1º Semestre

Fundamentos de Banco de Dados

Análise e Desenvolvimento de Sistemas
1º semestre

IMPACTA

Prof. Alexandre Leite Rangel

Doutor em Ciências e Mestre em Enfermagem pela Escola de Enfermagem de Ribeirão Preto da USP (2010), onde pesquisou o problema computacional da área de Otimização e Programação Linear Nursing Schedule Problem. Desenvolveu no Mestrado (2007), um Sistema de Apoio à Tomada de Decisão para Elaboração Automática da Escala de Trabalho dos Profissionais da Enfermagem, que foi avaliado durante o seu doutoramento. Anteriormente concluiu uma especialização Latu Sensu em Análise de Sistemas pela Universidade de Ribeirão Preto (1996). Graduou-se no Bacharelado em Análise de Sistemas, também pela UNAERP (1994). Obteve na FATEC a Licenciatura Plena com habilitação em Processamento de Dados (1999). É professor desde 1994, tendo lecionado na UNAERP, USP, UNIFEB, UNIP e FATEC e atualmente, atua no Claretiano Centro Universitário e na Faculdade Impacta de Tecnologia. Destaque para o projeto da Fábrica de Software do curso de Sistemas de Informação do UNIFEB, ao qual coordenou entre 2013 e 2016, onde implementou vários sistemas e coordenou alunos e bolsistas. Foi analista de sistemas do HCFMRP/USP - Hospital das Clínicas da Faculdade de Medicina de Ribeirão Preto da Universidade de São Paulo por 9 anos. É também autor de 6 livros, sendo 5 sobre Bancos de Dados e um sobre Desenvolvimento para Web.

O professor

Prof. Oswaldo Kotaro Takai

Mestre em Ciências de Computação e Matemática Computacional pela USP de São Carlos (ICMC-USP) e Graduado em Ciências da Computação também pela USP São Carlos. Consultor em desenvolvimento e modelagem de sistemas computacionais, mais de 20 anos de experiência como docente do ensino superior nas áreas de Computação, Engenharia de Software, Banco de Dados e Sistemas de Informação. Atuou como consultor do PNUD, foi especialista em Engenharia de Sistemas do Atech Tecnologias Críticas do Grupo EMBRAER e consultor interno da empresa VAGAS Tecnologia. Atualmente é prestador de serviço - Superintendência de Tecnologia da Informação - USP, professor convidado da Universidade Estadual de Campinas, professor da Faculdade Impacta de Tecnologia e desenvolve seu projeto de pesquisa junto ao grupo de Banco de Dados do IME-USP. Coordenador dos cursos de Bacharelado em Sistemas de Informação e do Curso Superior de Tecnologia em Gestão da Tecnologia da Informação. Certificado em Engenharia de Requisitos: CPRE-FL Certified Professional for Requirements Engineering - Foundation Level, dado pelo IBQTS ? Brasil, Licensed Examination Institute, como Licensed Examination Institute do IREB® (International Requirements Engineering Board) e ICAgile Certified Professional (ICP) Certificate.

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. A importância dos bancos de dados foi impulsionada nos últimos anos devido ao crescimento das aplicações web, das implantações de ERP's (Enterprise Resourcing Process), de BI (Businnes Intelligence), dentre outras.

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.

12 minutes ago	1 hour ago	2 hours ago
#VideoShowAoVivo	#CiteUmaMentira	#CiteUmaMentira
Viola Davis	Viola Davis	Ismaily
#CiteUmaMentira	Ismaily	Viola Davis
Ismaily	#Outono	#Outono
#Outono	Áries	Áries
Áries	#Kenzolovesbritney	#Encontro
#Kenzolovesbritney	#TercaDetremuraSDV	#TercaDetremuraSDV
#fsradiobrasil	#VídeoShowAoVivo	#TuristandoNoTwitter
UFSM	Leandro Pereira	Leandro Pereira
Shakhtar	Gonzalo Carneiro	Gabriel Valim

Figura 1 – Trending Topics Brazil, 20/03/2018, 15h19 (TRENDS24, 2018)

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. Um bom exemplo foram os inúmeros sistemas desenvolvidos em linguagem COBOL.

