Agenda 11: Modelo lógico de Banco de Dados

Conceitos trabalhados:

Sumário

Momento de Reflexão

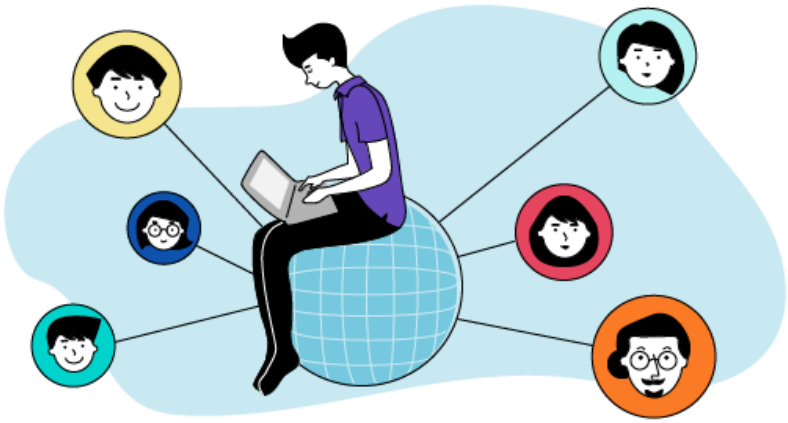
Olá, estudante! :)

Boas-vindas à décima primeira agenda do módulo 1!

Na agenda anterior você, aprofundou seus estudos sobre modelagem de dados, conhecendo o modelo conceitual MER e seus principais elementos. Viu ainda como fazer um Diagrama Entidade-Relacionamento, que é a representação do modelo lógico. Nessa agenda você avançará em seus estudos detalhando ainda mais seu modelo lógico por meio de uma etapa conhecida como **mapeamento.**

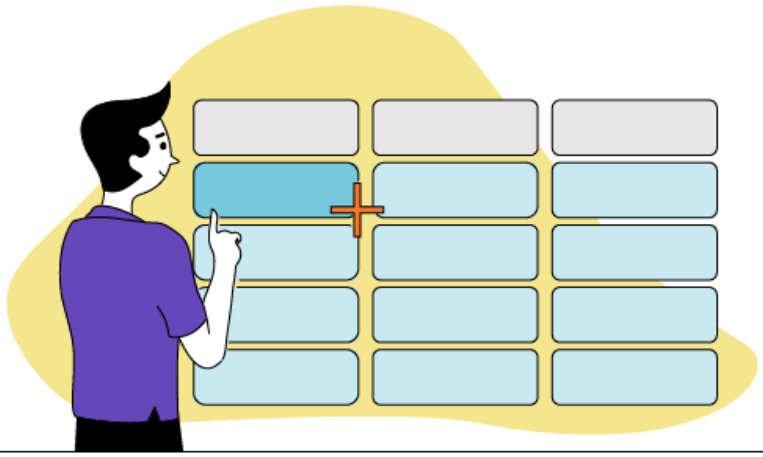
Para começar, que tal recapitular o processo que fez até aqui pensando em um exemplo prático?

Todos nós temos contato com muitas pessoas no dia a dia, não é mesmo? São pessoas da escola, da família, do trabalho, vizinhos do bairro...



Como você é muito organizado, decidiu agrupar os dados pessoais dos seus contatos, para ter acesso sempre.

A ideia é reunir informações como nome, endereço e telefone em um único arquivo. E qual a melhor maneira para fazer isso utilizando um editor de textos?



Sim, por meio de tabelas!

Na agenda 9 você viu que em um BD os dados estão organizados em campos (por exemplo, nome, endereço e telefone) que, por sua vez, estão organizados em **tabelas**. Depois, aprendeu a construir os modelos conceitual e lógico. Neste momento do curso, você tem em mãos um diagrama que representa seu futuro banco de dados, com as entidades, os atributos e os relacionamentos.

Mas, se um banco de dados é composto por tabelas, assim como a sua lista de contatos, como obter essas tabelas a partir do diagrama? Nesta agenda você irá aprender a organizar os dados em tabelas, utilizando conceitos de Banco de Dados.

Vamos lá?

Por que Aprender?

Você já sabe que, para selecionar SGBD, é necessário fazer uma **descrição dos tipos de informações que serão armazenadas em um banco de dados**. Para organizar essas informações, utilizamos modelos com diferentes níveis de abstração e objetivos (os modelos conceitual, lógico e físico). Cada descrição recebe o nome de Esquema de Banco de Dados.

Após selecionar o SGBD, o projetista deverá relacionar as características e restrições do modelo conceitual (MER) com os requisitos e conjuntos de regras do SGBD selecionado para implementação. Assim, o modelo lógico, que é uma representação do funcionamento do banco de dados, utilizará as estruturas suportadas pelo banco escolhido.

Porém, vale destacar que o diagrama ER permite registrar **quais** dados podem aparecer no banco, mas não registra **como** esses dados serão armazenados no SGBD. Para isso, é necessário detalhar o modelo lógico, ou seja, fazer seu mapeamento, para que, a partir desse detalhamento seja possível construir o **modelo físico,** assunto da próxima agenda.

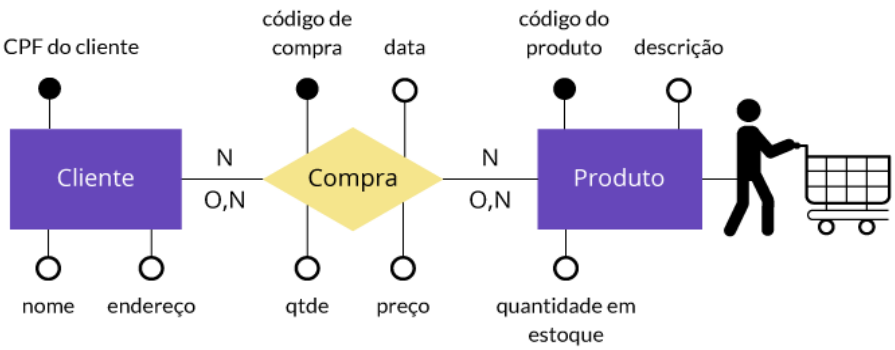
Para Começar o Assunto

Adriano é um rapaz empreendedor e está abrindo um minimercado em seu bairro. Já está quase tudo pronto, mas falta um ponto fundamental: um sistema informatizado que ajude Adriano a organizar e controlar seu negócio, principalmente pensando no estoque, clientes e vendas.

Para isso, contratou Ferrari, um desenvolvedor de sistemas que ficará responsável pela implantação do software. O primeiro passo de Ferrari foi levantar os dados e necessidades do minimercado, organizando um **modelo conceitual**, que classificou e organizou todos os requisitos levantados.

Num segundo momento, Ferrari deveria partir para o **modelo lógico**: fazer um diagrama ER que agrupasse os dados em entidades e relacionamentos. Foi quando explicou a Adriano que era necessário decidirem qual Sistema Gerenciador de Banco de Dados (SGBD) o minimercado utilizaria, já que as especificações do SGBD influenciam na construção do DER. Dessa forma, poderiam fazer os refinamentos necessários para alcançar maior desempenho das operações no banco de dados.

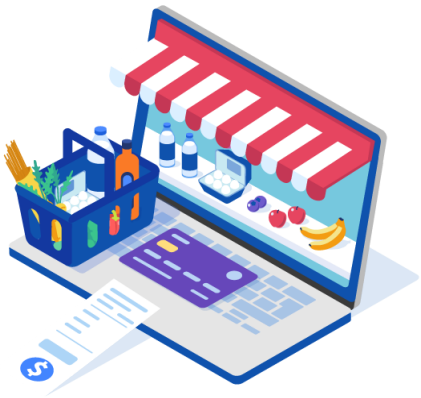
Sabendo que existem inúmeras opções de SGBD no mercado, cada qual com suas vantagens e desvantagens, para o contexto do mercado de Adriano decidiram implantar um Banco de Dados MySQL. Então, Ferrari deu continuidade a seu trabalho e construiu o DER:



Apesar de Ferrari ser um desenvolvedor, ele explicou a Adriano que não trabalha sozinho, mas com uma equipe dedicada à construção de bancos de dados. Assim, ele precisa garantir que as informações contidas no modelo lógico sejam representadas de acordo com o modelo relacional. Para restringir as opções dos desenvolvedores, ele detalhará o modelo lógico, que atualmente é um diagrama, em uma outra forma de representação. Essa etapa é chamada **mapeamento**, consiste em um conjunto efetivo de regras e permite que, posteriormente, Ferrari possa construir seu **modelo físico**.

