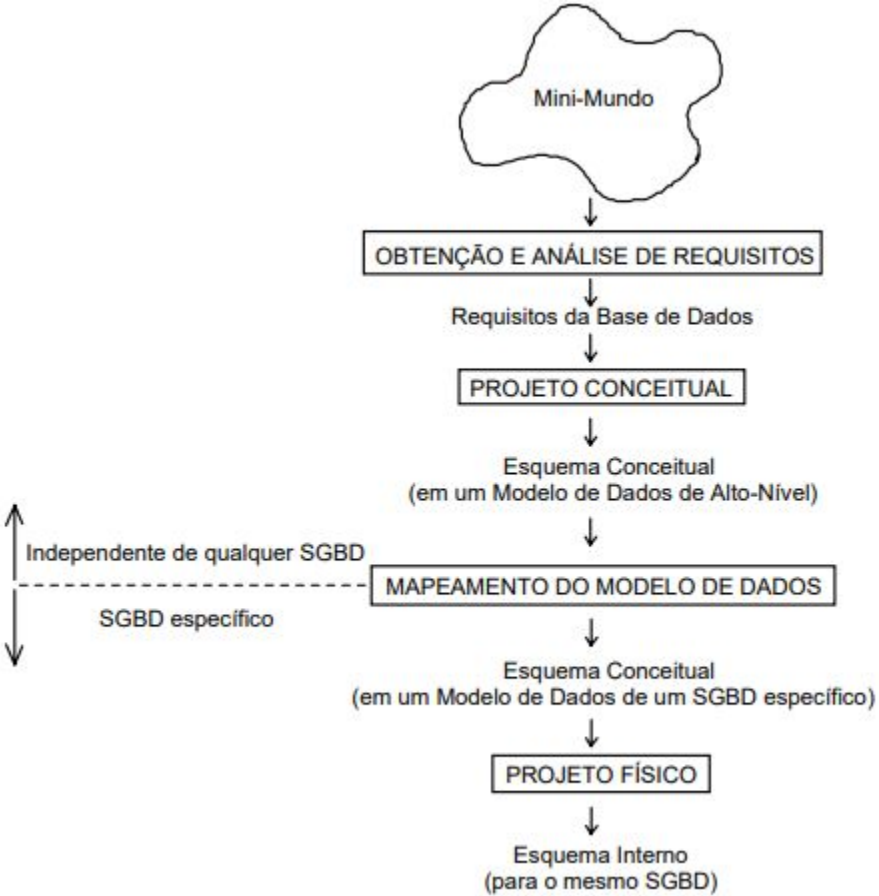
UVV

Design e Desenvolvimento de Banco de Dados I

Modelo lógico

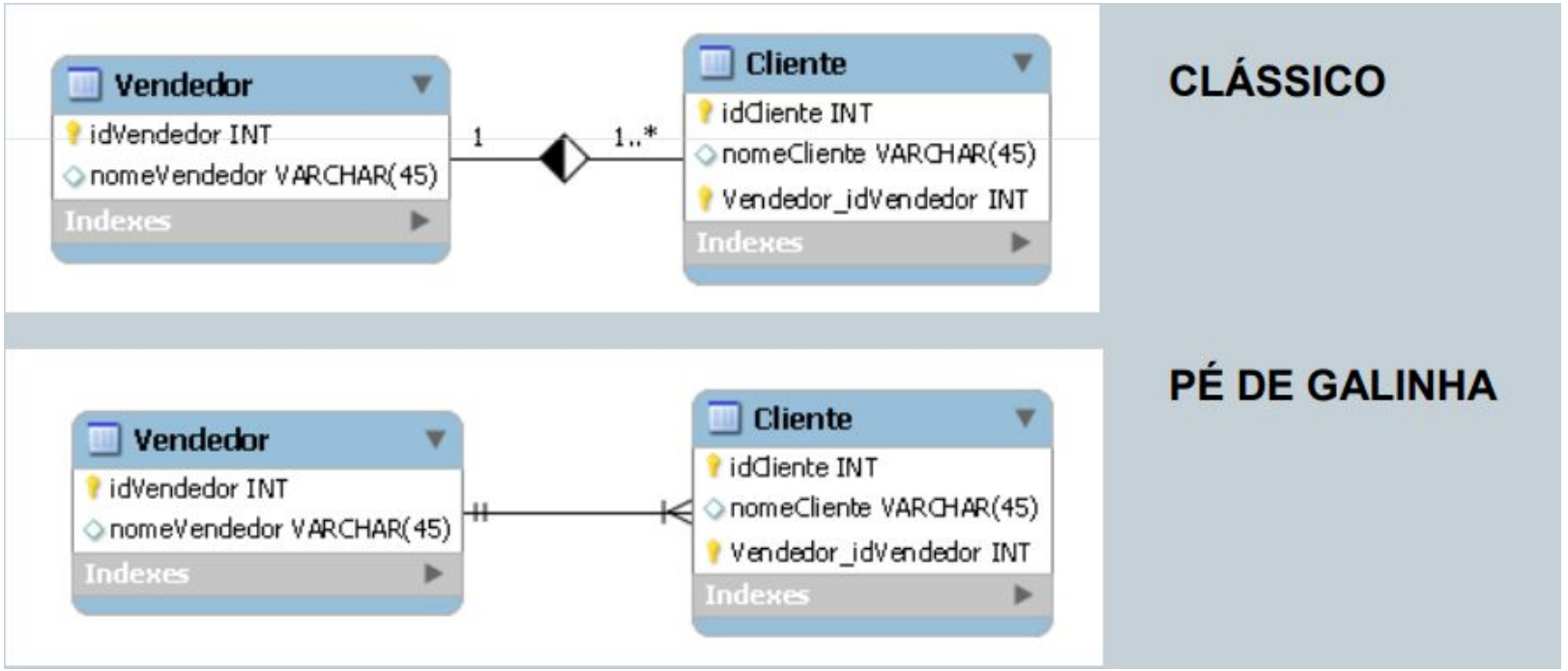
Prof. Me Renato Sousa Botacim
Email: renato.botacim@uvv.br

