemas gráficos e textuais de modelos ER

Conceito	Símbolo
Entidade	
Relacionamento	
Atributo	
Atributo identificador	
Relacionamento identificador	
Generalização/especialização	
Entidade associativa	

Símbolos usados na construção de esquemas ER. Fonte: Heuser (2010, pg: 62)

### Exercício 1 – possível correção:

A visão de dados de um sistema a ser desenvolvido com enfoque na análise para atender à área de Vendas de uma mercearia não deve ficar distante da realidade. *A análise abaixo está fundamentada no exemplo de Machado e Abreu (2012 / 2016).*

Partimos da premissa que não conhecemos nenhum detalhe ou experiência com sistemas de vendas.

Iniciando a modelagem:

- Existe uma abstração global definida como Venda, pela qual vamos iniciar o trabalho de modelagem, criando num primeiro momento, a entidade:

Venda

- O que caracteriza uma Venda? R: *essa resposta tem diversos atributos que a descreve além de outras entidades encapsuladas.*

- Vendedor: tira pedidos de produtos;

- A premissa acima desmembra a abstração Venda em três entidades:

Vendedor

Pedido

Cliente

- Pedido existe para armazenar dados no objeto pedido;

- Vendedor e Cliente existem para armazenar dados nos objetos.

As três entidades definidas participam do contexto vendas.

- Os objetos Vendedor, Pedido e Cliente pode ser definidos como entidades porque podemos identificar suas ocorrências individualmente e partir disso desenvolver suas tabelas com informações.

A análise sobre a abstração Venda nos remete a algumas reflexões:

- Se estamos efetuando uma Venda certamente estamos vendendo algo que normalmente se chamamos de entidade, 

Produto

 pois existirão várias ocorrências do objeto produto no contexto de uma Venda. Dessa maneira o DER pode ser da seguinte forma

Vendedor

Pedido

Cliente

Produto

- A partir dessa primeira modelagem é possível analisar o relacionamento entre as entidades definidas.

- Partindo da premissa que um pedido contém um ou vários produtos, fica clara a existência do relacionamento entre eles e consequentemente um entendimento melhor.

- Qual é a sua cardinalidade?

- Como os dados se relacionam?

- Durante essa análise ficou claro que um pedido pode conter vários produtos nos remetendo a cardinalidade de Um-para-Muitos.

- Acontece que um produto da mercearia é vendido diversas vezes para vários clientes diferentes através de vários pedidos.

- Considerando o parágrafo acima, analisando o sentido inverso de interpretação, de produto **para** pedido, encontramos uma cardinalidade de um produto para muitos pedidos, logo a cardinalidade é de Um-para-Muitos.

- Sabemos que num DER, quando temos um relacionamento entre duas entidades no qual ambos os sentidos de leitura do diagrama representam Um-para-Muitos, caracteriza que o relacionamento é Muitos-para-Muitos.

- A partir dessas constatações, devemos perguntar:

- Onde estão as informações de quantidade de produtos vendida?

- Qual o preço do produto pedido?

- Qual a sequência do produto pedido?

- Aprendemos que os relacionamentos com cardinalidade Muitos-para-Muitos são representados na forma de tabelas com campos que representam os atributos, no caso do relacionamento entre as entidades Pedido e Produto.

Entidade	Atributos	Relacionamento
Vendedor	id_vendedor nome_vendedor	
Cliente	id_cliente nome_cliente	
Produto	id_produto descricao_produto	Com Pedido 1:n (contém)
Pedido	nr_pedido valor_pedido data_pedido	Com Produto 1:n (contém)

Relacionamento	Atributos	Ligações
Contém	nr_pedido id_produto qtd_produto nr_sequencia_nr_pedido	Muitos com Pedido (nr_pedido) Muitos com Produto (id_produto)

- Vamos analisar se é preciso desmembrar as entidades, para tanto será preciso desenhar os atributos descritivos em cada entidade.

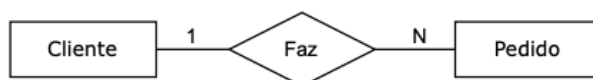
- O que caracteriza um Pedido? R: *Pode ser caracterizado por um número de identificação, por sua data, identificação do cliente, identificação do vendedor e pelos produtos que foram pedidos.*

- Quais seriam os relacionamentos existentes entre o restante das entidades definidas? R: *partindo da premissa que a modelagem é um retrato do mundo real. A premissa Cliente faz Pedido apresenta, além de duas entidades, mais um fato, que também é um objeto deste contexto, o ato de fazer um pedido.*

*O verbo faz é o relacionamento entre a entidade Cliente e a entidade Pedido, e efetua-se por intermédio de referência lógica em pedido de um cliente.*

*Observe que em Pedido temos o atributo "id\_cliente", que caracteriza uma chave estrangeira em Pedido, permitindo uma expressão de comparação: Cliente.id\_cliente = Pedido.id\_cliente.*

- O parágrafo acima possui cardinalidade de Um-para-Muitos no sentido de Cliente **para** Pedido, isto é, um Cliente faz muitos Pedidos.



Simulação entre as tabelas Cliente e Pedido. Fonte: Machado e Abreu (2012, p. 140).

### Cliente:

id_cliente	nome_cliente	fone_cliente	endereço_cliente
54321	Gestronildo Silva	9925-3011 9922-1803	Rua: São Paulo, 35 Setor 5 78.650-882 Ariquemes - RO
54322	Gertrudes João	9842-1115	Avenida das Tulipas Jardim das Palmerias 78.600.993 Ariquemes - RO
54323	Gerimunda Souza	9923-5135 9841-1355	Avenida Jk Setor 2 78.601-855



### Pedido:

nr_pedido	data_pedido	id_vendedor	Id_cliente	Id_produto
2023/001	04/04/2023	005	54323	54323
2023/002	04/04/2023	006	54322	54322
2023/003	04/04/2023	007	54321	54321

- Analisemos a possíveis ligações de Vendedor com outras Entidades:

- A entidade Vendedor está relacionada com o quê? R: *durante a análise descritiva notamos que o vendedor tira pedidos ao atender um cliente.*

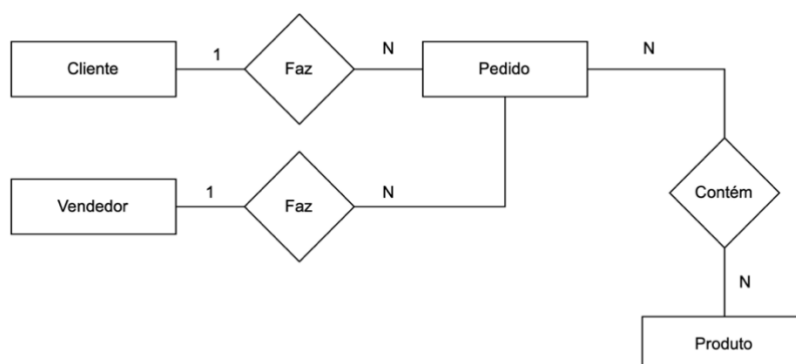
- Diante da resposta no parágrafo acima pergunta-se: a relação entre vendedor e cliente efetiva-se diretamente ou precisa de um meio para realizar-se? R: *Esse relacionamento efetiva-se por meio das ocorrências entre a entidade Vendedor com as ocorrências da entidade Pedido, e isso deve ser representado no modelo.*

### Quadro de entidades

Quadro entidades	Três relacionamentos
- Cliente;	- Faz;
- Pedido;	- Tira
- Produto;	- Contém.
- Vendedor;	

- Sendo Contém um relacionamento com campos, representa os itens de um pedido. Fonte: Machado e Abreu (2012, p. 140).

Reunindo o modelo de dados com as análises realizadas, definimos o dicionário de dados do projeto e o Diagrama de Entidade / Relacionamento.



A partir do DER apresentado é importante desenvolver uma simulação para entender e validar o modelo definido. Ao visualizar as tabelas de

dados, verifica-se o domínio do problema simulado, permitindo correções por interpretação do contexto.

Abaixo tabelas simulando a realidade com valores:

**Cliente:**

id_cliente	nome_cliente	fone_cliente	endereço_cliente
54321	Gestronildo Silva	9925-3011 9922-1803	Rua: São Paulo, 35 Setor 5 78.650-882 Ariquemes RO
54322	Gertrudes João	9842-1115	Avenida das Tulipas Jardim das Palmerias 78.600.993 Ariquemes RO
54323	Gerimunda Souza	9923-5135 9841-1355	Avenida Jk Setor 2 78.601-855 Ariquemes RO

**Vendedor:**

id_vendedor	nome_vendedor
005	Bizarro Assada
006	Esparadrapo Clemente
007	Rodometálico de Andrade

**Produto:**

id_produto	descricao_produto
222	Arroz barrigudinho
329	Refrigerante Oceânico
224	Sal Toca fundo

**Pedido:**

nr_pedido	data_pedido	id_vendedor	Id_cliente	Id_produto
2023/001	04/04/2023	005	54323	222
2023/002	04/04/2023	006	54322	329
2023/003	04/04/2023	007	54321	224

**Contém:**

nr_pedido	Id_produto	quantidade
2023/001	222	2
2023/002	329	10
2023/003	224	8

- O importante dessa etapa é determinar as entidades que coexistem no modelo, outros dados poderão ser colocados à parte como: comissão do vendedor, reserva de mercadorias etc.

- Pode ainda surgir adjetivos genéricos que induzem erroneamente a criar entidades a partir de grupos já definidos como: Pedido Pendente, pois, essa é uma adjetivação ou *status* de um pedido e não uma entidade.

Para dirimir esse tipo de problema na modelagem aplicamos as técnicas de normalização de dados. Porém a maneira ideal de utilizá-la é somente após o reconhecimento e identificação dos grupos principais de entidades.

### **NORMALIZAÇÃO DE DADOS:**

É um processo formal e passo a passo, que em banco de dados relacionais, examina o documento descritivo com o objetivo de purificar anomalias que podem causar problemas, tais como grupos repetidos, redundância de dados desnecessárias, dados multivalorados, perdas acidentais de informação, dificuldade de representação da realidade e dependências entre atributos, relacionamentos, chaves primárias e chaves estrangeiras principalmente durante a inclusão, exclusão, alteração e consulta de registros em um banco de dados.

Este processo utiliza regras fundamentadas embasadas em técnicas da **teoria dos conjuntos da matemática**. Desenvolvida por Edgar Frank Codd em 1970, ficou conhecida como Formas Normalização. É importante entender que um banco de dados relacional normalizado reduz o trabalho de manutenção e ajuda a evitar o desperdício do espaço de armazenamento.

A normalização é utilizada também como ferramenta de projeto do modelo de dados, usando relatórios, formulários e documentos utilizados pela realidade em estudo, constituindo uma ferramenta de levantamento.

Simplificando, o objetivo da normalização é evitar os problemas que podem provocar falhas no projeto do banco de dados, bem como eliminar a "mistura de assuntos" e as correspondentes redundâncias desnecessárias de dados.

Uma regra que deve ser observada em um do projeto de banco de dados orientado para o modelo relacional é nunca misturar assuntos em uma mesma tabela (MACHADO, 2020, p. 310)

Segundo o mesmo autor, o processo de normalização aplica uma série de regras sobre as tabelas de um banco de dados, para verificar se estão corretamente projetadas. Sabemos que as tabelas de um banco de dados relacional são derivadas de um MER e, muitas vezes, nessa derivação, temos muitos problemas de performance, integridade e manutenção de dados.

Normalmente, após a aplicação das formas normais, algumas tabelas acabam sendo divididas em duas ou mais tabelas, o que, no final, gera um número maior de tabelas do que originalmente existia. Esse processo causa a simplificação dos atributos de uma tabela, colaborando significativamente para a estabilidade do modelo de dados, reduzindo consideravelmente as necessidades de manutenção.

### **Anomalias:**

- Para saber quais são as anomalias em um banco de dados não normalizado será preciso pesquisar em toda a tabela, comparando linha a linha numa determinada coluna para encontrar um objeto pesquisado.

- Numa tabela de clientes, uma atualização do endereço de um cliente exigirá o mesmo tipo de pesquisa do parágrafo acima, esse problema se repetirá também na deleção de um pedido de um determinado cliente, e para piorar a situação será apagado a única cópia de seus dados de endereço.