Explore os materiais a seguir para conhecer as diretrizes e regras do mapeamento e aprender a criar o modelo físico.

Mergulhando no Tema



Adriano é um rapaz empreendedor e está abrindo um minimercado em seu bairro. Para permitir um bom gerenciamento, contratou o desenvolvedor Ferrari para criar o sistema informatizado do estabelecimento.

Ferrari levantou os dados e necessidades do pequeno negócio de Adriano, classificou e organizou os requisitos em um modelo conceitual. Depois, em conjunto definiram o SGBD a ser utilizado no minimercado: o MySQL. Com essa informação em mãos, o desenvolvedor partiu para o modelo lógico, criando um diagrama entidade-relacionamento.

Apesar de Ferrari ser um desenvolvedor, ele explicou a Adriano que não trabalha sozinho, mas com uma equipe dedicada à construção de bancos de dados. Assim, ele precisa garantir que as informações contidas no modelo lógico sejam representadas de acordo com o modelo relacional. Para restringir as opções dos desenvolvedores, ele **detalhará o modelo lógico**, que atualmente é um diagrama, em uma outra forma de representação. Essa etapa é chamada **mapeamento**, consiste em um conjunto efetivo de regras e permite que, posteriormente, Ferrari possa construir seu **modelo físico**.

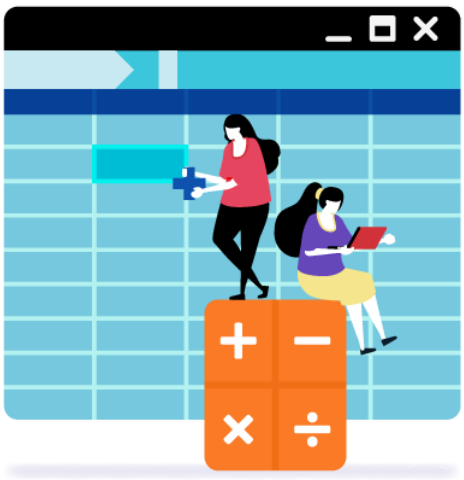
Você já sabe que, para selecionar SGBD, é necessário fazer uma descrição dos tipos de informações que serão armazenadas em um banco de dados, e para isso são utilizados diferentes tipos de modelos. Após selecionar o SGBD, o projetista deverá relacionar as características e restrições do modelo conceitual (MER) com os requisitos e conjuntos de regras do SGBD selecionado para implementação. Assim, o modelo lógico, que é uma representação do funcionamento do banco de dados, utilizará as estruturas suportadas pelo banco escolhido. Perceba que o **modelo lógico depende do software que será utilizado**. Qualquer alteração feita no SGBD exige que a estrutura interna do futuro banco seja alterada para adequar-se às características e exigências de implementação do modelo de BD selecionado.

É importante que você também note que o diagrama entidade relacionamento permite registrar **quais** dados podem aparecer no banco, mas não registra **como** estes dados serão armazenados no SGBD, ou seja, como se organizam nas tabelas existentes no banco de dados. Para esse detalhamento, é necessário desenvolver o **modelo físico**. Em um Banco de Dados Relacional (BDR), que é o tipo de BD que será trabalhado neste curso, o esquema interno é expresso utilizando **linguagem SQL**, por padrão.

Conheça agora alguns conceitos fundamentais para o detalhamento do modelo lógico, uma etapa conhecida como **mapeamento**.

Chave primária e chave estrangeira

Antes de continuar a construção do Diagrama ER para o minimercado do Adriano, vamos entender quais os elementos necessários para montar o modelo lógico.



Você já sabe que o modelo lógico parte para a construção de uma estrutura em formato adequado ao Sistema Gerenciador de Banco de Dados escolhido, onde o banco de dados será implementado. Sendo assim, os modelos são bastante dependentes da tecnologia a ser adotada. Essa tecnologia pode ser relacional, em redes ou orientada a objetos, e neste material iremos trabalhar com a relacional.

Para o processo de detalhamento de um modelo lógico a partir de um modelo conceitual, é preciso aplicar determinadas regras de derivação. Como nosso objetivo é explorar a derivação de modelos relacionais, trabalharemos com as regras para o **modelo relacional**, segundo Elmasri e Navathe (2011).

No modelo lógico, além de definir as entidades, atributos e relacionamentos, temos também que identificar as **chaves primárias**, **chaves estrangeiras** e tratar os relacionamentos entre as entidades.

Então, veja agora com maiores detalhes esses conceitos.

Chave primária

Na agenda anterior você conheceu o conceito de Chave Primária. Vamos retomá-lo e aprofundá-lo:

Uma chave primária é um atributo que possui um valor único para cada entidade individual. Em outras palavras, **Chaves Primárias (Primary Keys ou "PK")**, sob o ponto de vista de um banco de dados são representadas por palavras que garantem a unicidade de uma entidade e nunca podem ser nulas. Esta é uma restrição que proíbe que duas entidades possuam o mesmo valor para um determinado atributo ao mesmo tempo, ou seja, elas nunca se repetem.

Alguns tipos de entidades podem ter **mais que um atributo formando uma chave** ou mais que um atributo chave. São as chamadas **chaves compostas**.

Para compreender bem o que é uma chave primária, vamos para um exemplo: pense num cadastro que você faz numa loja, solicitando o seu nome, endereço, telefone e CPF. Você consegue identificar quais desses dados (atributos) podem garantir que durante uma consulta no banco de dados da loja o resultado traga somente o seu cadastro?

Nome?

Bom, vamos pensar.... e se você se chamar João de Souza ou Maria da Silva? Pode acontecer de haver mais de um cliente com o mesmo nome, por ser mais comum, não pode? Então, nome não é um atributo que garante que a consulta retorne a um único registro, portanto, não poderá ser escolhido como chave primária.

Endereço?

Hummm, você e seu irmão podem ser clientes da loja e, portanto, o endereço pode ser o mesmo, não é? Então, o endereço também não é um atributo que garante que a consulta retorne a um único registro, portanto, da mesma forma não poderá ser escolhido como chave primária.

Telefone?

O mesmo problema do endereço! Você e o seu irmão poderão informar o mesmo telefone no cadastro. Então, telefone não é legal para ser escolhido como chave primária.

E o CPF?

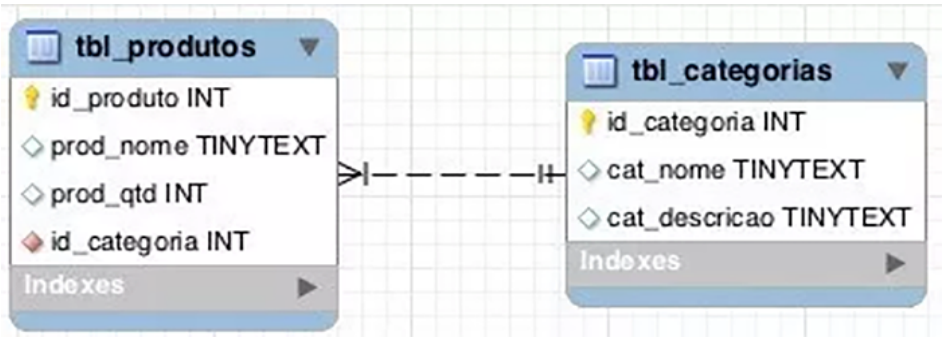
Já o Cadastro de Pessoa Física (CPF) **é único para cada cidadão brasileiro**, portanto, CPF pode ser escolhido como chave primária.

Na agenda anterior você viu que a chave primária pode ser representa pelo nome do campo sublinhado, certo? Você também pode representá-la por um círculo preenchido, veja:

O que é chave estrangeira?

