MODELING

DEFININDO AS **ESTRUTURAS FÍSICAS!**

7

LISTA DE FIGURAS

SUMÁRIO

1 DEFININDO AS ESTRUTURAS FÍSICAS!

Ao término do modelo entidade relacionamento (modelo lógico de dados), em que inserimos todas as necessidades de armazenamento e respectivas associações dentro da visão de negócio, devemos fazer a passagem desse modelo para o que chamamos de modelo relacional ou modelo físico de dados.

Neste capítulo abordamos o modelo físico, e você poderá se considerar apto a realizar a modelagem de dados completa de seu projeto Fintech!

1.1 Sobre o modelo físico

O modelo físico ou relacional representa os dados em um BD, como uma coleção de tabelas (relações), para gerência de bases de dados (SGBD), é um modelo de dados baseado em lógica na teoria dos conjuntos.

O modelo relacional ou físico de dados é derivado do modelo lógico, no qual se encontram detalhados os componentes de estrutura física do banco de dados, como: tabelas, campos, tipos de dados, índices, nomenclaturas, exigências (restrições) relativas a conteúdo, domínio dos campos, entre alguns outros elementos que veremos mais à frente.

Neste ponto estaremos prontos para a criação do banco de dados, utilizando um SGBD. No modelo físico ou relacional, devemos aplicar os elementos físicos mencionados, considerando o SGBD que será utilizado para implementação.

Exemplos de SGBDs relacionais: Oracle, SQL Server, DB2 (IBM), MySQL, PostgreSQL.

Esta é a última etapa do projeto de banco de dados. Após esta fase, utilizaremos uma linguagem de definição de dados reconhecida pelo SGBD (DDL – Linguagem de Definição de Dados, subdivisão da linguagem SQL). O conjunto de comandos DDL é denominado SCRIPT de criação de banco de dados e será executado no SGBD.

Definindo as estruturas físicas!

Figura 1 – Criação de banco de dados
Fonte: Shutterstock (2017)

1.2 Conversão de entidades e respectivos atributos

Converter entidade e respectivos atributos.

- Cada entidade é traduzida para uma tabela.
- Cada atributo define uma coluna.
- Cada atributo identificador define colunas que compõem a chave primária.

BOA PRÁTICA: A fim de facilitar o trabalho dos programadores, é conveniente manter os nomes das colunas curtos e procurar utilizar a mesma nomenclatura para todo o banco de dados. Normalmente, as empresas criam regras para composição dos nomes, que devem estar de acordo com as regras de criação de nomes dos SGBDs em relação aos caracteres e tamanho permitidos.

Exemplo:

SQL Developer Data Modeler

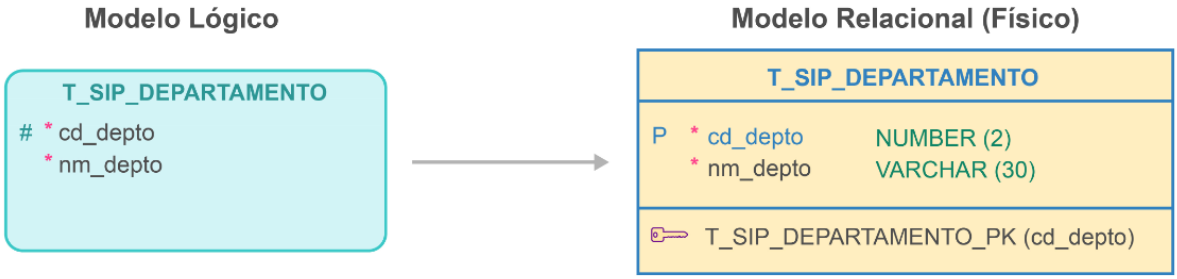

Figura 2 – Transição do modelo lógico para o modelo físico
Fonte: Elaborado pelo autor (2016)

Definindo as estruturas físicas!

ATENÇÃO: Todas as **tabelas devem possuir uma chave primária**. No caso de não existirem, é preciso criar um identificador, **preferencialmente com conteúdo numérico** por ser localizado mais rapidamente pelos SGBDs.

Exemplo: Modelo Relacional (Físico)

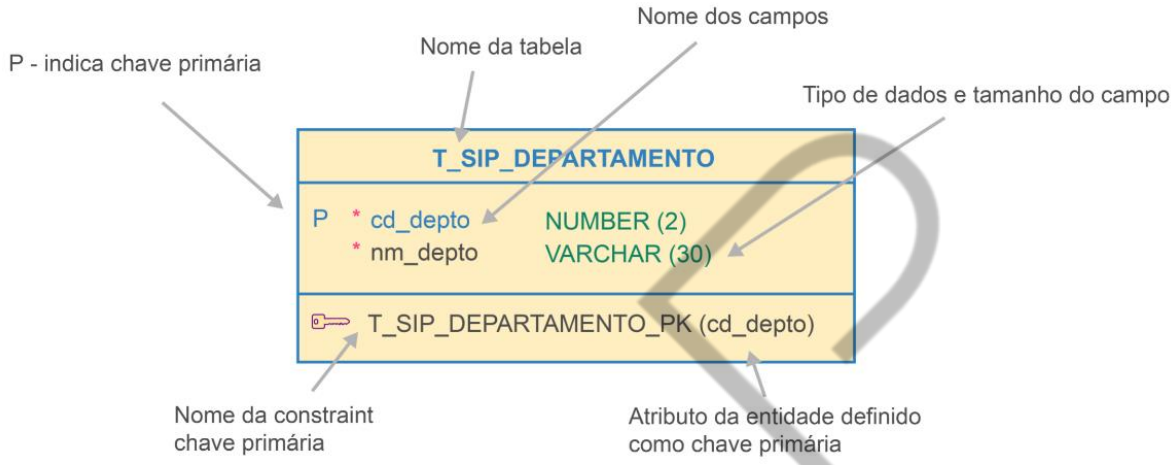

Figura 3 – Informações disponíveis em uma tabela do Modelo Físico
Fonte: Elaborado pelo autor (2016)

NOTA: *Constraints* são restrições (regras) associadas às colunas (campos) de uma tabela. No exemplo, a *constraint* do tipo chave primária é utilizada para identificar cada ocorrência (registro) de maneira única.

Regra para Mapeamento de entidade com relacionamento identificador

Para cada **relacionamento identificador** ( ), **é criada uma chave estrangeira** que implementa a entidade identificada pelo relacionamento identificador.

A **chave primária da tabela, que implementa** a entidade identificada pelo relacionamento identificador, será composta por:

- Colunas correspondentes aos atributos identificadores.
- Chaves estrangeiras que implementam os relacionamentos identificadores.

Exemplo: mapeamento de entidade com relacionamento identificador

Definindo as estruturas físicas!

