**Introdução - Fundamentos de Banco de Dados**

Esta disciplina se propõe ao estudo dos principais fundamentos de bancos de dados, para isto apresentando a modelagem de dados, operações relacionas aos bancos de dados e as técnicas de normalização de um modelo propiciando ao aluno o conhecimento para projetar um Banco de Dados desde o início.

Acompanhe a apresentação da disciplina, elaborada pelo Professor Marino Hilário Catarino - [vídeo](https://www.youtubeeducation.com/watch?v=G5w3lxeJZOE).

**Planejamento de estudos**

Esta disciplina é de 80 horas

Está previsto que você use cerca de 4 horas/semana

**Objetivos gerais**

Apresentar uma base sólida em fundamentos de banco de dados; introduzir como representar soluções de problemas utilizando modelos e diagramas de banco de dados, a fim de que seja possível organizar dados e informação em estruturas de banco de dados.

**Parte 1 -** [**Introdução aos sistemas de bancos de dados**](https://sites.google.com/faculdadeimpacta.com.br/fbd-p1?pli=1&authuser=2) **-** [**Vídeo**](https://www.youtubeeducation.com/watch?v=LiT1NoTy3ds)

**Apresentação**

Um banco de dados (BD) é uma coleção ou conjunto de dados inter-relacionados com uma estrutura regular que é armazenado de forma organizada para serem utilizados em situações específicas.

O Dado é toda a informação que possui um significado conforme um determinado contexto e que pode ser armazenado. Podemos exemplificar como um catálogo de filmes ou um sistema de controle de estoque em uma empresa.

**Objetivos**

A grande maioria dos bancos de dados são armazenados e acessados através de um software denominado Sistema Gerenciador de Banco de Dados (SGBD). O SGBD possui recursos que permitem uma interação do usuário e a manipulação das informações do banco de dados. Temos como exemplos de SGBD o SQL Server, Oracle, MySQL e o PostgreSQL. Em um SGBD, o modelo de dados mais utilizado para representar e armazenar os dados é o modelo relacional, no qual as estruturas são representadas por tabelas compostas por linhas e colunas formando as tuplas.

**Competências da disciplina**

Uma vantagem principal do SGBD é a persistência dos dados, porém ele possui outras propriedades que são muito importantes: abstrações de dados, escalabilidade, durabilidade, consistência e controle de concorrência, entre outras características.

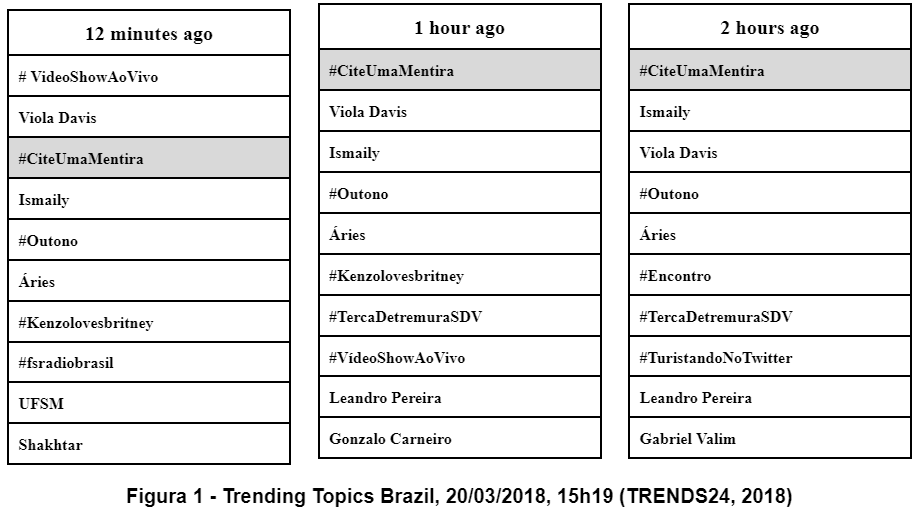
**Introdução**

Existem muitos tipos de banco de dados e eles estão presentes em nosso dia-a-dia: quando você vai ao supermercado, quando envia uma mensagem para alguém pelo celular, quando lê seus e-mails. Todas estas informações são armazenadas em bancos de dados e é o que estudaremos neste material.

Segundo Rangel et al (2014), os bancos de dados são amplamente utilizados e constituem a parte essencial de quase todas as empresas, independentemente do seu ramo de atividade.

Todas estas informações são dependentes de bancos de dados pois demandam grandes volumes de dados para serem armazenados e recuperados, sem contar mecanismos de recuperação em caso de falhas.

Mas, existe diferença entre informação e dados? É de fundamental importância conhecer a diferença entre estes dois conceitos. De acordo com Rangel et al (2014), “Os dados são considerados fatos brutos, o que indica que os fatos ainda não foram processados para revelar seu significado”. Por exemplo, todas as mensagens enviadas pelo Twitter em 1 minuto que, segundo Domo (2018), são da ordem de 456mil por minuto! Certamente, há muita informação nestas mensagens (twits) mas, primeiro é necessário que se faça um tratamento delas. Da mesma forma, pense em quantas compras são registradas por dia em um rede de lojas como o Carrefour ou o Walmart. Todos são dados brutos precisando ser lapidados. Estes dados “lapidados” são a informação, como, por exemplo: “Quanto se vendeu de sabão em pó hoje na rede de lojas?” ou, no caso do twitter, os “trending topics”. Um exemplo dos Trending Topics pode ser visto na figura 1.



Para se conseguir extrair tais informações de um número tão grande de dados, é preciso conhecer seu contexto e sua finalidade. As informações são essenciais para as empresas na tomada de decisão, independente da área que estejam inseridas: governamental, privado ou filantrópicas.

Antes do uso dos SGBDs (Sistemas Gerenciadores de Banco de Dados), as empresas armazenavam seus dados em sistemas de arquivos, que não tinham a capacidade nem de compartilhamento nem de proteção a estes dados.

Mas, e o que são os SGBDs? São os softwares que gerenciam e guardam os dados, dentre os quais podemos destacar o Oracle, SQL Server, Postgree, MySQL, Firebird, etc. Os softwares citados são apenas os relacionais, já que existem diversos tipos de SGBD, sendo estes os mais utilizados atualmente e fruto de avaliação neste material.

Para criar um banco de dados em um SGBDR (Sistema Gerenciador de Banco de Dados Relacional), precisamos criar um projeto que possui três níveis de abstração: Conceitual, Lógico e Físico.

O nível conceitual é composto pelo Projeto Conceitual que é o Modelo Entidade-Relacionamento utilizando-se o Diagrama Entidade-Relacionamento, onde estará o nível mais alto de abstração do projeto e, nenhum SGBD deve ser levado em consideração. Posteriormente, baseado neste projeto conceitual, será elaborado o projeto lógico que está situado no modelo Relacional, portanto, fazendo uso do Esquema de Relações do Banco de Dados, as quais devem ser aplicadas as regras de normalização para eliminação das redundâncias que possam haver. Neste nível, um SGBDR já pode ser considerado pois há pequenas diferenças entre eles que precisam ser levadas em conta.

“A função da normalização é atuar como um filtro sobre as entidades e os relacionamentos, eliminando alguns elementos sem causar perda de informação nas entidades e nas relações.” (RANGEL et al 2014). Estas regras de normalização são chamadas de formas normais e são em um total de 6, sendo:

1FN – Primeira Forma Normal

2FN – Segunda Forma Normal

3FN – Terceira Forma Normal

FNBC – Forma Normal de Boyce Codd

4FN – Quarta Forma Normal

5FN – Quinta Forma Normal

Findas estas fases, inicia-se o projeto físico que é elaboração dos scripts de construção das tabelas no SGBDR selecionado. Baseando-se então no Projeto Lógico que contém o esquema de relações do banco de dados, cria-se o script em Linguagem SQL, no dialeto do SGBDR escolhido para, em seguida aplicá-lo e criar efetivamente o conjunto de tabelas.

A Linguagem SQL (Structure Query Language ou Linguagem Estruturada de Consulta) é a que se utiliza para criação e manutenção dos bancos de dados relacionais. Ela está subdividida em três, a saber:

DDL – Data Definition Language ou Linguagem de Definição de Dados

DML – Data Manipulation Languagem ou Linguagem de Manipulação dos Dados

DCL – Data Control Language ou Linguagem de Controle de Dados

De acordo com Rangel et al (2014), a DDL (Data Definition Language) fornece recursos para definir objetos e controlar dados. São estes comandos que serão responsáveis pela estruturação do banco de dados, como, por exemplo, a criação das tabelas e índices.

Já o subconjunto DML (Data Manipulation Language), conforme afirmam Rangel et al (2014), objetivam mecanismos para manipular e gerenciar o banco de dados, permitindo assim que sejam inseridos, modificados, apagados e pesquisados quaisquer dados armazenados neles.

Por fim, a sub linguagem DCL (Data Control Language) é responsável pelo controle de acesso aos dados do banco de dados, com o gerenciamento de usuários e a criação de regras para realização de pesquisas, inserções, modificações e exclusões, por exemplo.

**Glossário de Conceitos**

1) Atributo: abstração de uma propriedade de uma entidade ou de um relacionamento.

2) Banco de Dados: sistema de armazenamento de dados cujo objetivo é registrar e guardar

informações importantes que poderão ser acessadas quando necessário.

3) BI (Business Intelligence ou Inteligência de Negócios): utiliza conceitos em que as informações são coletadas, armazenadas e analisadas, tendo como base fatos reais e/ou hipóteses. Esses sistemas auxiliam na gestão organizacional e no processo de tomada de decisões.

4) Dado atômico: tipo de dado considerado básico, ou seja, indivisível.

5) Dado não atômico: tipo de dado considerado complexo, divisíveis (fragmentados).

6) Entidade: abstração de um fato do mundo real para o qual se deseja manter seus dados no banco de dados.

7) ERP’s: são sistemas de gestão empresarial que possibilitam a integração de todos os dados e processos de uma empresa, melhorando o fluxo de informações.

8) Gerenciamento de Banco de Dados: utiliza Sistemas Gerenciadores de Banco de Dados (SGBDs).

9) Modelagem Conceitual: nível mais alto de abstração cujo objetivo é representar os requisitos de dados do domínio da aplicação (independente do modelo de banco de dados).

10) Modelagem Lógica: representação da modelagem conceitual em um modelo de banco de dados.

11) Modelagem Física: constitui um esquema SQL para a modelagem lógica (depende exclusivamente do SGBD).

12) Relacionamento: abstração de uma associação entre (ocorrências de) entidades.

13) Sistema Gerenciador de Banco de Dados (SGBD): coleção de programas responsáveis pela criação e manutenção de banco de dados. (RANGEL et al 2014)

**Perspectiva Histórica**

Com o passar do tempo, as empresas descobriram que a quantidade de dados gerados pelos sistemas de informação aumentava imensamente e que tornava-se cada vez mais dispendioso o processo de armazenagem destes dados. Desta forma, percebeu-se que era valido o esforço para descobrir uma forma mais eficiente de armazenar todos estes dados. (SANCHES, 2005)

Da mesma forma que muitas tecnologias da computação, os fundamentos de banco de dados relacionais surgiram na empresa IBM, em meados dos anos 1960 e 1970, como fruto de pesquisas por ela desenvolvidas. Diversas outras pesquisas desta época culminaram em outros modelos de bancos de dados como os hierárquicos e de rede. (RANGEL et al, 2014)

Em 1970, Ted Codd, então pesquisador da IBM publicou o primeiro artigo sobre bancos de dados relacionais, o qual tratava sobre o uso de cálculo e álgebra relacional. Ele procurava o desenvolvimento de um sistema que fosse capaz de acessar as informações com comandos em inglês. Como fruto deste artigo, a IBM idealizou o projeto System R. (SANCHES, 2005)

Este projeto visava criar um sistema de banco de dados relacional que deveria tornar-se um produto, o que posteriormente ocorreu, sendo primeiro o SQL/DS e depois o DB2, que ela produz até os dias atuais. A linguagem que foi criada pelo projeto System/R foi a linguagem SQL (Structure Query Language), que tornou-se o padrão para bancos de dados relacionais e atualmente é um padrão ISO (International Organization for Standardization). (RANGEL et al, 2014)

Os primeiros protótipos foram utilizados por muitas organizações, como o MIT Sloan School of Management. A IBM, no entanto, manteve o System/R em segundo plano por vários e decisivos anos. (SANCHES, 2005)

No final dos anos 1970, surge uma empresa, a Oracle, fruto do vislumbre de seu criador, Larry Elison que percebeu uma oportunidade onde outras empresas não haviam visto. Ele encontrou uma descrição de um protótipo funcional de um banco de dados relacional e descobriu que nenhuma empresa tinha se empenhado em comercializar essa tecnologia. (WIKIPEDIA, 2010)

Ellison e os cofundadores da Oracle Corporation, Bob Miner e Ed Oates, perceberam que havia um tremendo potencial de negócios ali, tornando-a assim uma das maiores empresas de software empresarial do mundo, cuja primeira versão comercial foi lançada em 1979 sob o nome de RSI. Atualmente chama-se Oracle Database e encontra-se na versão 12g. (FARIAS, 2018)

Vídeo - UNIVESP. Banco de dados - aula 01 - [visão geral sobre banco de dados e motivação](https://youtu.be/pmAxIs5U1KI?si=ay5wrMjtskeBYiOd).

**Parte 2 -** [**SGBD e modelos de bancos de dados**](https://sites.google.com/faculdadeimpacta.com.br/fbd-p2?pli=1&authuser=2) **-** [**Vídeo**](https://www.youtubeeducation.com/watch?v=_zJu06l9vbY)

**Fundamentos de Banco de Dados**

O que é um banco de dados? Por que utilizamos banco de dados? De onde surgiram os bancos de dados? O que são os Sistemas Gerenciadores de Bancos de Dados e para que eles servem? Estas são algumas das questões que se pretende responder neste capítulo. Segundo Silberschatz, Korth e Sudarshan (2008), um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados.

**Bancos de Dados**

Segundo Date (2000), em essência, um sistema de bancos de dados é apenas um sistema computadorizado de armazenamento de registros. O banco de dados pode, ele próprio, ser visto como o equivalente eletrônico de um armário de arquivamento.

O crescimento do uso de computadores está intimamente relacionado a utilização de bancos de dados pelos sistemas computacionais. O termo banco de dados é tão utilizado que precisa, primeiramente ser definido: “Um banco de dados é uma coleção de dados relacionados”.

Já por dados, podemos tomar quaisquer fatos conhecidos que possam ser armazenados e que possuam um significado implícito, como, por exemplo, nomes, números de documentos, telefones e e-mails.

Segundo Rangel et al (2014), “É importante que você entenda a diferença entre dados e informações. Os dados são considerados fatos brutos, o que indica que os fatos ainda não foram processados para revelar seu significado.”. Desta forma, ainda de acordo com o autor, “Para revelar seu significado, os dados brutos são processados de maneira apropriada, gerando, assim, as informações.”.