E por falar em COBOL, você já ouviu falar em "Bug do Milênio"? Foi uma falha no tratamento das datas nos programas desenvolvidos antes do ano 2000. Os programas desenvolvidos em

linguagem COBOL foram, muito provavelmente, os que tiveram a maior quantidade de erros a serem corrigidos. As datas eram armazenadas em texto no formato YYMMDD, ou seja, o réveillon de 2000 teríamos 991231 e 000101, tornando todos os cálculos com datas incorretos. Em um SGBD, modificações deste tipo são simples embora o tempo de execução dependa da quantidade de dados a serem modificados. Em arquivos, como no COBOL, as modificações precisam ser feitas uma-a-uma e em todos os programas que fazem acesso aquele dado.

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)

Capítulo 1 – Fundamentos de Bancos de Dados

1. Fundamentos de Bancos 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.

1.1 – 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.

De acordo com Elmasri e Navathe (2011), os bancos de dados são componentes essenciais em nossa vida cotidiana, presente na maioria as atividades e aplicativos que envolvem algum tipo de manipulação de informações. Por exemplo, quando você consulta as suas notas ou suas faltas, quando vai até a biblioteca e pega um livro emprestado ou quando compra um produto pela internet ou mesmo em várias lógicas físicas, tais como as grandes redes como Magazine Luíza ou Casas Pernambucanas.

Estas operações são exemplos do que podemos chamar de aplicações de bancos de dados tradicionais, em que a maior parte da informação é armazenada como textos ou números, de forma estruturada (EMASRI; NAVATHE, 2011).

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" (EMASRI; NAVATHE, 2011).

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. (EMASRI; NAVATHE, 2011).

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)

1.2 – 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  FD 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.

1.3 – 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)

Capítulo 2 – Sistema de Banco de Dados

2. 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).

2.1 – 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)

2.1.1 – Modelo hierárquico

O modelo hierárquico surgiu nos anos de 1960 e os dados são organizados em formato

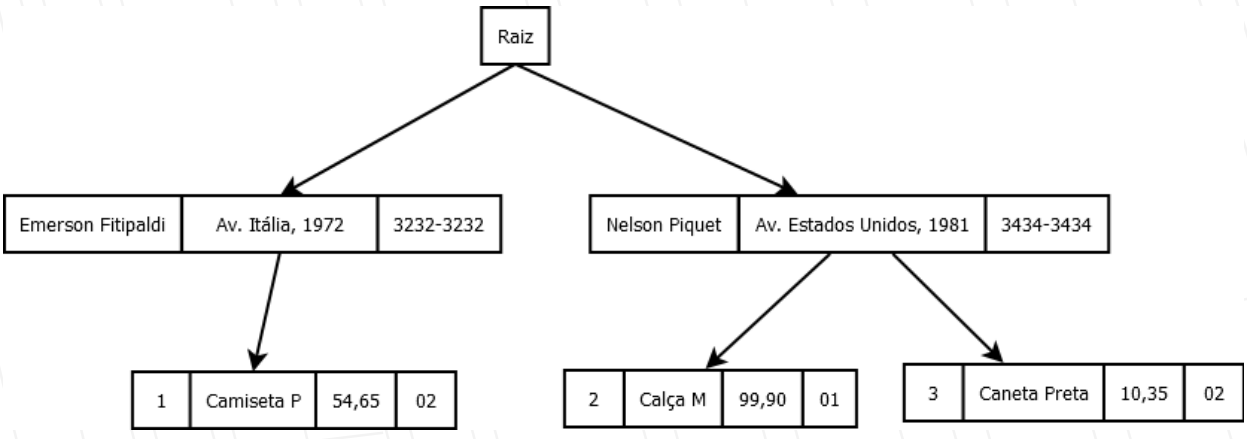


Figura 2 – Modelo de Banco de Dados Hierárquico

2.1.2 – 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. Um esquema deste tipo de banco de dados pode ser observado na figura 3.