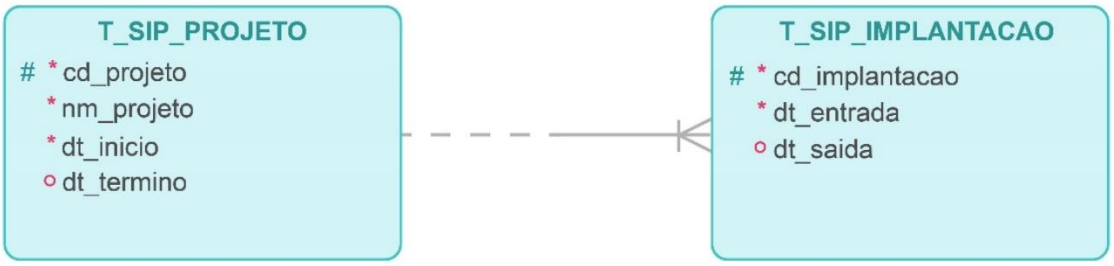

Figura 4 – Exemplo de relacionamento entre tabelas em notação de Barker
Fonte: Elaborado pelo autor (2016)

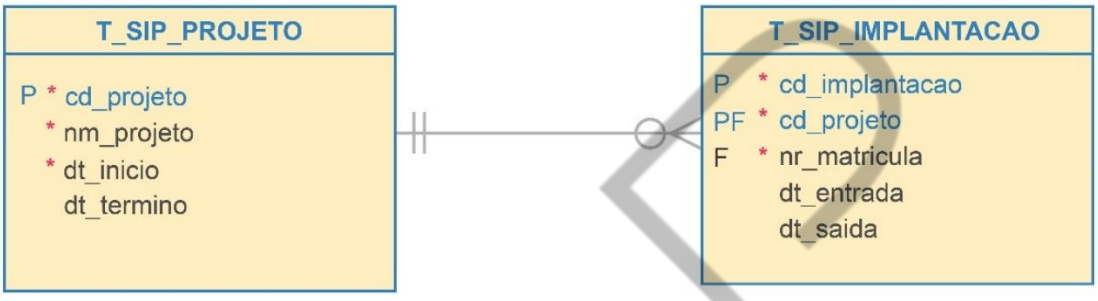

Figura 5 – Exemplo de relacionamento entre tabelas em notação da Engenharia da informação
Fonte: Elaborado pelo autor (2016)

SQL Developer Data Modeler: Modelo Relacional

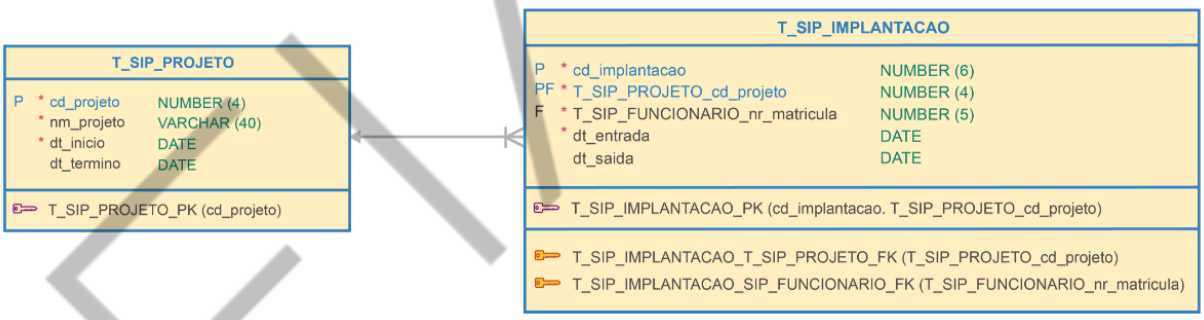

Figura 6 – Exemplo de relacionamento no modelo físico
Fonte: Elaborado pelo autor (2016)

Definindo as estruturas físicas!

Exemplo modelo relacional (Físico)

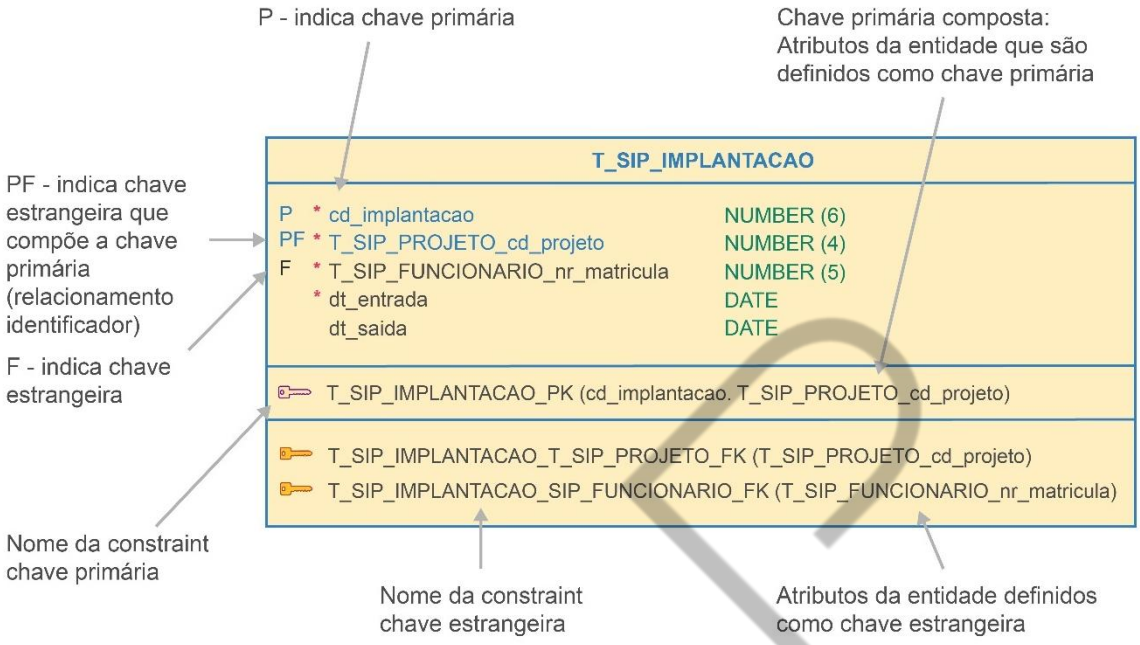

Figura 7 – Informações sobre o relacionamento presentes em uma tabela do modelo Físico
Fonte: Elaborado pelo autor (2016)

Atributos se transformam em campos de tabelas, **exceto quando são multivalorados ou compostos.**

- Atributos compostos devem ser desmembrados em atributos atômicos (único/indivisível).
- Atributos multivalorados devem dar origem a uma nova tabela.

1.3 Conversão de relacionamentos e respectivos atributos

A conversão é determinada pela cardinalidade mínima e máxima das entidades que participam do relacionamento.

Temos três opções para tradução de relacionamentos:

- Tabela própria (Entidade Associativa ou Agregação).
- Adição de colunas (atributos do relacionamento).
- Fusão de tabelas.

Tabela Própria (ENTIDADE ASSOCIATIVA)

Definindo as estruturas físicas!

O relacionamento é implementado através de uma tabela própria (**ASSOCIATIVA**), contendo as seguintes colunas:

- Colunas correspondentes aos identificadores das entidades relacionadas (**chaves estrangeiras**).
- Colunas correspondentes aos **atributos do relacionamento**.