Desta forma, os bancos de dados começaram a crescer e o volume de dados tornava trabalhoso e dispendioso sua manutenção. A IBM iniciou um grupo de pesquisa, denominado System/R para pesquisar sobre uma maneira mais eficiente de armazenar-se administrar-se estes dados, de forma que posteriormente pudesse se tornar um produto. Um de seus pesquisadores (Ted Codd) publicou um artigo descrevendo detalhadamente como deveriam ser o uso de cálculos e álgebra relacional. Um dos leitores deste artigo era Larry Elison que posteriormente seria um dos fundadores da Oracle, o primeiro SGBDR (Sistema Gerenciador de Banco de Dados Relacional) comercial e atualmente, um dos principais do mercado. (FARIAS, 2018)

**Sistema Gerenciador de Bancos de Dados**

Um SGBD (Sistema Gerenciador de Banco de Dados) é uma coleção de programas que permite aos usuários criar e manter um banco de dados, permitindo-se a definição, manipulação e compartilhamento de dados entre aplicações diversas e seus usuários. (EMASRI; NAVATHE, 2011)

São várias as características que distinguem o funcionamento de um SGBD em relação ao modus operandi anterior a eles com uso de arquivos. Quando se trabalha com arquivos, cada programa deve definir e implementar todos os arquivos necessários para a execução de uma aplicação. (EMASRI; NAVATHE, 2011)

Um bom exemplo destes tipos de sistemas foram os programas em COBOL (Common Business Oriented Language ou Linguagem Comum Orientada para os Negócios) que foi uma linguagem de programação orientada ao processamento de bancos de dados comerciais. O COBOL foi criado no segundo semestre de 1959! A linguagem possui várias versões de padronização, sendo a última em 2002, que substituiu a versão do COBOL-85 (1985). (WIKIPEDIA, 2004)

Os programas em COBOL, como a versão Microbase (brasileira) trabalhavam com arquivos próprios, basicamente, texto ASCII, que podia ser acessado por qualquer editor de texto e ter seus dados modificados. Não havia controle de transações e a segurança era a que a aplicação fosse capaz de fornecer.

Por exemplo, em COBOL, a organização de arquivos indica como os registros são organizados em um arquivo, podendo ser sequencial ou indexado. Cada modo de organização podia ter diferentes formas de acesso. Isso significa que, a cada programa, estas configurações precisavam ser especificadas e, em outras linguagens seriam completamente diferentes. No caso de uma modificação na estrutura de um arquivo, todos os programas que fazem acesso aquele arquivo, independente da forma de acesso, se para escrita ou leitura, deveriam ser modificados. Este foi o grande problema do “Bug do Milênio”, onde, mesmo arquivos que não utilizassem as datas, se faziam acesso a um arquivo que contivesse datas, precisavam ser modificados.

Agora, multiplique este trabalho por centenas de programas. Insano! Se uma modificação semelhante precisar ser feita em um banco de dados, o trabalho será infinitamente menor.

| 01 | IDENTIFICATION DIVISION. |
| --- | --- |
| 02 | PROGRAM-ID. HELLO. |
| 03 |  |
| 04 | ENVIRONMENT DIVISION. |
| 05 | INPUT-OUTPUT SECTION. |

| 06 | FILE-CONTROL. |
| --- | --- |
| 07 | SELECT STUDENT ASSIGN TO OUT1 |
| 08 | ORGANIZATION IS SEQUENTIAL |
| 09 | ACCESS IS SEQUENTIAL |
| 10 | FILE STATUS IS FS. |
| 11 |  |
| 12 | DATA DIVISION. |
| 13 | FILE SECTION. |
| 14 | FS STUDENT |
| 15 | 01 STUDENT-FILE. |
| 16 | 05 STUDENT-ID PIC 9(5). |
| 17 | 05 NAME PIC A(25). |
| 18 | 05 CLASS PIC X(3). |
| 19 |  |
| 20 | WORKING-STORAGE SECTION. |
| 21 | 01 WS-STUDENT. |
| 22 | 05 WS-STUDENT-ID PIC 9(5). |
| 23 | 05 WS-NAME PIC A(25). |
| 24 | 05 WS-CLASS PIC X(3). |
| 25 |  |
| 26 | PROCEDURE DIVISION. |
| 27 | OPEN EXTEND STUDENT. |
| 28 | MOVE 1000 TO STUDENT-ID. |
| 29 | MOVE ‘Tim”TO NAME. |
| 30 | MOVE ‘10’TO CLASS. |
| 31 | WRITE STUDENT-FILE |

| 32 | END-WRITE. |
| --- | --- |
| 33 | CLOSE STUDENT. |
| 34 | STOP RUN. |

**Listagem 1 – Programa em COBOL. (TUTORIALSPOINT, 2018)**

Percebe-se então que o controle de redundância de dados, de segurança da informação e a eficiência das consultas ficam seriamente comprometidas. Nestas situações, o SGBD apresenta-se como solução mais robusta e indicada. Entretanto, não se deve usar SGBDs indiscriminadamente.

**Quando não utilizar um Sistema Gerenciador de Bancos de Dados**

Em algumas situações, como em aplicações muito simples e estáveis, provavelmente, a utilização do SGBD torne-a demasiadamente complicada, fazendo com que se perca sua simplicidade.

Segundo Elmasri e Navathe (2011, p. 17), “Apesar das vantagens de usar um SGBD, existem algumas situações em que esse sistema pode envolver custos adicionais desnecessários, que não aconteceriam no processamento de arquivos tradicional”.

Os custos adicionais quando da utilização de um SGBD devem-se aos seguintes fatores:

Alto investimento inicial em hardware, software e treinamento.

A generalidade que um SGBD oferece para a definição e o processamento de dados.

Esforço adicional para oferecer funções de segurança, controle de concorrência, recuperação e integridade. (ELMASRI; NAVATHE, 2011)

Desta forma, pode ser mais vantajoso utilizar sistemas de arquivos nas seguintes situações:

* Aplicações de banco de dados simples e bem definidas, para as quais não se espera muitas mudanças.
* Requisitos rigorosos, de tempo real, para alguns programas de aplicação, que podem ser atendidos devido as operações extras executadas pelo SGBD.
* Sistemas embarcados com capacidade de armazenamento limitada, onde um SGBD de uso geral não seria indicado.
* Nenhum acesso de múltiplos usuários aos dados. (ELMASRI; NAVATHE, 2011)

Vídeo - [SGBD - Conceitos Básicos](https://youtu.be/NnG7f60XPkQ?si=XB8GdUHBpMWx0mHa)

**Parte 3 -** [**Transações**](https://sites.google.com/faculdadeimpacta.com.br/fbd-p3?pli=1&authuser=2) **-** [**Vídeo**](https://www.youtubeeducation.com/watch?v=1_jukkAYRsQ)

**Sistema de Banco de Dados**

A arquitetura dos SGBDs evolui constantemente. Essa evolução espelha as mudanças que ocorrem na computação, que evoluiu de grandes mainframes para servidores web e de banco de dados. Uma arquitetura básica de um SGBD é a cliente/servidor, onde um módulo é o cliente e é executado na estação do usuário (estação cliente) e outro módulo é executado no servidor (servidor de banco de dados).

**Modelos de dados, esquemas e instâncias**

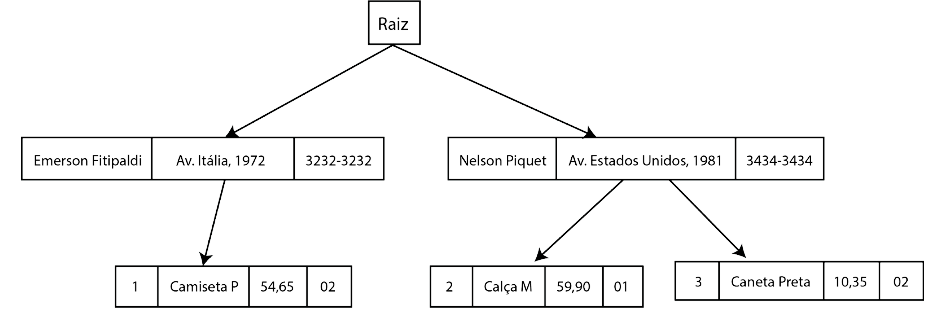
Uma característica fundamental de um banco de dados é a possibilidade de se obter abstração de dados em algum nível. Elmasri e Navathe (2011, p. 19) destacam que “A abstração de dados, geralmente, se refere a supressão de detalhes da organização e armazenamento de dados, destacando recursos essenciais para um melhor conhecimento desses dados.”. Assim, é possível, num primeiro momento, afastar-se das especificações de um SGBD para se pensar exclusivamente na organização dos dados. Outra característica é a de que os bancos de dados fornecem esta abstração de dados no nível de detalhamento ideal para cada tipo de usuário.

Heuser (2009, p. 16) define Modelo de Dados como “uma descrição dos tipos de informações que estão armazenadas em um banco de dados.”. Por exemplo, para uma escola, o modelo de dados poderia informar que o banco de dados armazena informações sobre alunos e que, para cada aluno, são armazenados o RA (Registro do Aluno), nome, endereço, telefone e e-mail. É importante salientar que o modelo de dados não informa quais alunos estão armazenados mas que o banco de dados armazena informações sobre os alunos.

Surgiram assim, vários modelos de dados, tais como, Rede, Hierárquico ou Relacional. Todo SGBD deve suportar um modelo que permita a representação dos dados de uma realidade. (FANDERUFF, 2003)

**Modelo hierárquico**

O modelo hierárquico surgiu nos anos de 1960 e os dados são organizados em formato de hierarquias ou árvores. Neste modelo os nós das hierarquias possuem ocorrências de registros,sendo que cada registro é uma coleção de atributos onde cada atributo contém somente uma informação. O primeiro registro da hierarquia é o registro-pai e todos os demais são denominados de registros-filhos.

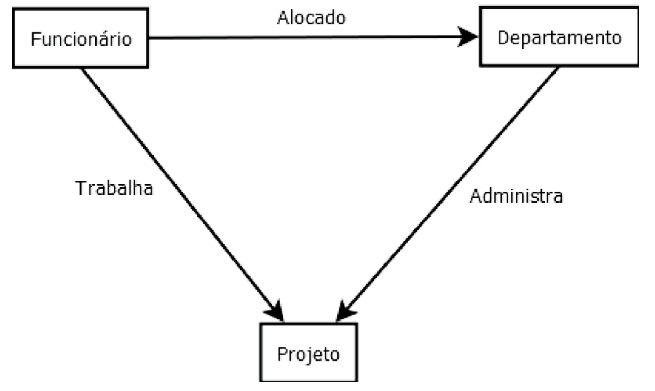


**Modelo de rede**

Este modelo foi utilizado principalmente no final dos anos de 1960 e no decorrer da década de 1970. Sua forma de organizar os dados é utilizando uma estrutura formada por várias listas, definindo assim uma intrincada rede de ligações. (FANDERUFF, 2003)

“Similar ao modelo hierárquico, os dados de rede são organizados em tipos de registros e ligações entre dois tipos de registro. Não existe restrição hierárquica.” (FANDERUFF, 2003, p. 5).

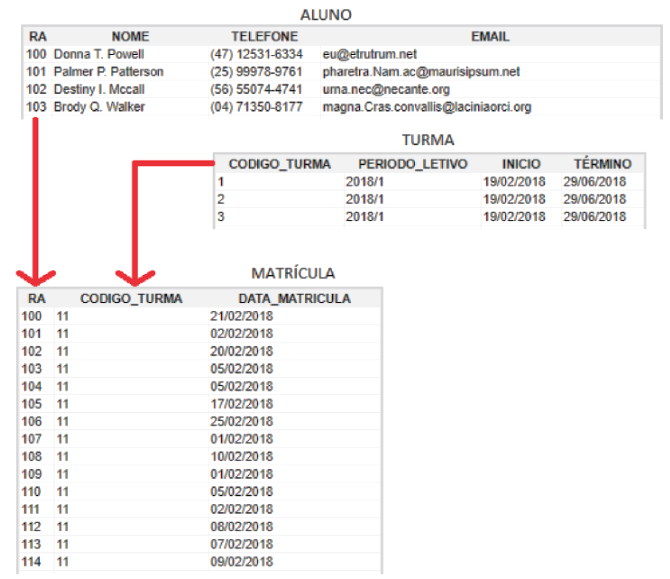
Ainda, segundo a autora, uma rede é, portanto, um conjunto ilimitado de nós. Não há o sentido de raiz e, uma desvantagem é o fato de que, caso o banco de dados tenha muitos tipos de entidades, pode resultar em esquemas complexos de relacionamentos.



**Modelo relacional**

O modelo relacional foi definido por Ted Codd. O nome inicial do projeto, dado pela IBM era System R. Baseado neste projeto surgiu a primeira versão da Linguagem SQL (Structured Query Language ou Linguagem Estruturada de Consulta) e que é utilizada até os dias atuais pelos SGBDs Relacionais, cujos exemplos mais famosos podemos citar o Oracle, o SQL Server e o DB2, dentre os de cunho proprietário e o MariaDB, Postgree e Firebird dentre os Free (Livres). (FANDERUFF, 2003)

Segundo Fanderuff (2003, p.6), “O objeto básico tratado pelo modelo relacional é a entidade ou relação, que pode ser definida como um objeto do mundo real, concreto ou abstrato”. Os bancos de dados relacionais visam eliminar a redundância e, é o mais utilizado pelo mercado nos dias atuais.



**Categorias de modelos de dados**

Muitos tipos de modelos foram propostos e eles são classificados de acordo com os tipos de conceitos que utilizam para descrever a estrutura do banco de dados. Por exemplo, para Elmasri e Navathe (2011, p.20), os “Modelos de dados de alto nível ou Modelos Conceituais oferecem conceitos que são próximos ao modo como muitos usuários percebem os dados”. Os autores ainda afirma que os “Modelos de dados de baixo nível ou físicos oferecem conceitos que descrevem os detalhes de como os dados são armazenados no computador.”.

Estes são os extremos mas, entre eles “está uma classe de modelos de dados representativos que oferece conceitos que podem ser facilmente entendidos pelos usuários finais, mas que não está muito longe do modo como os dados são organizados e armazenados pelo computador.” (ELMASRI; NAVATHE, 2011, p. 20)

Os modelos de dados conceituais utilizam os conceitos de entidades, atributos e relacionamentos. Segundo Emasri e Navathe (2011), uma entidade representa um objeto ou conceito do mundo real, como, por exemplo, um funcionário, um produto ou um veículo; os atributos representam as propriedades de interesse de cada entidade e os relacionamentos representam associações entre duas ou mais entidades (ELMASRI; NAVATHE, 2011).