Estes fatos são considerados anomalias de inclusão, alteração e exclusão de dados. Para a normalização dos dados escolhe-se uma tabela, um campo de chave primária que permitirá identificar os demais campos da estrutura.

Embora existam cinco formas normais (ou regras de normalização) 1FN, 2FN, 3FN, 4FN, 5FN, na prática normalmente aplicamos a três Formas Normais iniciais para considerar que um banco de dados está normalizado.

### **Primeira Forma Normal (1FN)**

A 1FN, define que cada ocorrência da chave primária deve corresponder a uma e somente uma informação de cada atributo na tabela, ou seja, a entidade não deve conter grupos repetitivos (multivalorados).

Em se tratando de uma tabela de banco de dados relacional, cada coluna deve ter exatamente uma definição na tabela para que ela esteja dentro da

norma 1FN. É importante entender que ao aplicar a primeira forma normal é preciso retirar a estrutura dos elementos repetitivos.

Resumindo, uma entidade estará na primeira forma normal (1FN) se todos os campos forem atômicos e **NÃO MULTIVALORADOS** (com múltiplos valores).

Exemplo:

**Tabela Cliente:**

id_cliente	nome_cliente	fone_cliente	endereço_cliente
54321	Gestronildo Silva	9925-3011 9922-1803	Rua: São Paulo, 35 Setor 5 78.650-882 Ariquemes RO
54322	Gertrudes João	9842-1115	Avenida das Tulipas, 20 Jardim das Palmeiras 78.600.993 Ariquemes RO
54323	Gerimunda Souza	9923-5135 9841-1355	Avenida Jk, 15 Setor 2 78.601-855 Ariquemes RO

Fonte: Fonte: o autor

Analisando as colunas da tabela acima temos: id\_cliente, nome\_cliente, fone\_cliente, endereço\_cliente. Nos campos temos os dados como: identificador, nome completo, alguns clientes possuem mais de um telefone e o campo endereço armazena a rua ou avenida, o setor, o CEP. Note que há nos dados da tabela que estão em desacordo com a 1FN:

- 1º: Os dados do campo fone\_cliente **são MULTIVALORADO**, há cliente que tem um telefone e outros que tem mais de um telefone;
- 2º: Os dados do campo endereço\_cliente **são MULTIVALORADO**, nele estão armazenados: a rua ou avenida e o número da casa, o bairro, CEP, a Cidade, e a UF;
- Vamos primeiro tentar resolver o problema do campo endereço:
  - Analisando os dados do campo verificamos que ele tem sempre cinco partes. Rua ou Avenida, Bairro, CEP, Cidade, UF, então podemos dividi-lo em **cinco campos na mesma tabela**.

- Lembre-se que para obter entidades 1FN, é imprescindível decompor cada entidade não normalizada em tantos **campos** ou **novas tabelas** quanto for o número de conjuntos de atributos repetidos.

- Se for criar uma tabela, a chave primária será a concatenação da chave primária da entidade original mais os atributos do grupo repetido visualizados como chave primária desse grupo.

#### Tabela Cliente:

id_cliente	nome_cliente	fone_cliente	Rua_avenida	Setor	CEP	Cidade	UF
54321	Gestronildo Silva	9925-3011 9922-1803	São Paulo, 35	Setor 5	78.650-882	Ariquemes	RO
54322	Gertrudes João	9842-1115	das Tulipas,20	Setor 3	78.600.993	Ariquemes	RO
54323	Gerimunda Souza	9923-5135 9841-1355	Juscelino Kub	Setor 2	78.601-855	Ariquemes	RO

Desdobrando da coluna endereco\_cliente - aplicada a norma 1FN. Fonte: Fonte: o autor.

A criação dos novos campos resolveu o problema do endereço multivalorado. Quanto ao campo telefone? Nele algumas entradas têm um número e em outras dois números. Podemos fazer o mesmo? Ou seja, criar dois campos nesta mesma tabela: telefone1 e telefone2?

- A resposta à última questão é **não devemos criar dois campos**. Se criarmos dois campos fone\_cliente\_1 e fone\_cliente\_2, algumas entradas ficarão vazias já que alguns clientes podem ter apenas 1 telefone ou ter mais de dois telefones e na tabela devemos evitar que campos fiquem vazios, pois, uma tabela que aceita campos vazios é uma tabela problemática.

- Partindo da premissa que todos os campos são importantes para estar na tabela e, portanto, não podem ficar vazios, a solução para esse problema será criar uma tabela.

#### Tabela Cliente:

id_cliente	nome_cliente	fone_cliente	Rua_avenida	Bairro	CEP	Cidade	UF
54321	Gestronildo Silva	9925-3011 9922-1803	São Paulo, 35	Setor 5	78.650-882	Ariquemes	RO
54322	Gertrudes João	9842-1115	das Tulipas,20	Setor 3	78.600.993	Ariquemes	RO
54323	Gerimunda Souza	9923-5135 9841-1355	Juscelino Kub	Setor 2	78.601-855	Ariquemes	RO

#### Tabela telefone\_cliente:

id_cliente	Fone_cliente
54321	9925-3011
54321	9911-1803

54322	9842-1115
54323	9223-5135
54323	9841-1355

Desdobrando da tabela Cliente - aplicada a norma 1FN. Fonte: o autor.

**Lembre-se:** o processo de normalização é sequencial, dessa maneira, a segunda forma normal 2FN só poderá ser aplicada se as tabelas estiverem na primeira forma normal 1FN.

**Exercícios:** responda as questões abaixo:

- Quais os problemas de uma tabela não normalizada com a 1FN? Explique?
- Toda tabela precisa obrigatoriamente ser normalizada com a 1FN? Explique?
- Diante do estudado em quantas parte pode ser dividir um roteiro de aplicação da forma normal 1FN? Explique cada parte?

----- //

### **Abordagens de modelagem – Banco de Dados II:**

O conjunto de conceitos usados na construção de um modelo é chamado de modelagem, entre eles temos: modelo conceitual, modelo lógico e modelo físico.

- O modelo lógico possui: **graus de relacionamento e cardinalidades**, que ao serem analisados se desdobram em: **modelo abstrato da organização** e **modelo abstrato do Banco de dados**.

- Para desenvolver a modelagem orientada a objetos para um banco de dados relacional aplicamos a abordagem E/R para linguagens impura, essa técnica foi desenvolvida originalmente por Peter Chen em 1976.

- Para reduzir de problemas de modelagem de banco de dados relacional existem técnicas de normalização de dados que precisam ser respeitadas e acatadas:

- A normalização de dados é um processo formal que parte do documento descritivo da análise dos requisitos para dirimir problemas de redundância de dados e possíveis problemas principalmente durante a inclusão, exclusão, alteração e consulta de registros em um banco de dados.

Este processo utiliza regras fundamentadas embasada em técnicas da **teoria dos conjuntos da matemática**. Desenvolvida por Edgar Frank Codd em 1970, ficaram conhecidas como Formas Normalização.

Normalmente, após a aplicação das formas normais, algumas tabelas acabam sendo divididas, simplificando a simplificação os atributos de uma tabela, estabilizando e reduzindo a manutenção de um banco de dados relacional.

- A Primeira Forma Normal (1FN), define que cada ocorrência da chave primária deve corresponder a uma e somente uma informação de cada atributo na tabela, ou seja, a entidade **NÃO** deve conter grupos repetitivos (multivalorados).

Preste muita atenção aos adjetivos genéricos que tendem a induzir erroneamente a criar entidades a partir de grupos já definidos, pois, normalmente uma adjetivação pode ser um ou *status* de uma entidade e não uma nova entidade.

Também é preciso ficar atento se a nova entidade caso criada demanda o desdobramento em uma nova classe na programação orientada a objetos, algumas vezes, definir um atributo do tipo tupla, lista, dicionário etc., pode resolver o problema de campos multivalorados.

- Lembre-se: o correto entendimento de uma informação depende da condição de interpretação dos fatos. Saber o que um dado caracteriza, ou a que/quem esse dado se refere, é de suma importância para o resultado correto do modelo de dados.

### **Variação temporal x histórico das ocorrências:**

Segundo Machado e Abreu (2012, p. 184), sempre que a decisão de armazenar o histórico de algum atributo for tomada, nasce um relacionamento de Um-para-Muitos (1-N) entre a entidade que contém o atributo e a entidade dependente. Também é preciso estar atento para que no mínimo a data da alteração do atributo, o novo valor do atributo para cada alteração seja registrado, concatenando como chave dessa nova entidade a data de referência.

O armazenamento do histórico de ocorrências após a aplicação da 1FN, é uma atitude que facilita a auditoria de ocorrências e a análise de dados para tomada de decisões futuras (*forecast*) por parte da alta administração. Por isso é importante entender quais atributos vão se transformar ao longo do tempo e criar uma regra de acordo com as normas fiscais de cada empresa, quanto tempo que os históricos devem permanecer armazenados.

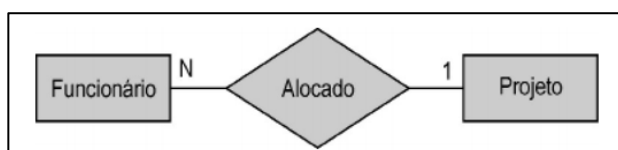
Exemplo proposto por Machado (2020), adaptado pelo autor:



### Quando os fatos podem confundir:

Saber interpretar o que um dado caracteriza, ou a quem esse dado se refere, é de suma importância para o resultado correto do modelo de dados (p.210-211).

Num cenário de uma empresa de projetos de engenharia os colaboradores são alocados a um projeto até o seu encerramento. O registro dessa alocação envolve a data de início das atividades no novo projeto e quantos tempo em meses que ele vai ficar alocado.



DER representando o fato. Fonte Machado (2020, p. 211)

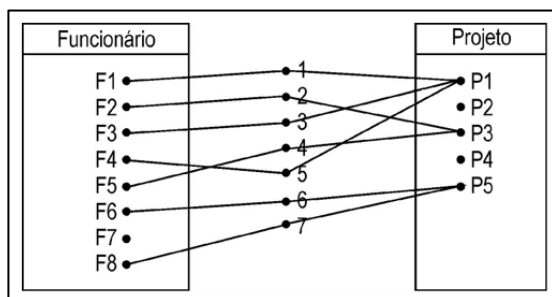
Analisando o diagrama temos:

- Um funcionário é alocado a um projeto e um projeto aloca muitos funcionários;
- Sobre a data de início e o tempo de alocação, como ela se caracteriza?
  - Como esses dados são inerentes ao fato alocação, pergunta-se: são campos, dados do relacionamento alocado?
  - A resposta é **não**, pois, seriam de fato, se um funcionário fosse alocado em mais de um projeto ao mesmo tempo.
  - Para que a entidade funcionário, estabeleça o relacionamento com a entidade projeto, precisará do atributo Código\_do\_Projeto em sua estrutura:

Entidade	Atributos	Relacionamento
Funcionário	Matrícula do Funcionário Nome do Funcionário Endereço Código do Projeto (FK) Data de Início no Projeto Tempo Previsto de Alocação	Aloca N:1
Projeto	Código do Projeto Nome do Projeto	Aloca 1:N

Quadro representando o relacionamento. Fonte Machado (2020, p. 212)

Segundo o autor, utilizamos o padrão FK (*foreign key*) para indicar uma chave estrangeira e o diagrama de instâncias para o relacionamento Um-para-Muitos entre funcionário e projeto ficaria da seguinte forma:



Note que o autor, não aloca o funcionário F7 e que os projetos P4 e P2 não possuem nenhum funcionário alocado. Nesse contexto, a condicionalidade, do relacionamento tem seu grau de importância no contexto, pois permitirá que uma ocorrência de projeto seja inserida na entidade projeto sem a necessidade, de ter funcionários alocados e versa.

A característica para efetivação lógica de um relacionamento um-para-muitos é o fato de ele necessitar da existência de chave estrangeira em uma das entidades. Lembre-se: sempre que existir um relacionamento com cardinalidade de um-para-muitos, a referência lógica chave estrangeira estará colocada na entidade que possuir o lado muitos da cardinalidade (MACHADO, 2020, p.214).

- Então como devemos entender a condicionalidade do relacionamento com relação aos valores dos atributos?

- Para as ocorrências da entidade funcionário que não estiverem relacionadas com nenhuma ocorrência de projeto, o atributo também existirá, porém seu valor será NULO, inexistente.