Exemplo: SQL Developer Data Modeler: Notação de Barker

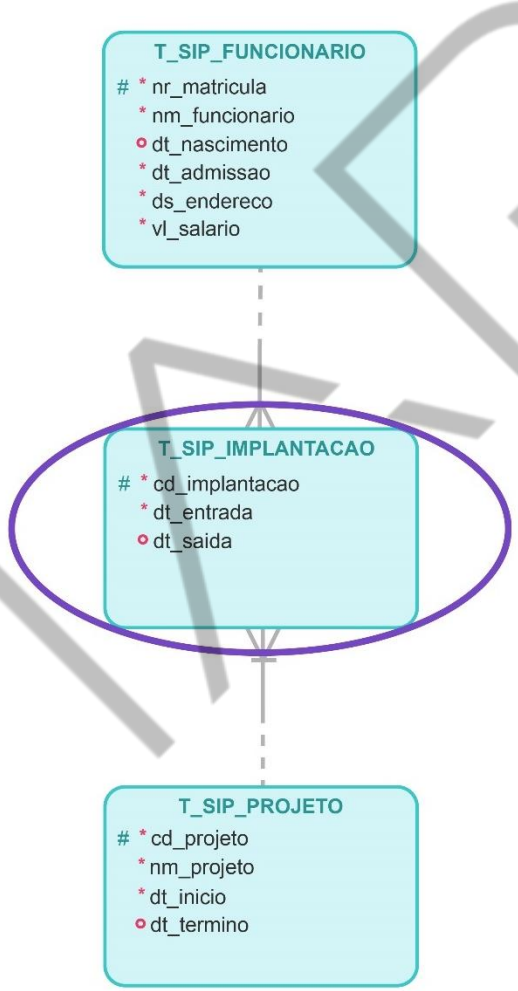

Figura 8 – Exemplo de entidade associativa em notação de Barker
Fonte: Elaborado pelo autor (2016)

SQL Developer Data Modeler: Notação da Engenharia da Informação

Definindo as estruturas físicas!

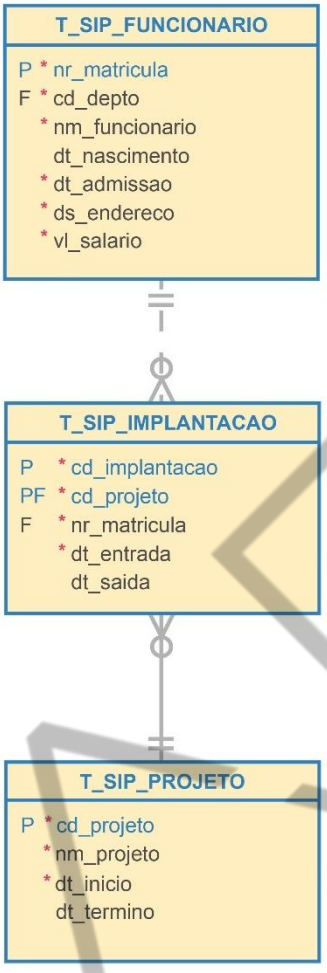

Figura 9 – Exemplo de entidade associativa em notação da Engenharia da informação
Fonte: Elaborado pelo autor (2016)

SQL Developer Data Modeler: Modelo Relacional

Definindo as estruturas físicas!

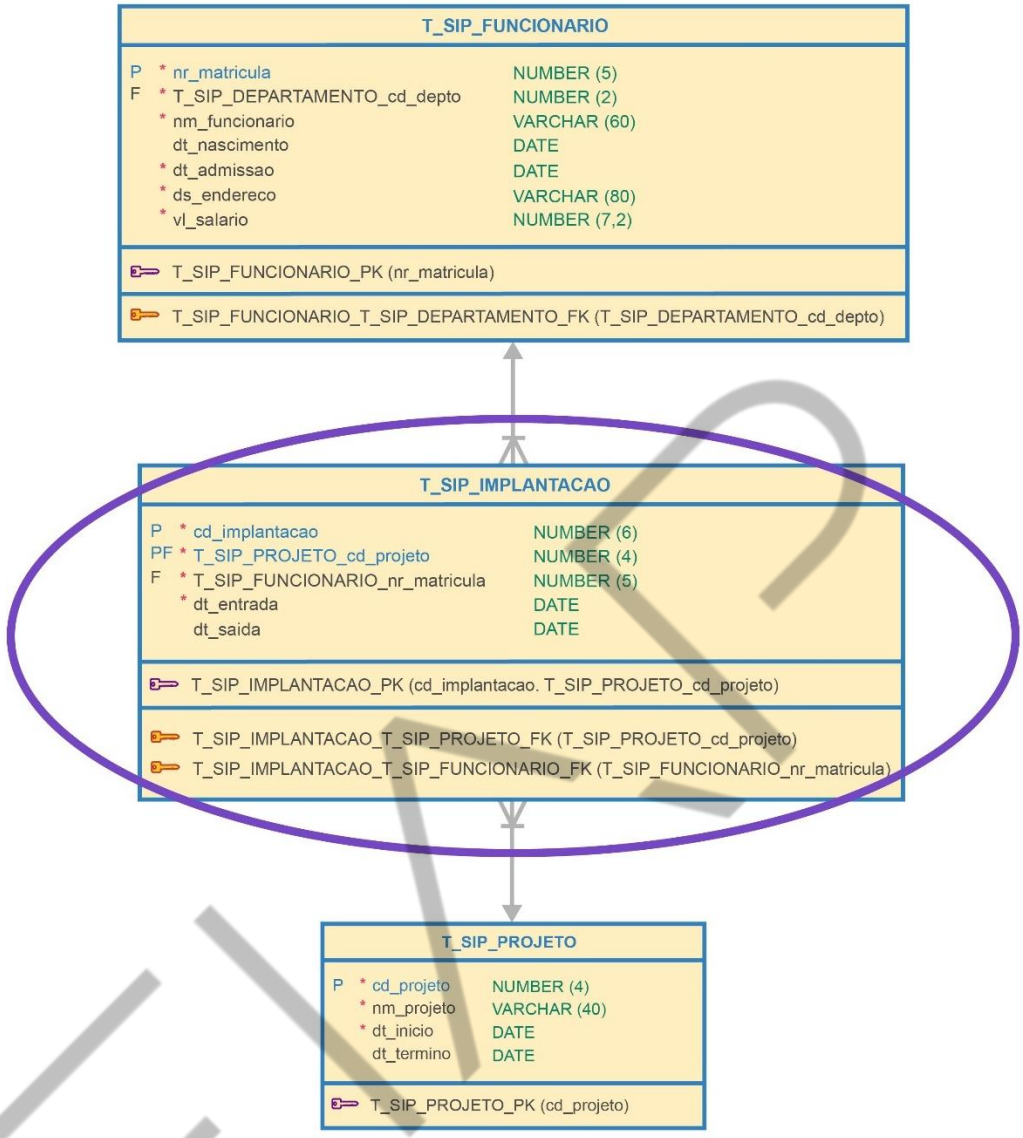

Figura 10 – Exemplo de entidade associativa se tornando uma tabela associativa no modelo físico
Fonte: Elaborado pelo autor (2016)

1.3.1 Adição de Colunas

Adicionar na tabela, cuja cardinalidade máxima é 1 (relacionamento), as seguintes colunas:

- Colunas correspondentes ao identificador da entidade relacionada, formando uma chave estrangeira em relação à tabela que implementa a entidade relacionada.
- Colunas correspondentes aos atributos do relacionamento.

Exemplo:

Definindo as estruturas físicas!

O atributo data da lotação é um atributo do relacionamento entre as entidades “DEPARTAMENTO” e “EMPREGADO”.

O **atributo do relacionamento** deve ser **conduzido** à TABELA com **cardinalidade máxima igual a 1**, neste exemplo “**EMPREGADO**”.