**Transações**

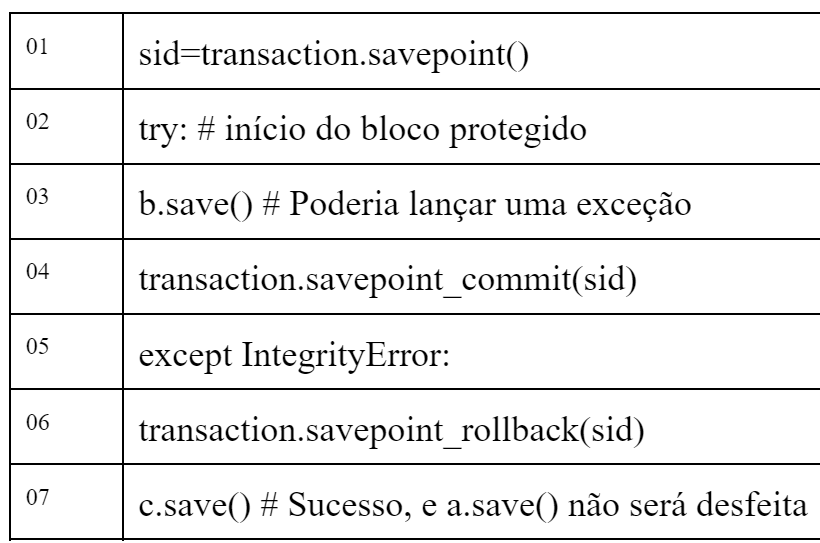
Segundo Silberschatz, Korth e Sudarshan (2006), “uma transação é uma unidade de execução do programa que acessa e possivelmente atualiza vários itens de dados”.

Para garantir a integridade dos dados, é necessário que o sistema de banco de dados mantenha as seguintes propriedades das transações:

* **Atomicidade:** Em uma transação envolvendo duas ou mais partes de informações discretas, ou a transação será executada totalmente ou não será executada, garantindo assim que as transações sejam atômicas.
* **Consistência:** A transação cria um novo estado válido dos dados ou em caso de falha retorna todos os dados ao seu estado antes que a transação foi iniciada.
* **Isolamento:** Uma transação em andamento mas ainda não validada deve permanecer isolada de qualquer outra operação, ou seja, garantimos que a transação não será interferida por nenhuma outra transação concorrente.
* **Durabilidade:** Dados validados são registados pelo sistema de tal forma que mesmo no caso de uma falha e/ou reinício do sistema, os dados estão disponíveis em seu estado correto. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006)

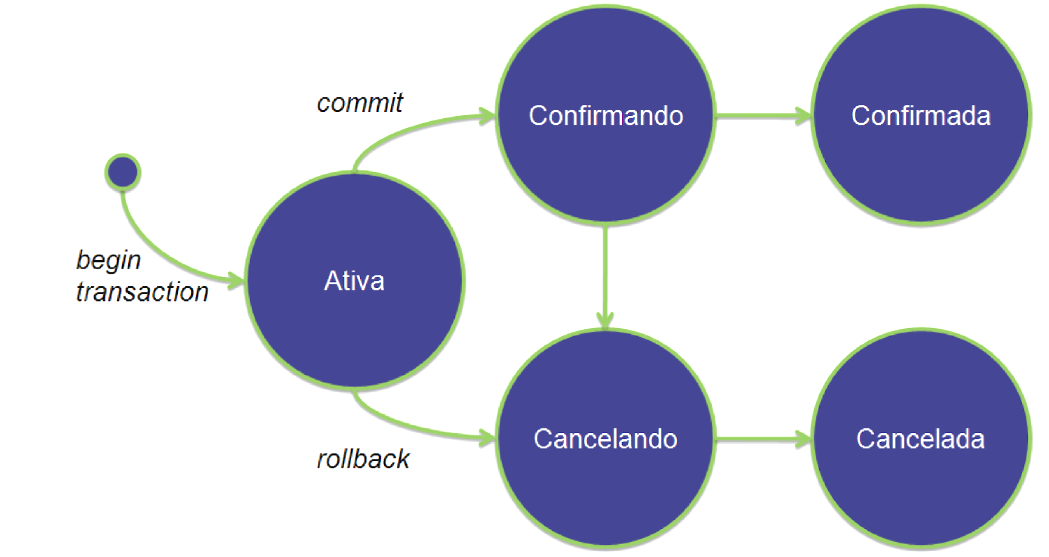
Uma transação é criada quando se executam comandos de manipulação de dados da Linguagem SQL (INSERT, DELETE ou UPDATE), que inserem, apagam ou modificam dados. Em uma transação podem haver um ou vários destes comandos combinados. Neste caso, a transação deve ser formalmente iniciada com o comando START TRANSACTION que somente pode ser encerrada com os comandos COMMIT ou ROLL BACK.

Quando todos os comandos da transação são executados com sucesso, o comando COMMIT encerra a transação, efetivando todas as modificações feitas no banco de dados. Entretanto, se algum erro acontecer, o comando ROLL BACK é emitido e o banco de dados retorna ao estado inicial, antes do início da transação, efetivando, portanto, todas as modificações ou nenhuma delas.



Observando o código na imagem acima, percebe-se que existe um bloco protegido (try… except). Na linha 1 está o comando que inicia a transação. Na linha 2, o comando de início do bloco protegido. Na linha 3, o comando que salva as modificações no banco de dados e na linha 4 o comando COMMIT, que efetiva as modificações realizadas no banco de dados. Perceba que, caso aconteça algum problema ou erro durante a execução da linha 4, como, por exemplo, falha da rede, o bloco protegido desviará o processamento para a linha 6 (exceção) e o comando ROLL BACK então será emitido.

Em uma transação, seja qual for o seu encerramento, com COMMIT ou com ROLL BACK, o resultado dela é perene, ou seja, não poderá ser modificado mais, salvo outra transação altere os mesmos dados, assim, se dados forem incluídos e transação for encerrada com COMMIT, eles estarão inseridos no banco de dados e disponíveis para os demais usuários e, por outro lado, se a transação for encerrada com ROLL BACK, os dados não terão sido inseridos. O mesmo raciocínio vale para comandos de modificação ou exclusão de dados.



**Arquitetura de três esquemas**

De acordo com Emasri e Navathe (2011), as três das quatro características importantes da abordagem de banco de dados são:

* Uso de um catálogo para armazenar a descrição (esquema) tornando-o assim, autodescritivo.
* Isolamento de programas e dados.
* Suporte para múltiplas visões do usuário.

A arquitetura de três esquemas tem por objetivo separar as aplicações do usuário do banco de dados físico, como pode ser visto na figura abaixo. Quando é usado o termo descrição do banco de dados, entendemos como a chamada de “esquema de uma banco de dados” que é especificada durante um projeto de banco de dados.

O nível interno tem um esquema interno, que descreve a estrutura do armazenamento físico do banco de dados. O esquema interno usa um modelo de dados físico e descreve os detalhes completos do armazenamento de dados e caminhos de acesso para o banco de dados.

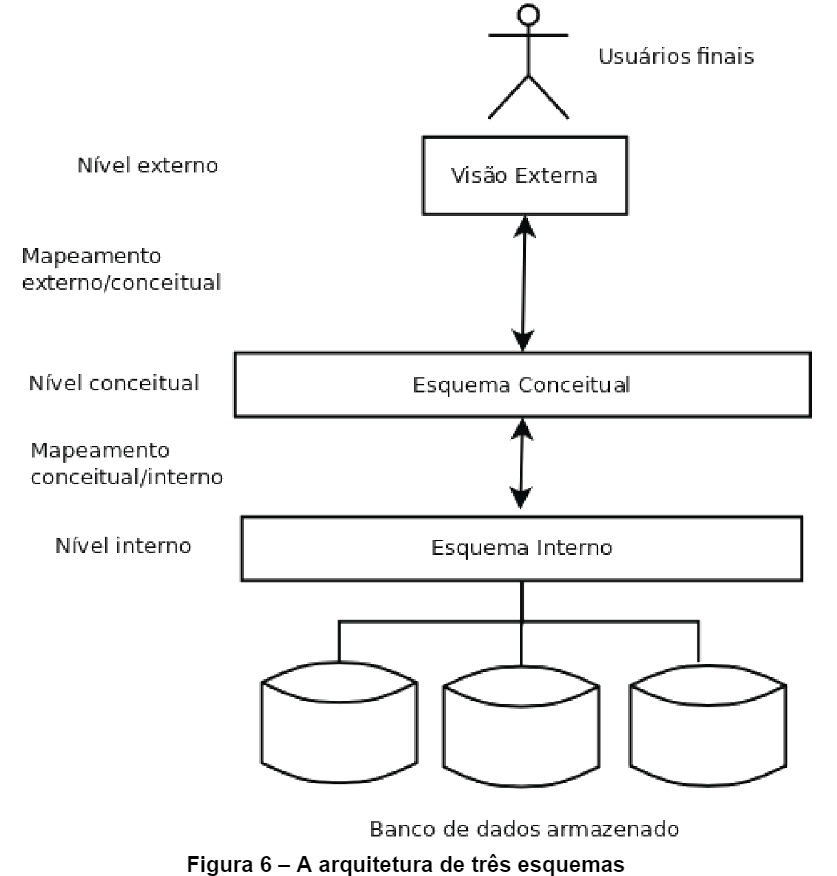
O nível conceitual tem um esquema conceitual, que descreve a estrutura do banco de dados inteiro para uma comunidade de usuários. O esquema conceitual oculta os detalhes das estruturas de armazenamento físico e se concentra na descrição de entidades, tipos de dados, relacionamentos, operações do usuário e restrições. Normalmente, um modelo de dados representativo é usado para descrever o esquema conceitual quando um sistema de banco de dados é implementado.

O nível externo ou de visão inclui uma série de esquemas externos ou visões do usuário. Cada esquema externo descreve a parte do banco de dados em que um grupo de usuários em particular está interessado e oculta o restante do banco de dados do grupo de usuários. De forma semelhante ao nível conceitual, cada esquema externo é comumente implementado usando um modelo de dados representativo, baseado em um projeto de esquema externo. (ELMASRI; NAVATHE, 2011, p. 22)

Devido a arquitetura de três esquemas permite que um usuário possa visualizar os níveis de esquema em um sistema de banco de dados. Segundo Elmasri e Navathe (2011), “A maioria dos SGBDs não separa os três níveis completa e explicitamente, mas dá suporte a eles de alguma forma”.

É importante destacar que os três esquemas são apenas descrições dos dados pois os dados realmente armazenados estão no nível físico. “Se a solicitação for uma recuperação, os dados extraídos do banco de dados armazenado devem ser reformatados para corresponder à visão externa do usuário”. (ELMASRI; NAVATHE, 2011)

Os processos de transformação e os resultados entre os níveis são chamados de mapeamentos.



**Independência dos dados**

Pode-se dizer que é possível efetuar alterações no esquema ou no nível de um banco de dados, sem alterar um nível superior. A arquitetura de três esquemas pode ser utilizada para ser usada para explicar melhor o conceito de independência de dados.

Independência lógica de dados: Permite alterar apenas o nível conceitual, não havendo nenhuma alteração no nível externo ou nas aplicações do usuário.

Independência física de dados: Permite alterar o nível interno sem ter que alterar o nível conceitual, nível externo ou as aplicações do usuário.

[Vídeo](https://youtu.be/tA8OaWGaBxg?si=9zGSS3PxkR_xJXhq) - BISPO, José. Banco de dados e SGBD.

**Parte 4 -** [**Introdução ao modelo Entidade-Relacionamento**](https://sites.google.com/faculdadeimpacta.com.br/fbd-p4?pli=1&authuser=2) **-** [**Vídeo**](https://www.youtubeeducation.com/watch?v=LUrVu3ve2O0)

**Modelo Entidade-Relacionamento**

“A modelagem conceitual é muito importante no projeto de uma aplicação de banco de dados bem-sucedida”. (ELMASRI; NAVATHE, 2011, p. 131).

O Modelo Entidade-Relacionamento é um modelo de alto nível criado com o objetivo de representar a semântica associada aos dados do minimundo. O ME-R é utilizado na fase de projeto conceitual, onde se cria o esquema conceitual do banco de dados. Utiliza conceitos intuitivos, permitindo assim aos projetistas de banco de dados capturar os conceitos associados aos dados da aplicação, sem qualquer interferência da tecnologia específica de implementação do banco de dados.

O esquema conceitual é criado usando-se o ME-R chama-se Diagrama Entidade-Relacionamento (DE-R).

* **ME-R:** conjunto de conceitos e elementos de modelagem que o projetista do banco de dados precisa conhecer.
* **DE-R:** Resultado do processo de modelagem executado pelo projetista de dados que conhece o ME-R.

O ME-R é a técnica de modelagem de dados mais difundida e utilizada. o DE-R foi criado em 1976 por Peter Chen e pode ser considerada de fato como um padrão para modelagem conceitual. (HEUSER, 2009)

**Entidades**

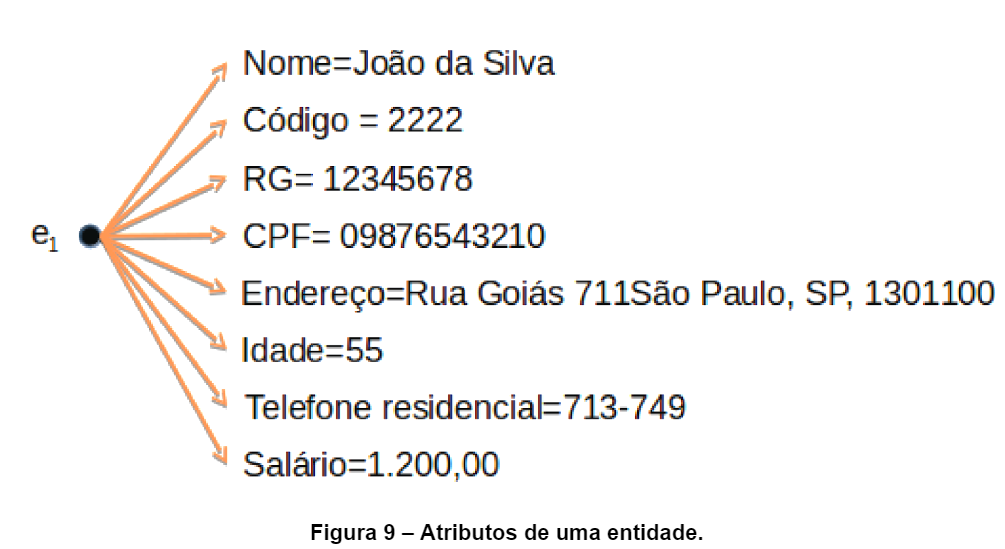
O conceito fundamental da abordagem ER é o conceito de entidade. (HEUSER, 2009). Entretanto, o ME-R descreve os dados, além das entidades, como relacionamentos e atributos. (ELMASRI; NAVATHE, 2011)

Uma entidade representa, no modelo conceitual, um conjunto de objetos, ou de entidades como sugere o nome, da realidade modelada. Estas entidades tem também uma existência independente. (ELMASRI; NAVATHE, 2011) (HEUSER, 2009). A entidade pode ser uma “coisa” física, como um carro ou uma pessoa ou conceitual, como um cargo, uma empresa ou um pedido de compras. (ELMASRI; NAVATHE, 2011)

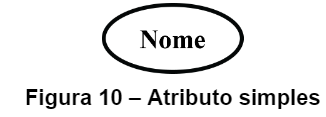
Cada entidade possui atributos que, nada mais são, do que suas características. Por exemplo, uma entidade PRODUTO pode ser descrita pelo seu nome, fabricante, preço de custo, preço de venda e quantidade em estoque e, para cada atributo a entidade terá um valor. Os valores de cada atributo de entidade tornam-se parte importante dos dados que serão armazenados no banco de dados. (ELMASRI; NAVATHE, 2011)

Na figura 9 é exibido uma entidade e os valores para seus atributos. A entidade e1 tem 8 atributos: nome, código, RG, CPF, endereço, idade, telefone residencial e salário. Seus valores são, respectivamente: ‘João da Silva’, 2222, ‘12345678’, ‘09876543210’, ‘Rua Goiás, 711, São Paulo, SP, 1301100’, 55, ‘713-749’ e 1200,00.