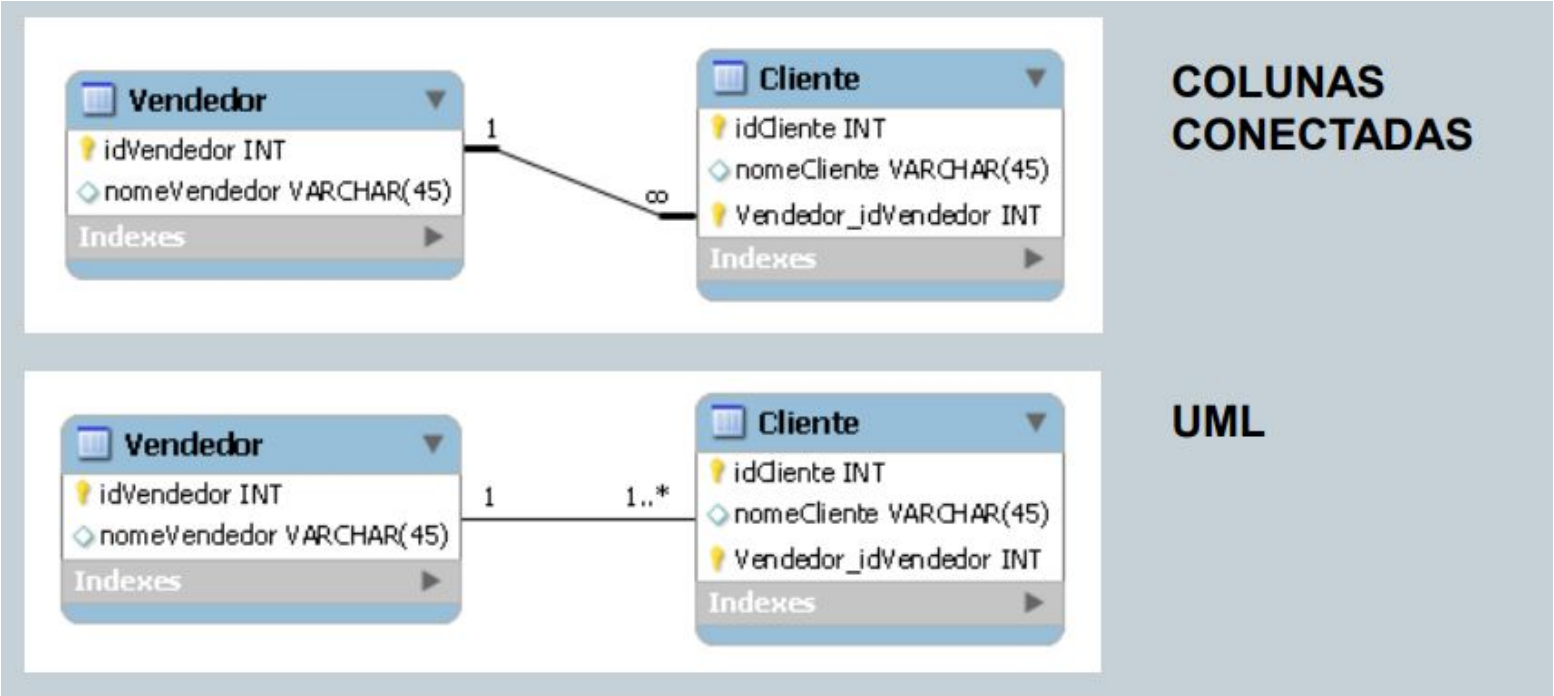

Mini-Mundo
OBTENÇÃO E ANÁLISE DE REQUISITOS
Requisitos da Base de Dados
PROJETO CONCEITUAL
Esquema Conceitual
(em um Modelo de Dados de Alto-Nível)
Independente de qualquer SGBD
MAPEAMENTO DO MODELO DE DADOS
SGBD específico
Esquema Conceitual
(em um Modelo de Dados de um SGBD específico)
PROJETO FÍSICO
Esquema Interno
(para o mesmo SGBD)

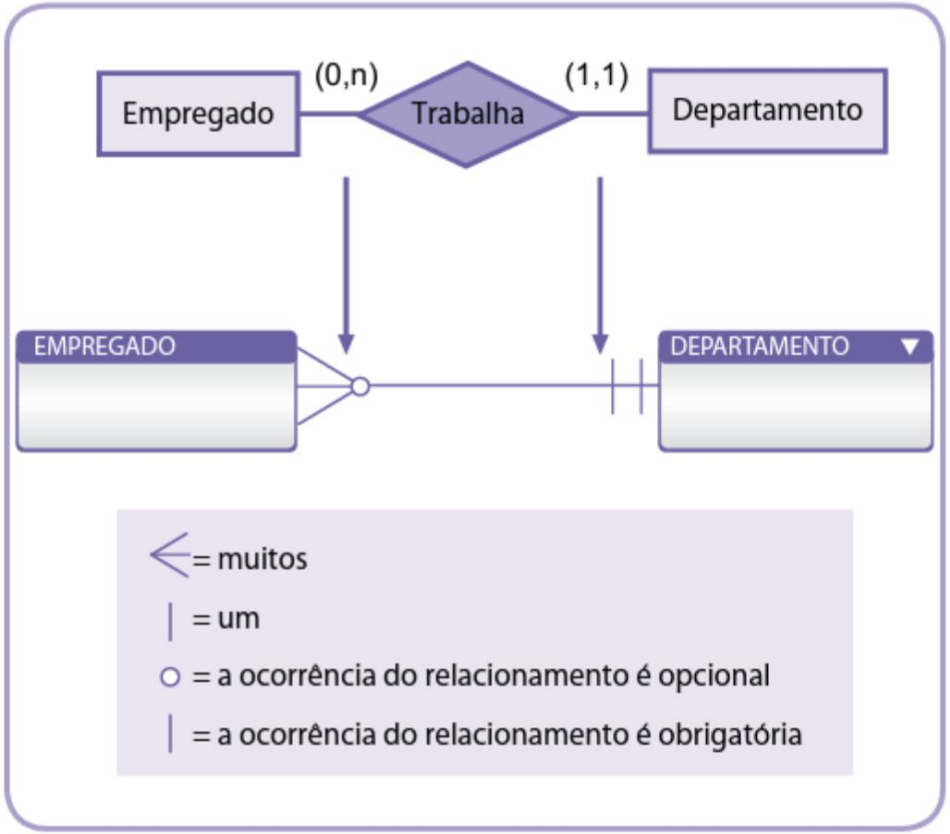
Modelo Lógico

Compreende uma descrição das estruturas que serão armazenadas no banco e que resulta numa representação gráfica dos dados de uma maneira lógica, inclusive nomeando os componentes e ações que exercem uns sobre os outros.

Existem várias notações para o modelo lógico. Em nossa disciplina, vamos utilizar a notação de James Martin (Engenharia de Informações) que foi criada em 1980 e é uma das mais utilizadas no mercado de trabalho.