Notação de Peter Chen:

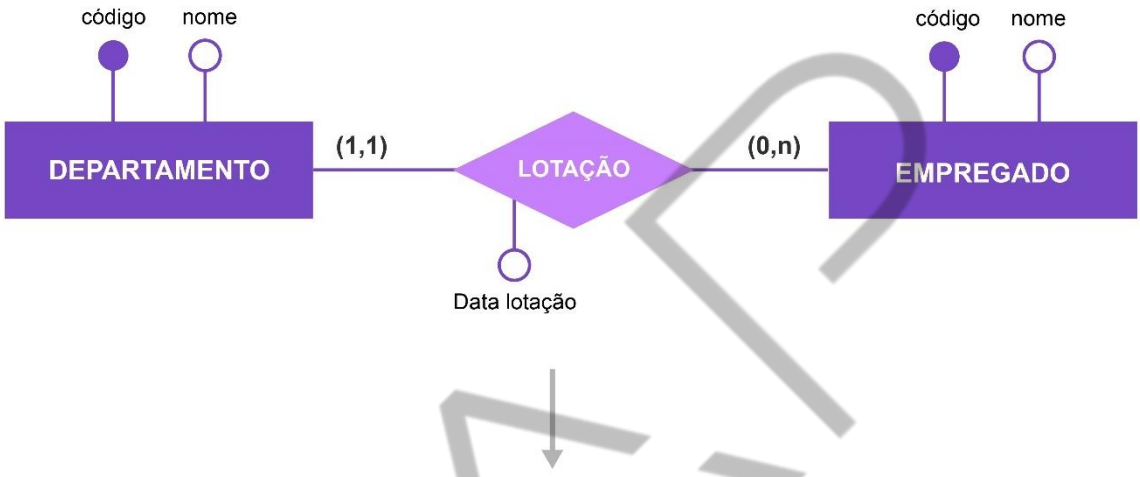

Figura 11 – Modelo conceitual do relacionamento “Departamento x Empregado” na notação de Peter Chen
Fonte: Elaborado pelo autor (2016)

SQL Developer Data Modeler: Notação da Engenharia da Informação

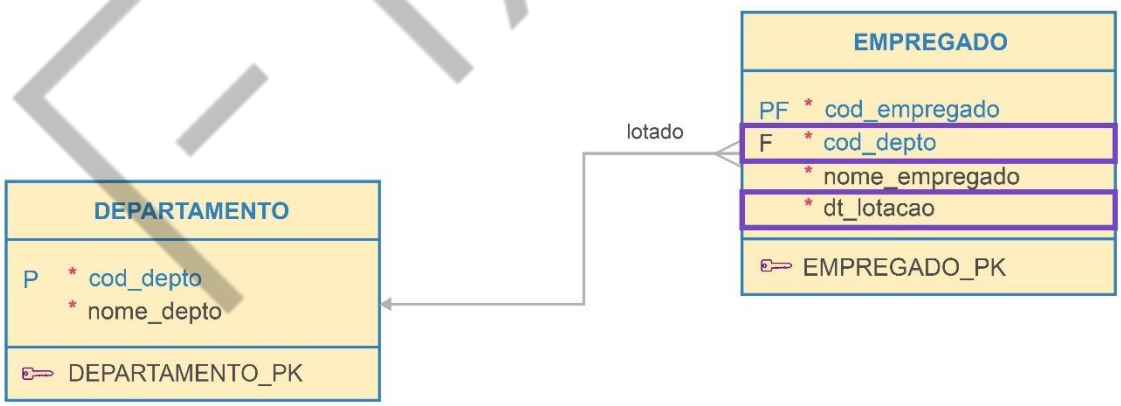

Figura 12 – Modelo físico do relacionamento “Departamento x Empregado” na notação da Engenharia da Informação
Fonte: Elaborado pelo autor (2016)

Definindo as estruturas físicas!

1.3.2 Fusão de Tabelas

Implementar em uma única tabela todos os atributos de ambas as entidades e atributos eventualmente existentes no relacionamento. Esta conversão somente pode ser aplicada para relacionamentos do tipo 1:1.

Teremos a implementação de um relacionamento recursivo.

Exemplo:

SQL Developer Data Modeler: Notação de Barker

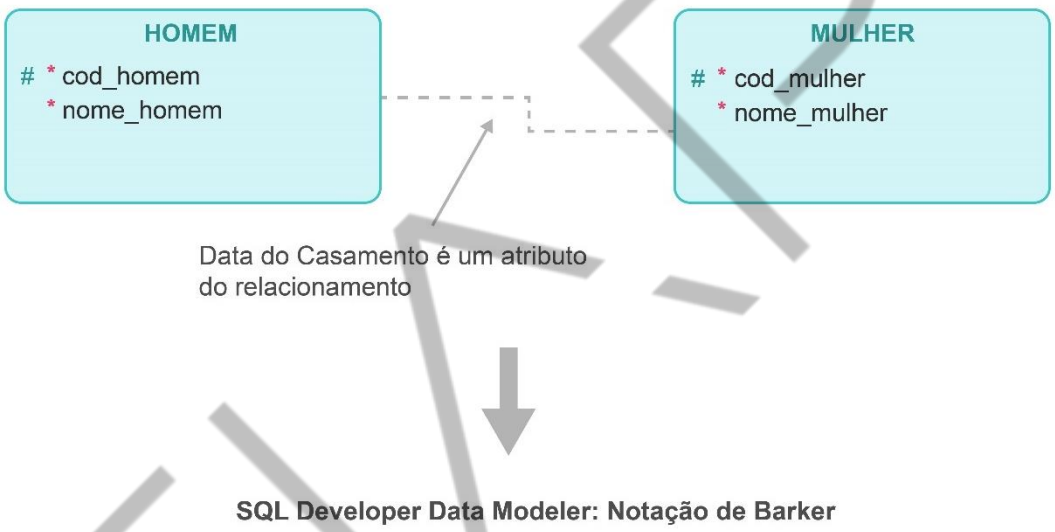

Figura 13 – Exemplo de fusão de tabelas
Fonte: Elaborado pelo autor (2016)

SQL Developer Data Modeler: Notação de Barker

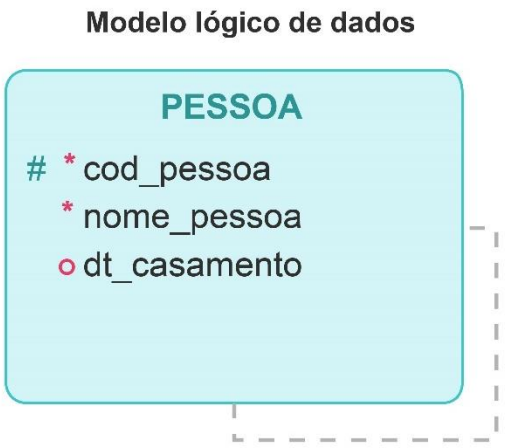

Figura 14 – Modelo lógico com autorrelacionamento
Fonte: Elaborado pelo autor (2016)

Definindo as estruturas físicas!

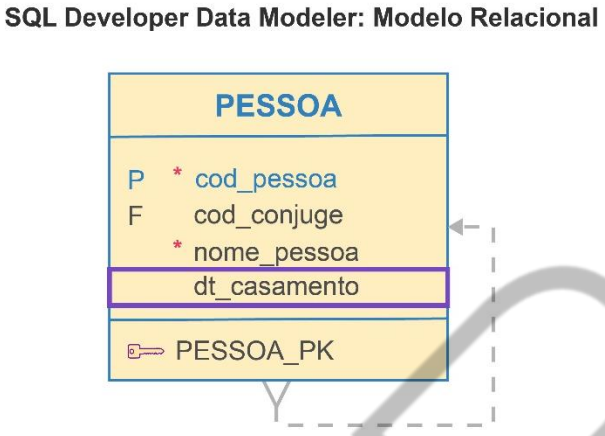

Figura 15 – Destaque para o campo data de casamento na tabela pessoa
Fonte: Elaborado pelo autor (2016)

1.4 Conversão de especialização e generalização

Temos três alternativas para implementação:

- Uso de uma única tabela para toda hierarquia de generalização/ especialização.
- Uso de uma tabela para cada entidade.
- Uso de uma tabela genérica e tabelas especializadas.