E o conceito de Chave Estrangeira. Você imagina o que é?

Quando uma entidade apresenta um atributo **importado de outra entidade**, esse atributo é denominado **chave estrangeira**. As chaves estrangeiras são o resultado de associações entre entidades.



Imagine que Adriano queira cadastrar vários produtos que sejam de uma determinada categoria, por exemplo, vários chocolates que serão vendidos em seu mercado. Essa lista de produtos com seus atributos, no banco de dados, dá origem a uma tabela “Produtos” (tbl_produtos). **Cada entidade resulta em uma tabela própria** e, como a lista de categoria dos produtos também é uma entidade, tem seus próprios atributos e está organizada em outra tabela (tbl_categorias). Toda vez que Adriano for cadastrar um produto de chocolate, precisará indicar a **chave primária da tabela categoria que indique a categoria** a que o produto pertence (chocolates).

Tabela categoria	
Id_categoria (PK)	Prod_nome
001	chocolates
002	mercearia
003	Limpeza

O atributo que é chave primária (PK) na tabela "categoria dos produtos"...

Tabela produtos			
Id_produto	Prod_nome P	rod_qtd	Id_categoria (FK)
002345	Chocolate Gar 80g	300	001
002346	Caixa de bombom Lac 110g	20	001
002347	Arroz TI 5kg	50	003

...torna-se "chave estrangeira" (FK) quando é importada para outra entidade, no caso "produtos".

Ou seja, quando inserirmos um registro na tabela de produtos com o "id_categoria", essa chave primária da tabela "categorias" **será uma chave estrangeira (FK) dentro da tabela "produtos"**. É uma chave que vem de fora, de outra tabela, por isso o nome "estrangeira"[1].

Mas... porque é importante utilizar chaves estrangeiras?

Com a chave estrangeira, podemos facilitar as consultas e fazer cruzamentos de dados por meio destas referências. Dessa forma, uma chave estrangeira sempre será necessária quando houver o relacionamento entre duas tabelas.

Você verá exemplos de chave estrangeira nas regras de mapeamento!

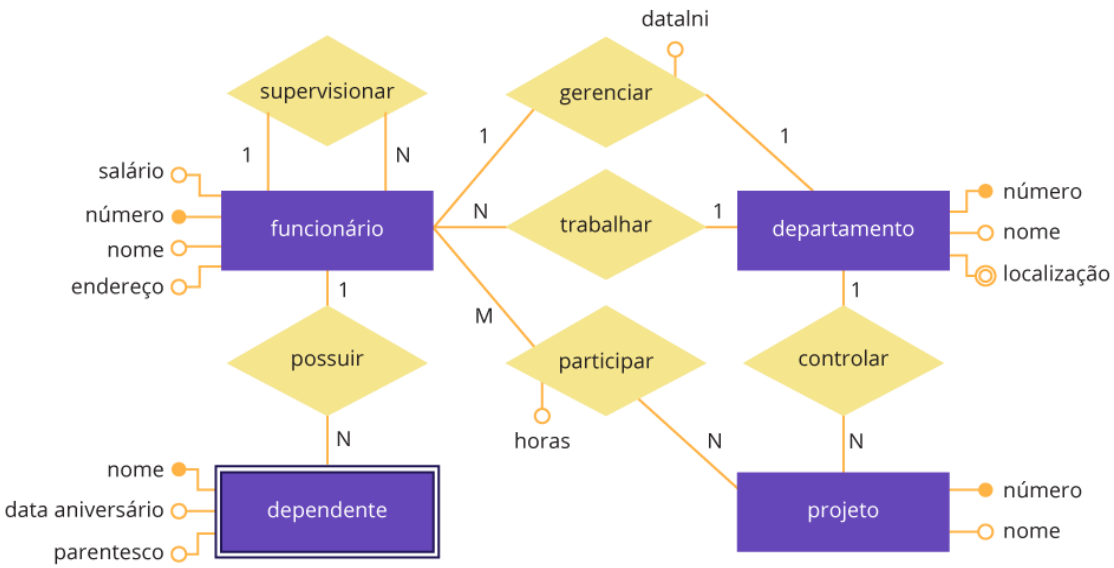
Regras de Derivação–Mapeamento

O mapeamento é um detalhamento do modelo lógico, feito a partir do Diagrama Entidade Relacionamento. Orientando-se pelas seis regras indicadas a seguir, você fará a leitura do DER e registrará as informações nele contidas em um outro formato, que já indica a quantidade e estrutura das tabelas do BD.

Uma vez que se consegue representar a semântica de uma aplicação por meio de modelos de dados, em especial pelo Modelo Entidade Relacionamento, é importante garantir que as informações contidas no MER sejam representadas corretamente no Modelo Relacional. Para garantir uma representação fiel das informações no Modelo Relacional é preciso seguir algumas diretrizes, com objetivo de **restringir as opções dos desenvolvedores em um conjunto efetivo de regras**.

Vamos ao mapeamento:

#0 | Algumas informações importantes para o mapeamento



Como você já sabe, o mapeamento é um detalhamento do modelo lógico, feito a partir do Diagrama Entidade Relacionamento. Isso quer dizer que você traduzirá o seu diagrama em outro formato, textual, mais próximo do que será utilizado no modelo físico.

No mapeamento, cada linha representa uma tabela. A sintaxe do modelo é grafada da seguinte maneira:

Entidade1 = {chave_primaria, demais_atributos, relacionamentos}

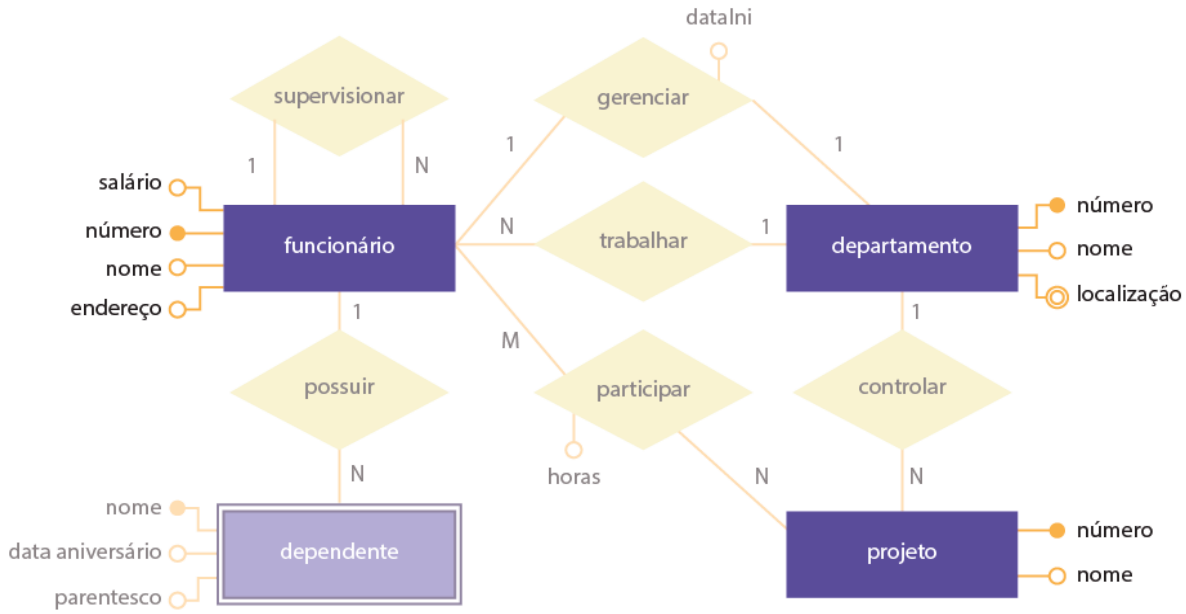
Entidade2 = {chave_primaria, demais_atributos, relacionamentos}

Você pode utilizar qualquer editor de texto para fazer o mapeamento.

No exemplo deste material, o diagrama representa o modelo lógico de um banco de dados que organiza as informações de pessoal, departamento e projeto de uma empresa.