Matrícula	Nome	Código do Projeto	Data de Início no Projeto	Tempo de Alocação
1466	Pedro Antônio	P-25	12/12/1991	16 meses
2322	Luiz Paulo Diniz	P-18	05/01/1992	4 meses
7712	Carlos Estevão	NULO	NULO	NULO
4415	Sílvia Cristina	P-18	18/04/1992	5 meses

Tabela representando

condicionalidade funcionário-projeto. Fonte Machado (2020, p. 214)

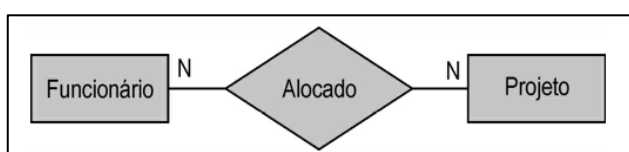
Código do Projeto	Nome do Projeto
P-18	Projeto Phoenix
P-25	Projeto Minerva
P-32	Projeto Corrup
P-55	Projeto Nova Ponte
P-203	Orçamento 95

Tabela entidade os projetos. Fonte Machado (2020, p. 214)

O autor, destaca que existem ocorrências que justificam a cardinalidade muitos-para-um, independentemente de existirem ocorrências com cardinalidade um-para-um.

Na Tabela de ocorrências funcionários / projeto, há um projeto que somente o funcionário Pedro Antônio está com código de projeto P-25, mas, existe ao menos um caso de um projeto ter seu valor de `Codigo_do_projeto` referenciado em mais de uma ocorrência de funcionário, o que é suficiente para estabelecer essa cardinalidade de Muitos-para-Um entre funcionário e projeto.

Utilizando o mesmo cenário de projeto, analisemos a situação do sentido que um funcionário pode atuar em mais de um projeto simultaneamente.



DER representando novo sentido de análise fato. Fonte Machado (2020, p. 216)

Note que a cardinalidade é de Muitos-para-Muitos. Esse tipo de relacionamento tem a característica de possuir campos para representar informações que podem ser inerentes ao fato ou evento, que é o relacionamento.

- A lógica desse tipo de relacionamento, precisa de pelo menos dois atributos, que são as chaves primárias das entidades participantes do relacionamento.

- Neste novo sentido do estudo de caso a `Data_de_Início_no_Projeto` e `Tempo_de_Alocação_no_Projeto` são informações do evento que une funcionário ao projeto, e como pode existir mais de um evento desses para cada ocorrência de funcionário, assim em projeto, esses dados passam a ser únicos para cada ocorrência do relacionamento, passando a serem múltiplos no contexto do modelo (MACHADO, 2020, p. 217).

- Como o número de ocorrências do relacionamento é indeterminado, não podemos mais mantê-los como atributos de funcionário, pois não saberíamos quantas ocorrências colocar desses atributos em funcionário, sendo necessário o desdobramento em múltiplos e indefinidos atributos.

- Logo, o relacionamento alocado passa a ter existência lógica, ou seja, possui dados ou pode ser transformado em uma entidade associativa, e a estrutura de dados do modelo ficará da seguinte forma:

Entidades	Atributos
Funcionário	Matrícula_Funcionário Nome_Funcionário
Projeto	Código_Projeto Nome_Projeto

Relacionamento ou Entidade Associativa	Atributos
Alocado	Matrícula_Funcionário Código_Projeto Data_Início_no_Projeto Tempo_de_Alocação

Quadros representando a existência lógica do relacionamento. Fonte Machado (2020, p. 216)

- Segundo o autor, relacionamento apresentado acima efetiva-se por meio de uma expressão relacional, que indica como deve ser feita a comparação entre os campos comuns as entidades, porém, com características diferentes:

- A comparação é realizada entre campos das entidades e dos campos do relacionamento, formando uma expressão composta.

Já as Expressão do relacionamento:

- Funcionário.Matricula-Funcionário = Alocado.Matricula-Funcionário
- Alocado.Código-Projeto = Projeto.Código-Projeto

Esta expressão diz que o valor do campo matrícula na entidade funcionário deve ser igual ao valor do campo matrícula no relacionamento alocado, e que o valor do campo Código\_do\_Projeto no relacionamento alocado deve ser igual ao valor de Código\_do\_Projeto na entidade projeto, conjuntamente.

Quando isso acontecer com uma ocorrência de funcionário, uma ocorrência de alocado e uma ocorrência de projeto, estaremos relacionando as duas entidades que são funcionário e projeto (MACHADO, 2020, p. 219).

Matrícula	Nome	Data_Admissão
1466	Pedro Antônio	12/05/1990
2322	Luiz Paulo Diniz	18/06/1991
7712	Carlos Estevão	24/05/1990
4415	Sílvia Cristina	05/05/1991
1216	Sandra Chi Min	01/02/1992
1401	Maurício Abreu	15/05/1992

Tabela funcionários

Código_do_Projeto	Nome_do_Projeto
P-18	Projeto Phoenix
P-25	Projeto Minerva
P-32	Projeto Corrup
P-55	Projeto Nova Ponte
P-203	Orçamento 95

Tabela projetos - Fonte Machado (2020, p. 220)

Matrícula Funcionário	Código_do_Projeto	Data_Início_no_Projeto	Tempo_de_Alocação no_Projeto
1466	P-18	24/05/1990	24 meses
1466	P-25	12/11/1991	06 meses
1466	P-32	02/01/1992	12 meses
7712	P-18	10/06/1991	04 meses
7712	P-79	12/12/1991	12 meses
4415	P-18	15/01/1992	03 meses
1216	P-25	01/03/1992	05 meses

Tabela relacionando os fatos estudados - Fonte Machado (2020, p. 220)

Analisando as tabelas acima temos:

- A ocorrência do funcionário com matrícula 1466 é de muitos, pois, ele está alocado em três projetos, P-18, P-25 e P-32.

- A ocorrência do funcionário com matrícula 7712 é de muitos, pois, está alocado a dois projetos.

- Já as ocorrências dos funcionários de matrículas 4415 e 1216 estão alocados cada um em um projeto, logo a cardinalidade é de um-para-um, o que não invalida a cardinalidade a cardinalidade muitos-para-muitos.

A análise e leitura do modelo de dados nos dois sentidos é imprescindível para compreensão perfeita da realidade.

Note o que acontece ao analisarmos a situação por outro sentido de leitura do relacionamento:

- O projeto de código P-18 possui muitas ocorrências de funcionários alocados: 1466, 7712 e 4415, assim como o projeto P-25: funcionários 1466 e 1216. Os projetos P-32 e P-79 possuem somente uma ocorrência de funcionário.

**DEPENDÊNCIA FUNCIONAL:** para entender as próximas formas normais, é necessário conhecer o conceito de dependência funcional que fundamentam tais teorias.

Dada uma entidade qualquer, dizemos que um atributo ou conjunto de atributos A é dependente funcional de outro atributo B da mesma entidade, se a cada valor de B existir, nas linhas da entidade em que aparece, um único valor A. (MACHADO e ABREU, 2012, p. 185)

- B é funcionalmente dependente de A ou B depende de A
- A determina B;

Matrícula	Nome	Sobrenome	Departamento
1021	João	Barros	900
1145	Pedro	Silva	700
1689	Antonio	Jardim	900

- Na tabela acima levanta-se alguns questionamentos:

- Departamento determina matrícula?

A matrícula depende de departamento?

- A resposta é **não**, pois departamento 900 está em 1021 e 1689.

- Matrícula determina departamento?

- A resposta é **sim**, pois, se conhecer a matrícula, é possível saber o departamento, já um funcionário só tem um departamento.

O exemplo acima nos mostra que as informações de uma entidade devem ser dependentes da informação que é a sua identificadora, sua chave primária.

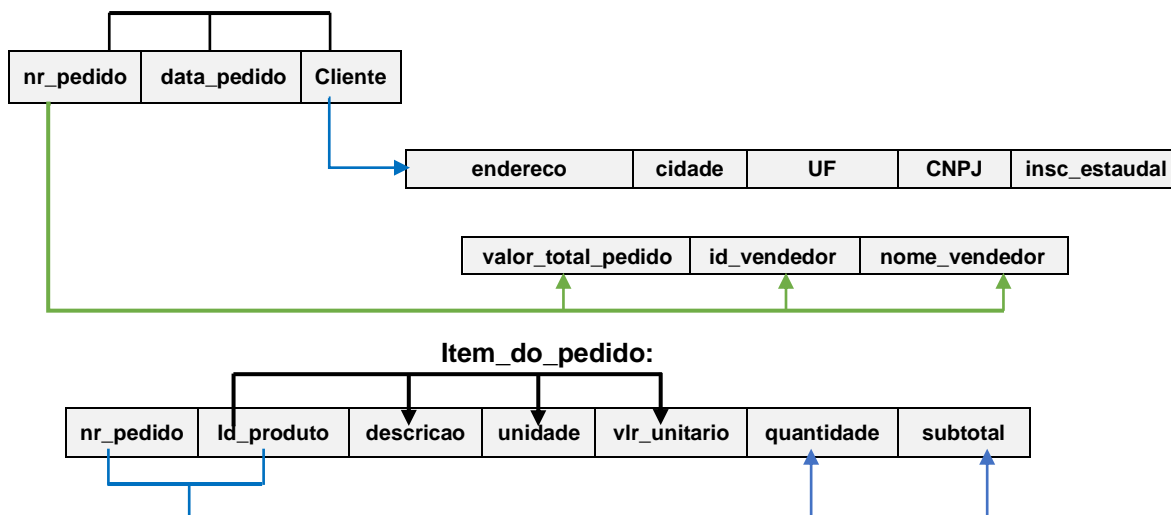
**Dependência Funcional Total (completa) ou parcial:** quando ocorre uma chave primária concatenada, dizemos que um atributo ou conjunto de atributos depende totalmente dessa chave se, e somente se, cada valor da chave (e não parte dela) estiver associado a um valor para cada atributo (MACHADO e ABREU, 2012, p. 185).

**Dependência Funcional Transitiva:** ocorre quando um atributo ou conjunto de atributos A depende de outro atributo B, que não pertence a chave primária, mas é dependente funcional dela, Então dizemos que A é dependente de B. (MACHADO e ABREU, 2012, p. 185)

Observe a tabela de **Vendas** abaixo, que ao aplicar a 1FN à entidade Pedido, vamos obter uma nova entidade chamada Item\_Pedido, que herdará os atributos repetitivos e destacados da entidade Pedido.

## Entidades na 1FN:

### Pedido

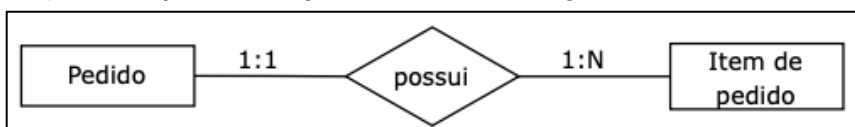


- No exemplo da modelagem de Vendas, acima, a dependência total ocorre na entidade **Item\_do\_pedido**, o atributo **quantidade** depende de forma total da chave primária concatenada **nr\_pedido + id\_produto**. Por isso, é importante conhecer a realidade a ser modelada.

- Segundo o autor, a dependência total só ocorre quando a chave primária for composta por vários (concatenados) de atributos, ou seja, em uma entidade chave primária composta de um único atributo esse tipo de dependência não ocorre.

- Ainda no exemplo da modelagem de Vendas, acima, a dependência transitiva ocorre quando na entidade **Pedido**, os atributos do **endereco**, **cidade**, **UF**, **CNPJ**, **insc\_estadual** são dependentes transitivos do da entidade **Cliente**. Na entidade **Pedido**, o atributo **nome\_vendedor** é dependente transitivo do atributo **id\_vendedor** (MACHADO, 2012 e MACHADO, 2020).

### Representação no Diagrama ER – Modelagem de Vendas



Fonte: Machado e Abreu (2012, p. 183)

- Uma entidade **Pedido** possui no mínimo um e no máximo N elementos na entidade **Item\_do\_pedido** e uma entidade **Item\_do\_pedido** pertence a uma e somente uma entidade **Pedido**, (MACHADO E ABREU, p. 182-184).