Exemplo com uso de uma <u>única tabela</u> para toda hierarquia de generalização/especialização:

Por meio do uso de uma única tabela para toda hierarquia de generalização/especialização, as colunas CREA e CRM, no exemplo, devem ser definidas como opcionais.

Isso é necessário pois, se um funcionário não pertencer a nenhuma das classes especializadas, terá as colunas vazias, enquanto um médico terá a coluna CREA vazia e um engenheiro terá a coluna CRM vazia.

Definindo as estruturas físicas!

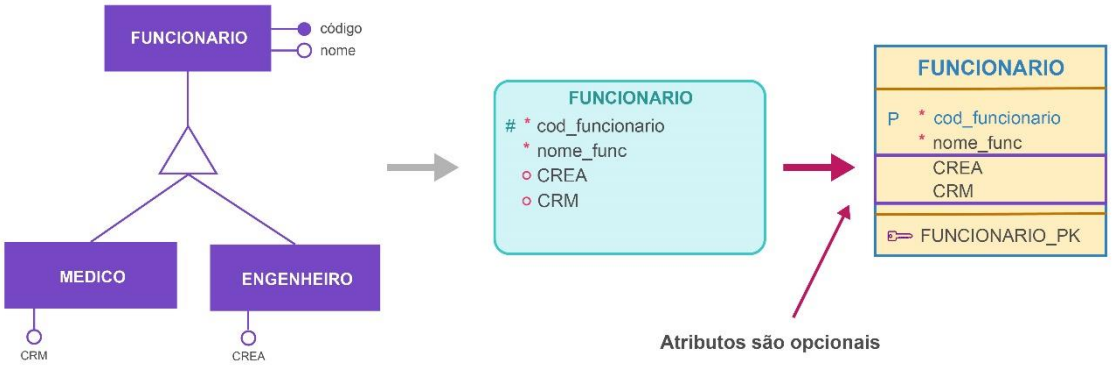

Figura 16 – Transposição do modelo conceitual para os modelos lógico e físico
Fonte: Elaborado pelo autor (2016)

Exemplo com uso de <u>uma tabela por entidade especializada</u>:

Nesta implementação, temos uma tabela para cada necessidade. Observe que as colunas CREA e CRM, por exemplo, são mandatórias para cada tabela específica implementada. Implementando dessa forma, as tabelas representam papéis ou necessidades específicas dentro de um contexto de negócio.

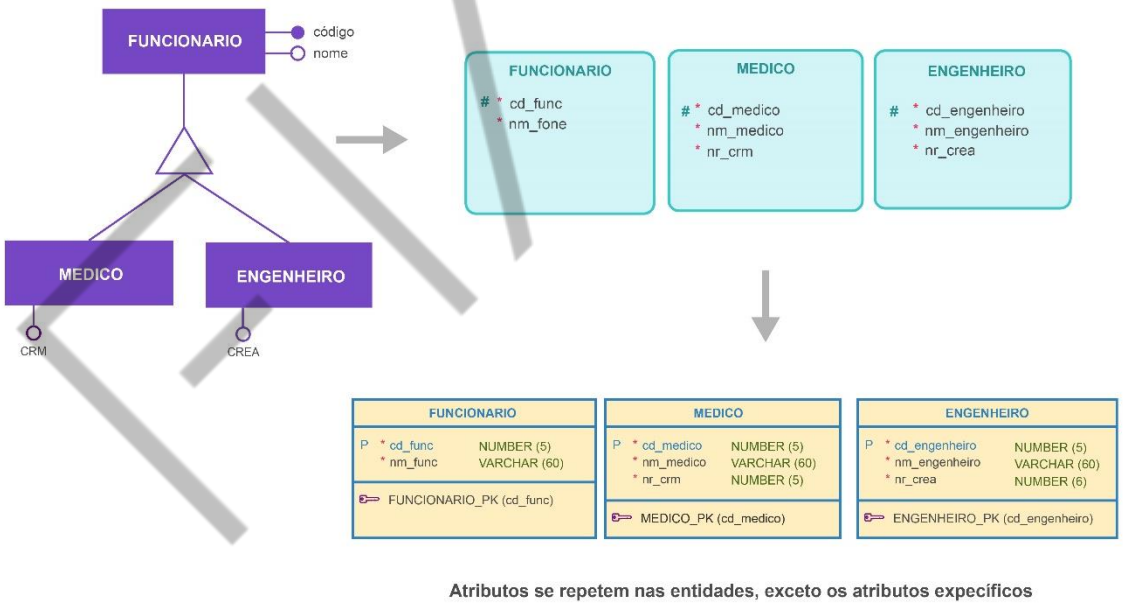

Figura 17 – Transposição do modelo conceitual para os modelos lógico e físico (2)
Fonte: Elaborado pelo autor (2016)

Outro exemplo com uso de <u>uma tabela por entidade especializada</u>:

Neste caso, devemos aplicar todas as regras relativas à implementação de entidades e relacionamentos, acrescendo a inclusão da chave primária da tabela correspondente à entidade genérica, em cada entidade especializada correspondente. (herança de atributos).

As tabelas especializadas herdam a chave primária da tabela genérica.

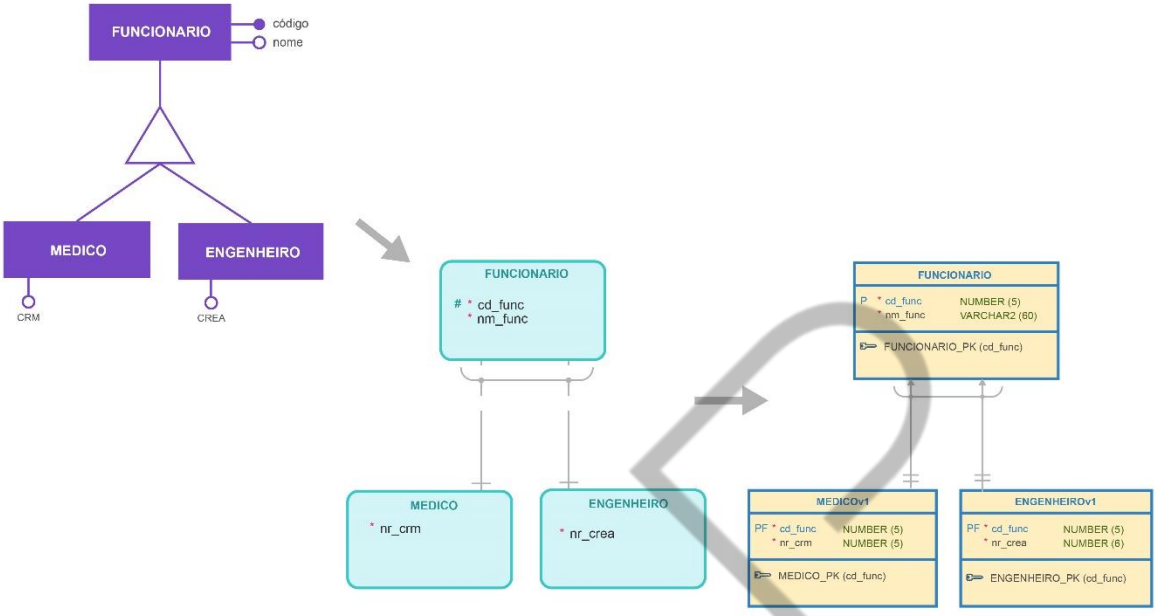

Figura 18 – Transposição do modelo conceitual para os modelos lógico e físico (3)
Fonte: Elaborado pelo autor (2016)

Exemplo com uso de uma tabela por entidade especializada:

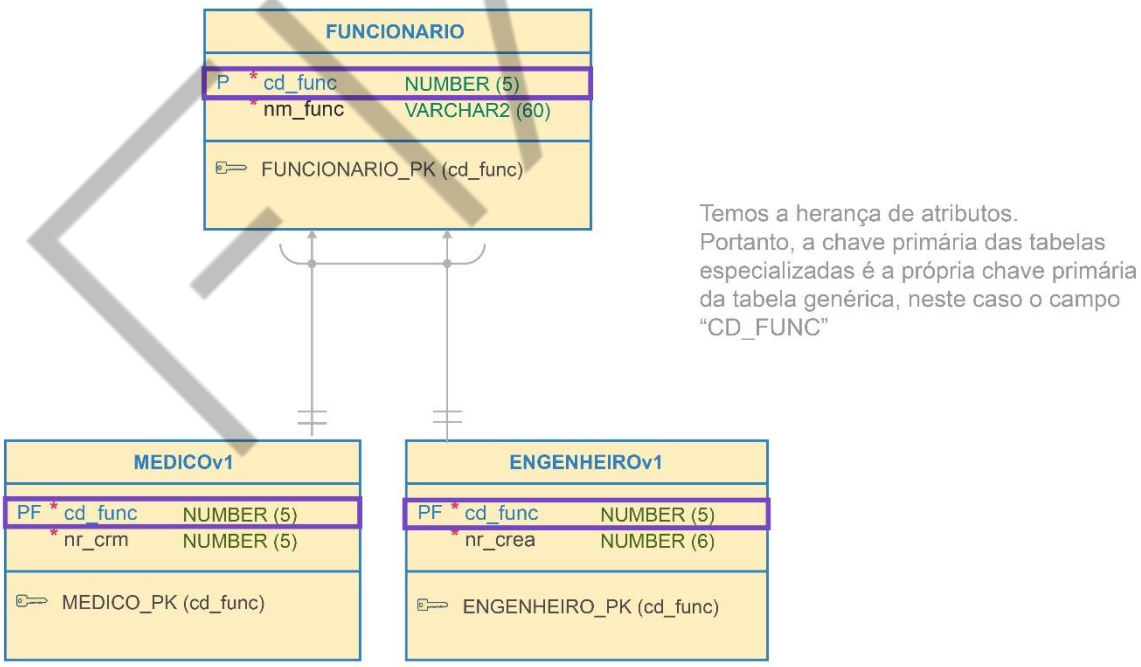

Figura 19 – Exemplo de herança no modelo físico
Fonte: Elaborado pelo autor (2016)

Definindo as estruturas físicas!

2 ANÁLISE ENTRE OS TIPOS DE IMPLEMENTAÇÃO

Tabela única: Os dados estão em uma única linha, não sendo necessário realizar junções para obter informações da entidade genérica e especializada. As colunas opcionais são referentes aos atributos que podem ser vazios de acordo com as características de cada funcionário.

A chave primária é armazenada uma única vez.

Uma tabela por entidade especializada: Faremos junções para obter as informações necessárias, pois parte dos dados está armazenada na tabela genérica e parte na tabela especializada.

Uma tabela para cada entidade: Teremos replicação dos dados em cada tabela.

2.1 Mapeamento de entidades fracas

Para cada tabela fraca, a chave primária será composta pela chave primária da tabela Forte e mais um atributo (chave estrangeira na tabela fraca), formando, assim, uma chave primária composta.

SQL Developer Data Modeler: Notação de Barker

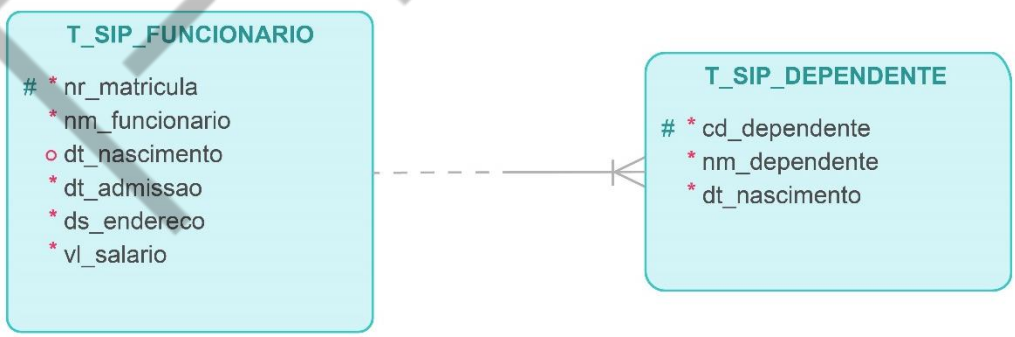

Figura 20 – Exemplo de modelo lógico
Fonte: Elaborado pelo autor (2016)

SQL Developer Data Modeler: Modelo Relacional

Definindo as estruturas físicas!

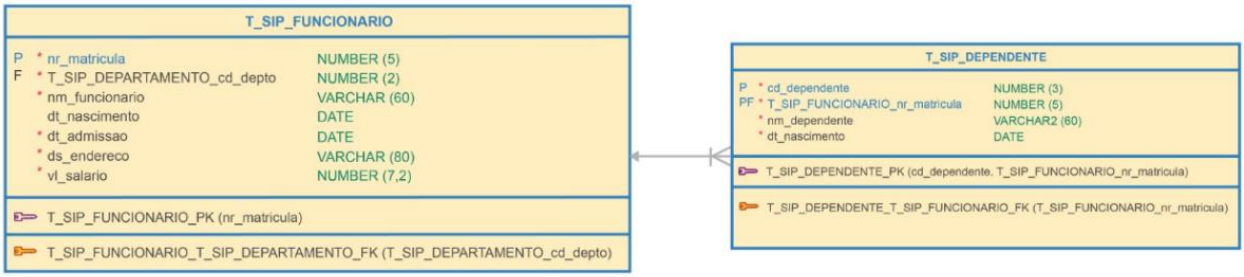

Figura 21 – Exemplo de modelo físico
Fonte: Elaborado pelo autor (2016)

SQL Developer Data Modeler: Modelo Relacional

SQL Developer Data Modeler: Modelo Relacional

Chave primária composta na TABELA FRACA, pela chave estrangeira mais um campo da própria tabela.

Figura 22 – Detalhe da tabela dependente
Fonte: Elaborado pelo autor (2016)

2.2 Tipos de Dados

O SQL Developer Data Modeler permite definir o tipo de dado dos atributos, isto é, se aquele atributo poderá armazenar valores numéricos, texto, data, imagens, sons ou algum outro valor. Os tipos de dados mais comuns são aqueles usados para armazenar texto, números e datas. Outros tipos de dados serão discutidos posteriormente.

O tipo de dado texto é usado para armazenar caracteres alfanuméricos, ou seja, palavras e texto sem formatação. Campos como nome de um cliente, endereço de uma clínica ou cidade natal de um aluno, normalmente, são armazenados como

dados alfanuméricos. VARCHAR2 é a forma mais comum de representar esse tipo de dado.

A Figura “Tela com as Propriedades da Tabela” mostra o atributo **nm_dependente VARCHAR2(60)**, indicando que o campo é do tipo alfanumérico – o número **60,** entre parênteses, indica o número máximo de caracteres que pode ser armazenado – ou simplificando o tamanho do campo. É obrigatório informar o tamanho do campo e, caso ocorra uma tentativa de inserção de um valor com mais caracteres que o definido em seu tamanho, ocorrerá um erro. O banco de dados armazenará o valor informado para os campos do tipo VARCHAR2 exatamente como foi informado, sem acrescentar espaços em branco ao final do texto.