Regra #1 | Mapear Conjuntos de Entidades Regulares

Identificar todas as entidades regulares (não fracas) e seus atributos:



Primeiro, você deverá localizar todas as entidades regulares. Cada uma dará origem a uma linha nova. Depois, escreverá seus atributos, com a nomenclatura devidamente padronizada.

Você se lembra que, na unidade 1, falou-se sobre a importância dessa normalização? Por padrão, adotamos um **prefixo** no nome dos atributos para identificar a qual entidade esse atributo pertence. Por isso, os atributos da entidade **Funcionário** iniciam-se sempre com a letra "**f**", da entidade; os atributos de **Departamento** se iniciam com a letra "**d**", da entidade; e os de **Projeto** iniciam com a letra "**p**".

Funcionário = {**f**número, **f**nome, **f**endereço, **f**salário}

Departamento = {**d**número, **d**nome}

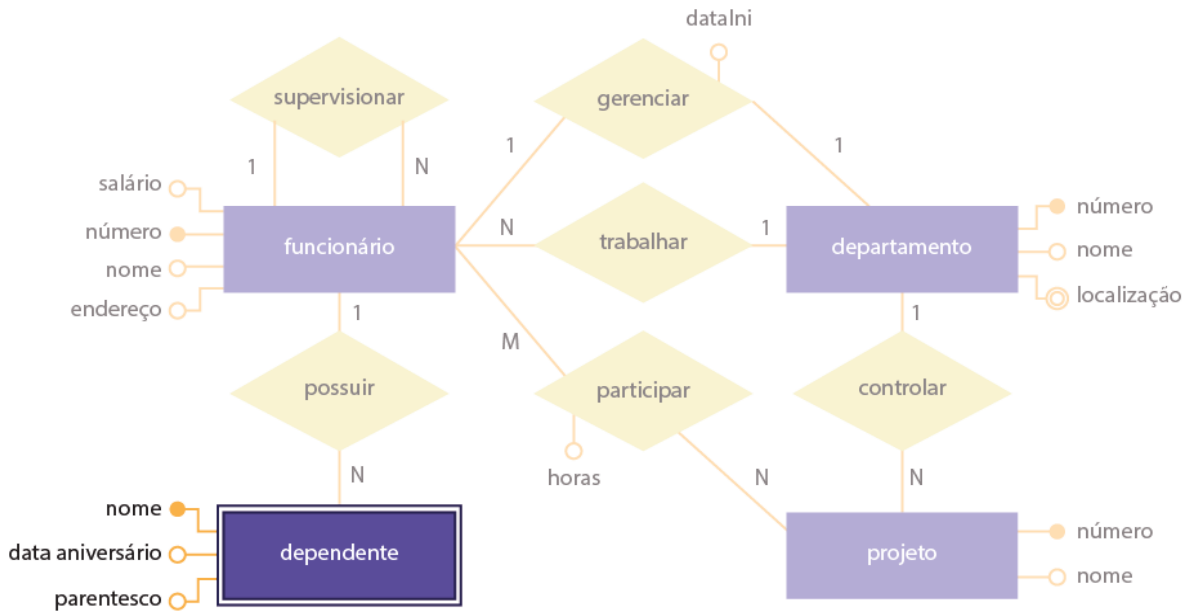
Projeto = {**p**número, **p**nome}

Você percebeu que o atributo **localização**, da entidade Departamento, não foi representado no mapeamento acima? Isso porque é um **atributo multivalorado** e será tratado na 6ª Regra.

A entidade **Dependente** também não está no nosso mapeamento: por ser uma entidade fraca, o caso será tratado na regra 2, a seguir.

Regra #2 | Mapear Conjuntos de Entidades Fracas

Relacionar todos os atributos da entidade fraca, acrescentando também o atributo chave da entidade da qual ela depende.

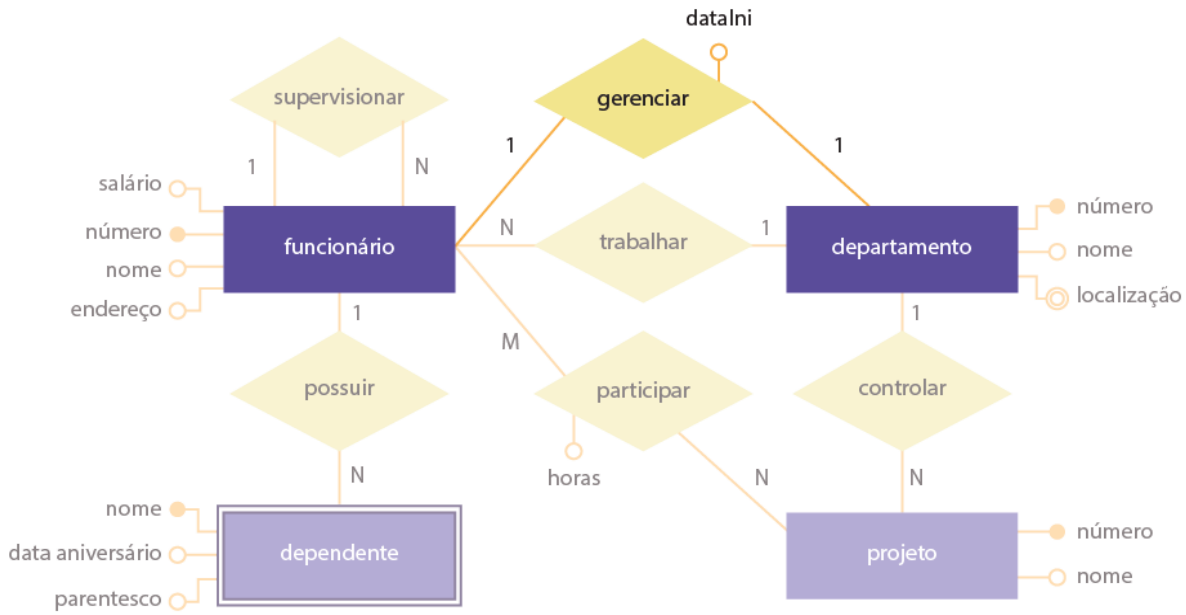


Agora, acrescente ao seu mapeamento a entidade fraca. Observe que o atributo chave da entidade Funcionário (fnumero), vai para a entidade Dependentes como um atributo chave! Portanto, a entidade Dependente terá uma **chave composta** pelos atributos **nome do dependente e número do funcionário** do qual depende.

Funcionário = {fnúmero, **f**nome, **f**endereço, **f**salário}
Departamento = {**d**número, **d**nome}
Projeto = {**p**número, **p**nome}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}

Regra #3 | Mapear Conjuntos de Relacionamentos Binários 1:1

No mapeamento dos relacionamentos 1:1, deve-se **escolher uma entidade para receber o atributo chave da outra entidade relacionada**.



Mas como escolher a entidade que receberá o atributo? Você sempre deve fazer uma pergunta envolvendo essas duas entidades relacionadas:

Todo funcionário gerencia um departamento?

Resposta: Não necessariamente!

Sabemos que nem todo o funcionário é gerente de um departamento, não é? Então, não é interessante que a entidade Funcionário receba a chave primária do departamento gerenciado.

Agora fazemos a pergunta ao contrário:

Todo o departamento é gerenciado por um funcionário?

Resposta: Sim, todo o departamento deve ter alguém responsável por ele.

Sendo assim, departamento receberá a chave primária do funcionário gerente.