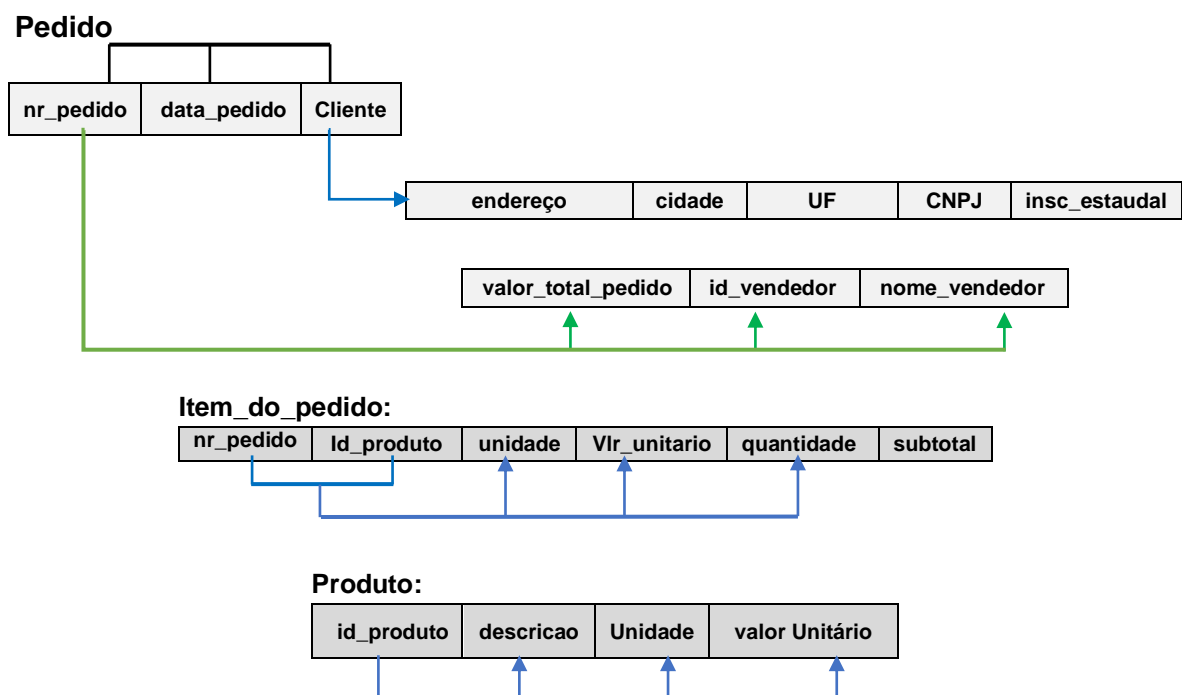
----- //

Uma estrutura estará na forma normal 2FN se a forma normal 1FN não possuir campos que sejam **funcionalmente dependentes** de parte da chave.

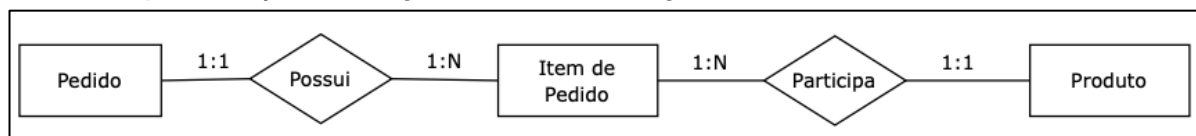
No exemplo de **Vendas** abaixo, vamos analisar se alguma entidade possui chave primária concatenada:

- Se sim, passamos a analisar se existe algum atributo ou conjunto de atributos com dependência total e/ou parcial em relação a algum elemento da chave primária (MACHADO e ABREU, 2012)

### Entidades na 2FN:



### 2FN: Representação no Diagrama ER – Modelagem de Vendas



Fonte: Adaptado de Machado e Abreu (2012, p. 197)

Analisando as tabelas e os diagrama ER acima:

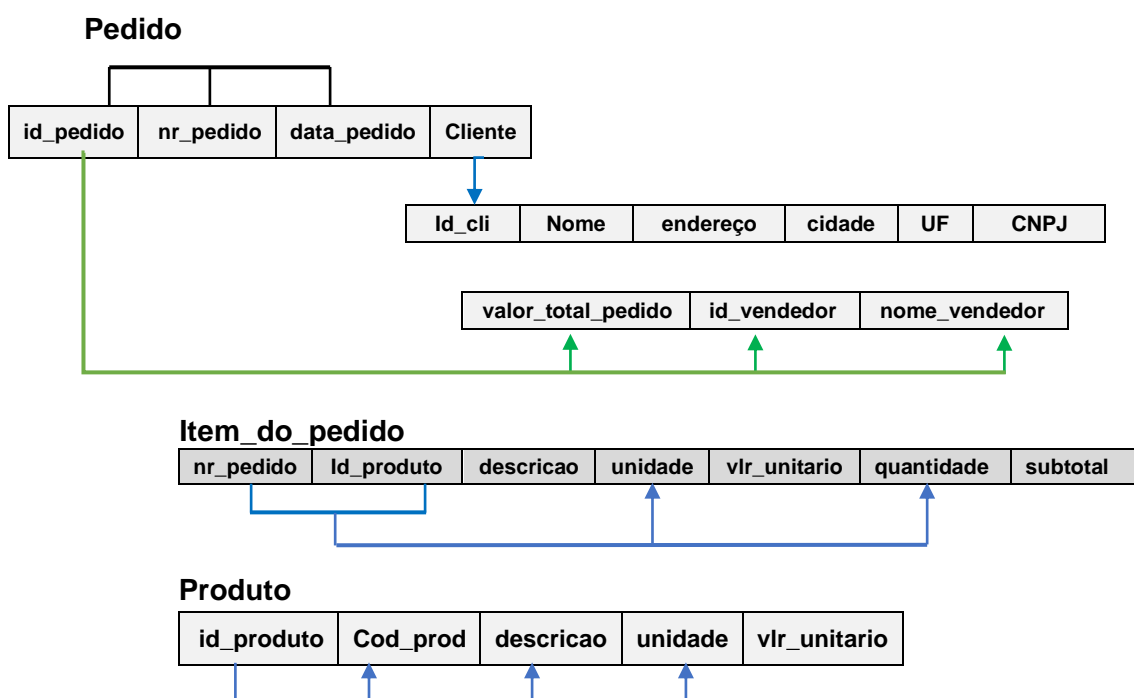
- Uma entidade Produto participa de no mínimo um e no máximo N elementos da entidade **Item\_do\_pedido**, e o atributo **id\_produto** pode pertencer somente a entidade **Produto**, essa análise é o que justifica o relacionamento do tipo 1:N.



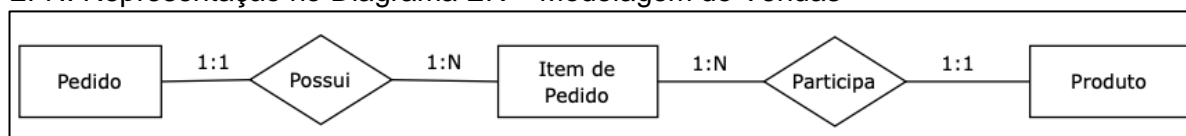
- Segundo Machado (2020, p. 326), Quando um atributo ou conjunto de atributos A de uma tabela depende de outro atributo B, que não pertence à chave primária, mas é dependente funcional deste, dizemos que A é dependente transitivo de B. As dependências funcionais podem existir para atributos que não são chaves em uma tabela. Este fato denominamos dependência funcional transitiva.

### - Considerações sobre a 2FN:

Parta sempre da premissa que o processo de normalização é sequencial, ou seja, só devemos aplicar as regras 2FN se todas as tabelas do banco de dados já se encontrarem na 1FN e todos os atributos não chave forem totalmente dependentes da chave primária. Exemplo:



2FN: Representação no Diagrama ER – Modelagem de Vendas



Fonte: Adaptado de Machado e Abreu (2012, p. 197)

### Anomalias:

#### Item\_do\_pedido

id_produto	nr_pedido	descricao	unidade	vlr_unitario	quantidade	subtotal
001	001-2023	Mouse	Unitário	50.00	3	150.00
002	002-2023	Teclado	Unitário	150.00	5	750.00
003	003-2023	Monitor	Unitário	500.00	2	1000.00

Note que a entidade **Item\_do\_pedido** está na 1FN, no entanto, existem algumas anomalias (problemas) que a impedem de estar na norma 2FN:

Sabemos que uma tabela de uma entidade de um banco de dados só pode ter uma chave primária. Analisando os atributos da tabela da entidade **Item\_do\_pedido**, na intenção de deixá-la na forma 2FN temos:

**Chave primária:**

id_pedido
-----------

**Não chave primária:**

id_produto	descricao	unidade	vlr_unitario	quantidade	subtotal
------------	-----------	---------	--------------	------------	----------

- Note que os atributos não chave primária são totalmente dependentes da chave primária? *Para o Python atributos não chave são os campos que não são chave primária.*

- Analise se há dependência entre os atributos. *Para saber se um atributo é dependente ou não é preciso entender a regra de negócio.*

- Estamos trabalhando na tabela da entidade **Pedido**. Um pedido é feito por um **Cliente** que comprou um produto, logo o **Pedido** é feito sobre um determinado **Produto**. Dentro desse raciocínio é importante ter o atributo **id\_produto** na tabela da entidade **Item\_do\_pedido**? R: **Sim**.

- É importante ter o atributo **descricao** na entidade **Item\_do\_pedido**?

R: **Não**, pois, o atributo **descricao** não é dependente da **chave primária** (**nr\_pedido**) desta tabela e sim do **id\_produto**. Portanto, a descrição do produto está diretamente ligada ao **id\_produto** da entidade **Produto**. Portanto este deveria estar em uma entidade separada.

Entidades quase na **2FN**:

Tabela da entidade **Item\_do\_pedido**

id_produto	nr_pedido	vlr_unitario	quantidade	subtotal
001	001-2023	50.00	3	150.00
002	002-2023	150.00	5	750.00
003	003-2023	500.00	2	1000.00

Tabela da entidade **Produto**

id_produto	descricao	descricao	unidade	vlr_unitario
001	Mouse	Mouse	Unitário	50.00
002	Teclado	Teclado	Unitário	150.00
003	Monitor	Monitor	Unitário	500.00

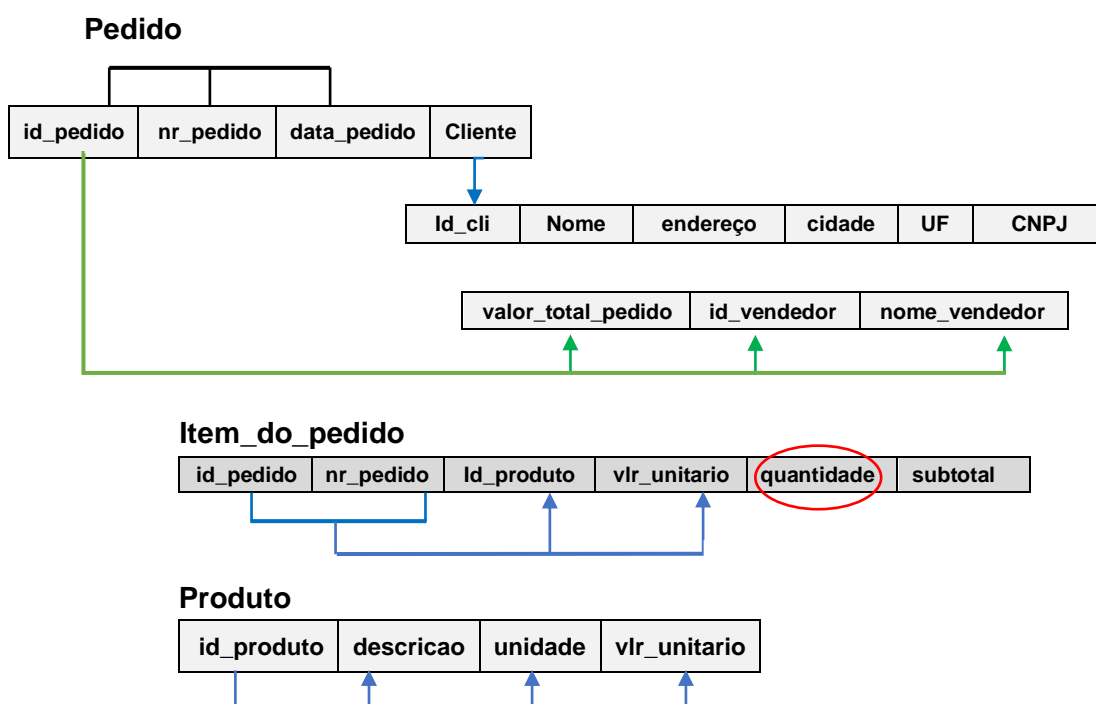
- Ao aplicar a 2FN, houve a necessidade de criar a entidade **Produto**, o que torna o atributo **id\_produto** **chave estrangeira** na entidade

**Item\_do\_pedido** e **chave primária** na entidade **Produto**. Mas, a tabela da entidade **Item\_do\_pedido** está pronta para passar para 3FN?