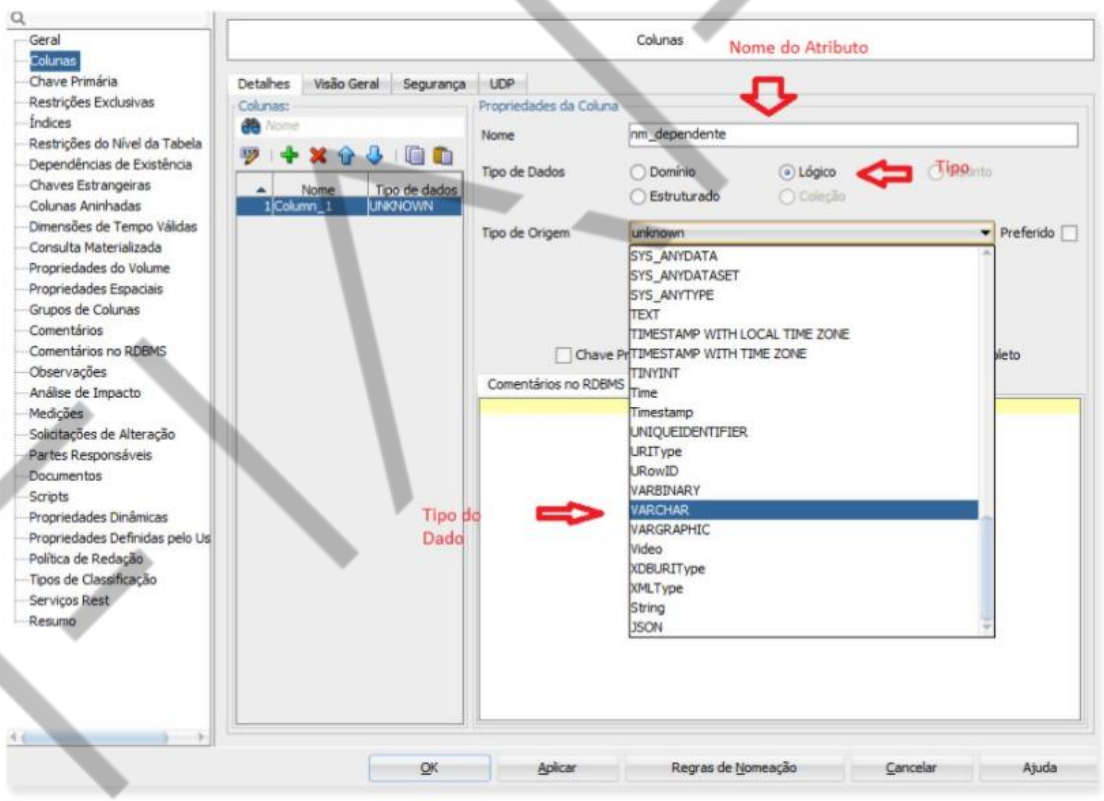

Figura 23 – Tela com as Propriedades da Tabela
Fonte: Oracle SQL Developer Data Modeler (2021)

O tipo de dado numérico é usado para armazenar números positivos ou negativos, sejam eles inteiros, com ponto flutuante ou com tamanho fixo de casas decimais. Campos como altura, peso e salário normalmente são armazenados como dados numéricos. NUMBER é a forma mais comum de representar esse tipo de dado.

A Figura “Exemplo de Modelo Físico” mostra o atributo **nr_matricula NUMBER(5)**, indicando que o campo é do tipo numérico, inteiro, com tamanho de 5 bytes, permitindo armazenar valores de até cinco dígitos, isto é, até o valor máximo de 99999 (noventa e nove mil e novecentos e noventa e nove). Ocorrerá um erro caso ocorra a tentativa de inserção de um valor maior do que o tamanho definido. A Figura “Exemplo de Modelo Físico” também mostra o atributo **vl_salario NUMBER(7,2)**, indicando que o campo é do tipo numérico, com duas casas decimais. O número 7 indica que o tamanho do campo é de sete bytes e o número 2 indica que dois bytes estão sendo reservados para armazenar as casas decimais, nesse caso, o maior valor que pode ser armazenado nesse campo é 99999,99.

O tipo de dado data é usado para armazenar datas. Esse tipo de dado contém data e hora e o seu formato pode ser definido no momento da instalação do banco de dados ou em tempo de execução. O formato permite armazenar dia, mês, ano, hora, minuto e segundo. Não aceita frações de segundo, nem fuso horário.

A Figura “Detalhe da tabela dependente” mostra o atributo **dt_nascimento DATE**, indicando que é um campo do tipo data. Não é necessário informar o tamanho desse tipo de dado.

2.3 Gerando um *script*

O Oracle SQL Developer Data Modeler permite gerar um arquivo de *script* com os comandos de Data Definition Language (DDL), que podem ser usados para criar as tabelas no banco de dados. O atalho Alt+Shit+L pode ser usado para executar o Editor de Arquivos DDL. Outro caminho é selecionar o menu “Exibir” e escolher a opção “Editor de Arquivos DDL”.

Definindo as estruturas físicas!

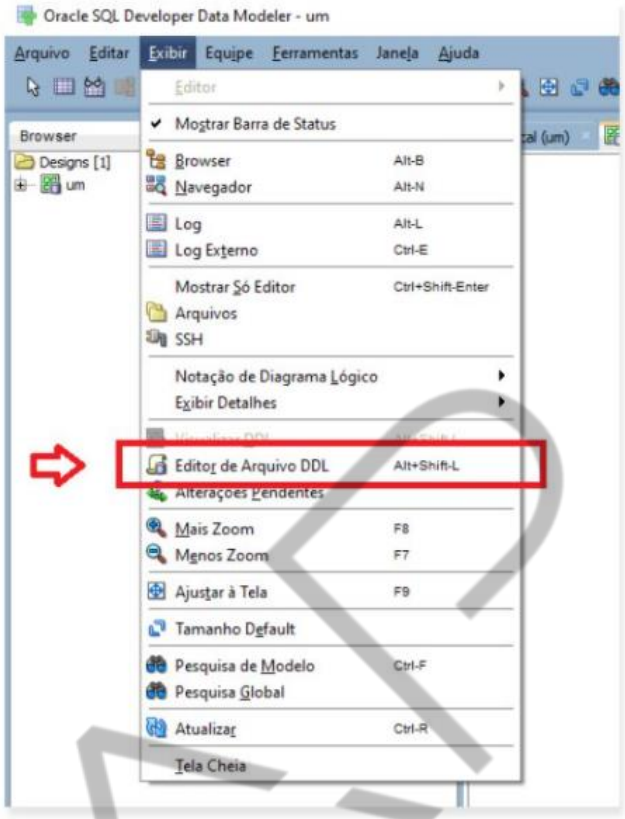

Figura 24 – Caminho para o Editor de Arquivo DDL
Fonte: Oracle SQL Developer Data Modeler (2021)