Vendedor
idVendedor INT
nomeVendedor VARCHAR(45)
Indexes
1
1..*
Cliente
idCliente INT
nomeCliente VARCHAR(45)
Vendedor_idVendedor INT
Indexes
CLÁSSICO
Vendedor
idVendedor INT
nomeVendedor VARCHAR(45)
Indexes
Cliente
idCliente INT
nomeCliente VARCHAR(45)
Vendedor_idVendedor INT
Indexes
PÉ DE GALINHA


Vendedor
idVendedor INT
nomeVendedor VARCHAR(45)
Indexes
1
∞
Cliente
idCliente INT
nomeCliente VARCHAR(45)
Vendedor_idVendedor INT
Indexes
COLUNAS CONECTADAS
Vendedor
idVendedor INT
nomeVendedor VARCHAR(45)
Indexes
1
1..*
Cliente
idCliente INT
nomeCliente VARCHAR(45)
Vendedor_idVendedor INT
Indexes
UML


Empregado
(0,n)
Trabalha
(1,1)
Departamento
EMPREGADO
DEPARTAMENTO
= muitos
= um
= a ocorrência do relacionamento é opcional
= a ocorrência do relacionamento é obrigatória

Os relacionamentos já não são mais nomeados. Só será necessário nomear um relacionamento quando houver mais de um relacionamento entre duas tabelas.

A grande vantagem dessa proposta é que, a partir do modelo conceitual gerado, poderemos aplicar regras predefinidas em função da tecnologia a ser empregada e, assim, obter os modelos necessários.

Isso fará com que, a partir de um mesmo modelo conceitual, possamos gerar modelos lógicos para bancos de dados baseados na abordagem “relacional” ou até mesmo em novas abordagens, tais como bancos de dados orientados a objetos.

Assim, podemos definir que o processo de obtenção de um modelo lógico a partir de um modelo conceitual segue os seguintes passos, segundo Cougo (1997):

1. obter o modelo conceitual;
2 definir o tipo de implementação (relacional, O-O);
3. aplicar as regras de derivação específicas;
4. adaptar o modelo às necessidades.

REGRAS DE DERIVAÇÃO

A obtenção de um modelo lógico deveria ser feita a partir de um modelo conceitual previamente gerado.

Etapa 1: Mapeamento de tipos de entidade regular

Para cada tipo de entidade regular (forte) “E” no modelo conceitual, crie uma tabela “T” que inclua todos os atributos simples de “E”.

Inclua apenas os atributos de componente simples de um atributo composto.

Escolha um dos atributos chave de “E” como chave primária para “T”.

Se a chave escolhida de “E” for composta, então o conjunto de atributos simples que a compõem juntos formarão a chave primária de “T”.

Se várias chaves fossem identificadas para “E” durante o projeto conceitual, a informação que descreve os atributos que formam cada chave adicional é mantida a m de especificar chaves secundárias (únicas) da tabela “T”.

O conhecimento sobre as chaves também é mantido para ns de indexação e outros tipos de análises.

Etapa 2: Mapeamento de tipos de entidade fraca

Para cada tipo de entidade fraca “F” no modelo conceitual: com tipo de entidade proprietária forte “E”, crie uma tabela “T” e inclua todos os atributos simples (ou componentes simples dos atributos compostos) de “F” como atributos de “T”.

Além disso, inclua como atributos chave estrangeira de “T” os atributos de chave primária da tabela que correspondem aos de entidade proprietária “E”.

Isso consegue mapear o tipo de relacionamento de identificação de “F”.

A chave primária de “T” é a combinação das chaves primárias do proprietário “E” e a chave parcial do tipo de entidade fraca “F”, se houver.

Etapa 3: Mapeamento dos tipos de relacionamento binários 1:1

- já criamos duas tabelas ("A" e "B"), uma para cada conjunto de entidades que participa do relacionamento;
- delas, incluir como chave estrangeira a chave primária da outra;
- se o relacionamento for total em um dos dois conjuntos de entidades, então é melhor incluir a chave estrangeira no "lado total"

Etapa 4: Mapeamento de tipos de relacionamento binário 1:N

Para cada tipo de relacionamento “R” binário regular 1:N, identifique a tabela “T” que representa o tipo de entidade participante no lado N do tipo de relacionamento.

Inclua como chave estrangeira em “T” a chave primária da tabela “P” que representa o outro tipo de entidade no lado N (tabela “T”) está relacionada, no máximo, a uma instância de entidade do lado 1 (tabela “P”) do tipo de relacionamento.

Inclua quaisquer atributos simples (ou componentes simples dos atributos compostos) do tipo de relacionamento 1:N como atributos de “P”.

- já temos duas tabelas ("T","P"), uma para cada conjunto de entidades que participa do relacionamento;
- incluir como chave estrangeira, na tabela do "lado muitos" (o "lado N"), a chave primária da tabela do "lado um".

Etapa 5: Mapeamento de tipos de relacionamento binário N:N

Para cada tipo de relacionamento “R” binário N:N, crie uma nova tabela “S” para representar “R”.

Inclua como atributos de chave estrangeira em “S” as chaves primárias das tabelas que representam os tipos de entidade também, quaisquer atributos simples do tipo de relacionamento N:N (ou componentes simples dos atributos compostos) como atributos de “S”.

Observe que não podemos representar um tipo de relacionamento N:N por um único atributo de chave estrangeira em uma das tabelas participantes (como zemos para os tipos de relacionamento 1:1 ou 1:N) devido à razão de cardinalidade, por isso temos de criar uma tabela de relacionamento “S” separada.

Complementando:

- já temos duas tabelas, uma para cada conjunto de entidades que participa do relacionamento;
- criar uma nova tabela contendo, como chaves estrangeiras, as chaves primárias dessas duas tabelas;
- a combinação dessas chaves estrangeiras forma a chave primária da nova tabela;
- incluir também, se houver, colunas com os atributos do relacionamento

Etapa 6: Mapeamento de atributos multivalorados

Para cada atributo multivalorado “A” na tabela “S”, crie uma tabela “T”. Essa tabela “T“ incluirá um atributo correspondente a “A”, mais o atributo da chave primária da tabela “S”.

A chave primária de “T” é a combinação do atributo da chave primária da tabela “S” (esse atributo também será chave estrangeira) e o atributo “A”.

Se o atributo multivalorado for composto, incluímos seus componentes simples.