Em um DE-R podem ocorrer vários tipos de atributos: simples versus composto, valor único versus valor multivalorado e armazenado versus derivado. (ELMASRI; NAVATHE, 2011)



De forma geral, os atributos univalorados (atributos simples), são representados no DE-R utilizando-se uma elipse de borda simples, como mostrado na figura 10.



**Atributos Compostos**

Atributos compostos podem ser subdivididos em subpartes menores. Cada uma destas partes representam atributos mais básicos que têm significados independentes. (ELMASRI; NAVATHE, 2011)

O exemplo mais clássico deste tipo de atributo é, muito provavelmente o endereço que, muitas pessoas chamam de “endereço completo”. O que vem a ser um endereço? É um atributo que pode ser decomposto em atributos menores, como, por exemplo, nome do logradouro, número do imóvel, complemento (no caso de apartamentos e similares), nome do bairro, nome da cidade, nome do estado e o CEP.

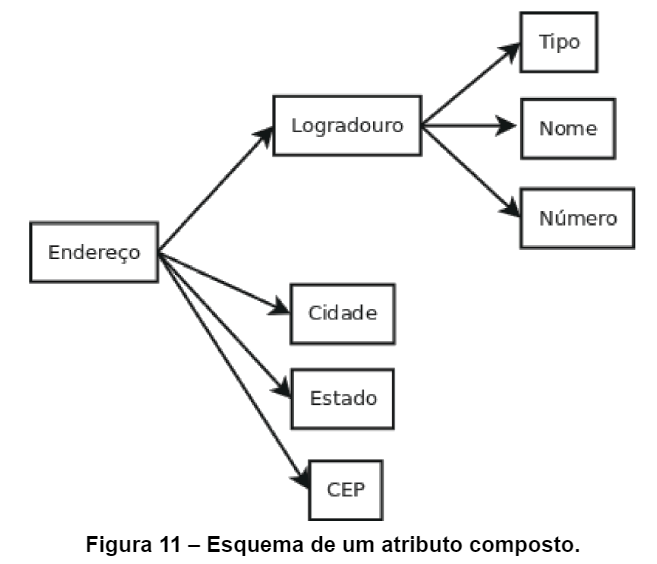
No exemplo da figura 9, pode-se observar que o endereço contém ‘Rua Goiás, 711, São Paulo, SP, 1301100’ que pode ser decomposto em Tipo do Logradouro (Rua), Nome do Logradouro (Goiás), Número (711), Cidade (São Paulo), Estado (SP) e o CEP (1301100).

Os atributos compostos podem formar uma hierarquia, como, por exemplo, o Logradouro ser subdividido em três atributos simples: Tipo do Logradouro, Nome do Logradouro e Número. O valor de um atributo composto é a concatenação dos valores de todos os seus atributos simples. (ELMASRI; NAVATHE, 2011)

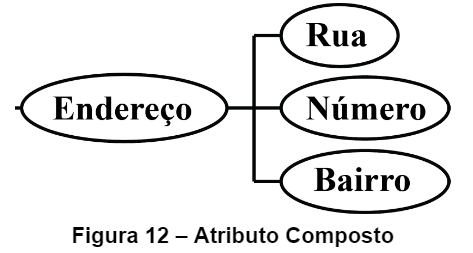
Concatenação é um termo usado em computação para designar a operação de unir o conteúdo de duas strings. Por exemplo, considerando as strings “Alex” e “Andre” a concatenação da primeira com a segunda gera a string “Alexandre”. (WIKIPEDIA, 2008)

Modela-se um atributo como composto quando o projetista refere-se a ele hora como uma unidade, hora como um dos seus subcomponentes. Se a referência ao atributo for sempre como unidade, não há a necessidade modelá-lo como atributo composto. (ELMASRI; NAVATHE, 2011)

A figura 11 demonstra o esquema do atributo composto endereço, apresentado na figura 9.



Um atributo composto é representado no DE-R por um conjunto de elipses, com borda simples, interligadas, como mostrado na figura 12.



**Atributos Multivalorados**

A grande maioria dos atributos possui um único valor para uma determinada entidade, como pode ser vista na figura 9. Observe que a entidade e1 possui um código (2222) para o atributo código e um nome (‘João da Silva’) para o atributo nome. Estes valores são chamados de valores únicos. Por exemplo, o CPF é um valor único de uma pessoa. Entretanto, em alguns casos o atributo pode ter mais de um valor, sendo então um atributo multivalorado. Pode-se citar como exemplo, as cores para um carro (figura 14), as referências comerciais para um cliente (figura 15) ou, os mais comuns, telefone e email que uma mesma pessoa pode ter mais de um.

Um atributo multivalorado, no DE-R é representado por uma elipse de bordas duplas, como mostrado na figura 13.



**Atributos Derivados**

Em alguns casos, dois ou mais valores de atributos estão relacionados, como, por exemplo, a idade e a data de nascimento de uma determinada pessoa. Para uma pessoa específica, sua idade pode ser determinada (calculada) utilizando-se a data atual (data de hoje) e a data de nascimento, obtendo assim a idade em anos, meses e dias. Desta forma, o atributo idade é chamado de atributo derivado, enquanto que a data de nascimento é considerada um atributo armazenável. (ELMASRI; NAVATHE, 2011)

Alguns atributos derivados podem ser obtidos com o relacionamento de atributos de diferentes entidades, como, por exemplo, obter o número de funcionários que trabalham em um determinado departamento. (ELMASRI; NAVATHE, 2011)

No DE-R, um atributo derivado é representado por uma elipse de bordas tracejadas, como mostrado na figura 16.



**Valores Nulos**

Certas vezes, uma determinada entidade pode não ter um valor para ser aplicado a um atributo específico. Por exemplo, o atributo email. Não é garantia de que todas as pessoas tenham e-mail, sobretudo as mais idosas. Outro exemplo pode ser a Formação Acadêmica, uma vez que ainda existem pessoas analfabetas! Para estas situações foi criado um valor especial chamado NULL. (ELMASRI; NAVATHE, 2011)

Atributos que não possuam valores específicos aplicáveis, como email e Formação Acadêmica podem receber o valor NULL. NULL pode ser aplicado quando a informação está faltando, quando se desconhece a informação ou quando a informação não é aplicável. Na primeira situação, por exemplo, se uma pessoa não tem o número de telefone no momento do preenchimento do cadastro; o segundo, poderia ser um paciente que chega desacordado a uma unidade de emergência e sem documentos, seu nome é desconhecido das pessoas que o resgataram e, finalmente o terceiro, solicitar-se o número da reservista (serviço militar) para uma mulher, que pode mas não é obrigada a alistar-se, portanto, a grande maioria não teria este documento.

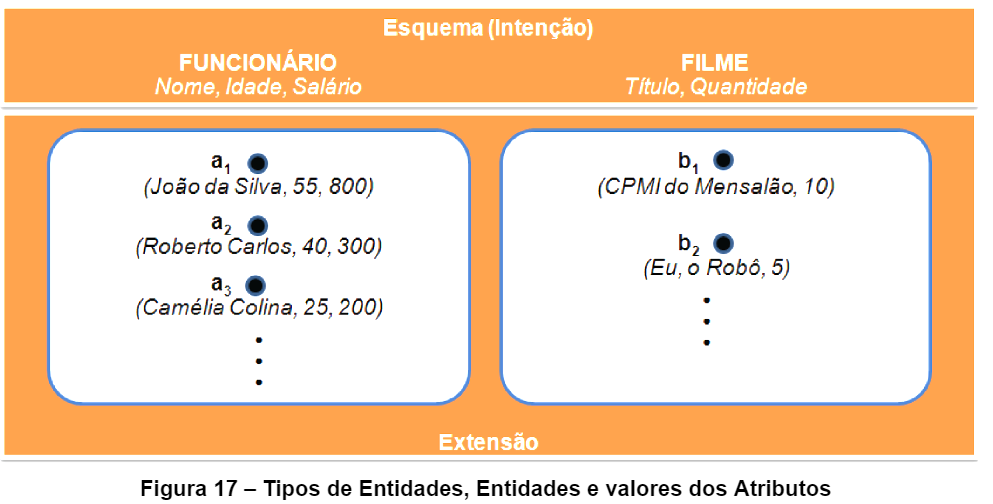
**Tipos de Entidades**

“Um banco de dados em geral contém grupos de entidades que são semelhantes”. (ELMASRI; NAVATHE, 2011, p. 136) Por exemplo, uma escola que possui centenas de alunos pode desejar armazenar informações semelhantes relacionadas a seus alunos. As entidades dos alunos, compartilham os mesmos atributos mas, cada uma tem o(s) próprio(s) valor(es) para cada atributo.

Dessa forma, “Um Tipo de Entidade define uma coleção (ou conjunto) de entidades que têm os mesmos atributos. Cada Tipo de Entidade no banco de dados é descrito por seu nome e atributos”. (ELMASRI; NAVATHE, 2011, p. 136)

A figura 17 mostra dois tipos de entidade: FUNCIONÁRIO e FILME, e uma lista de alguns atributos para cada uma. Nesta figura, também aparecem algumas entidades individuais de cada Tipo de Entidade junto com os valores para cada um de seus atributos. “A coleção de todas as entidades individuais de determinado tipo de entidade no banco de dados, em qualquer ponto no tempo, é chamada de conjunto de entidades”. (ELMASRI; NAVATHE, 2011, p. 136)

No DE-R, um Tipo de Entidade é representada por um retângulo e deve ter como nome um substantivo no singular, como Funcionário ou Filme.



**Atributos-chave de um Tipo de Entidade**

Um Tipo de Entidade possui uma restrição importante chamada de “restrição de exclusividade” sobre os atributos. Esta restrição é obtida através do atributo-chave.

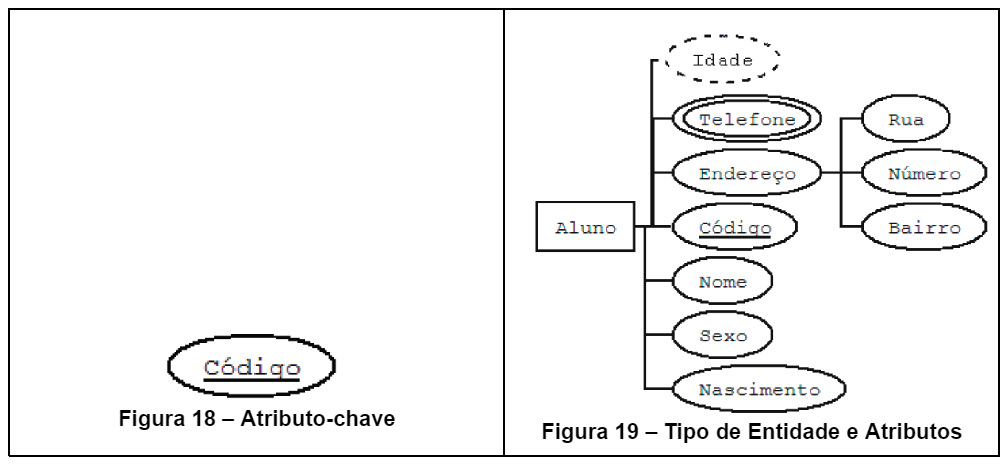
De acordo com (ELMASRI; NAVATHE, 2011, p. 137):

Um tipo de entidade normalmente tem um ou mais atributos cujos valores são distintos para cada entidade individual no conjunto de entidades. Esse atributo é denominado atributo-chave, e seus valores podem ser usados para identificar cada entidade de maneira exclusiva.

Voltando ao exemplo do Tipo de Entidade ALUNO, citado no item 4.2; pode-se utilizar o RA (Registro Acadêmico ou Registro do Aluno) como atributo chave do Tipo de Entidade ALUNO, uma vez que cada aluno tem seu próprio RA e eles não se duplicam. Da mesma forma, outro atributo que poderia ser usado como atributo-chave é o CPF do aluno. Entretanto, como pode haver um aluno que não possua o CPF ou, em caso de irmão que utilizassem o CPF do pai ou da mãe, este atributo poderia causar erros ao ferir a restrição de exclusividade exigida.

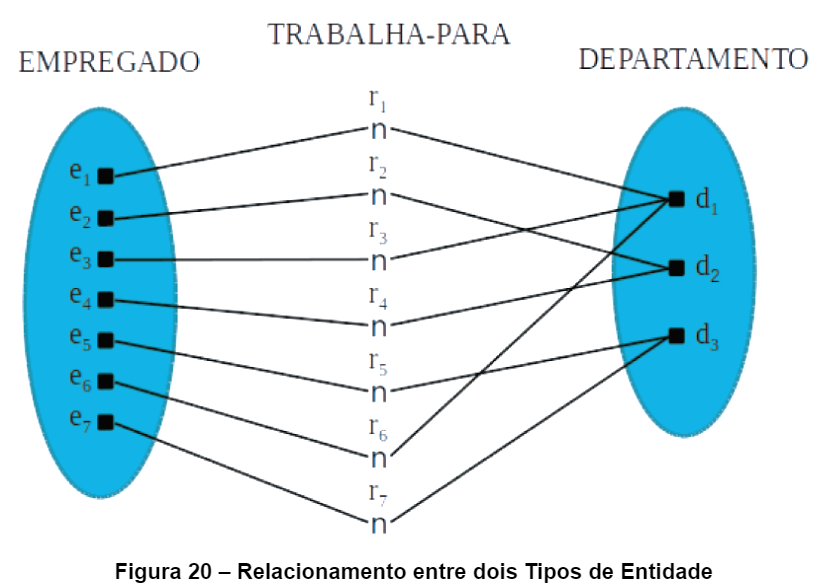
Em algumas situações, existem Tipos de Entidades que podem ter mais de um atributo-chave. Como exemplo, pode-se ter um Tipo de Entidade PESSOA, que tenho como atributos o RG, o Local de Expedição, Nome da Pessoa e Telefone. Nenhum dos atributos pode identificar sozinho, de forma única, cada uma das entidades (Pessoas) do Tipo de Entidade. Desta forma, se faz necessário a utilização de dois atributos chave (RG e Local de Expedição), pois o RG é um documento estadual e seu número pode ser duplicado em diferentes estados mas, ao uni-lo ao Local de Expedição, torna-se uma chave unívoca, ou seja, capaz de identificar de forma única cada uma das entidades (pessoas). (ELMASRI; NAVATHE, 2011)

Cada um dos atributos-chave de um Tipo de Entidade, no DE-R, são representados por uma elipse de bordas simples sublinhadas, como mostrado na figura 18. A figura 19 ilustra um Tipo de Entidade com seus atributos.