O editor de Arquivo DDL será iniciado, clique em **Gerar:**

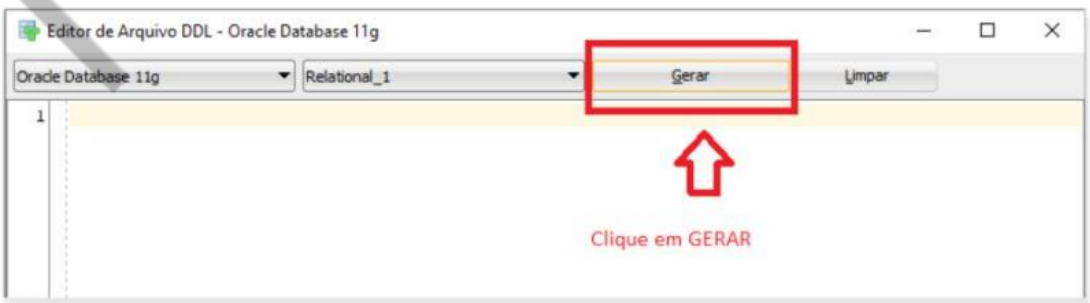

Figura 25 – Tela do Editor de Arquivo DDL
Fonte: Oracle SQL Developer Data Modeler (2021)

Na tela “Opções de Geração de DDL”, clique em “OK”. Opcionalmente desmarque as caixas “Atribuído a Esquemas”, “Esquemas”, “Tipos Estruturados” e “Tipos de Coleção”:

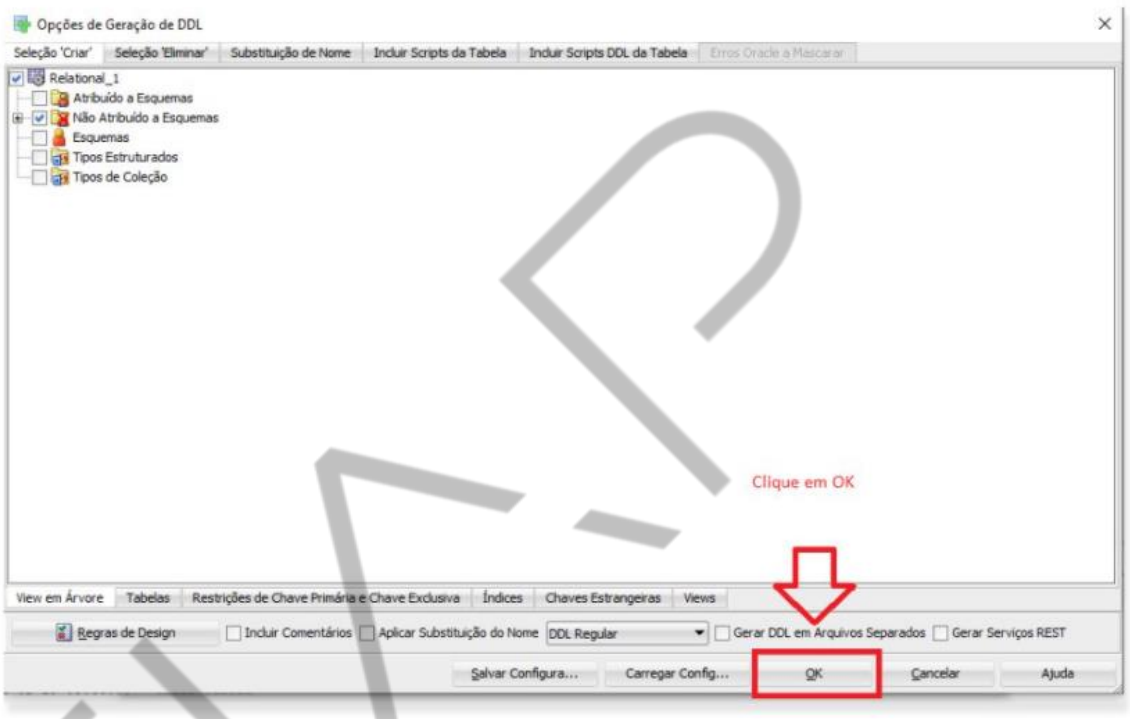

Figura 26 – Tela de Opções de Geração de DDL
Fonte: Oracle SQL Developer Data Modeler (2021)

As instruções DDL foram geradas. Ao clicar em Salvar, será gerado um arquivo com o *script* com os comandos DDL, que podem ser usados para criar as tabelas no banco de dados:

Definindo as estruturas físicas!

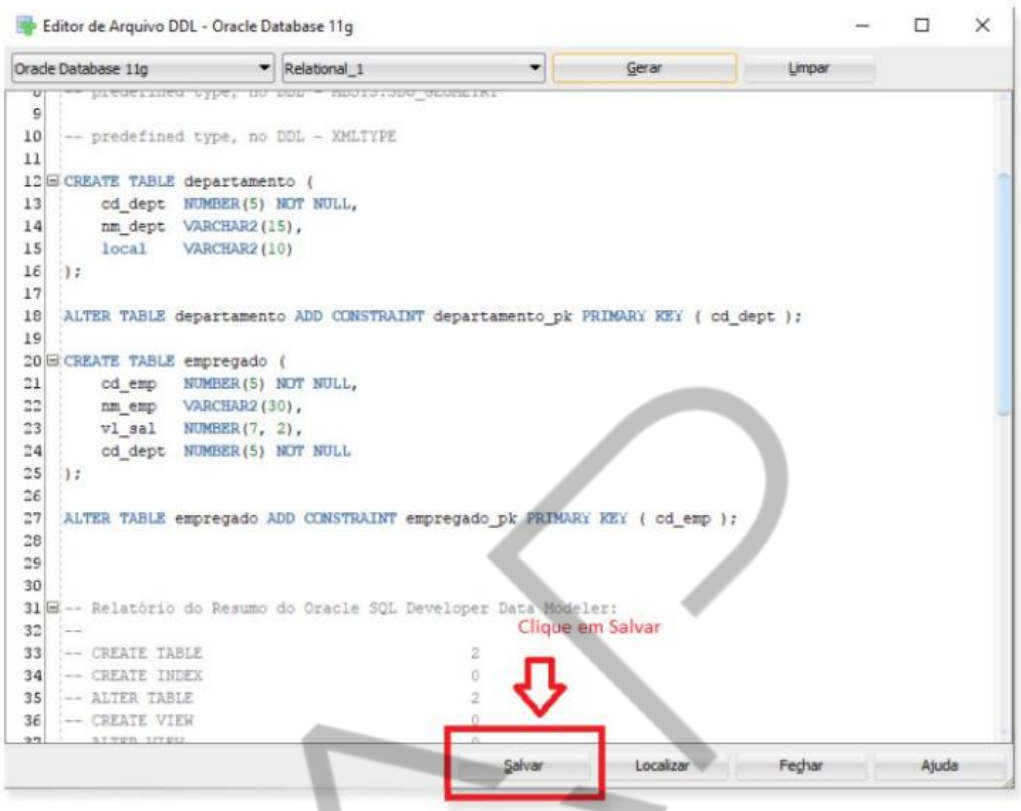

Figura 27 – Tela do Editor de Arquivo DDL com a opção de salvar habilitada
Fonte: Oracle SQL Developer Data Modeler (2021)

Definindo as estruturas físicas!

REFERÊNCIAS

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**: Fundamentos e Aplicações. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

HEUSER, C. A. **Projeto de Banco de Dados**. 6. ed. Rio Grande do Sul: Bookman, 2009. v. 4 (Série Livros Didáticos).

MACHADO, F. N. R. **Banco de Dados** - **Projeto e Implementação**. 2. ed. São Paulo: Érica, 2008.

MENDES, J. C. **Banco de Dados Transformação ER – Relacional**. 2011. Disponível em: https://jeancarlomendes.files.wordpress.com/2011/02/9-transformac3a7c3a3o-entre-modelos.pdf. Acesso em: 08 mar. 2021.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de Dados**: Implementação em SQL, PL/SQL e Oracle 11g. São Paulo: Pearson Education do Brasil, 2013.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 5. ed. Rio de Janeiro: Campus, 2006.