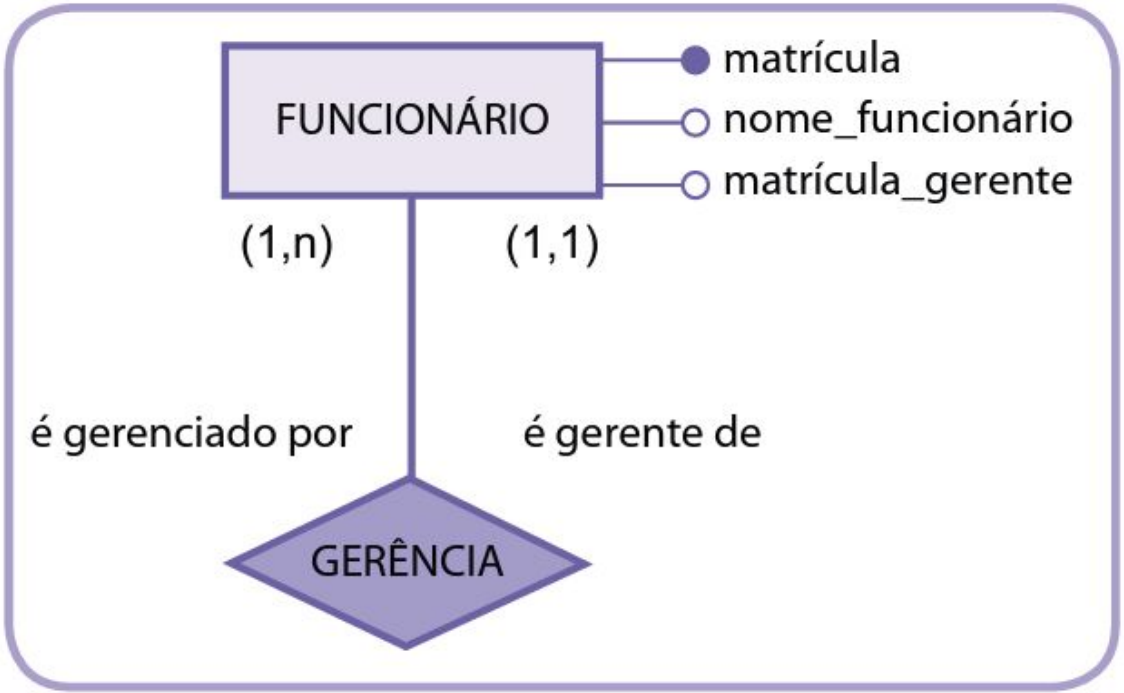
Para cada atributo multivalorado, criar uma tabela contendo:

- como chave estrangeira, a chave primária da tabela que possuía o atributo multivalorado mais o próprio atributo que era multivalorado;
- a nova chave primária também será uma chave estrangeira para a tabela que possuía o atributo;
- a chave primária da nova tabela é a combinação da chave estrangeira e do valor do atributo;
- se o atributo multivalorado for composto, incluímos seus componentes simples.

ETAPA 7: MAPEAMENTO DE AUTO RELACIONAMENTO 1:N

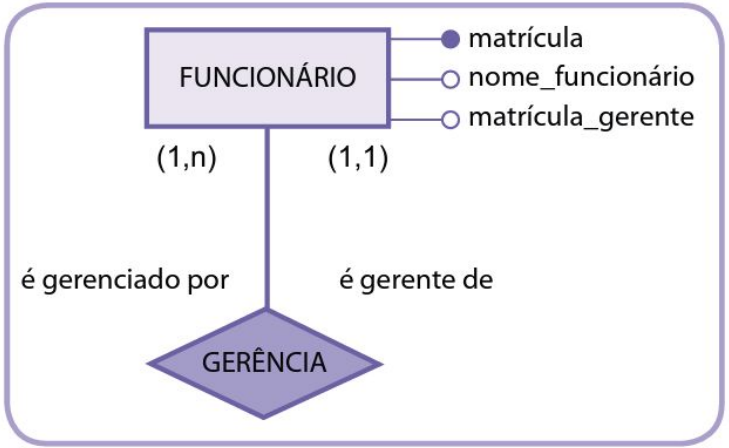
Para os auto relacionamentos 1:N, como existe um relacionamento entre a mesma entidade, deve se criar a chave estrangeira na própria tabela, no atributo que tem o relacionamento.

Nesse caso, o atributo que terá a chave estrangeira não terá o mesmo nome da chave primária, pois não é possível repetir o nome do atributo.


FUNCIONÁRIO
matrícula
nome_funcionário
matrícula_gerente
(1,n)
(1,1)
é gerenciado por
é gerente de
GERÊNCIA

Ao final, teremos a tabela assim:

FUNCIONARIO (matricula, nome_funcionario, matricula_gerente) sendo que o atributo matricula_gerente terá uma chave estrangeira para FUNCIONÁRIO.