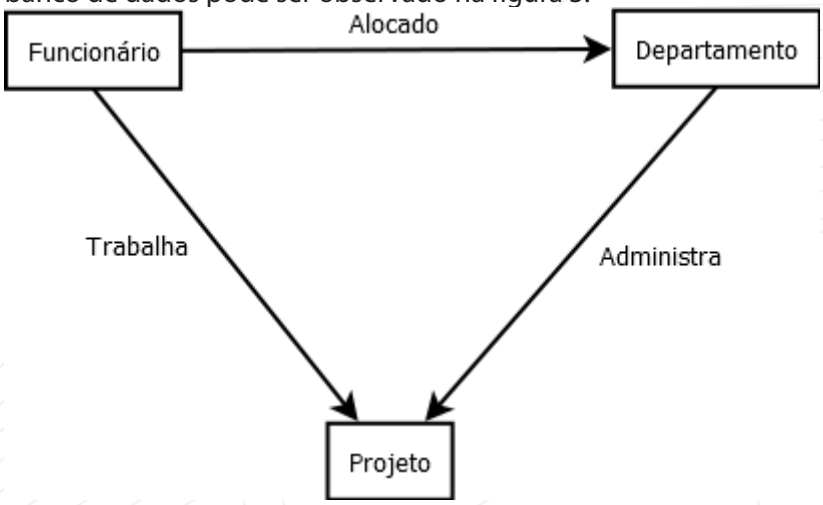


Figura 3 – Banco de Dados de Rede

2.1.3 – 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. Um exemplo simplificado disso pode ser observado na figura 4.

ALUNO

RA	NOME	TELEFONE	EMAIL
100	Donna T. Powell	(47) 12531-6334	eu@etrutrum.net
101	Palmer P. Patterson	(25) 99978-9761	pharetra.Nam.ac@maurisipsum.net
102	Destiny I. Mccall	(56) 55074-4741	urna.nec@necante.org
103	Brody Q. Walker	(04) 71350-8177	magna.Cras.convallis@laciniaorci.org

TURMA

CODIGO_TURMA	PERIODO_LETIVO	INICIO	TÉRMINO
1	2018/1	19/02/2018	29/06/2018
2	2018/1	19/02/2018	29/06/2018
3	2018/1	19/02/2018	29/06/2018

MATRÍCULA

RA	CODIGO_TURMA	DATA_MATRICULA
100	11	21/02/2018
101	11	02/02/2018
102	11	20/02/2018
103	11	05/02/2018
104	11	05/02/2018
105	11	17/02/2018
106	11	25/02/2018
107	11	01/02/2018
108	11	10/02/2018
109	11	01/02/2018
110	11	05/02/2018
111	11	02/02/2018
112	11	08/02/2018
113	11	07/02/2018
114	11	09/02/2018

Figura 4 – Banco de Dados Relacional

2.2 – 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).

2.3 – 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.

```
01  sid = transaction.savepoint()
```

02	try: # início do bloco protegido
03	b.save() # Poderia lançar uma exceção
04	transaction.savepoint_commit(sid)
05	except IntegrityError:
06	transaction.savepoint_rollback(sid)
07	c.save() # Sucesso, e a.save() não será desfeita

Listagem 2 – Transação em Python (django)

Observando o código na listagem 02, 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. A figura 5 representa o esquema de uma transação.

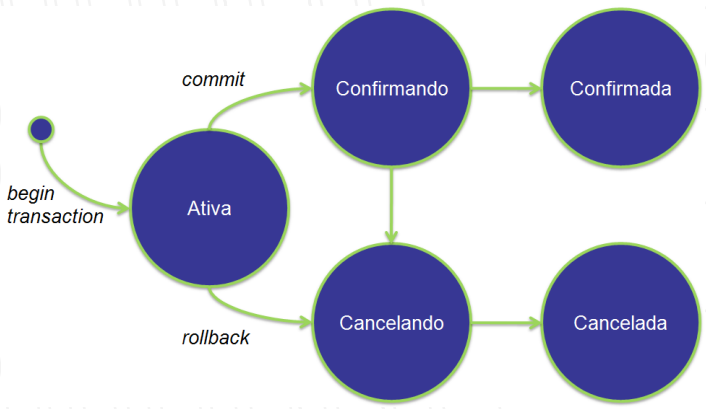


Figura 5 – Esquema de Transações no Banco de Dados

2.4 – 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 6. 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.