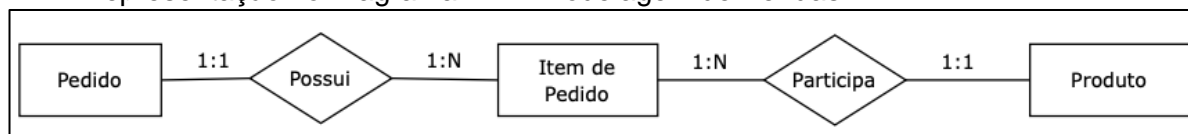
R: Não, pois os atributos **vlr\_unitario** da entidade **Item\_do\_pedido** deve estar na entidade **Produto**, e o atributo subtotal não devia existir na entidade **Item\_do\_pedido**, pois, esse atributo faz parte de um cálculo. Porém, no momento vamos deixar como está, entenderemos quando aplicamos a 3FN.

- Quando aplicamos a 1FN e a 2FN para normalizar os dados é comum criar tabelas de entidades, pois, o objetivo é colocar o banco de dados dentro das normativas.

Entidades na **2FN**:



2FN: Representação no Diagrama ER – Modelagem de Vendas



Fonte: Adaptado de Machado e Abreu (2012, p. 197)

----- //

Segundo Machado (2020), Machado e Abreu (2012) e Heuser (2009), a terceira forma normal ou 3FN determina que **não devem existir atributos com dependência funcional transitiva em uma tabela**, pois eles podem provocar anomalias de inclusão, manutenção e deleção.

Uma entidade estará na 3FN se em sua estrutura nenhum de seus atributos possuir dependência transitiva em relação a outra entidade que não participe da chave primária.

Os autores afirmam que, ao ser retirada a dependência transitiva, deve-se criar uma entidade que contenha os atributos que dependem transitivamente de outro e sua chave primária deve ser o atributo que causou essa dependência.

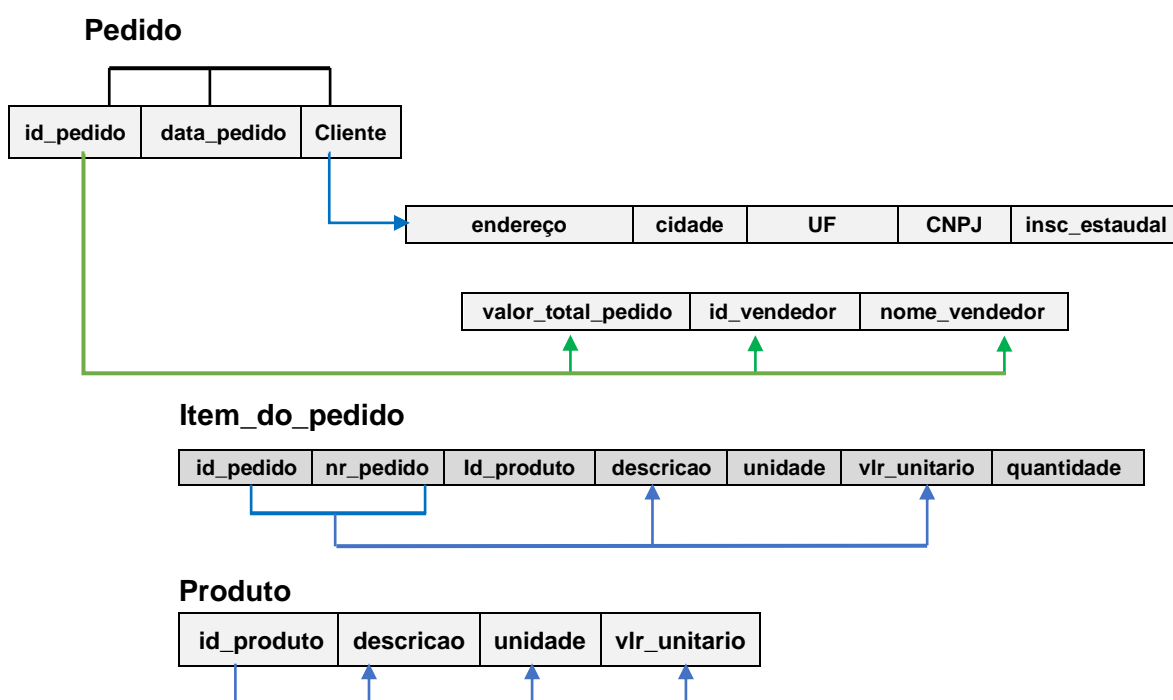
Entidades na 3FN, além de não terem atributos com dependência transitiva elas não devem conter atributos que sejam resultados de algum cálculo de outros atributos.

O objetivo de formalizar tabelas de banco dados é refinar a modelagem deixando a estrutura de dados mais íntegra, exclusiva e sem repetições desnecessárias, o que pode causar possível sobrecarga no gerenciador de banco de dados.

Sabemos que uma tabela estará na 3FN se ela estiver na 2FN e se nenhuma coluna não-chave depender de outra coluna não-chave, ficando para a 3FN a tarefa de eliminar os atributos que podem ser obtidos através dos cálculos de outra equação.

Portanto para que as tabelas de banco de dados referente a vendas fiquem dentro da normal 3FN temos que:

1º - Remover o atributo **subtotal**.



2º - Na tabela da entidade **Pedido** trocar a palavra **Cliente** que representa a entidade Cliente para o atribuo **id\_cliente**. A partir dessa ação criamos a entidade Cliente.

- No entanto, muito cuidado ao com os atributos que dependem transitivamente da entidade **Cliente** Cliente. Neste caso, criar a entidade **Cliente** que terá como chave primária **id\_cliente** e os atributos não chaves: **id\_cliente** nome, rua\_avenida, bairro, CEP, cidade e UF (MACHADO, 2020; MACHADO E ABREU, 2012 E HEUSER, 2009).

#### Pedido

id_pedido	nr_pedido	data_pedido	id_cliente
-----------	-----------	-------------	------------

#### Cliente

Id_cliente	nome	rua_avenida	CEP	cidade	UF	CNPJ	i_e
------------	------	-------------	-----	--------	----	------	-----

Ainda na entidade **Pedido**, o atributo **nome\_vendedor** depende transitivamente do atributo do **atributo id\_vendedor** que não pertence à chave primária. Para eliminar esta anomalia, se deve criar a entidade **Vendedor**, com o atributo **id\_vendedor** como chave primária e o atributo **nome\_vendedor**. Com isso a entidade **Pedido** recebe o **id\_vendedor**.

- As tabelas das entidades **Pedido**, **Clientes**, **Vendedor** e **Produtos** já estão na 3FN:

#### Pedido

id_pedido	nr_pedido	data_pedido	id_cliente	id_vendedor
-----------	-----------	-------------	------------	-------------

#### Cliente

Id_cliente	nome	rua_avenida	CEP	cidade	UF	CNPJ	i_e
------------	------	-------------	-----	--------	----	------	-----

#### Vendedor

id_vendedor	nome_vendedor
-------------	---------------

#### Produto

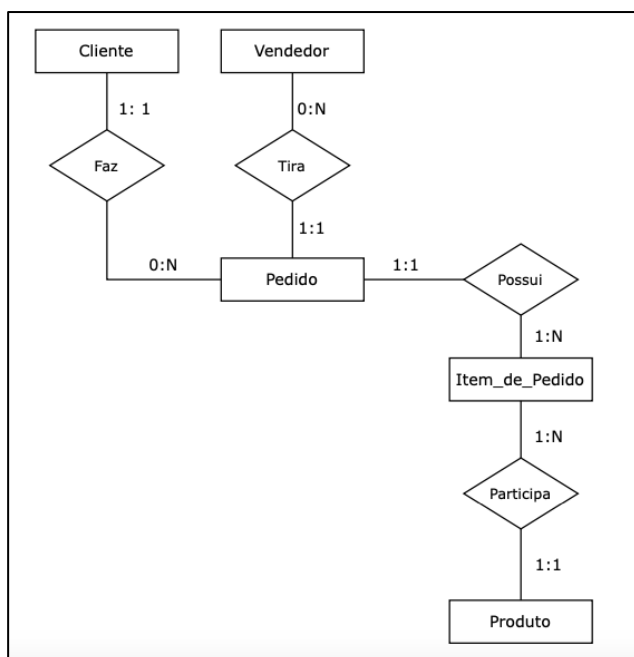
id_produto	descricao	unidade	vlr_unitario
------------	-----------	---------	--------------

- Normalizaremos a tabela **Item\_do\_Pedido** na 3FN:

#### Item\_do\_pedido

id_pedido	nr_pedido	Id_produto	vlr_unitario	quantidade
-----------	-----------	------------	--------------	------------

### 3FN: Representação no Diagrama ER – Modelagem de Vendas



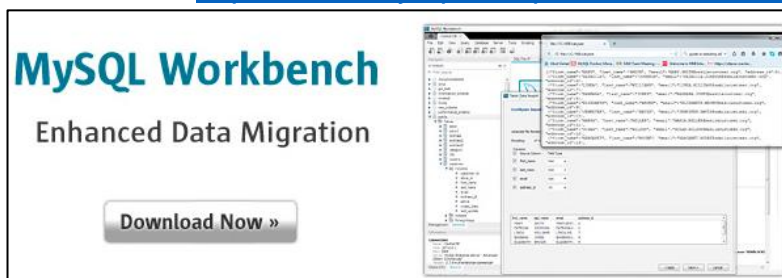
----- //

**Ferramentas Case - Computer-Aided Software Engineering:** são ferramentas utilizadas para a modelagem de dados visual - *MySQL WordkBenck*.

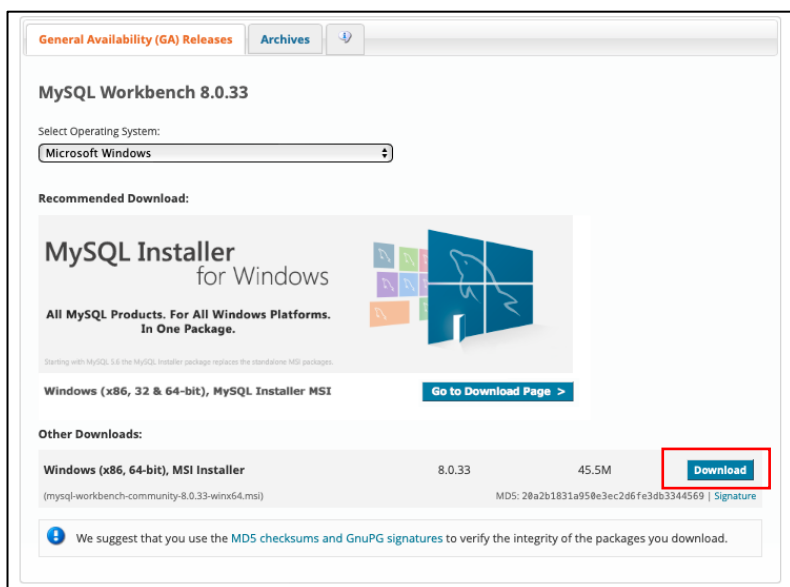
Inicialmente vamos utilizar o *MySQL WordkBenck* para modelar os dados. Sugiro não fazer a instalação pela loja e sim direto do site do MySQL WorkBenck