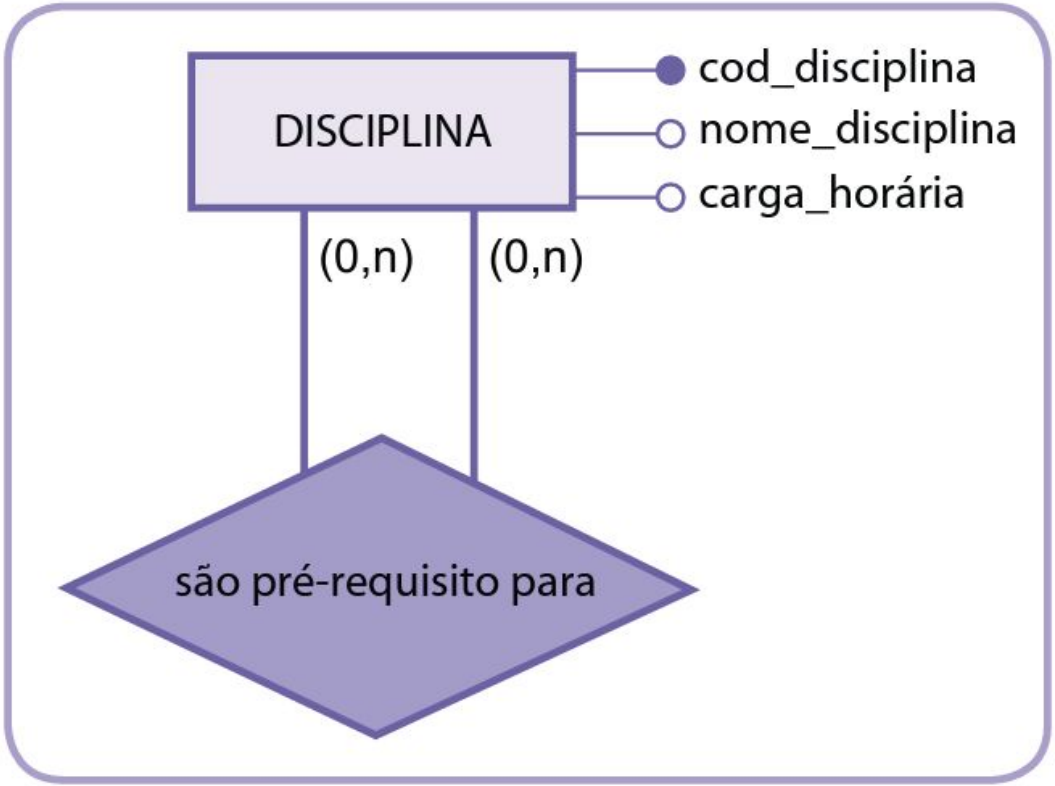


q q

Para os auto relacionamentos N:N, faremos da mesma forma que um relacionamento N:N tradicional, ou seja, criamos uma nova tabela para o relacionamento.

Esta nova tabela terá, ao menos, duas colunas as quais serão, ao mesmo tempo, chave primária e chaves estrangeiras.

Neste caso, as duas chaves estrangeiras estarão se referenciando à mesma tabela base.


DISCIPLINA
cod_disciplina
nome_disciplina
carga_horária
(0,n)
(0,n)
são pré-requisito para

DISCIPLINA (cod_disciplina, nome_disciplina, carga_horaria) PRE_REQUISITO (cod_disciplina, cod_disciplina_pre)

cod_disciplina terá uma chave estrangeira apontando para DISCIPLINA.
cod_disciplina_pre terá uma chave estrangeira apontando para DISCIPLINA.

Etapa 9: Mapeamento de tipos de relacionamento n-ário (qualquer relacionamento maior que o binário).

Para cada tipo de relacionamento nário R, onde n > 2, crie uma tabela T para representar R.

Inclua como atributos de chave estrangeira em T as chaves primárias das tabelas que representam as entidade participantes.

Inclua, também, quaisquer atributos simples do tipo de relacionamento nário (ou componentes simples de atributos compostos) como atributos de T.

A chave primária de T normalmente é uma combinação de todas as chaves estrangeiras que referenciam as tabelas representando as entidades participantes.

Porém, se as restrições de cardinalidade sobre qualquer um dos tipos de entidade E participantes em R for 1, então a chave primária de T não deve incluir o atributo de chave estrangeira que referencia a relação E’ correspondente a E.

Complementando:

- criar uma nova tabela contendo, como chaves estrangeiras, as chaves primárias das tabelas que representam os conjuntos de entidades participantes.
- normalmente, a combinação dessas chaves estrangeiras forma a chave primária da nova tabela, mas se a cardinalidade máxima de uma das entidades participantes for 1, então a chave estrangeira que referencia essa entidade não fará parte da chave primária da nova tabela.

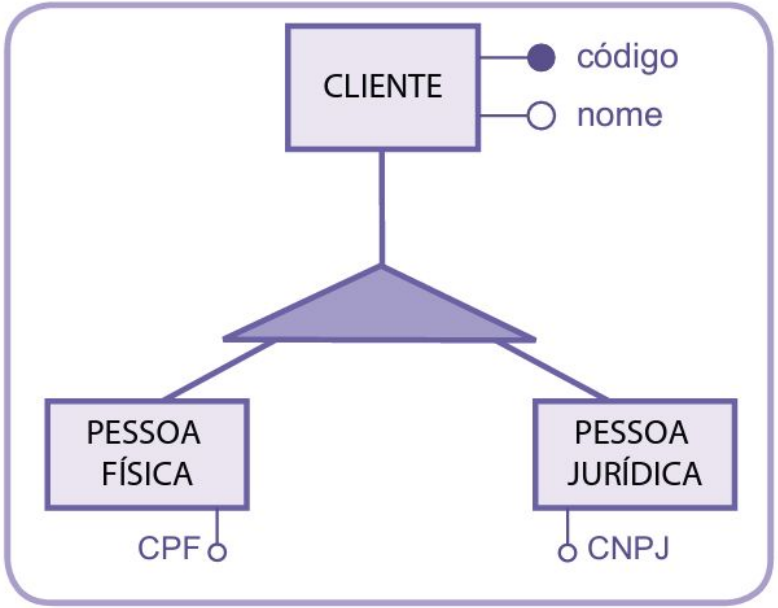
correspondências entre as construções e restrições do modelo conceitual lógico

MODELO CONCEITUAL	MODELO LÓGICO
Tipo de entidade	Tabela
Tipo de relacionamento 1:1 ou 1:N	Chave estrangeira (ou tabela)
Tipo de relacionamento N:N	Nova tabelae duaschaves estrangeiras
Tipo de relacionamento n-ário	Nova tabela
Atributo simples	Atributo
Atributo composto	Conjunto de atributos componentes simples
Atributo multivalorado	Tabela e chave estrangeira
Conjunto de valores	Domínio
Atributo-chave	Chave primária

Etapa 10: Mapeamento da especialização / generalização

Tomemos como base este modelo conceitual.

Para os casos em que há generalização / especialização, há três opções de projeto que podem ser adotadas:



Opção 1

Criar uma tabela para cada entidade (CLIENTE, FÍSICA, JURÍDICA). Nesse caso, a chave primária das tabelas especializadas (FÍSICA e JURÍDICA) é a mesma da entidade generalista (CLIENTE) e também devem ser chaves estrangeiras para a tabela CLIENTE.

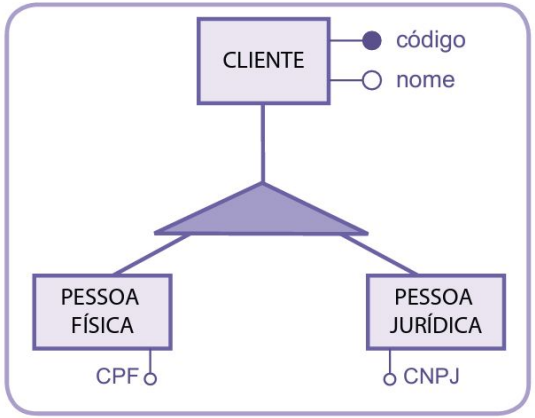
Logo, ao final, teremos três tabelas, assim:

Opção 1

CLIENTE (cod_cliente, nome_cliente)

FISICA (cod_cliente, cpf, sexo)

JURIDICA(cod_cliente, cnpj, nome_fantasia)



Lembre-se de que:
A coluna cod_cliente na tabela FISICA deverá ter chave estrangeira para CLIENTE.
A coluna cod_cliente na tabela JURIDICA deverá ter chave estrangeira para CLIENTE.