Você percebeu que existe um **atributo no relacionamento**? O atributo **dataini** é a data em que o funcionário começou a gerenciar o departamento. Ele está representado no relacionamento porque não pertence apenas à entidade funcionário e nem somente à entidade departamento; mas, sim, ao relacionamento. Ou seja, ele existirá na relação funcionário gerencia departamento, indicando quando isso começou a acontecer. Neste caso, o mapeamento do relacionamento entre essas duas entidades ficará assim:

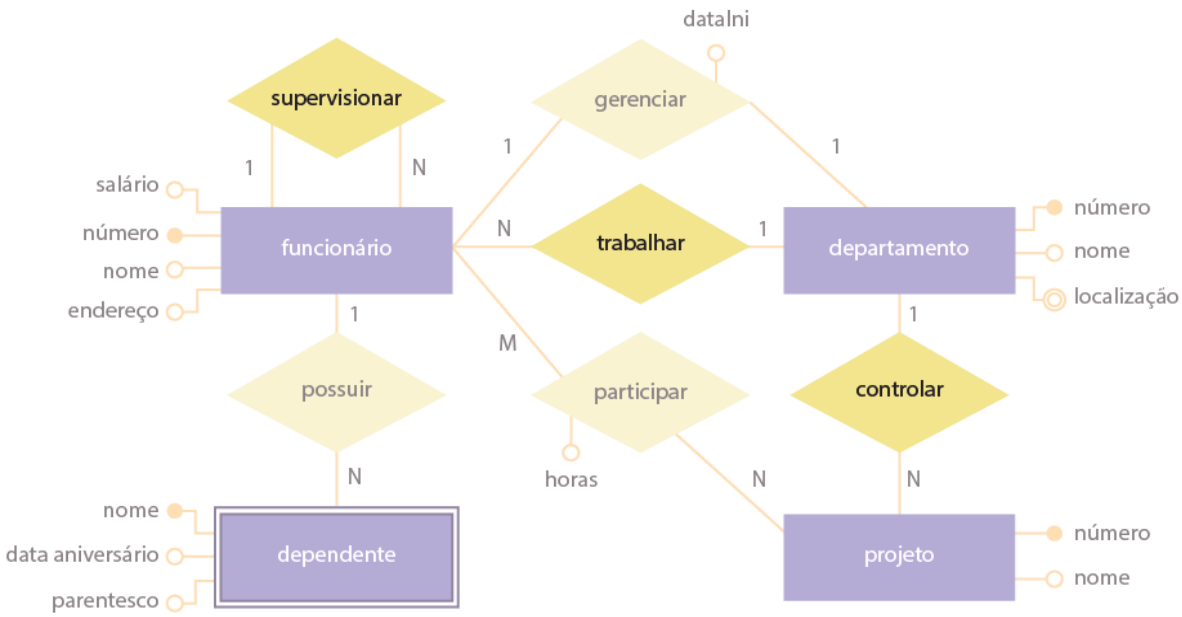
Funcionário = {fnúmero, **f**nome, **f**endereço, **f**salário}
Departamento = {**d**número, **d**nome, **f**numerogerente, **gerencia**dataini}
Projeto = {**p**número, **p**nome}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}

Você se lembra que todo atributo recebido por outra entidade é chamado de **Chave Estrangeira**? Portanto, **fnumerogerente** do empregado é a chave estrangeira da entidade **Departamento**.

Note que o atributo chave da uma entidade **Funcionário** vai para a entidade **Departamento** como um atributo simples!

Regra #4 | Mapear Conjuntos de Relacionamento Binário Regular 1:N

No mapeamento dos relacionamentos 1:N, deve-se acrescentar à entidade de lado N, o atributo chave da entidade de lado 1.



Neste passo, mapearemos os relacionamentos: **supervisionar, trabalhar e controlar**.

Esse relacionamento é um caso especial em que uma entidade se relaciona com si própria.

É chamado de relacionamento **recursivo** ou auto-relacionamento. É raro de ocorrer, mas sua utilização tem grande importância em alguns casos. No exemplo, na classe **funcionário** existe um

único funcionário que é **supervisor** dos demais funcionários. Assim temos uma relação 1:N.

Um funcionário supervisiona vários funcionários e os funcionários são supervisionados por um único supervisor.

Sendo assim, o atributo **SuperNúmero** é adicionado na entidade **Funcionário**:

Funcionário = {**f**número, **f**nome, **f**endereço, **f**salário, **super**numero}
Departamento = {**d**número, **d**nome, **f**numerogerente, **gerencia**dataini}
Projeto = {**p**número, **p**nome}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}

- **Avaliando o relacionamento supervisionar;**

O mesmo acontece com o relacionamento **trabalhar**: a entidade de lado N da relação recebe a chave primária do atributo chave da entidade de lado 1. Sendo assim, o atributo **DNumero** deve ser adicionado à entidade **Funcionário**. Ficando representado dessa forma:

Funcionário = {**f**número, **f**nome, **f**endereço, **f**salário, **super**numero**, d**numero}
Departamento = {**d**número, **d**nome, **f**numerogerente, **gerencia**dataini}
Projeto = {**p**número, **p**nome}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}

- **Avaliando o relacionamento trabalhar;**

Da mesma forma, no relacionamento **controlar**, das entidades **Departamento** e **Projeto**, o lado N da relação (a entidade **Projeto**) recebe a chave primária do lado 1 do relacionamento, no caso (ou seja, da entidade **Departamento**):

Funcionário = {**f**número, **f**nome, **f**endereço, **f**salário, **super**numero**, d**numero}
Departamento = {**d**número, **d**nome, **f**numerogerente, **gerencia**dataini}
Projeto = {**p**número, **p**nome, **d**numero}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}

Note que os atributos chave das entidades do lado 1 foram para as entidades chave do lado N como atributos simples!

- **Avaliando o relacionamento controlar.**

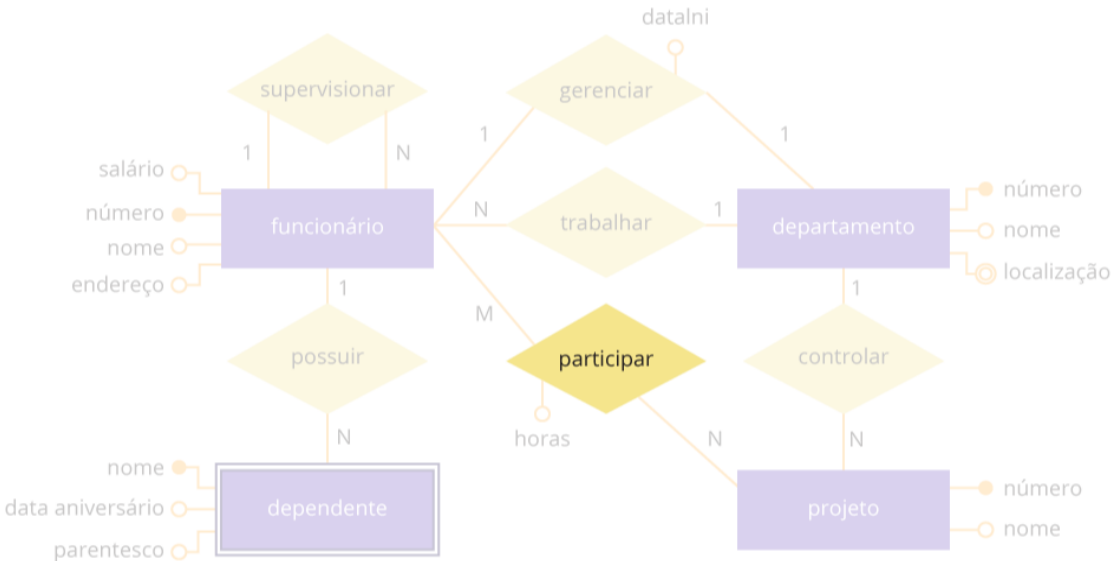
Da mesma forma, no relacionamento **controlar**, das entidades **Departamento** e **Projeto**, o lado N da relação (a entidade **Projeto**) recebe a chave primária do lado 1 do relacionamento, no caso (ou seja, da entidade **Departamento**):

Funcionário = {**f**número, **f**nome, **f**endereço, **f**salário, **super**numero**, d**numero}
Departamento = {**d**número, **d**nome, **f**numerogerente, **gerencia**dataini}
Projeto = {**p**número, **p**nome, **d**numero}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}

Note que os atributos chave das entidades do lado 1 foram para as entidades chave do lado N como atributos simples!

Regra #5 | Mapear Relacionamento Binário M:N

O relacionamento binário envolve dois conjuntos de entidades. A maior parte dos conjuntos de relacionamentos modelados em um BD é do tipo binário. Para cada relacionamento binário M:N cria-se uma relação. Os atributos da relação são os **atributos do conjunto de relacionamento, juntamente com os atributos chave das relações que mapeiam os conjuntos de entidades envolvidas**. A chave da relação é a concatenação dos atributos chave das relações que mapeiam os conjuntos de entidades envolvidos.