### Donwload instalação e Configuração do MySQL WordkBenck – no Windows:

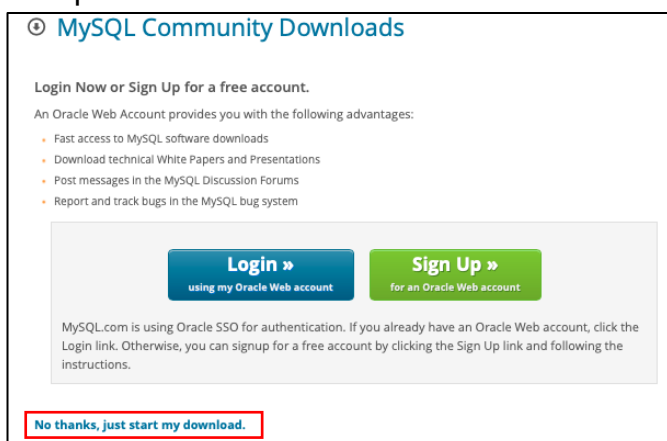
- Abra o site: <https://www.mysql.com/products/workbench/>



- Clique em Donwload now
- *General Availability (GA) Releases*
- Selecione o sistema operacional de seu computador



- Clique no botão *downloads*



- Clique no botão “*no thanks, just start my download*”;
- Assim que terminar o *download* – abra a pasta onde baixou o arquivo e execute-o para que ele possa ser instalado em seu equipamento, siga os passos cliente em next até o final da instalação;
- Após instalado não esqueça de apagar o arquivo de instalação. Observe que após a instalação o *MySQL WorkBench* vai apresentar uma mensagem que não há nenhum servidor rodando. **Deixe como está.** Pois o objetivo de instalar o MySQL Workbenck é inicialmente fazer as modelagens.
- Ao abrir o *MySQL WorkBench* ele vai continuar apresentando a mensagem que não há nenhum servidor rodando. **Deixe como está.**

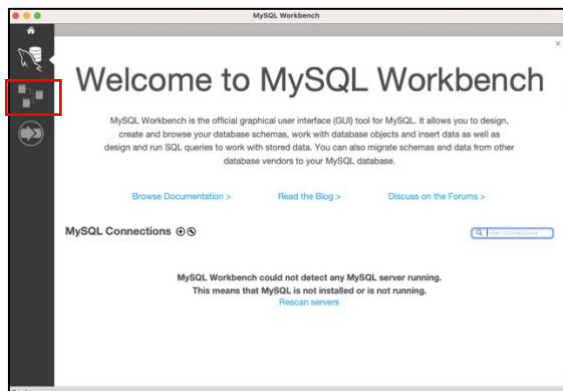


## Conhecendo como funciona o MySQL Workbench:



- Criar um modelo;
- Dar nome ao modelo (futuro banco de dados);
- Salvar o modelo;
- Criar o diagrama - as tabelas (entidades);
- Selecionar os tipos de dados para os atributos;
- Definir chave primária (primari key, pk);
- Definir chave estrangeira (foreing key, pk);
- Exportar o diagrama (imagem ou PDF).

- Criar um modelo – interface inicial do *Workbench*;



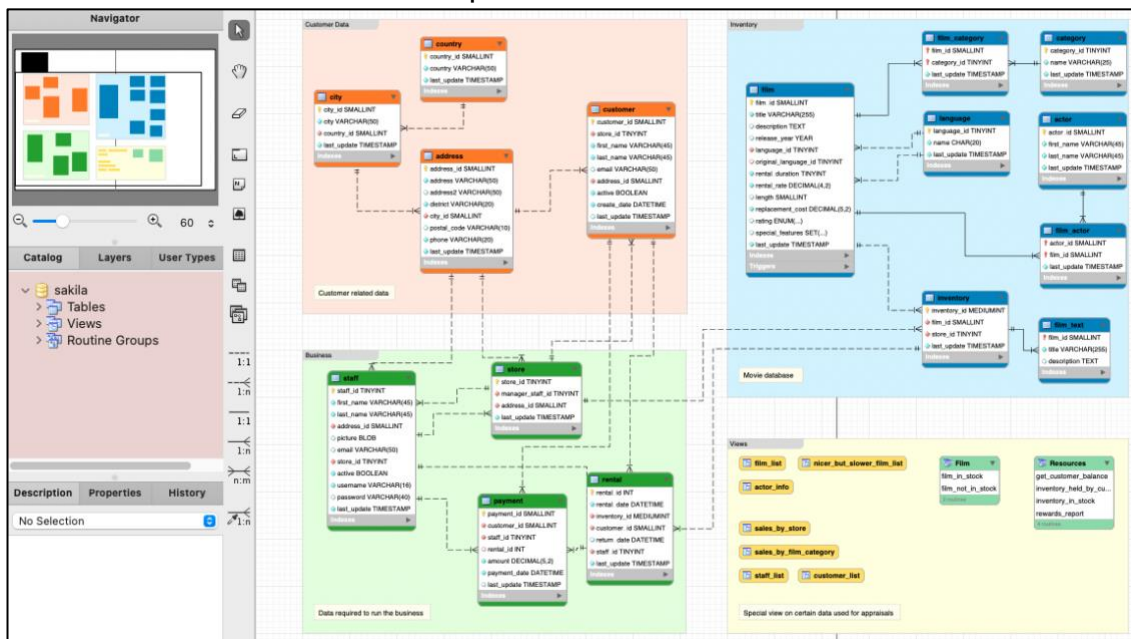
- Para iniciar clique no botão em vermelho;
- Abrirá a tela de modelo – *models*;



- O *WorkBench* já vem com um modelo



- Para abrir o modelo dê dois cliques:

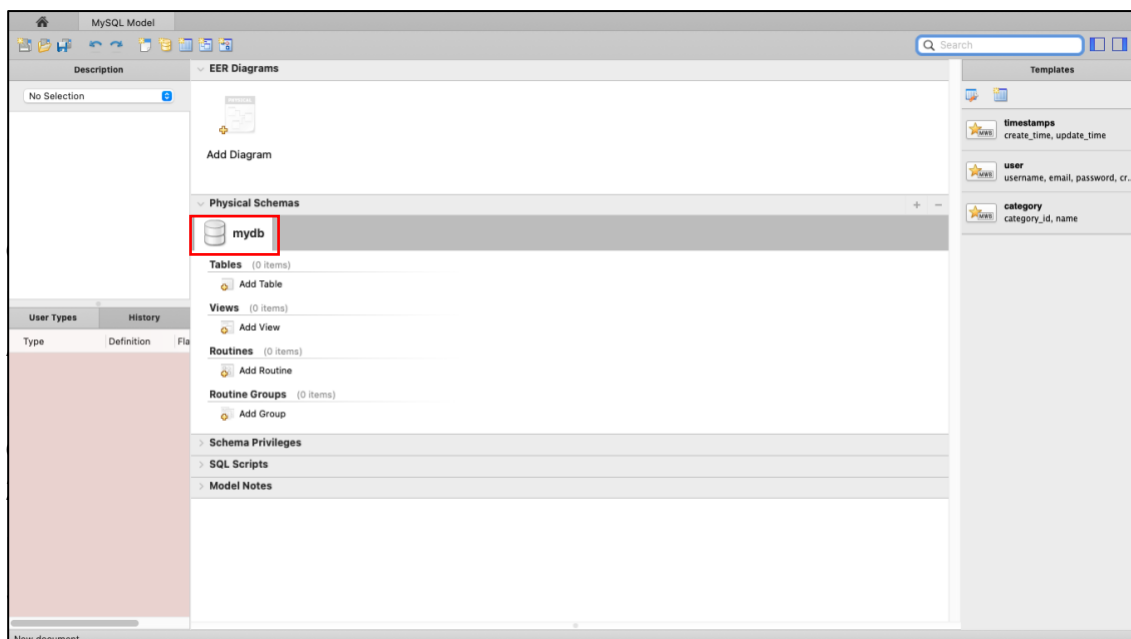


- O modelo apresenta um esboço de como o *Workbench* irá funcionar;
- Clique no botão fechar para fechar o modelo aberto até voltar a tela inicial;
- Clique no botão de + (mais) da tela inicial para criar um modelo;

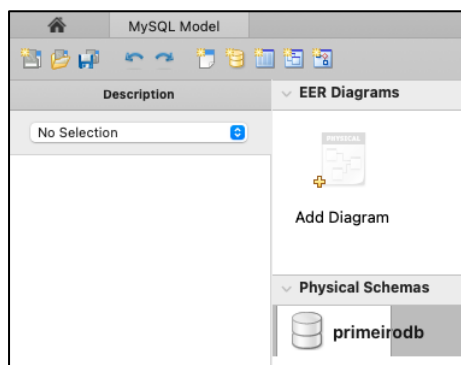




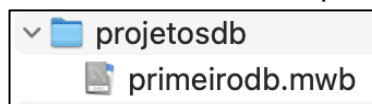
- Primeira ação a ser tomada será dar um nome para o modelo, que será o mesmo nome que será gerado o banco de dados.
- Para fazer isso, localize na tela do modelo o item **mybd** e dê um clique duplo sobre ele – **mybd** é um nome temporário que *WorkBench* dá ao novo modelo;
- Dê o nome ao exemplo que vamos trabalhar de *primeirodb* – tudo minúsculo;
- Clique no ícone em forma de disquete para salvar escolha a pasta do projeto



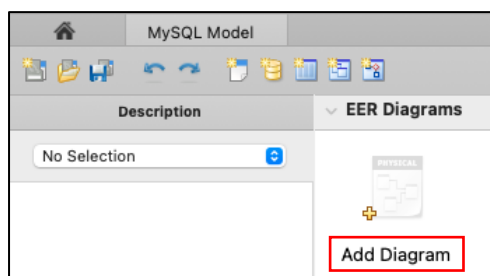
- Após salvar localize no WorkBench o item *Physical Schemas*



- Note que no diretório do projeto onde salvou foi criado o arquivo físico.

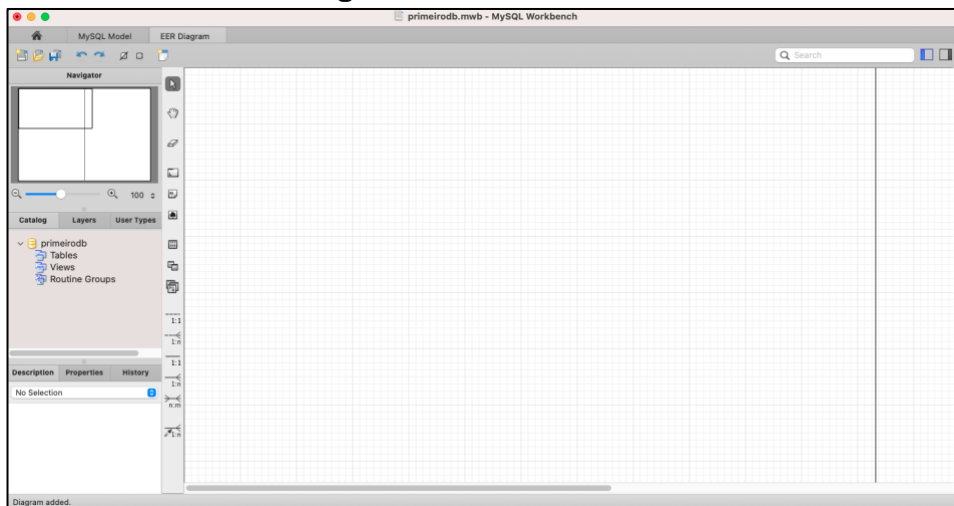


- Criar o diagrama - as tabelas (entidades);
- Para criar o diagrama dê um duplo clique em **add diagram** ;

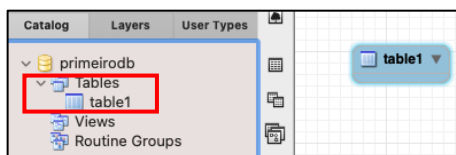
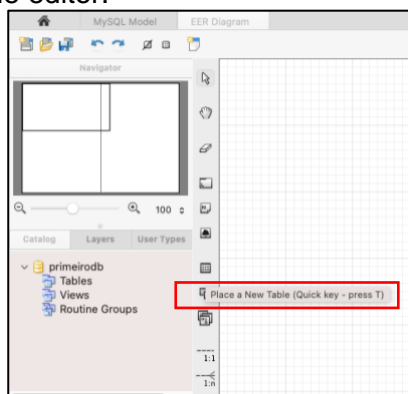