Opção 2

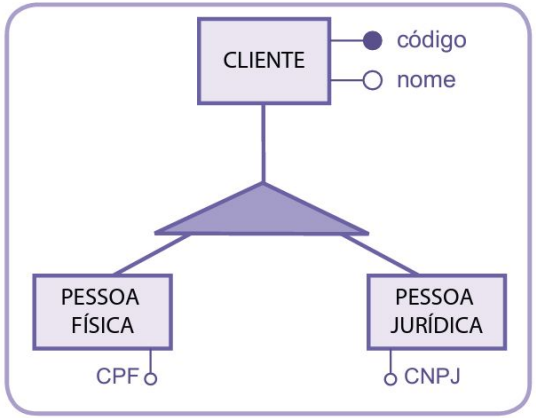
Criar uma tabela para cada entidade da especialização, como FÍSICA e JURIDICA. Nesse caso, os atributos da entidade genérica (em nosso exemplo CLIENTE) devem ser incluídos em ambas as tabelas criadas.

Ao Final, teremos duas tabelas, assim:

FISICA (cod_cliente, nome_cliente, cpf, sexo)

JURIDICA (cod_cliente, nome_cliente,

CNPJ,nome_fantasia)

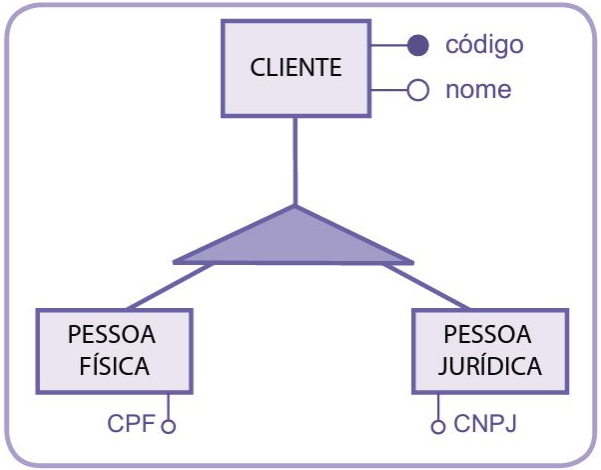


Opção 3

Criar uma única tabela, incluindo os atributos das 3 entidades. Nesse caso, os campos que eram das entidades especializadas deverão ser opcionais, ou seja, deverão aceitar valores nulos. Ao nal, teremos a única tabela, assim:

CLIENTE (cod_cliente, nome_cliente, cpf,

sexo, cnpj, nome_fantasia)



Qual escolher?

Cada uma das opções apresentadas traz vantagens e desvantagens.

Tipos de Dados

https://www.ibm.com/docs/pt-br/cmofm/9.5.0?topic=SSEPCD_9.5.0/com.ibm.ondemand.mp.doc/arsa0094.html

Tipos de Valores Numéricos

podem ser tipos inteiros, de ponto-flutuante ou de ponto-fixo, ocupando de um a oito bytes por valor em disco, com exceção do tipo bit que armazena de zero a 64 bits por valor. São eles: BIT, TINYINT, SMALLINT, INT, MEDIUMINT e BIGINT. Tipos de numéricos de ponto-flutante FLOAT e DOUBLE que requem quatro e oito bytes, por valor, respectivamente, e de ponto-fixo DECIMAL, que requer quatro bytes aproximadamente para cada lado, precisão e escala.

Tipos de Valores String

Podem ser binários ou não-binários, podendo ocupar espaços de tamanhos fixos ou variáveis. São eles: CHAR, VARCHAR, TEXT, BINARY, VARBINARY, BLOB, ENUM e SET.

Não-binários: CHAR, VARCHAR e TEXT, este último que se divide em mais outros três, que são: TINYTEXT, MEDIUMTEXT e LONGTEXT.

Binários: BINARY, VARBINARY e BLOB, este último que se divide em mais outros três, que são: TINYBLOB, MEDIUMBLOB e LONGBLOB.

Tipos de Valores Temporais

são tipos de dados utilizados normalmente para armazenar informações de data e hora, podendo ser somente a data ou a hora, de acordo com o contexto e pode ocupar de um a oito bytes por valor. São eles: DATE, TIME, DATETIME, TIMESTAMP e YEAR.

Normalização

Finalizado o esquema relacional, passa-se ao processo de normalização. Este processo baseia-se no conceito de forma normal. Uma forma normal é uma regra que deve ser obedecida por uma tabela para que esta seja considerada "bem projetada". Há diversas formas normais, isto é, diversas regras, cada vez mais rígidas, para verificar tabelas relacionais. Em nossa disciplina, vamos considerar três formas normais. As formas normais são denominadas simplesmente primeira, segunda e terceira forma normal, abreviadamente 1FN, 2FN e 3FN.

Em geral, quanto mais alta a forma normal, mais operações de junção são necessárias para produzir a saída especificada e mais recursos são exigidos pelo sistema de banco de dados para responder a consultas do usuário final.

Um projeto bem- sucedido deve considerar a demanda desse usuário por empenho rápido. Portanto, em algumas ocasiões, será preciso desnormalizar certas partes de um projeto do banco de dados, de modo a atender às exigências de desempenho.

Primeira Forma Normal (1FN)

O primeiro passo da normalização consta da transformação do esquema de tabela não normalizada em um esquema relacional na primeira forma normal (1FN). Segundo Rob e Coronel (2011), uma tabela encontra-se na 1FN quando:

- todos os atributos de chave estão definidos;
- não há grupos de repetição na tabela. Em outras palavras, cada intersecção de linha/coluna contém um e somente um valor, não um conjunto de valores;
- todos os atributos são dependentes da chave primária.