Observe o relacionamento **Funcionário "participa" de Projeto**. Nesse caso, um funcionário pode participar de vários projetos e um projeto pode ter a participação de vários funcionários, portanto, dizemos que o relacionamento é N:M.

Assim, devemos **criar uma nova entidade**, a partir do relacionamento **Participar**, adotando como chave primária a **concatenação** da chave primária de **Funcionário** e da chave primária de **Projeto**, acrescentando-se o(s) atributo(s) que existir(em) no relacionamento:

Funcionário = {**f**número, **f**nome, **f**endereço, **f**salário, **super**numero**, d**numero}
Departamento = {**d**número, **d**nome, **f**numerogerente, **gerencia**dataini}
Projeto = {**p**número, **p**nome, **d**numero}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}
Participar = {**f**num, **p**num, horas}

Por que isso? Porque para você conseguir cadastrar um mesmo funcionário em vários projetos, deverá fazer um registro para cada projeto do qual o funcionário participa, vinculando o número de horas a serem trabalhadas. Veja o exemplo:

MARIA trabalha no projeto de desenvolvimento do sistema **MAP** com carga horária de **20** horas semanais.

MARIA trabalha no projeto de desenvolvimento do sistema **GIGA** com carga horária de **10** horas semanais.

MARIA trabalha no projeto de desenvolvimento do sistema **EMPLACAR** com carga horária de **14** horas semanais.

Da mesma forma:

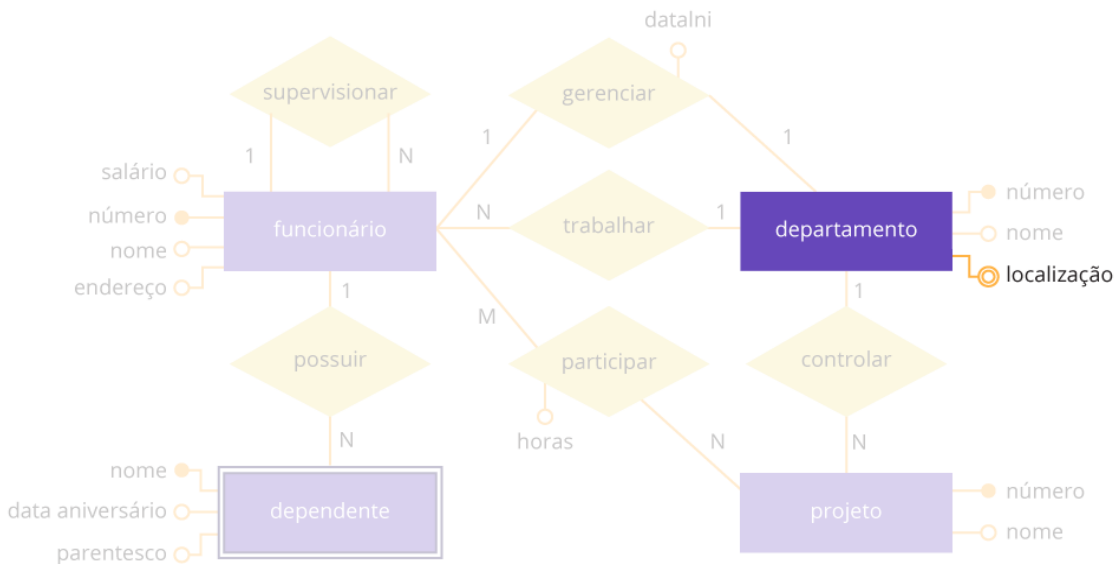
JOÃO trabalha no projeto de desenvolvimento do sistema **MAP** com carga horária de **20** horas.

JOÃO trabalha no projeto de desenvolvimento do sistema **EMPLACAR** com carga horária de **22** horas.

Perceba que você poderá repetir o número da matrícula da Maria para vários números de projetos diferentes e poderá repetir os números de projetos para vários funcionários diferentes. A única coisa que você não poderá fazer é **cadastrar duas vezes o mesmo funcionário para o mesmo projeto**, não é? Ora, e como você poderá fazer para que isso não aconteça? A única forma é **colocar os dois atributos como chave primária** (**número da matrícula do funcionário e número do projeto**).

Regra #6 | Mapear Atributos Multivalorados

A 6ª regra identifica os atributos multivalorados, ou seja, aqueles que possuem um ou mais valores para o mesmo atributo.



Para este caso, deve-se criar uma **nova entidade**, inserindo o atributo multivalorado e o atributo chave da entidade a qual ele pertence. **Ambos os atributos devem ser chaves primárias, compondo uma chave concatenada**.

Funcionário = {**f**número, **f**nome, **f**endereço, **f**salário, **super**numero**, d**numero}
Departamento = {**d**número, **d**nome, **f**numerogerente, **gerencia**dataini}
Projeto = {**p**número, **p**nome, **d**numero}
Dependente = {**depend**nome, **f**número, **depend**dataniver, **depend**parentesco}
Participar = {**f**num, **p**num, horas}
LocalDep = {**d**número, localização}

Entendeu o motivo?

Imagine que um departamento de uma universidade, por exemplo, esteja localizado em vários polos, em cidades diferentes. Para que não tenha o problema de cadastrar o mesmo polo, em uma mesma localização, mais de uma vez, definimos os dois atributos como chave. Veja um exemplo:

LocalDep: (**d**número, **d**localização)

Na tabela **LocalDep**, poderiam constar os seguintes dados:

- Departamento número 10, local São Paulo.
- Departamento número 10, local Campinas.
- Departamento número 10, local Jundiaí.
- Departamento número 20, local São Paulo.
- Departamento número 20, local Curitiba.

Perceba que apenas podemos dizer que um mesmo departamento tem polos em localidades diferentes se o **número do departamento** e o **local** forem definidos como chave. Observe que dessa forma não é permitido cadastrar um mesmo departamento em um mesmo local mais de uma vez, evitando duplicidade de registros.

Entendeu agora o porquê temos que definir o número do departamento e a sua localização como chave?

Mapeamento feito!

Sendo assim, o mapeamento do Modelo ER representado ficou dessa forma:

Funcionário = {fnúmero, fnome, fendereço, fsalário, supernúmero, dnúmero}

Regra #1 Regra #4

Departamento = {dnúmero, dnome, fnúmerogerente, gerenciadataini}

Regra #1 Regra #3

Projeto = {pnúmero, pnome, dnúmero}

Regra #1 Regra #4

Dependente = {dependnome, fnúmero, dependdataniver, dependparentesco}

Regra #2

Participar = {fnum, pnum, horas}

Regra #5

LocalDep = {dnúmero, localização}

Regra #6

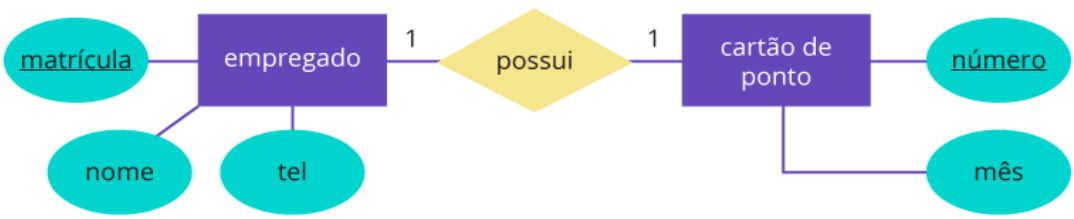
Com esse detalhamento em mãos, você conseguirá partir para o modelo físico!

Praticando o mapeamento do diagrama ER

Agora que você já conhece as seis regras do mapeamento, é hora de praticá-las.

Procure fazer o mapeamento dos três diagramas ER a seguir, cada qual pensado para um BD diferente.

Situação A



Relacionamento 1:1, entre empregado e cartão de ponto:

Um empregado possui um único cartão de ponto e um cartão de ponto pertence a apenas um empregado.