- Clique no ícone em forma de disquete para salvar escolha a pasta do projeto

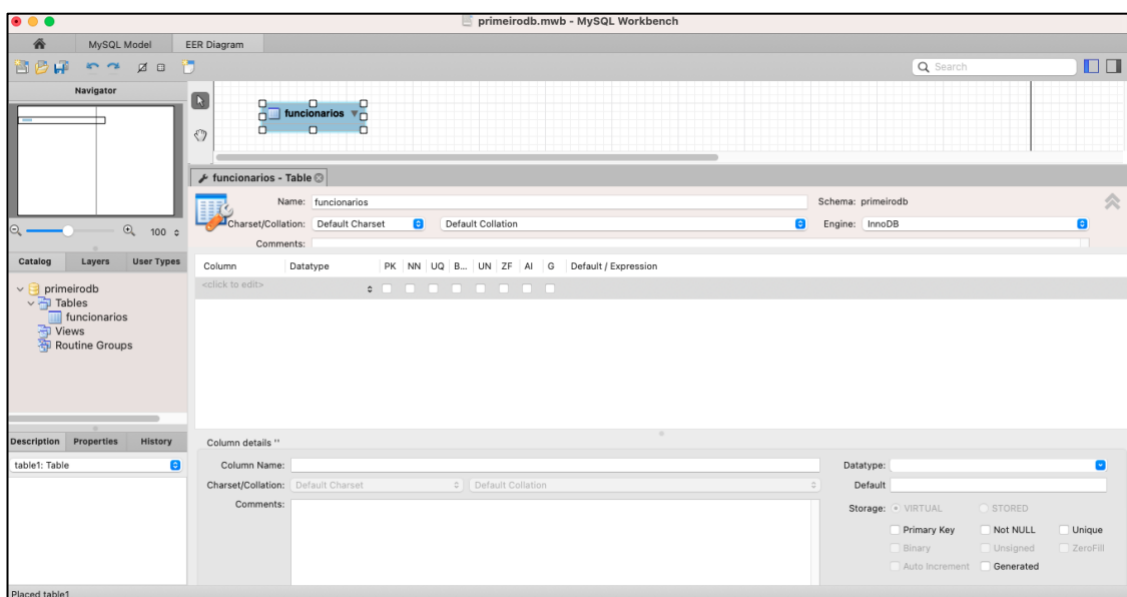
## - Tela do editor de diagramas



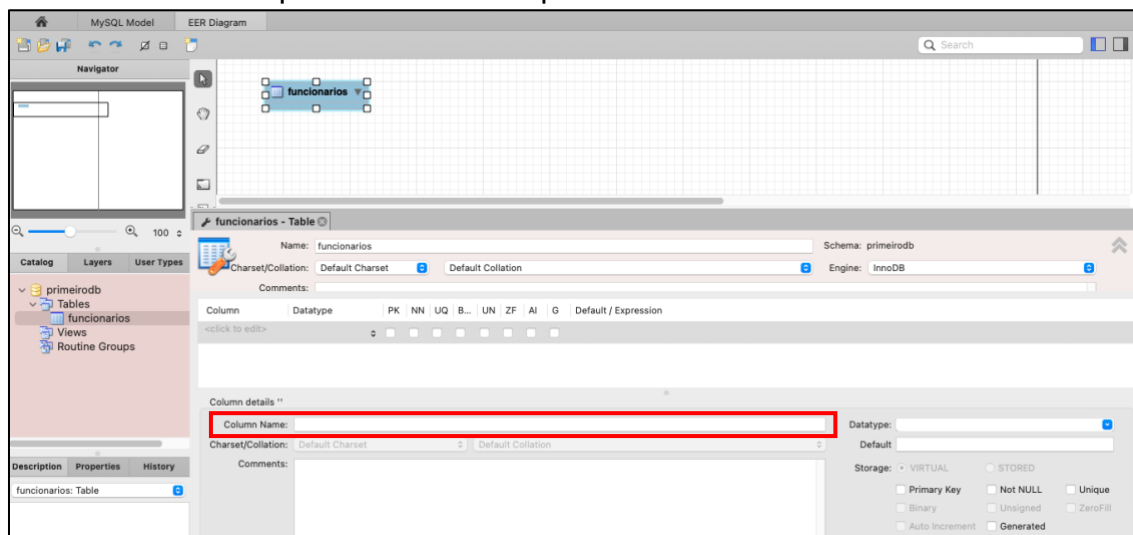
- Inicialmente faremos tudo de forma manual para entender como a interface funciona.
- Criamos e nomeamos o modelo (nome do banco de dados), o próximo passo será:
- Criar as entidades, ou seja, as tabelas de nosso banco de dados;
- Para criar uma tabela clique sobre o botão em destaque e após clique em uma área de editor:



- Assim que a tabela foi criada no *navigator catalog* foi criada a tabela **table1** como nome genérico.
- Para editar a **table1**, dar um novo nome para ela e inserir atributos dê dois cliques sobre o nome da tabela na área do editor.
- Feito isso abrirá a tela de edição conforme abaixo

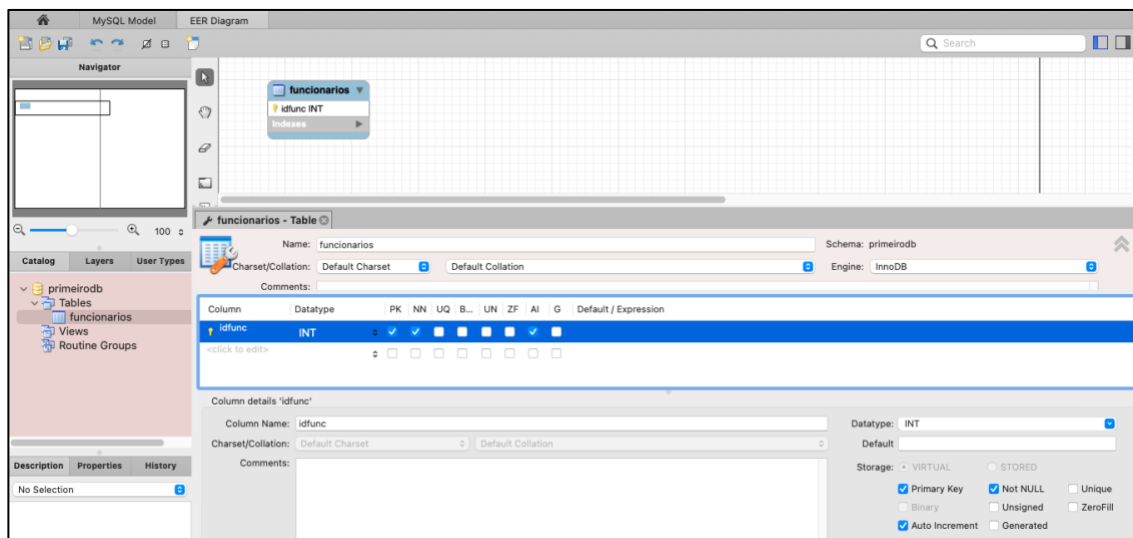


- Note que na tela de edição acima existe outras configurações que no momento vamos deixar como padrão.
- Outro detalhe é que, quando alteramos o nome da tabela para **funcionarios** na área de edição, a tabela **table1** do *navigator catalog* foi teve seu nome alterado para o novo nome.
- Sempre que fizer alterações no seu modelo clique sobre o ícone que representa salvar (disquete), pois o *Workbench* é uma ferramenta complexa e pode ocorrer **bugs** durante o desenvolvimento e a ferramenta fechar sem salvar suas alterações, consequentemente as alterações serão perdidas.
- Até o momento criamos uma tabela chamada **funcionarios**, que nos abre o assunto **nomenclatura de tabelas**:
  - Recomenda-se manter os nomes das tabelas em letras minúsculas;
  - O nome da tabela pode estar no plural ou singular – mas siga um padrão;
  - Se a tabela tiver um nome composto separe o nome por underline;
  - Evite acentuar os nomes, colocar espaços ou caracteres especiais da língua portuguesa no nome das tabelas;
- Após criada a tabela o próximo passo será criar as **colunas da tabela**:
  - Quando você clica no nome da tabela na área na área de edição várias abas de configuração serão abertas na área parte inferior do *Workbench*
  - Localize o campo **column Name**: - para nomear a coluna da tabela



- Por convenção a primeira coluna de uma tabela recebe o identificador;
  - A primeira coluna da table funcionários terá o nome idfunc
- Localize o campo Datatype e o defina com int. Por padrão esse é o tipo de dados de uma coluna **id**
- Por ser o identificador, por padrão ative a checkbox PKcampo Datatype e o defina com int. Por padrão esse é o tipo de dados de uma coluna **id**;

- Localize os campos ou checkbox e ative: **PK** = primary Key, **NN** = Not Null; **AI** = Auto Increment, essa opção deve ser marcada para todos os campos de nossa tabela. Essas são características padrão dos campos identificador. observe também que na tabela apareceu uma chave indicando que esse é um campo do tipo identificador.

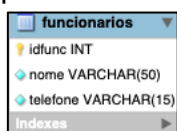


- Abaixo da coluna idfun clique na opção **<click to edit>** e crie as:

- **Column Name: nome**, **Datatype: VARCHAR(50)**, defina entre os parênteses o valor 50 = caracteres – clicando no **Data Type**. Ative a **checkbox**: Not Nul = NN. Realizada as configurações note que na área de edição dentro da tabela funcionário o campo Nome ficou com uma marcação em **azul** indicando que o campo não aceita valores nulos;



- **Column Name: telefone**, **Datatype: VARCHAR(15)**, defina entre os parênteses o valor 15 = caracteres – clicando no **Data Type**. Ative a **checkbox**: Not Nul = NN. Realizada as configurações note que na área de edição dentro da tabela funcionário o campo telefone ficou com uma marcação em **azul** indicando que o campo não aceita valores nulos;

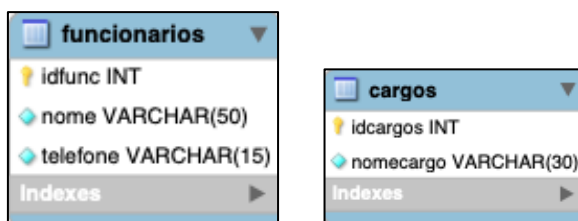


- Vamos conhecer como definir uma chave estrangeira ou **foreign key**. Partindo do princípio de que uma chave estrangeira é um campo que se relaciona com outra **entidade**, no entanto, em nosso exemplo temos apenas uma entidade.

- Diante da situação crie uma outra entidade (tabela) com o nome **cargo** que deve ter os seguintes campos:

- **Column Name: idcargo** que será o campo identificador do registro da tabela; **Datatype: int**; Ative as **checkbox**: **Primary Key** ou **PK** e a **Not Nul** ou **NN** e **Auto Increment** ou **IA**;

- *Column Name*: **nomecargo**, *Datatype*: **VARCHAR(30)**, defina entre os parênteses o valor 30 = caracteres – clicando no *Data Type*. Ative a *checkbox*: **Not Nul = NN**.

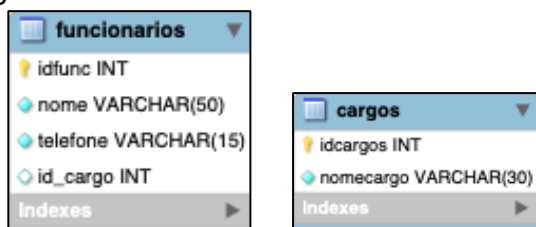


- A questão agora é como faremos para relacionar a tabela **funcionarios** com a tabela **cargos**? Pois sabemos que todo funcionário tem um cargo em uma empresa.

- Como nesse primeiro momento estamos trabalhando de forma manual, para conhecermos e entendermos como funciona a criação e o relacionamento de tabelas. Vamos voltar a tabela **funcionarios** e editá-la (dois cliques):

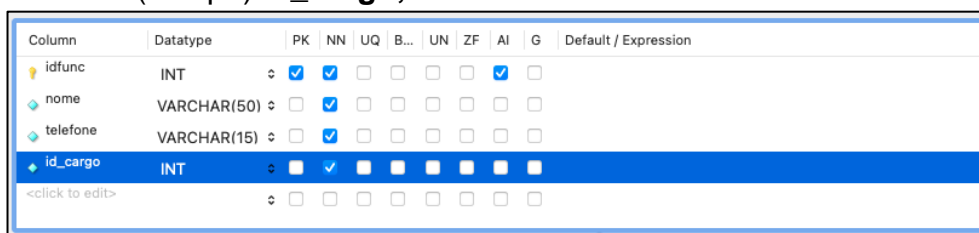
- Crie a *Column Name*: **id\_cargo** - *Datatype*: **int** - **Not Nul** ou NN, que será o atributo da tabela **funcionário** que fará referência ao **idcargos** da tabela **cargo**.