CÓDIGO	NOME	GERENTE DEP.	LOCALIZAÇÕES DO DEPARTAMENTO
5	Pesquisa	1	Cachoeiro, Castelo, Muqui
4	Adm	2	Marataízes, Guarapari
1	Sede	3	Iconha

Segunda Forma Normal (2FN)

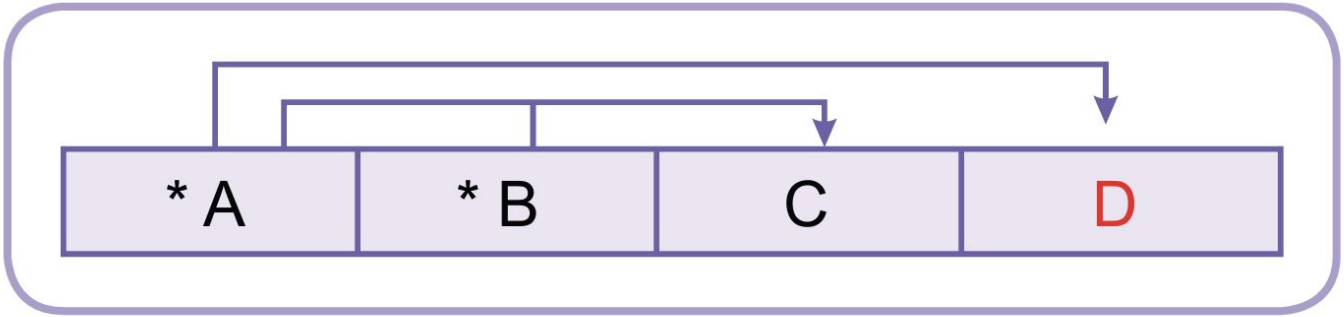
Segundo Rob e Coronel (2011), uma tabela encontra-se na 2FN quando:

- está em 1NF. e
- não inclui dependências parciais; ou seja, nenhum atributo é dependente apenas de uma parte da chave primária.

Se a 2FN trata das dependências parciais de uma chave primária, se a chave primária for simples (formada apenas por um atributo) ela já estará na 2FN.

Observe que ainda é possível uma tabela em 2NF apresentar dependência transitiva, ou seja, um ou mais atributos podem ser funcionalmente dependentes de atributos não relacionados à chave.

Veja um exemplo de quando uma tabela não está na 2FN:

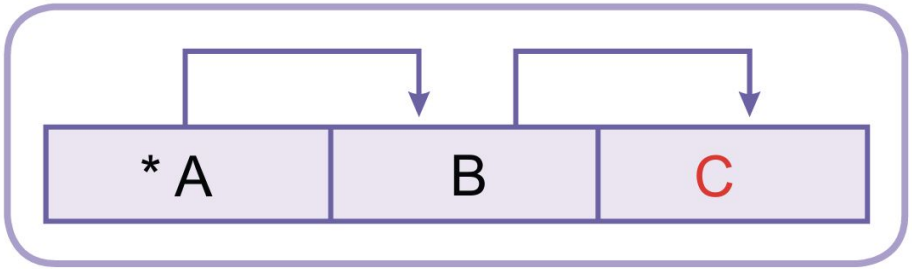


Ddepende só de A e não da chave inteira (A,B)

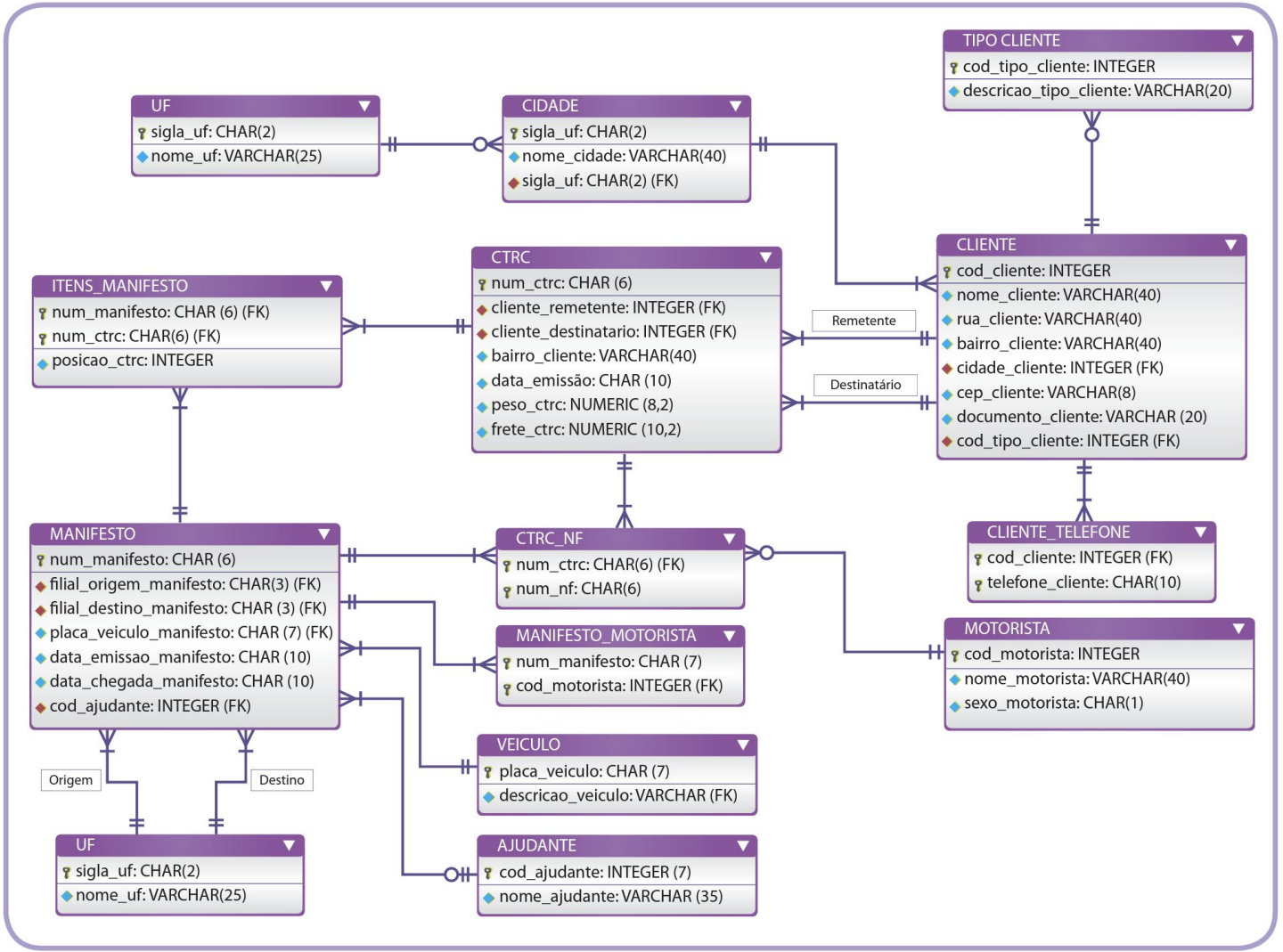
TERCEIRA FORMA NORMAL (3FN)

Segundo (Rob e Coronel, 2011), uma tabela encontra-se na 3FN quando:

- está em 2NF e
- não contém dependências transitivas.