Para escolher qual entidade deverá receber o atributo da outra entidade, devemos sempre fazer a seguinte pergunta:

Todo empregado possui um cartão de ponto?

Resposta: Sim!

Por isso, a entidade cartão de ponto poderá receber a chave primária de empregado.

Agora fazemos a pergunta ao contrário:

Todo o cartão de ponto pertence a um empregado?

Resposta: Sim, pois se existe o cartão de ponto é sinal que pertence a um empregado da empresa, não é?

Sendo assim, a entidade cartão de ponto também poderá receber a chave primária de empregado.

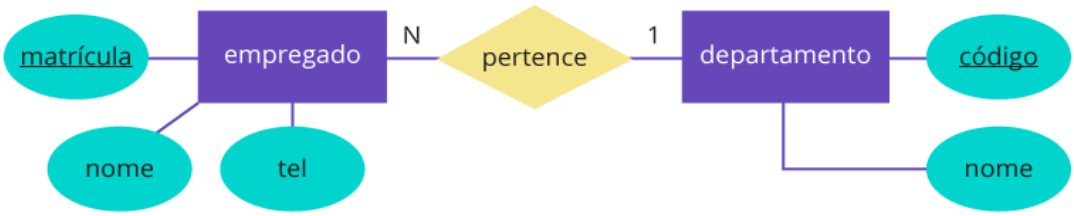
Em casos desse tipo, você pode escolher qual entidade receberá a chave primária da outra. Vamos, então, levar o atributo chave da entidade empregado para a entidade cartão de ponto. Nesse caso, o mapeamento do relacionamento entre essas duas entidades ficará assim:

Empregado – EmpMatrícula, EmpNome, EmpTelefone

Cartão de Ponto – CpNúmero, CpMês, EmpMatrícula

Dizemos que o atributo recebido pela outra entidade é chamado de **Chave Estrangeira**. Portanto, matrícula do empregado (**EmpMatrícula**) é a chave estrangeira de cartão de pontos.

Situação B



Relacionamento 1:N (“um para N ou 1 para Muitos”)

Um empregado pertence a um único departamento, mas um departamento possui muitos empregados.

Em um relacionamento 1:N, o lado N sempre recebe a chave primária do lado 1. Para não esquecer, pense sempre nisso: muitos (N) são mais fortes do que 1 e por isso o lado N rouba a chave primária do lado 1! É uma brincadeirinha, mas ajuda a não errar!

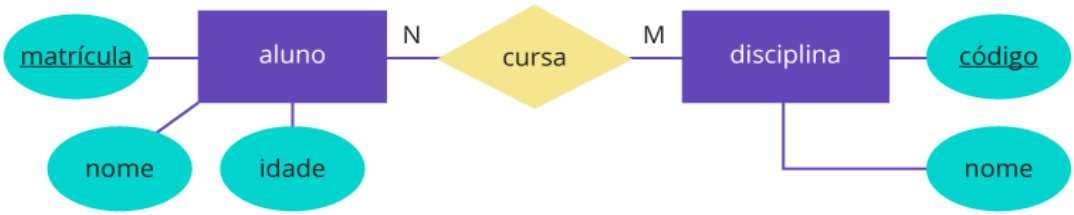
Nesse caso, o mapeamento do relacionamento entre essas duas entidades ficará assim:

Empregado – EmpMatrícula, EmpNome, EmpTelefone, DepCodigo

Departamento – DepCodigo, DepNome

Dizemos que o atributo recebido pela outra entidade é chamado de **Chave Estrangeira**. Portanto, código do departamento (**DepCodigo**) é a chave estrangeira da entidade empregado.

Situação C



Relacionamento N:M (“N para M ou Muitos para Muitos”)

Um aluno pode cursar mais de uma disciplina e uma disciplina pode ser cursada por mais de um aluno.

Para cada relacionamento binário M:N, cria-se uma nova relação.

Os atributos da relação são os atributos do conjunto de relacionamento juntamente com os atributos chave das relações que mapeiam os conjuntos de entidades envolvidos. A chave da relação é a concatenação dos atributos chave das relações que mapeiam os conjuntos de entidades envolvidos.

Por que isso? Veja que para conseguir cadastrar um mesmo aluno em várias disciplinas, deverá ser feito um registro para cada uma das disciplinas que o aluno cursa, concorda? Exemplo:

- Maria cursa matemática
- Maria cursa português
- Maria cursa história
-

E assim por diante.... Da mesma forma:

- João cursa matemática
- João cursa português
- João cursa história
-

Perceba que você poderá repetir a matrícula da Maria para vários códigos de disciplinas diferentes e poderá repetir os códigos das disciplinas para vários alunos diferentes. O que você não poderá fazer é cadastrar duas vezes o mesmo aluno para a mesma disciplina, não é?

Ora, e como você poderá fazer para que isso não aconteça?

A única forma é **colocar os dois atributos como chave primária** (**matrícula do aluno** e **código da disciplina**). Concorda?

Então, a terceira entidade criada pelo relacionamento N:M terá como chave primária tanto a chave de **aluno** quanto a chave de **disciplina**. Veja como ficará o mapeamento:

Aluno – AlMatrícula, AlNome, AlIdade

Disciplina – DiscCodigo, DiscNome

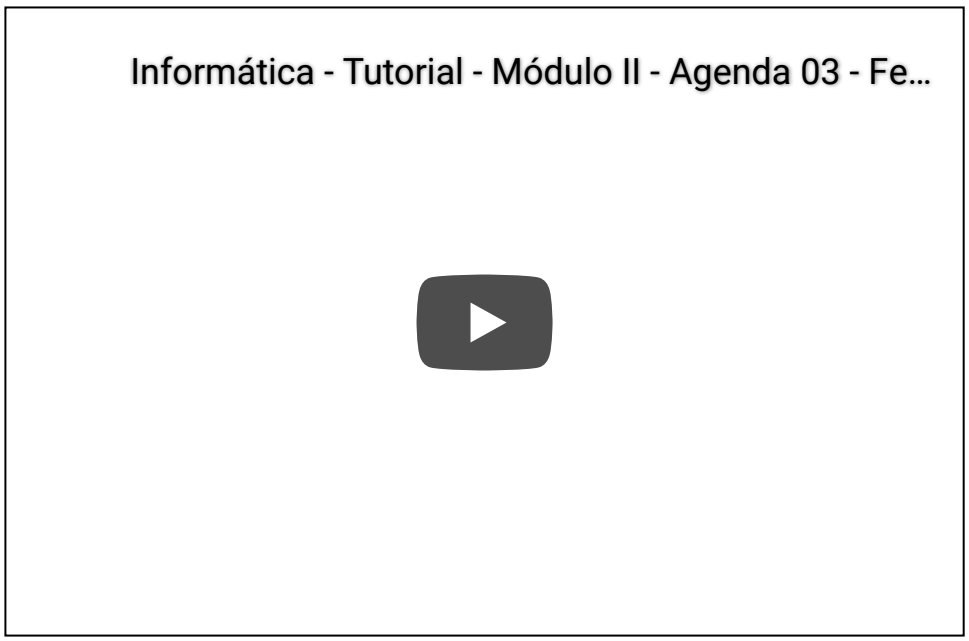
Curso – CurMatriculaAluno, CurCodDisciplina

Dizemos que os atributos da classe criada pelo relacionamento aluno cursa disciplina é uma concatenação (junção) das chaves primárias de aluno e disciplina.