- **Atenção** no Windows o **WorkBench** não gera a chave estrangeira de forma automática, o desenvolvedor deverá saber qual o nome da chave estrangeira que ele criou nas outras tabelas. Até o momento nossas tabelas estão da seguinte forma:

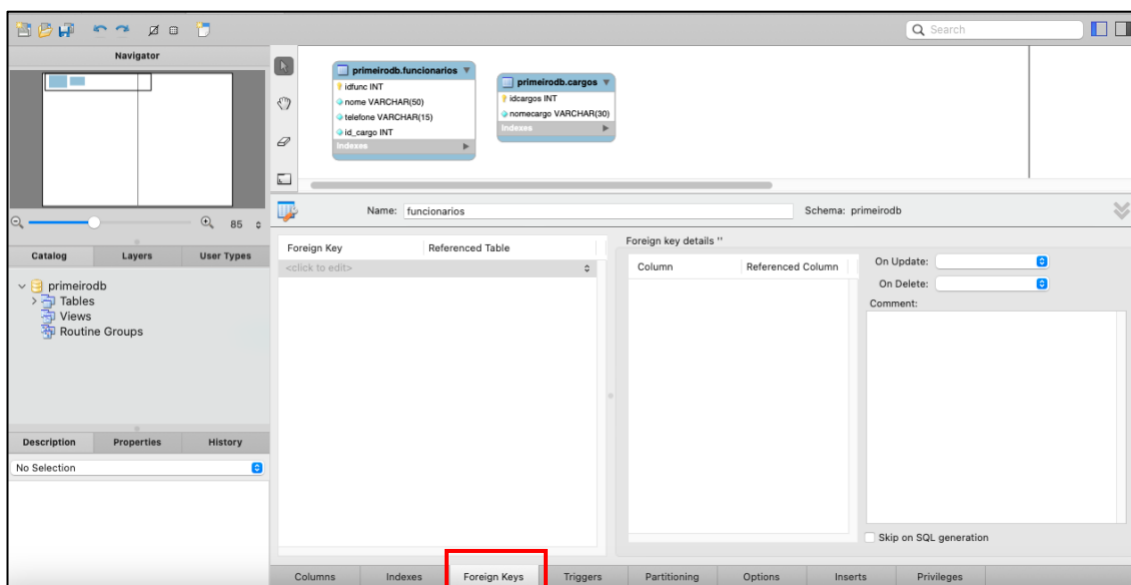


- Para definir uma chave estrangeira no Windows serão necessárias quatro ações:

**1ª** Configurar o *Workbench* de forma que ele entenda que a coluna (campo) **id\_cargo** é uma chave estrangeira. Para isso no *Workbench*, na área de edição de dois cliques na tabela **funcionario**. Depois caixa *column* selecione a coluna (campo) **id\_cargo**;



**1ª** - Leve o ponteiro do mouse na parte inferior da tela “última linha” do *Workbench* clique na guia **Foreign Keys**

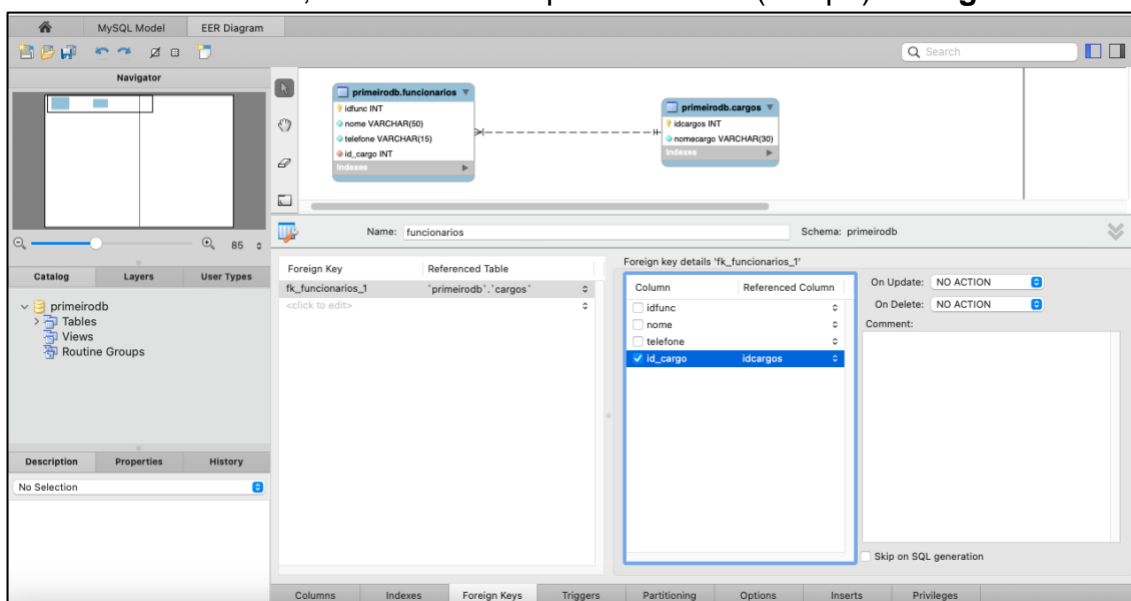


- Clique que na caixa **Foreign Key** e defina uma nome para sua chave estrangeira.

- Ao definir um nome para a chave estrangeira. Comece o nome da chave estrangeira com a sigla **fk** seguida de **underline** mais o **nome da tabela** mais o número da chave estrangeira daquela tablea Por exemplo **fk\_funcionarios\_1**, pois, uma tabela pode se relacionar com mais várias tabelas, sempre siga um padrão para os nomes de chave estrangeira para não se perder.

**2ª** Seleciona a tabela na qual a tabela **funcionarios** se relaciona, em nosso exemplo é a **tabela cargos**.

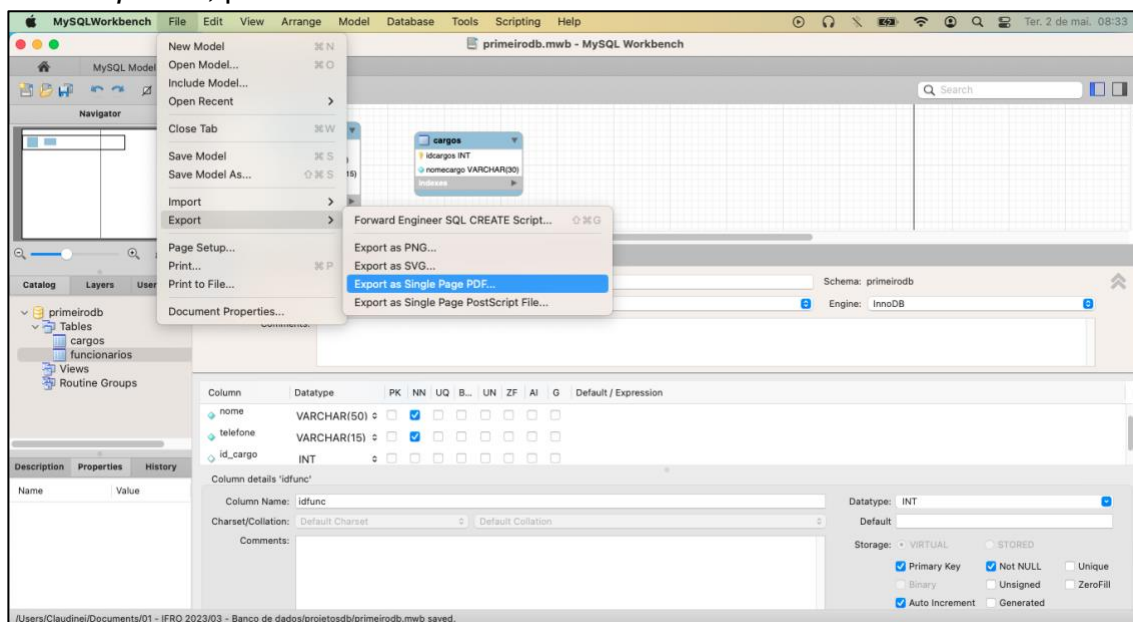
**3ª** Selecionar qual a coluna (campo) da tabela **cargo** irá se relacionar com a tabela **funcionário**, em nosso exemplo é a coluna (campo) **idcargos**.



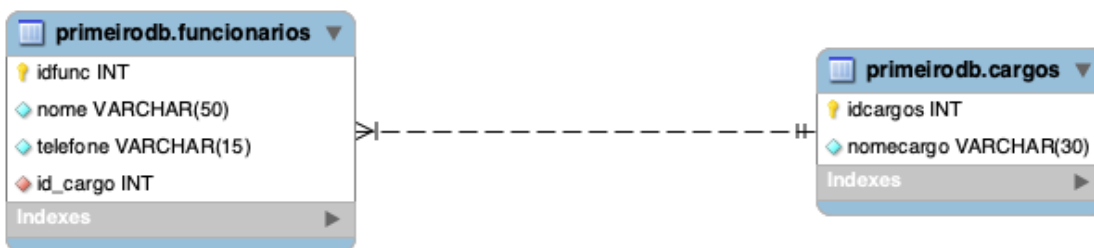
- Se tudo der certo haverá uma linha pontilhada seguindo normalmente o estilo pé-galinha. A linha pontilhada é chamada de relacionamento fraco, pois,

só conseguimos criar um funcionário se já houver um cargo definido, isso significa que a tabela **funcionario** depende da criação da tabela **cargo**.

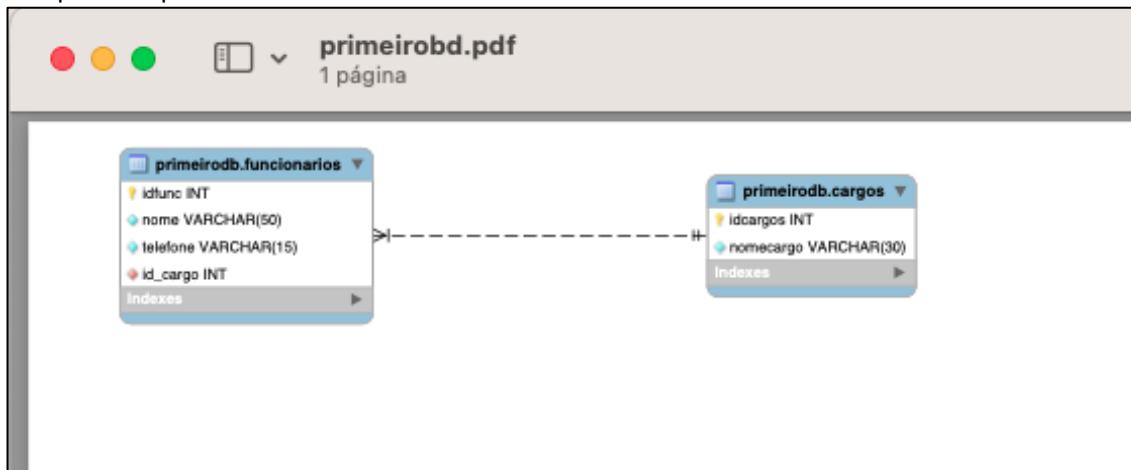
**4ª** A última ação a ser feita é a **exportação do modelo**. Partindo do princípio de que a modelagem do banco de dados está terminada e agora é o momento de criarmos uma documentação do banco de dados para o cliente, ou criar o *scrip SQL*, para isso:



- Arquivos exportados



- Arquivos exportados





## Referências

F. V. C. Santos, Rafael. Python: **Guia prático do básico ao avançado** (Cientista de dados Livro 2). 2020. 2ª Edição - Kindle.

HEUSER, Carlos Alberto. **Banco de Dados Relacional: Conceitos, SQL e Administração**. 1ª Edição, Editora Porto Alegre, 2019).

MACHADO, Felipe Nery Rodrigues; ; ABREU, Maurício. **Projeto de banco de dados: uma visão prática**. Edição revisada. São Paulo. Editora Érica, 2012 / 2016. ISBN: 978-85-365-0252-6

MACHADO, Felipe Nery Rodrigues. **Banco de dados**. 4ª Edição. Editora Érica. São Paulo. 2020. ISBN 9788536532691