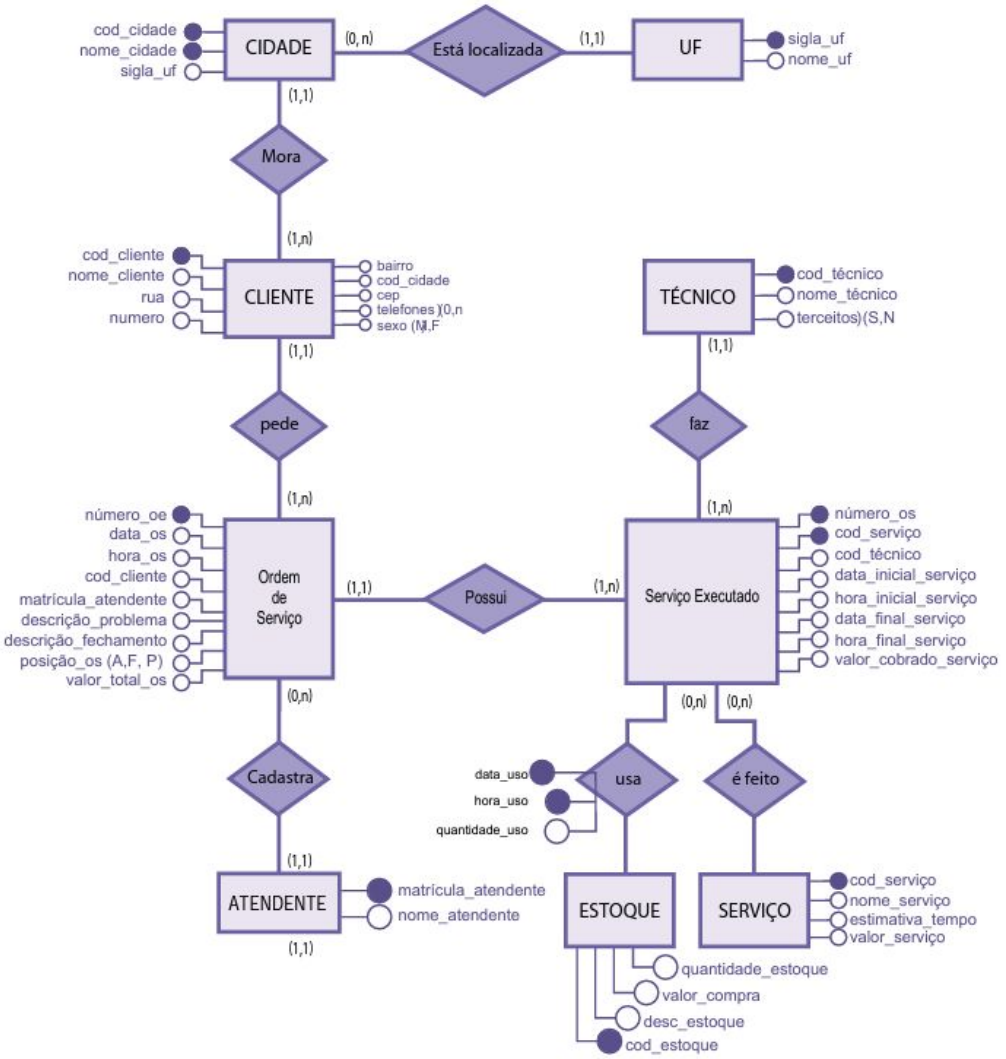


C depende de B que não é chave


TIPO CLIENTE
cod_tipo_cliente: INTEGER
descricao_tipo_cliente: VARCHAR(20)
UF
sigla_uf: CHAR(2)
nome_uf: VARCHAR(25)
CIDADE
sigla_uf: CHAR(2)
nome_cidade: VARCHAR(40)
sigla_uf: CHAR(2) (FK)
CLIENTE
cod_cliente: INTEGER
nome_cliente: VARCHAR(40)
rua_cliente: VARCHAR(40)
bairro_cliente: VARCHAR(40)
cidade_cliente: INTEGER (FK)
cep_cliente: VARCHAR(8)
documento_cliente: VARCHAR (20)
cod_tipo_cliente: INTEGER (FK)
ITENS_MANIFESTO
num_manifesto: CHAR (6) (FK)
num_ctrc: CHAR(6) (FK)
posicao_ctrc: INTEGER
CTRC
num_ctrc: CHAR (6)
cliente_remetente: INTEGER (FK)
cliente_destinatario: INTEGER (FK)
bairro_cliente: VARCHAR(40)
data_emissão: CHAR (10)
peso_ctrc: NUMERIC (8,2)
frete_ctrc: NUMERIC (10,2)
Remetente
Destinatário
CLIENTE_TELEFONE
cod_cliente: INTEGER (FK)
telefone_cliente: CHAR(10)
MANIFESTO
num_manifesto: CHAR (6)
filial_origem_manifesto: CHAR(3) (FK)
filial_destino_manifesto: CHAR (3) (FK)
placa_veiculo_manifesto: CHAR (7) (FK)
data_emissao_manifesto: CHAR (10)
data_chegada_manifesto: CHAR (10)
cod_ajudante: INTEGER (FK)
CTRC_NF
num_ctrc: CHAR(6) (FK)
num_nf: CHAR(6)
MANIFESTO_MOTORISTA
num_manifesto: CHAR (7)
cod_motorista: INTEGER (FK)
MOTORISTA
cod_motorista: INTEGER
nome_motorista: VARCHAR(40)
sexo_motorista: CHAR(1)
VEICULO
placa_veiculo: CHAR (7)
descricao_veiculo: VARCHAR (FK)
Origem
Destino
UF
sigla_uf: CHAR(2)
nome_uf: VARCHAR(25)
AJUDANTE
cod_ajudante: INTEGER (7)
nome_ajudante: VARCHAR (35)

Exercício

Baseado no modelo conceitual abaixo, crie o modelo lógico.



Bibliografia

- DATE, C. J. Introdução a sistemas de bancos de dados. 8. ed. Rio de Janeiro, RJ: Elsevier, 2004. 865 p. ISBN 9788535212730
- ELMASRI, Ramez; NAVATHE, Sham. Sistemas de banco de dados. 7. ed. São Paulo, SP: Pearson Education do Brasil, 2019. xxvi, 1126 p. ISBN 9788543025001.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de banco de dados. 7. ed. Rio de Janeiro, RJ: LTC, 2020. xx, 762 p. ISBN 9788595157330.