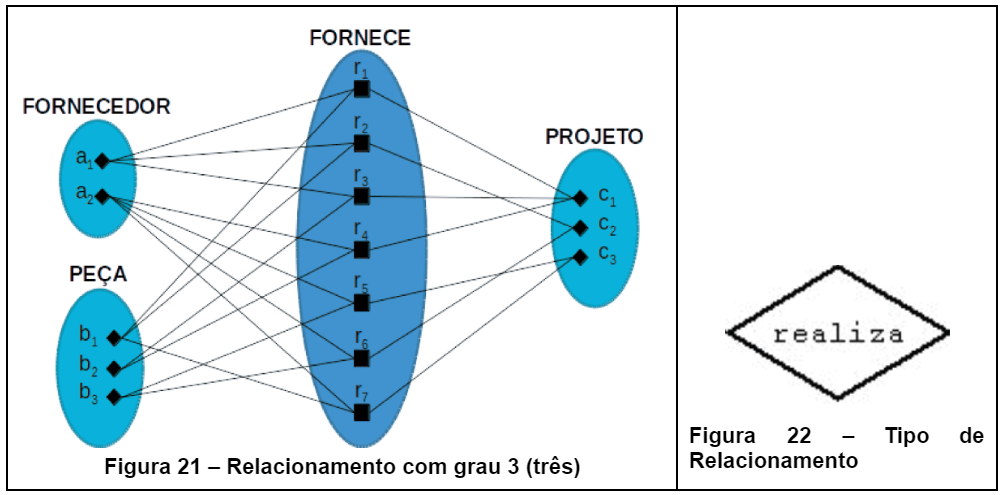


**Tipos de Relacionamento**

Em um projeto de banco de dados, existem vários relacionamentos explícitos entre os diversos tipos de entidade pois, toda vez que um atributo de um tipo de entidade se refere a outro atributo de outro tipo de entidade, existe um relacionamento. Como exemplo, podemos citar o DEPARTAMENTO em que o EMPREGADO trabalha. Ele referencia em qual departamento um determinado empregado trabalha. Desta forma, um tipo de relacionamento R entre n tipos de entidade E1, E2,... En, define um conjunto de associações ou um conjunto de relacionamento. Na figura 20 está demonstrado um relacionamento entre dois tipos de entidades. (ELMASRI; NAVATHE, 2011



“O grau de um tipo de relacionamento é o número do tipo de entidades participantes”. (ELMASRI; NAVATHE, 2011, p. 141) Assim, pode-se afirmar que o grau de relacionamento do relacionamento demonstrado na figura 20 é 2 (dois), pois contém dois tipos de entidade: EMPREGADO e DEPARTAMENTO. Um tipo de relacionamento de grau dois é chamado de binário e um de grau três é chamado de ternário. Um exemplo de relacionamento ternário pode ser visto na figura 21. No DE-R, o tipo de relacionamento é representado por um losango, como pode ser observado na figura 22.



**Relacionamentos como atributos**

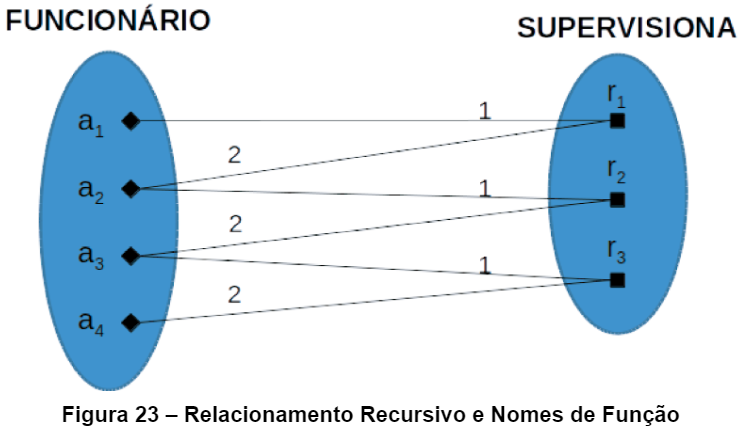
Em algumas situações, é conveniente pensar em um tipo de relacionamento em termos de atributos. Como exemplo, o relacionamento binário TRABALHA-PARA, demonstrado na figura 22, pode-se pensar em um atributo chamado Departamento do tipo de entidade FUNCIONÁRIO, em que cada valor do atributo Departamento é uma referência à entidade DEPARTAMENTO onde o funcionário trabalha. Logo, o conjunto de valores possíveis para esse atributo (Departamento) é o conjunto de todas as entidades DEPARTAMENTO. Entretanto, quando existe um relacionamento binário, sempre há duas opções, então, o raciocínio for invertido e se colocar um atributo Empregado no tipo de entidade DEPARTAMENTO, ele será um atributo multivalorado, uma vez que os valores possíveis para este atributo são o conjunto de todas as entidades EMPREGADO. (ELMASRI; NAVATHE, 2011)

**Nomes de Função e Relacionamentos Recursivos**

Cada tipo de entidade que participa de um tipo de relacionamento possui um papel específico. No caso do relacionamento EMPREGADO trabalha-para DEPARTAMENTO, o papel de EMPREGADO é do de empregado ou trabalhador e o de DEPARTAMENTO é o papel de empregador. Entretanto, nem sempre a escolha do nome é simples. (ELMASRI; NAVATHE, 2011)

“O nome da função significa a função que uma entidade participante do tipo de entidade desempenha em cada instância de relacionamento, e ajuda a explicar o que o relacionamento significa”. (ELMASRI; NAVATHE, 2011, p. 141)

O nome da função não é uma informação obrigatória, uma vez que o nome do Tipo de Entidade pode servir como nome de função. Entretanto, há situações em que um mesmo Tipo de Entidade participa mais de uma vez do mesmo tipo de relacionamento executando funções diferentes. Nestes casos, a utilização do nome de função torna-se obrigatória. Este tipo de relacionamento recebe o nome de relacionamentos recursivos, como demonstrado na figura 23.

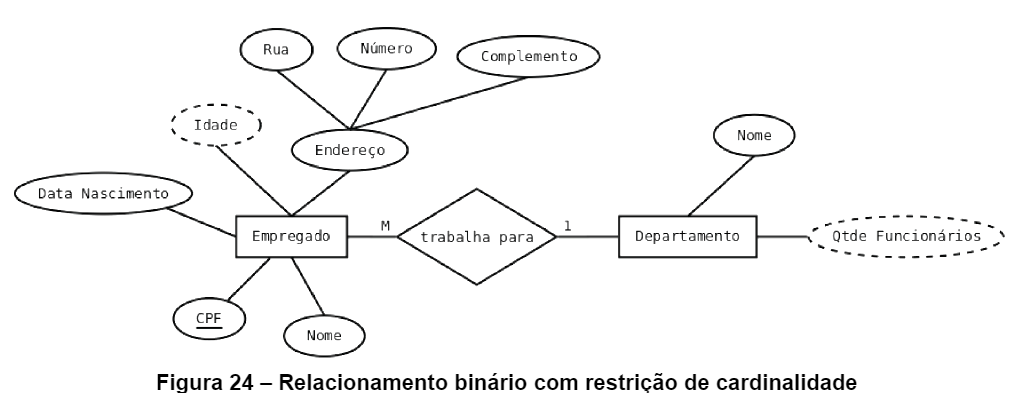


**Razão de Cardinalidade e Restrição de Participação**

Para um relacionamento binário, a razão de cardinalidade especifica a quantidade máxima de instâncias de relacionamentos em que uma entidade pode participar. As razões de cardinalidade possíveis são:

* 1:1 – Um-para-um
* 1:N – Um-para-N
* N:1 - N-para-Um
* N:N – N-para-N

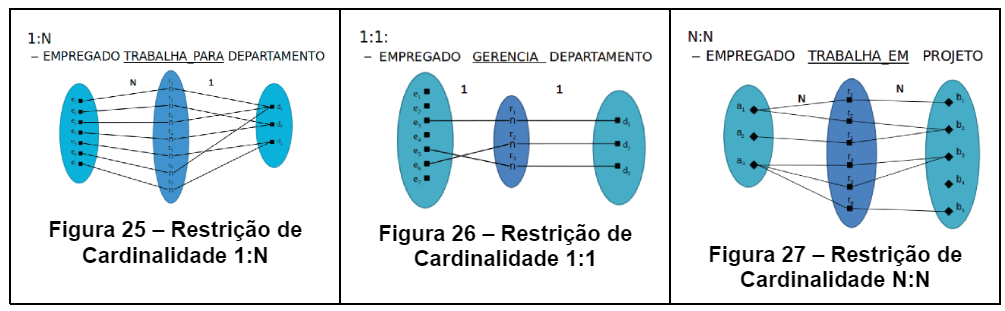
No caso do relacionamento TRABALHA-PARA, DEPARTAMENTO:FUNCIONÁRIO tem razão de cardinalidade 1:N (Um-para-N), indicando assim que um Departamento pode estar relacionado com quaisquer número de Empregados mas que, um Empregado pode estar relacionado com apenas um Departamento. A restrição de cardinalidade para relacionamentos binários são representadas no DE-R exibindo 1, M e N nos losangos com pode ser observado na figura 24. (ELMASRI; NAVATHE, 2011)



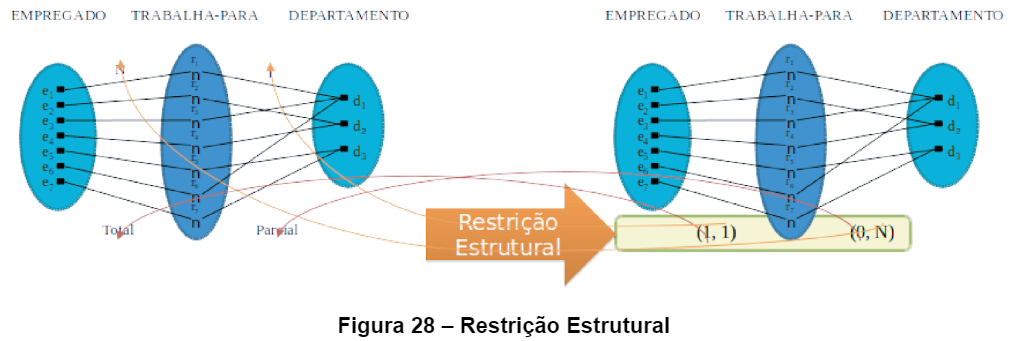
A restrição de participação especifica se a existência depende dela estar relacionada a outra entidade por meio do tipo de relacionamento. Essa restrição especifica o número mínimo de instâncias de relacionamento em que cada entidade pode participar e, as vezes, é chamada de restrição de cardinalidade mínima. Existem dois tipos de restrição de participação – total e parcial. [...] Se a política de uma empresa afirma que todo funcionário precisa trabalhar para um departamento, então uma entidade de funcionário só pode existir se participar em, pelo menos, uma instância de relacionamento TRABALHA\_PARA. (ELMASRI; NAVATHE, 2011, p. 143)

Neste caso, a participação de FUNCIONÁRIO em TRABALHA\_PARA é total, já que o conjunto total de funcionários devem estar relacionados e uma entidade DEPARTAMENTO.

A restrição estrutural define o mínimo é o máximo que uma entidade pode participar de um relacionamento. Nas figuras 25, 26 e 27, estão representados, respectivamente, os relacionamentos com restrição de cardinalidade 1:N, 1:1 e N:N e, na figura 28 um exemplo de como é a restrição estrutural.

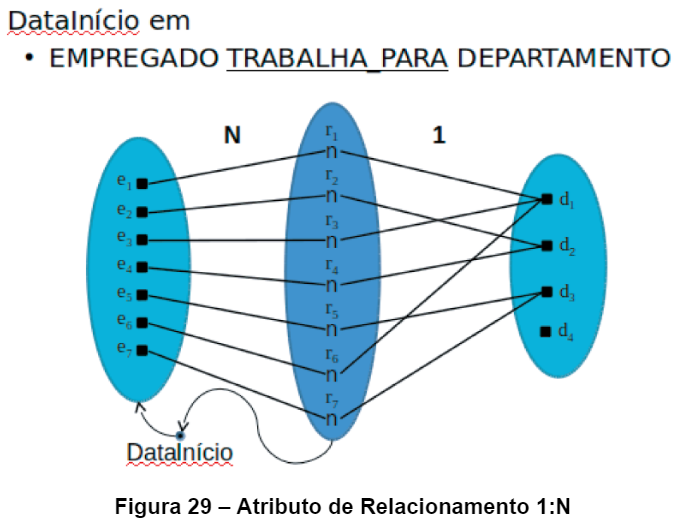


No relacionamento GERENCIA, não é esperado que todo funcionário gerencie um departamento, tendo assim, parte das entidades de FUNCIONARIO relacionadas a uma parte de DEPARTAMENTO. Desta forma, a participação é parcial. Em diagramas Entidades-Relacionamento, a participação total é exibida como uma linha dupla que conecta o tipo de entidade participante ao relacionamento, enquanto que a participação parcial é representada por uma linha simples. (ELMASRI; NAVATHE, 2011)



Os Tipos de Relacionamentos também podem ter Atributos. Por exemplo, para registrar a quantidade de horas trabalhadas por um empregado em um dado projeto (Horas), pode ser representado como um atributo do relacionamento TRABALHA\_EM; a data em que um gerente começou a gerenciar um departamento (DataInício), pode ser representado como um atributo do relacionamento GERENCIA. Os atributos dos tipos de relacionamento 1:1 e 1:N podem ser migrados para um dos tipos de entidades participantes. Em relacionamento 1:N, o atributo pode ser migrado somente para o lado N do relacionamento, como pode ser visto na figura 29. Nos relacionamentos 1:1, a decisão de onde colocar o atributo de relacionamento é determinada de maneira subjetiva pelo projetista. (ELMASRI; NAVATHE, 2011)

“Para relacionamento M:N, alguns atributos podem ser determinados pela combinação de entidades participantes em uma instância de relacionamento, e não pode haver qualquer entidade isolada”. (ELMASRI; NAVATHE, 2011, p. 144) Estes atributos devem ser especificados como atributos de relacionamento.