Ferramenta Case BR Modelo para modelo lógico

Até agora você utilizou softwares editores de texto para preparar seus diagramas ER, correto? Saiba que existem ferramentas que podem auxiliar na construção desse tipo de representação gráfica, caso do **Case BR Modelo**. O tutorial a seguir pode ser de grande ajuda, caso você deseje utilizá-la:

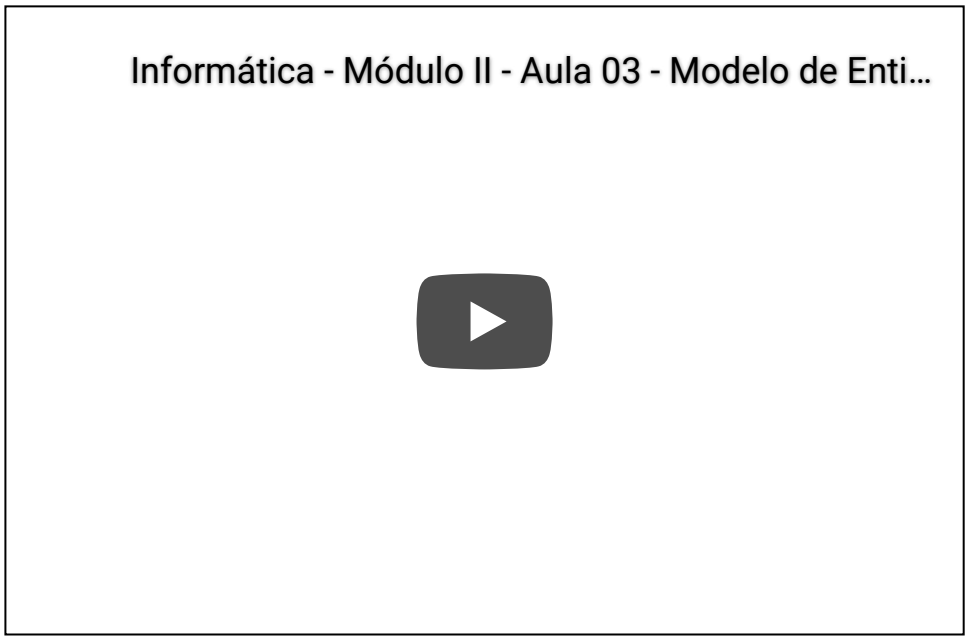
No video a seguir, você irá retormar alguns conceitos importantes já estudados.

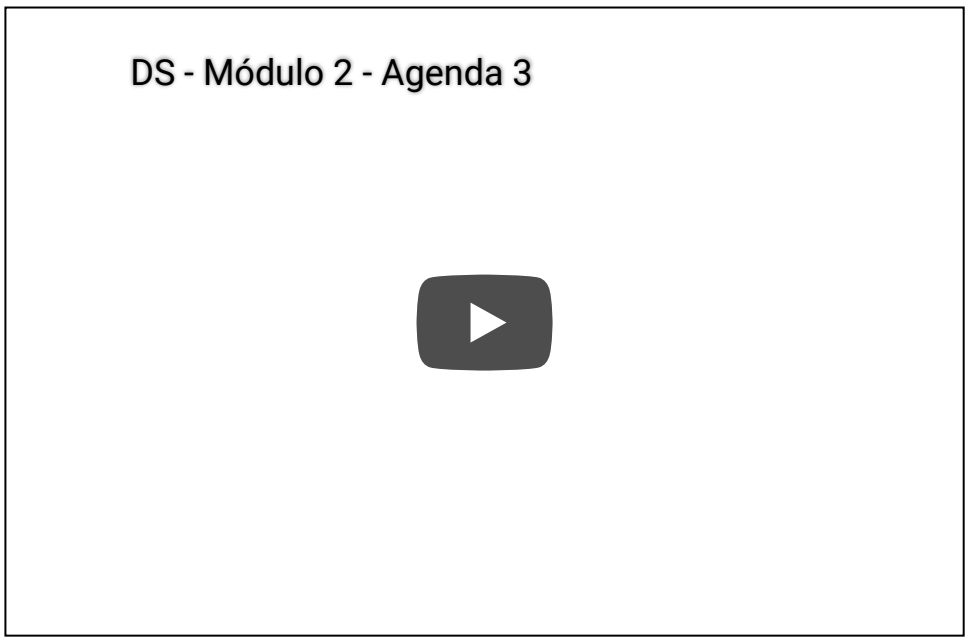


O próximo passo é **implementar** o banco de dados, ou seja, desenvolver o **modelo físico**, também conhecido como projeto físico. Neste curso, trabalharemos com o **modelo relacional** e o SGBD será o MySQL. O importante agora é instalarmos o Sistema Gerenciador de Banco de Dados. Vamos lá?

Recapitulando...

Esta foi uma agenda intensa, com diversos conceitos novos! Para sintetizar e retomar os conceitos estudados, assista às videoaulas





[1]Adaptado de: https://www.diegomacedo.com.br/entendendo-as-chaves-dos-bancos-de-dados/

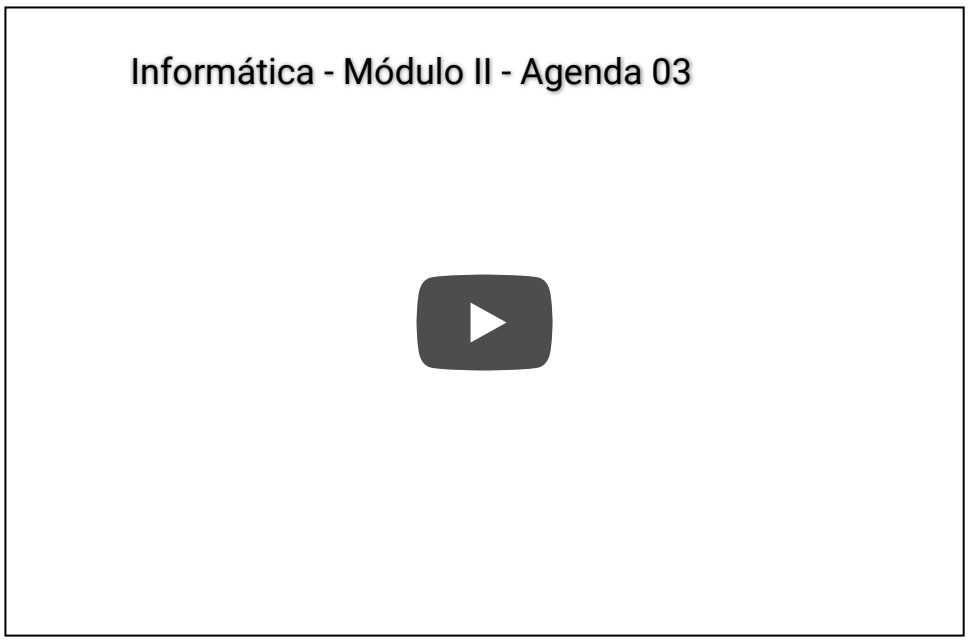
Ampliando Horizontes

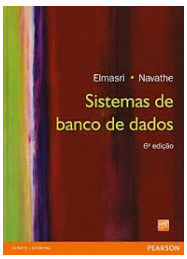
Onde encontro mais informações sobre modelos lógicos e físicos?

Caso deseje complementar seus estudos, recomendamos a consulta aos seguintes materiais:

Modelo Entidade-Relacionamento – Parte II

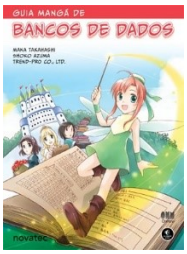
Nesta videoaula, o professor Rogério Silva continua o detalhamento do Modelo ER, retomando e aprofundando aspectos que você viu nessa agenda.

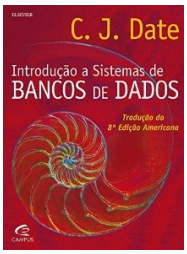




Sistemas de bancos de dados. ELMASRI, R.; NAVATHE, S. B. 6ª edição. Editora Pearson, 2012.

Guia mangá de banco de dados. TAKAHASHI, M.; AZUMA, S. 2ª edição. Editora Novatec, 2011.





Introdução a Sistema de Banco de Dados. DATE, C. J. 8ª edição. Editora Campus, 2013.

Resumindo o Estudo

Nesta agenda, além de aprofundar os conceitos de chave primária e chave estrangeira, você aprendeu as regras para **mapeamento** e para **criação do modelo físico**. Com o exemplo do mercado de Adriano, você compreendeu as etapas necessárias para que um banco de dados seja criado.

Na próxima agenda você aprofundará seus estudos na construção do Projeto Físico, utilizando o Sistema Gerenciador de Banco de Dados (SGBD) MySQL.

Até lá!