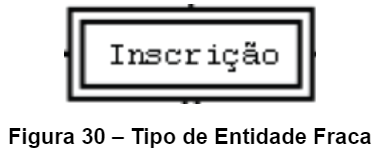


**Tipo de Entidade-Fraca**

“Tipos de entidade que não possuem atributos-chave próprios são chamado tipos de entidade-fraca” (ELMASRI; NAVATHE, 2011, p. 144) As entidades destes tipos de entidades estão relacionadas a outras entidades específicas. Estas outras entidades recebem o nome de tipo de entidade de identificação ou proprietário e o tipo de relacionamento que relaciona um tipo de entidade fraca ao seu proprietário recebe o nome de relacionamento de identificação do tipo de entidade fraca.

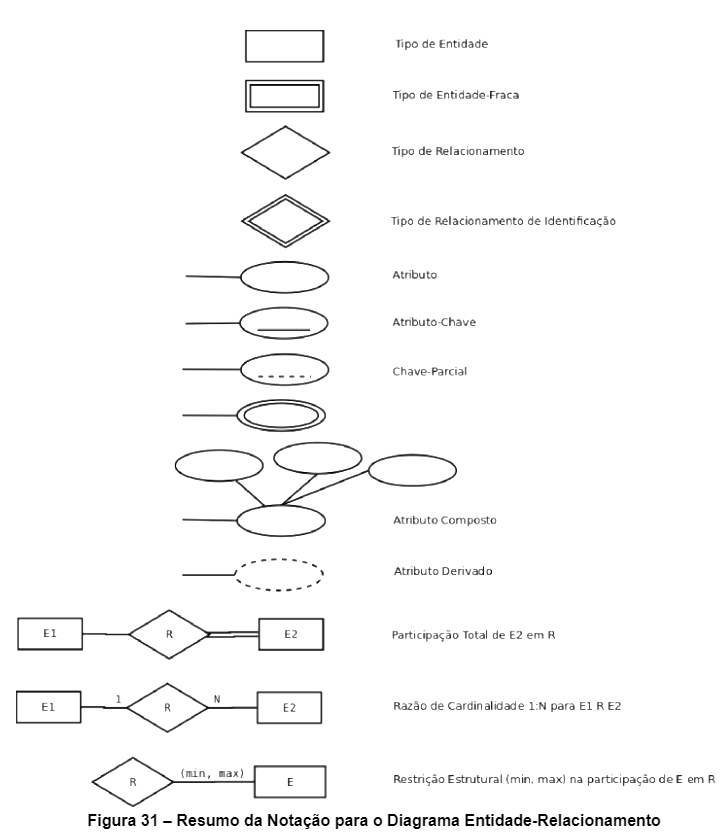
Um tipo de entidade-fraca sempre tem restrição de participação total (dependência existencial) com respeito ao seu tipo de relacionamento de identificação, uma vez que não é possível identificar uma entidade-fraca sem o correspondente tipo de entidade proprietária. Os tipos de entidade-fraca podem ter uma chave-parcial, que é um conjunto de atributos que pode univocamente identificar entidades-fracas relacionadas à mesma entidade proprietária. (ELMASRI; NAVATHE, 2011)

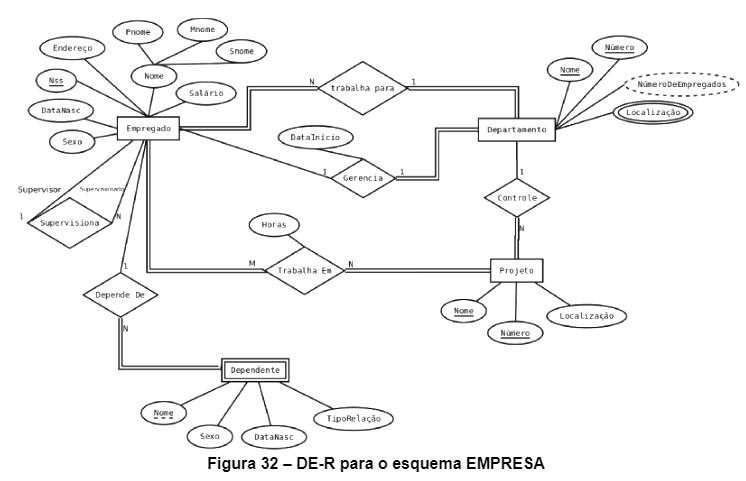
No DE-R, um tipo de entidade-fraca é representado por um retângulo de bordas duplas, como pode ser observado na figura 30.



**Resumo da Notação para Diagramas Entidade-Relacionamento**

Na figura 31 pode ser visto um resumo de todas as representações abordadas até aqui no diagrama Entidade-Relacionamento e, na figura 32, observa-se um exemplo de diagrama entidade-relacionamento abordando todas as possibilidades de representação neste tipo de diagrama.





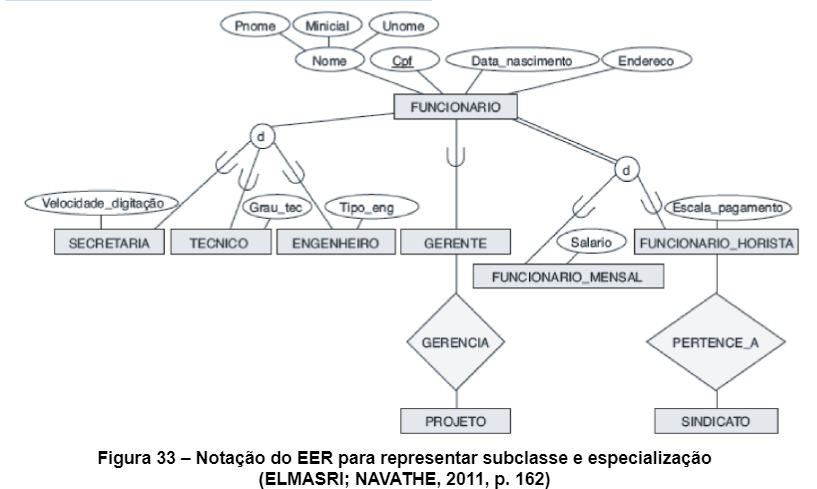
**Parte 5 -** [**Continuação ao modelo Entidade-Relacionamento**](https://sites.google.com/faculdadeimpacta.com.br/fbd-p5?pli=1&authuser=2) **-** [**Vídeo**](https://www.youtubeeducation.com/watch?v=jep21MryfNI)

**Modelo Entidade-Relacionamento Estendido**

Quase todas as aplicações de bancos de dados podem ser representadas por meio do modelo entidade-relacionamento. Entretanto, alguns bancos de dados são mais complexos e exigem, para sua melhor representação, alguns aspectos adicionais.

Para solucionar esse problema, o modelo ER é expandido para o modelo EER. Esse modelo inclui, além dos novos conceitos, todos os conceitos de modelagem do modelo ER. (RANGEL et al, 2014, p. 69)

O Modelo Entidade-Relacionamento Estendido inclui todos os conceitos de modelagem existentes no Modelo Entidade-Relacionamento. Além disso, ele acrescenta os conceitos de subclasse e superclasse que estão relacionados aos conceitos de generalização / especialização, como pode ser visto na figura 33.



O relacionamento entre uma superclasse e qualquer uma de suas subclasses recebe o nome de relacionamento superclasse/subclasse. Como exemplo, pode ser visto na figura 33 dois destes tipos de relacionamento: FUNCIONÁRIO/SECRETÁRIA e FUNCIONÁRIO/TÉCNICO. Em um relacionamento assim, uma entidade da subclasse representa a mesma entidade do mundo real que a entidade da superclasse. Desta forma, pode-se citar, como exemplo que, o funcionário EMERSON FITIPALDI também é o técnico EMERSON FITIPALDI. Então, embora sendo a mesma entidade, o papel específico que exercem é o que os diferencia. Entretanto, quando este tipo de relacionamento é implementado no banco de dados é representada como objetos distintos, relacionados através do atributo chave. (ELMASRI; NAVATHE, 2011)

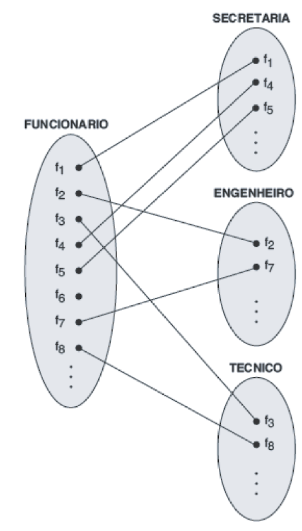
Em um banco de dados, uma entidade não pode existir pelo simples fato de pertencer a uma subclasse, assim, ela deve pertencer também à superclasse. Esta entidade pode ou não ser incluída como membro de uma ou mais subclasses. Uma subclasse então, herda os atributos existentes na superclasse e, desta forma representa a mesma entidade do mundo real. (ELMASRI; NAVATHE, 2011) “Ao conceito de superclasse e subclasse está associado o conceito de herança”. (RANGEL et al, 2014, p. 70)

**Especialização e Generalização**

“Entende-se por especialização a definição de um conjunto de subclasses de um tipo de entidade, a partir de uma superclasse”. (RANGEL et al, 2014, p. 70) Este tipo de classe é chamado de superclasse e o conjunto de classes que a formam é definido baseado em alguma característica distinta das entidades da superclasse. (ELMASRI; NAVATHE, 2011)

Como exemplo, pode-se observar na figura 33 que ENGENHEIRO é uma especialização de FUNCIONÁRIO e, que possui um atributo TIPO\_ENG que não existe em FUNCIONÁRIO, uma vez que somente engenheiros precisam desta informação.

**Especialização**



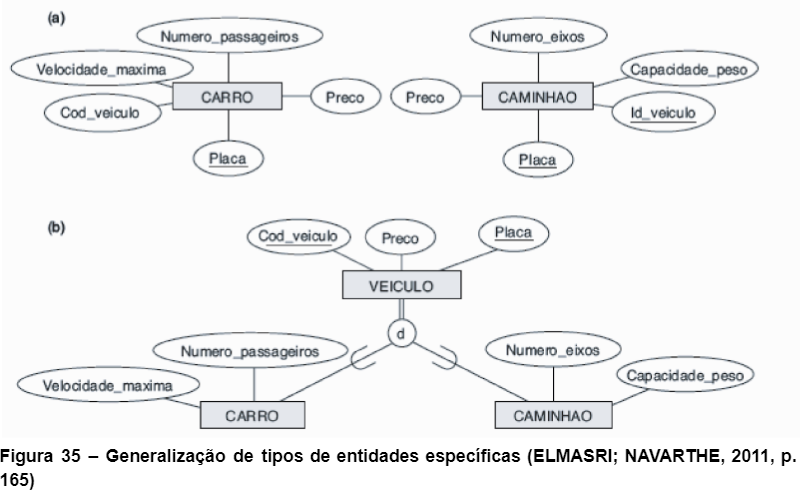
“Uma especialização pode ser definida por predicado, quando há uma condição no valor de algum atributo da superclasse”. (RANGEL et al, 2014, p. 70) Isto acontece no exemplo da figura 33 onde um funcionário pertencerá a uma das subclasses de FUNCIONÁRIO. Na figura 34 estão demonstradas algumas instâncias de entidades que pertencem às subclasses da especialização de FUNCIONÁRIO. Pode-se observar também como uma entidade da superclasse é a mesma entidade da subclasse que representa uma entidade do mundo real.

Existem dois motivos principais para incluir relacionamentos de classe/sub-classe e especializações em um banco de dados, sendo que o primeiro motivo deve-se ao fato de quecertos atributos podem ser aplicados a algumas entidades da superclasse mas, não a todas e, o segundo motivo é que em alguns relacionamentos podem participar apenas algumas das subclasses! (ELMASRI; NAVATHE, 2011)

**Generalização**

“Generalização é a definição de um tipo entidade (superclasse) a partir de suas subclasses”. (RANGEL et al, 2014, p. 70) Ela pode ser vista ou pensada como o processo reverso da abstração, onde as diferenças entre vários tipos de entidades são suprimidas, suas características comuns reunidas e então generalizadas em uma superclasse, da qual os tipos de entidades específicas são subclasses. (ELMASRI; NAVATHE, 2011)

Por exemplo, se na especialização ao se deparar com um um tipo de entidade VEÍCULO, o projetista o específica, criando subclasses como, por exemplo, CARRO e CAMINHÃO, na generalização, caso o projetista obtenha as classes específicas CARRO e CAMINHÃO, ele poderá fazer o raciocínio invertido, agrupando-as em uma classe mais genérica e, portanto, menos específica VEÍCULO, como pode ser observado na figura 35.



Segundo Elmasri e Navathe (2011, pp. 165-166):

*Uma notação dramática para distinguir generalização de especialização é usada em algumas metodologias de projeto. Uma seta apontando para a superclasse generalizada representa uma generalização, ao passo que setas apontando para subclasses especializadas representam uma especialização.*

Entretanto, a utilização deste tipo de notação não é muito adotada, “porque a decisão sobre qual processo é seguido em determinada situação costuma ser subjetiva”. (ELMASRI; NAVARTHE, 2011, p. 166)

**Restrições sobre especialização e generalização**

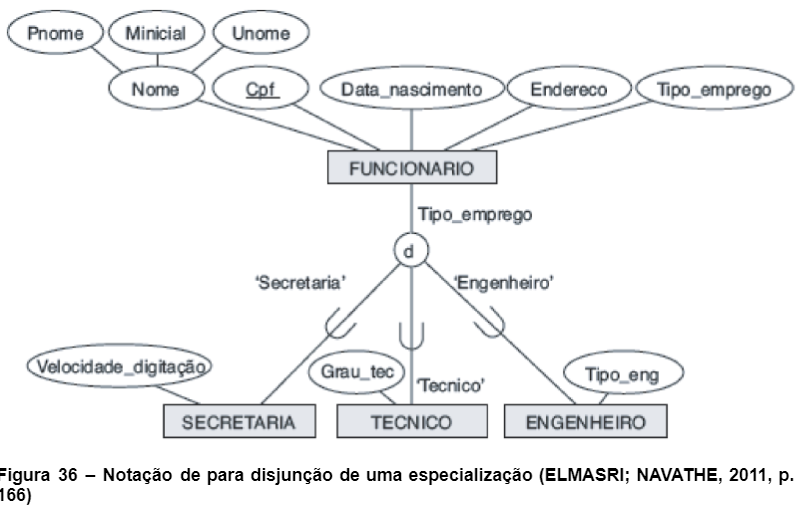
De modo geral, podem haver diversos tipos de entidade especializadas de uma outra como, da mesma forma, uma especialização pode consistir em apenas um tipo de entidade. É possível determinar a quantidade de entidades que se tornarão membros das subclasses ou colocar uma condição sobre algum atributo da superclasse. Essas subclasses são chamadas de subclasses definidas por predicado. Essa condição é uma restrição que especifica exatamente quais entidades farão parte da subclasse. (ELMASRI; NAVATHE, 2011)

Quando todas as subclasses tiverem sua condição definidas pelo mesmo atributo da superclasse esta especialização recebe o nome de especialização definida por atributo e o atributo é chamado de atributo de definição mas, quando não se tem uma condição para determinar os membros das subclasses, diz-se que esta é definida pelo usuário. (ELMASRI; NAVATHE, 2011)

Há ainda dois outros tipos de restrição. A restrição de disjunção que especifica que as classes devem ser disjuntas e pode ser disjunta ou de sobreposição. Então, isto significa que uma entidade pode ser membro de no máximo uma das subclasses. (ELMASRI; NAVATHE, 2011)

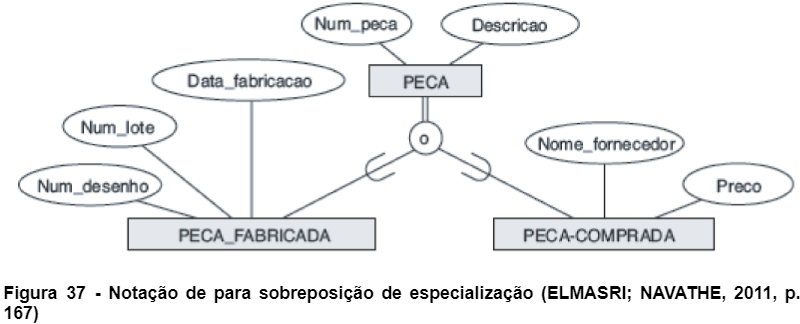
“Em matemática, dois conjuntos são ditos disjuntos se não tiverem nenhum elemento em comum. [...] Os conjuntos: { 1 , 2 , 3 } e { 6 , 7 } são disjuntos pois não possuem elementos em comum”. (WIKIPEDIA, 2007)

A figura 36 ilustra este caso, onde o “d” no círculo significa disjunção.



Outra restrição possível de ser aplicada a uma especialização é a restrição de completude ou de totalidade e que pode ser total ou parcial. Quando é total, indica que toda entidade da superclasse precisa ter um correspondente em, pelo menos uma das subclasses.

Por outro lado, quando é parcial, permite que uma entidade da superclasse não pertença a nenhuma das subclasses. Uma restrição parcial é indicada por uma linha simples. Um exemplo pode ser visto na figura 37. (ELMASRI; NAVATHE, 2011)

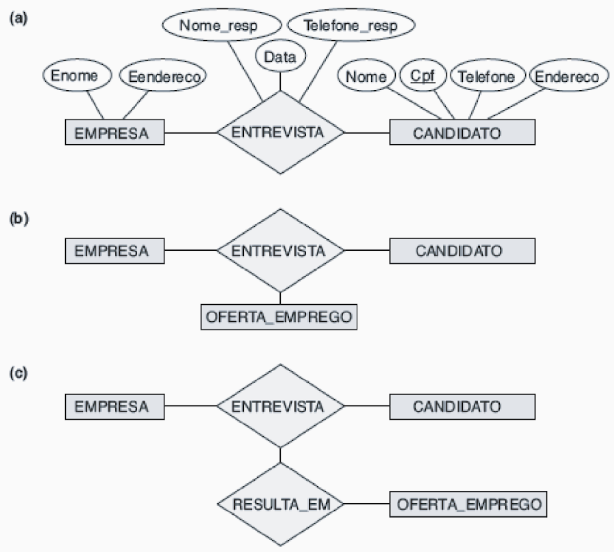


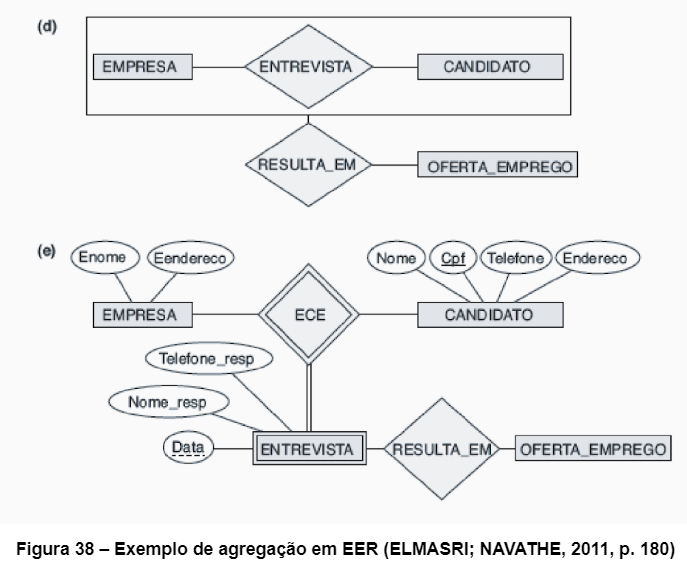
É importante salientar que restrições de disjunção e completude são independentes, podendo então haver 4 possibilidades, sendo: disjunção, total; disjunção, parcial; sobreposição, total e sobreposição, parcial. Todavia, a restrição correta será determinada baseando-se no mundo real.

**Agregação**

“É um conceito de abstração para a criação de objetos compostos com base em seus objetos componentes”. (ELMASRI; NAVATHE, 2011, p. 178)

No modelo EER, normalmente a necessidade de agregação ocorre quando se tem a necessidade de ligar um tipo de relacionamento a outro. A notação do DE-R não prevê este tipo de situação, assim, a solução é a agregação, criando-se uma “entidade virtual” em um nível mais alto de abstração. Entretanto, uma maneira mais “elegante” de solucionar este problema é transformar o tipo de entidade que se deseja relacionar com o relacionamento em um tipo de entidade-fraca que depende do relacionamento das outras duas entidades. Como exemplo, tem-se o caso de um CANDIDATO a procura de emprego em uma determinada EMPRESA e o relacionamento entre eles é ENTREVISTA. Do resultado da entrevista, surgirá, ou não, uma oferta de emprego que, para ser melhor descrita precisa estar ligado a um relacionamento RESULTA-EM mas, como não é possível ligar dois relacionamentos (ENTREVISTA e RESULTA-EM), agrega-se o relacionamento ENTREVISTA e a agregação é então ligada ao relacionamento RESULTA-EM. A solução mais adequada e final pode ser vista na figura 38, item e, como fruto desta “evolução” do relacionamento. (ELMASRI; NAVATHE, 2011)





[Vídeo](https://youtu.be/OVBFFe4-jSM?si=6Fphb3x6_35O0oSn) - **Modelagem de Dados - O que são Cardinalidades** - Boson Treinamentos

**Parte 6 -** [**Notação diagrama Entidade Relacionamento e UML**](https://sites.google.com/faculdadeimpacta.com.br/fbd-p6?pli=1&authuser=2) **-** [**Vídeo**](https://www.youtubeeducation.com/watch?v=GFvXEP7JMGk)

**Diagrama de Classes**

“O diagrama de classes é um dos mais importantes e mais utilizados da UML”. (GUEDES, 2009, p. 101)

Segundo a Wikipedia (2004):

*A UML - Linguagem de Modelagem Unificada (do inglês, UML - Unified Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software. Em outras palavras, na área de Engenharia de Software, a UML é uma linguagem de modelagem que permite representar um sistema de forma padronizada (com intuito de facilitar a compreensão pré-implementação). A UML é adequada para a modelagem de sistemas, cuja abrangência poderá incluir desde sistemas de informação corporativos a serem distribuídos a aplicações baseadas na Web e até sistemas complexos embutidos de tempo real. É uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas*

O principal enfoque do diagrama de classes é permitir a visualização das classes que irão compor o sistema e seus atributos e métodos, assim como demonstrar como elas se relacionam. O diagrama de classes é composto basicamente por classes e suas associações. (GUEDES, 2009)

**Atributos e Métodos**

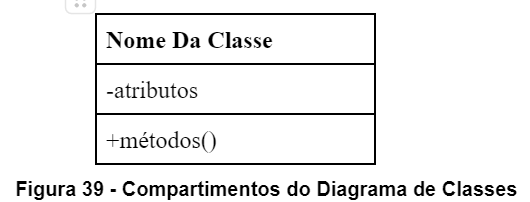
“Classes costumam ter atributos, que [...], armazenam os dados dos objetos da classes”. (GUEDES, 2009, p. 101). Além dos atributos, as classes também podem ter métodos, entretanto, esta parte não será abordada pois, para modelar um banco de dados são necessários apenas os atributos. Quanto aos atributos, seus valores podem variar de uma instância para outra, assim, torna-se possível identificar cada objeto individualmente. (GUEDES, 2009)

As Classes representam objetos do mundo real e, estes objetos têm características similares. As classes são representadas por um retângulo que, segundo a notação básica da UML, pode estar subdividido em três partes ou compartimentos que contém, Nome da Classe, a lista de atributos e a lista de operações. Destes, apenas o compartimento com o Nome da Classe é obrigatório. (GÓES, 2014)

“Não é realmente obrigatório que uma classe apresente as três divisões, pois pode haver classes que não tenham atributos ou classes que não contenham métodos, [...]” (GUEDES, 2009, p. 103)

**Nomes das Classes**

De acordo com GÓES (2014, p. 134), os nomes das Classes devem sempre estar no singular, em negrito e centralizado dentro do seu compartimento. Ele pode ser simples ou composto e, sempre a primeira letra do nome deve estar em letra maiúscula e seguida de letras minúsculas.



**Atributos das classes**

Segundo Góes (2009, p. 135), os atributos devem ser escritos em formatação normal e começar por letras minúsculas e posicionadas à esquerda do seu compartimento, e podem ter nomes simples (saldo) ou compostos (fotosAluno). Nesses casos, o segundo termo do nome do atributo, assim como os demais termos, deve iniciar com letra maiúscula.

Esta notação para os nomes dos atributos é também chamada de Notação Camel Case, que é a denominação em inglês para a prática de escrever palavras compostas ou frases, onde cada palavra é iniciada com maiúsculas e unidas sem espaços. (WIKIPEDIA, 2006)

Ao se modelar bancos de dados com o diagrama de classes, deve-se colocar os atributos chave no início da lista de atributos. Um exemplo de classe com os atributos pode ser observado na figura 40.

| **Médico** |
| --- |
| -crm: key  -nome |

**Figura 40 - Classe com atributos**

**Visibilidade**

Segundo Góes (2014, p. 136), “A visibilidade de um atributo define por quem uma propriedade desse atributo pode ser utilizada”. Existem três tipos de visualização, a saber:

• público, representado pelo símbolo “+”

• protegido, representado pelo símbolo “#”

• privado, representado pelo símbolo “-”

A Programação Orientada a Objetos – POO possui uma característica que é a Encapsulação. Visando não quebrá-la, atributos e métodos devem ser visíveis apenas onde é estritamente necessário, desta forma, os atributos de uma classe devem ter visibilidade privada (“-”), pois somente a classe deve ter acesso à eles. Para acessar os atributos de uma classe, utilizam-se métodos públicos. Entretanto, este não é o objetivo deste material e, portanto, abordaremos somente atributos, devendo todos eles serem classificados com visibilidade privada.

**Multiplicidade do atributo**

No caso dos atributos, a multiplicidade determina o número mínimo e máximo de calores que um atributo pode conter. Se nenhuma multiplicidade for indicada, assume-se que ela é 1. Desta forma, atributos multivalorados devem ter sua multiplicidade indicada. (GÓES, 2014) As multiplicidades possíveis podem ser vistas na figura 41.

| **Multiplicidade** | **Significado** |
| --- | --- |
| **0..1** | Zero ou um |
| **1** | Somente um |
| **0..\*** | Zero ou mais |
| **1..\*** | Um ou mais |
| **n** | Somente n (n > 1) |
| **0..n** | Zero a n (n > 1) |
| **1..n** | Um a n (n > 1) |

**Figura 41 – Multiplicidades possíveis**

**Atributo derivado**

De acordo com Góes (2014, p. 137), Atributos derivados são aqueles cujos valores podem ser deduzidos (calculados) em função dos valores de outros atributos derivados e/ou de atributos essenciais, também conhecidos por armazenados. [...] Atributos derivados são representados por uma barra (/) à esquerda do seu nome. A barra indica que o atributo é calculado em função de uma operação.

Um exemplo de uma classe com um atributo derivado de um atributo composto pode ser observado na figura 42.

| **Departamento** |
| --- |
| - número: key  - nome: key  - localização 1..\*  - /numeroEmpregados |

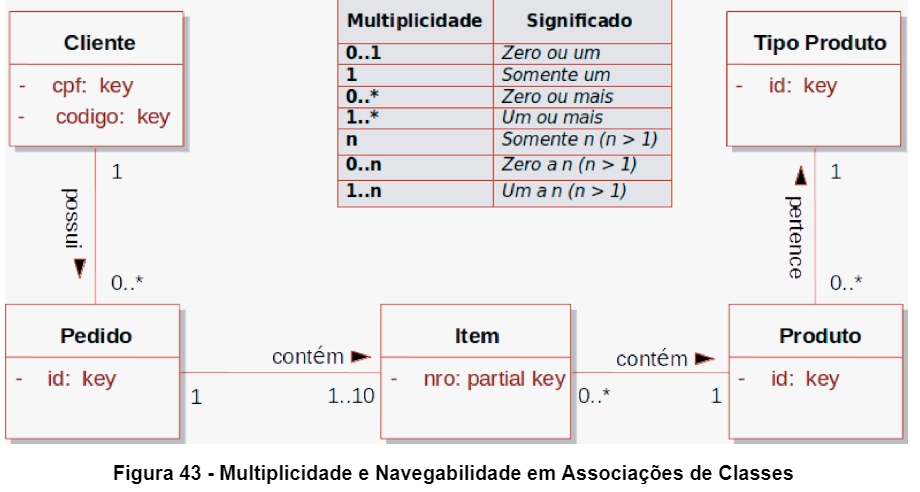
**Figura 42 – Atributo derivado e atributo composto.**

**Relacionamentos ou Associações**

“As classes costumam ter relacionamentos entre si, [...], que permitem que elas compartilhem informações” (GUEDES, 2009, p. 106) “Uma associação é um relacionamento entre objetos de uma mesma classe ou de classes distintas”. (GÓES, 2014, p. 139) “As associações são representadas por linhas ligando as classes envolvidas. Essas linhas podem ter nomes ou títulos para auxiliar a compreensão de vínculo estabelecido entre os objetos das classes envolvidas nas associações”. (GUEDES, 2009, p. 106)

Outra informação importante em uma associação é a multiplicidade que representa o número de objetos de cada classe que participa da associação.

“A navegabilidade é representada por uma seta em uma das extremidades da associação, identificando o sentido em que as informações são transmitidas entre os objetos das classes envolvidas”. (GUEDES, 2009, p. 109) Um exemplo de multiplicidade e navegabilidade pode ser observado na figura 43.



**Associação Binária**

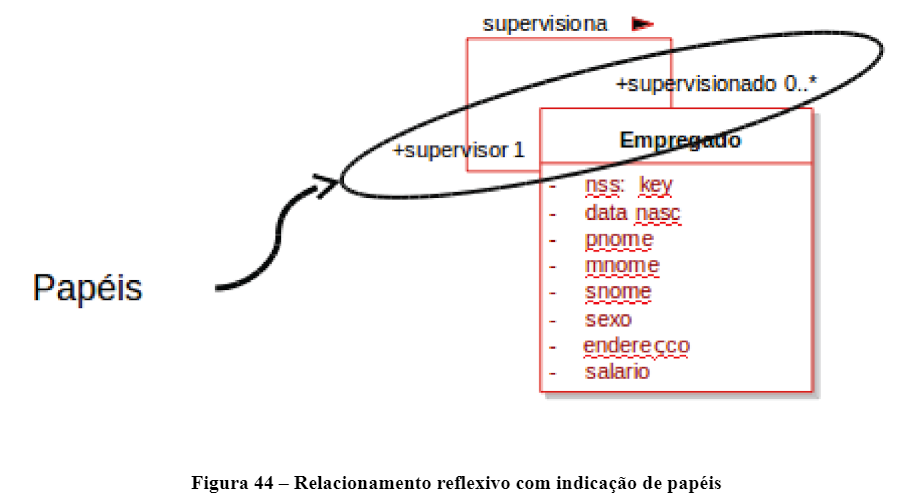
“Ocorrem quando são identificados relacionamentos entre objetos de duas classes distintas. Este tipo de associação é, em geral, a mais comumente encontrada”. (GUEDES, 2009, p. 109). Exemplos de associações binárias podem ser observados na figura 36.

**Associação Unária ou Reflexiva e atribuição de Papéis**

“São relacionamentos que ocorrem entre objetos de uma mesma classe”. (GÓES, 2014, p. 141) Em outras palavras, “este tipo de associação ocorre quando existe um relacionamento de um objeto de uma classe com objetos da mesma classe”. (GUEDES, 2009 p. 107)

Nestes casos, pode-se perceber que uma única classe é encontrada no diagrama e que seus objetos relacionam-se com outros objetos dela. Um exemplo clássico deste tipo de relacionamento é a classe de EMPREGADO que pode ter um relacionamento reflexivo consigo mesma onde a associação é de chefia ou supervisão.

Em casos como estes pode haver confusão ou falta de clareza em quem cumpre qual papel e, portanto, torna-se necessário a indicação de papéis. Entenda que a indicação de papéis não é obrigatória e é utilizada apenas quando necessária para aumentar a clareza sobre o relacionamento. Pode também ser utilizada em outros tipos de relacionamento como binário ou N-ário. Na figura 44 pode ser visto um exemplo de relacionamento reflexivo com indicação de papéis.

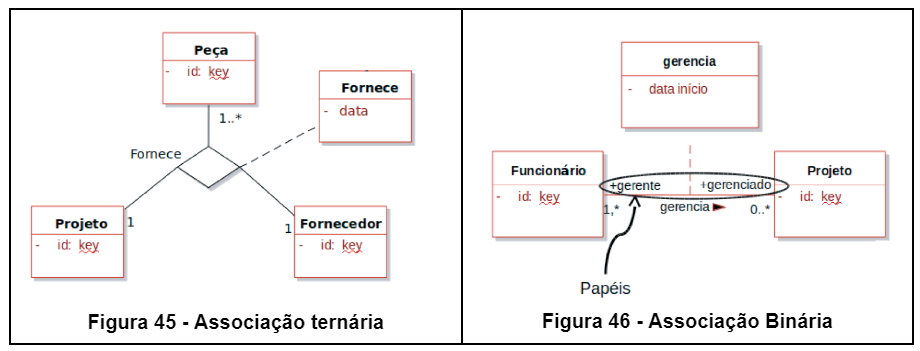


**Associação Ternária ou N-ária**

“Associações ternárias ou n-árias são associações que conectam objetos de mais de suas classes. São representadas por um losango para onde convergem todas as ligações da associação”. Um exemplo de um relacionamento ternário com classe associativa pode ser observado na figura 45

**Classe Associativa**

“Este tipo de relação ocorre normalmente quando existe um relacionamento cuja multiplicidade entre as classes é ‘muitos (\*)’ em ambas as extremidades da relação” (GÓES, 2009, p. 147). “As classes associativas são necessárias nos casos em que existem atributos relacionados à associação que não podem ser armazenados por nenhuma das classes envolvidas” (GUEDES, 2009, p. 115) É importante frisar que Classes Associativas não tem atributo chave! Um exemplo de um relacionamento binário pode ser visto na figura 46.



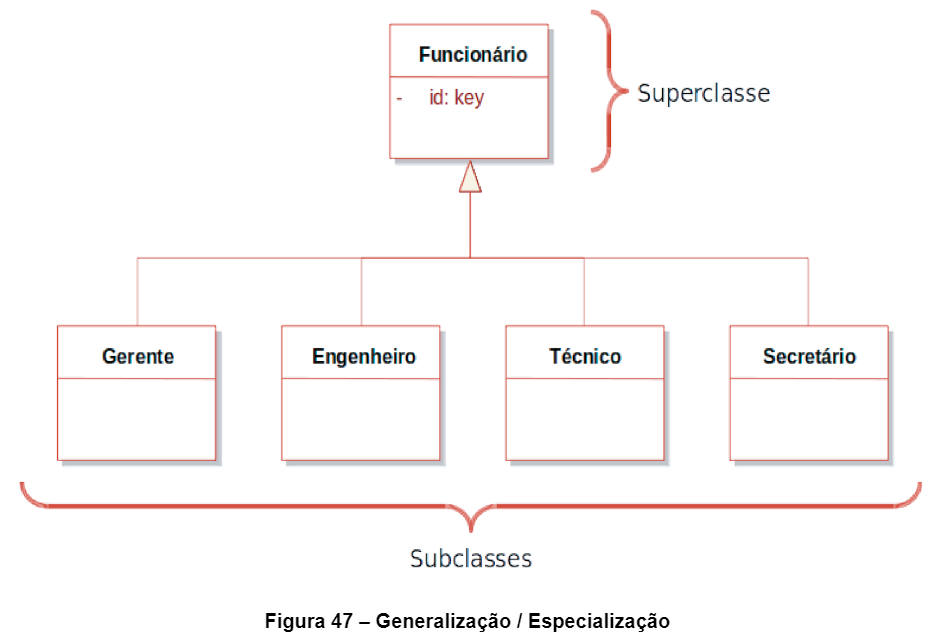
**Generalização / Especialização**

Especialização é o processo de classificar uma classe de objetos em subclasses mais especializadas. Generalização é o processo inverso de generalizar várias classes em uma classe abstrata de mais alto nível [...]. A Especialização é o refinamento conceitual, enquanto a generalização é a síntese conceitual. (ELMASRI; NAVATHE, 2011, p. 178)

Este é um tipo especial de associação. É utilizada quando ocorre herança entre as classes, identificando as classes-mãe chamadas gerais e as classes-filhas ou subclasses, chamadas especializadas, demonstrando assim a hierarquia entre elas. (GUEDES, 2009)

A generalização/especialização ocorre quando existem duas ou mais classes com características muito semelhantes. Desta forma, para evitar ter de atributos e/ou métodos idênticos e, assim, conseguir reutilizar código, se cria uma classe geral onde são declarados os atributos e métodos comuns a todas as classes envolvidas no processo e depois declara-se as classes especializadas ligadas a ela, que então, herdam duas características, podendo também ter métodos ou atributos próprios. (GUEDES, 2009)

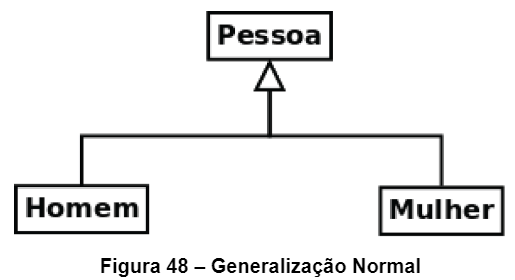
Este tipo de associação também pode ser chamada de “é um” pois, uma classe geral “é um” objeto da classe especializada, como, pode ser observado na figura 47. Neste exemplo, um funcionário pode ser um gerente, um engenheiro, um técnico ou um secretário, além é claro, de qualquer outro funcionário que não o destas quatro subclasses. (ELMASRI; NAVATHE, 2011)



**Generalização Normal**

Na generalização normal a classe mais específica, chamada de subclasse, herda tudo da classe mais geral, chamada de superclasse. Os atributos, operações e todas as associações são herdados.

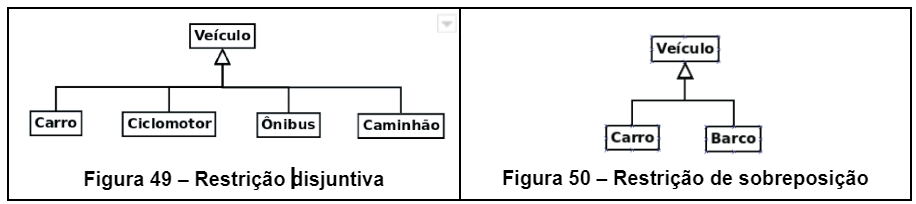
A generalização normal é representada por uma linha entre as duas classes que fazem o relacionamento, sendo que se coloca uma seta no lado da linha onde se encontra a superclasse indicando a generalização, como pode ser observado na figura 48.

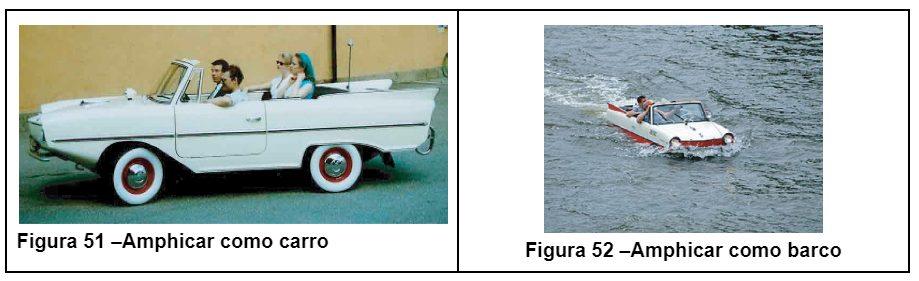


**Restrições de Sobreposição e Disjuntiva**

A Restrição de disjunção especifica que as subclasses da especialização devem ser mutuamente exclusivas, ou seja, uma entidade pode pertencer a somente uma subclasse da especialização. Este é o caso do exemplo da figura 49, já que um veículo somente pode ser carro ou ciclomotor ou ônibus ou caminhão, exclusivamente. (GUEDES, 2009)

Contrária a essa situação, temos que uma entidade pode pertencer a mais de uma subclasse ao mesmo tempo. Nesse caso, dizemos que há uma sobreposição, ou seja, uma mesma entidade pode pertencer a mais de uma subclasse da especialização. (GUEDES, 2009) Este seria o caso da figura 50, onde um veículo anfíbio pode ser carro e barco simultaneamente, caso do Amphicar model 770, veículo alemão anfíbio fabricado nos anos de 1961 e 1965, num total de 3.878 unidades, e que pode ser visto nas figuras 51 e 52. Seu nome é a vem dos conceitos de “Anfíbio” e “carro”, no inglês. (WIKIPEDIA, 2004)





**Restrição de Integralidade**

A restrição de integralidade pode ser total ou parcial. Ela será total quando toda entidade de uma superclasse pertencer a, pelo menos, uma das subclasses dessa superclasse, ou seja, no exemplo da figura 48, todas as pessoas são homens ou mulheres, no sentido biológico. Não há uma terceira opção. (GUEDES, 2009)

A parcialidade da restrição ocorre quando uma entidade da superclasse não pertence a nenhuma das subclasses definidas. Isso pode ocorrer, por exemplo, na figura 49 onde, se houver a possibilidade de existir um veículo anfíbio, ele não se classifica em nenhuma das sub classes. (GUEDES, 2009)

Desta forma, pode haver quatro possibilidades de restrições, considerando-se que os dois tipos (total e parcial) de restrições são independentes:

a) Disjunção total.

b) Disjunção parcial.

c) Sobreposição total.

d) Sobreposição parcial.

Uma destas possibilidades deve ser informada em toda generalização/especialização. Quando há apenas uma subclasse, subentende-se que é uma Disjunção Parcial. (GUEDES, 2009)

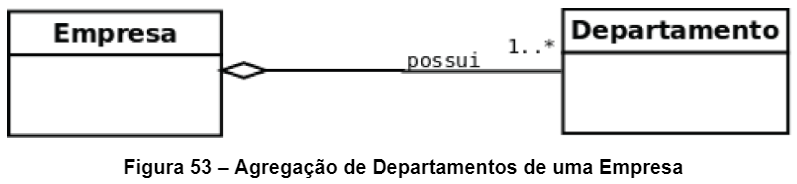
A UML permite que se construa uma hierarquia múltipla, que é quando uma classe herda características de duas ou mais classes, entretanto, não utilize esta solução pois não pode ser implementada em SGBDs relacionais e nem todas as linguagens de programação tem essa capacidade de implementação.

**Agregação**

A agregação é um tipo de relacionamento no qual duas classes estão inseridas em um contexto “todo-parte”, entre pares (um objeto não é mais importante que o outro). É um relacionamento em que um objeto contém o outro, numa associação de posse. (GUEDES, 2009)

As classes deste tipo de relacionamento podem viver de forma independente, de forma que os objetos da parte constituinte ou da agregada sejam independentes com relação à vida, porém ambas pertencem a um único todo. As palavras chaves usadas para identificar uma agregação são: “consiste em”, “contém”, “é parte de”. (GUEDES, 2009)

O símbolo de agregação difere do de associação por conter um losango na extremidade da classe que representa o todo. Um exemplo de agregação pode ser observado na figura 53.

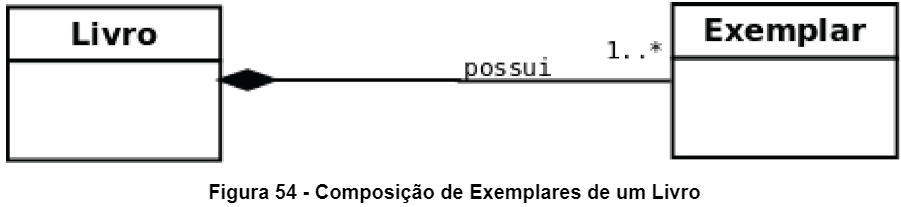


**Composição**

Uma associação de composição nada mais é do que uma variação da agregação, ocorrendo quando duas classes só têm sentido no momento em que estiverem associadas. Isso

implica que, se uma instância da classe deixar de existir, todas as outras associadas por composição “deixarão” de existir também. Este é o conceito aplicado às entidades fracas. (GUEDES, 2009)

O símbolo de composição diferencia-se graficamente do símbolo de agregação por utilizar um losango preenchido. Um exemplo pode ser visto na figura 54, onde a entidade fraca “Exemplar” está associada por composição à classe “Livro”.



Na figura 55 (Diagrama de Classes) está representado o mesmo banco de dados da figura 32 (DE-R)