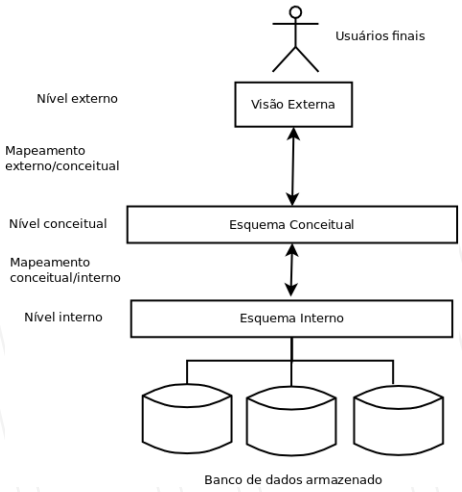


Figura 6 – A arquitetura de três esquemas

2.4.1 – 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.

Capítulo 3 – Modelagem Conceitual

3. Modelagem Conceitual

A modelagem conceitual é a representação de uma porção da realidade através de um modelo conceitual que a represente e mantenha conexão com a realidade existente.

Neste capítulo serão estudados os conceitos de modelo conceitual e sua representações.

3.1 – Modelos em Geral

Dentre os vários significados para modelo, pode-se destacar que modelo é "esquema teórico que representa um fenômeno ou conjunto de fenômenos complexos e permite compreendê-los e prover-lhes a evolução". (HOUAISS, 2009)

Por outro lado, um dos significados de conceito e que melhor se aplica aqui é "representação mental de um objeto abstrato ou concreto, que se mostra como um instrumento fundamental do pensamento em sua tarefa de identificar, descrever e classificar os diferentes elementos e aspectos da realidade" (HOUAISS, 2009).

Desta forma, é possível definir que um modelo conceitual é um esquema teórico que representa de forma simplificada uma parte da realidade. Como exemplo clássico, pode-se citar o mapa do metrô. Embora represente todas as linhas, ele é uma simplificação da realidade pois não contém todos os detalhes delas, sequer contém o traçado idêntico aos das linhas reais, como pode ser observado na figura 7. Esta representação, com raras exceções, pode ser entendida por praticamente todas as pessoas.

Um modelo conceitual pode ou não ter um padrão visual. Um mapa mental, por exemplo, é uma simplificação da realidade uma vez que representa os conceitos apresentados em uma aula ou reunião, de forma simplificada e que também tem uma ligação com a realidade pois os conceitos ali representados são reais e não foram inventados. Entretanto, um mapa mental não contém um padrão visual já que ele pertence a uma única pessoa e as pessoas os criam de formas diferentes.

Houaiss (2009) define padrão como sendo uma "base de comparação consagrada como modelo por consenso geral ou por determinado órgão oficial". Desta forma, os bancos de dados devem ser projetados utilizando-se um modelo conceitual em notação que possua um padrão visual.

Figura 7 – Mapa do Metrô de São Paulo (METRO, 2018)

3.2 – Modelos Conceituais

Segundo Heuser (2009, p. 25), "um modelo conceitual é uma descrição do banco de dados de forma independente de implementação em um SGBD. O modelo conceitual registra que dados podem aparecer no banco de dados mas não registra como estes dados estão armazenados".

Modelar significa criar um modelo que explique as características de funcionamento e comportamento de um software. A partir deste modelo é que o software será criado o que facilita o seu entendimento e seu projeto, uma vez que, com características principais, erros de programação, de projeto e de funcionamento serão evitados. (WIKIPEDIA, 2004)

De acordo com Elmasri e Navathe (2011, p. 132),

A modelagem conceitual é uma fase muito importante no projeto de uma aplicação de banco de dados bem-sucedida. Geralmente o termo aplicação de banco de dados refere-se a um banco de dados em particular e aos programas relacionados que implementam as consultas e atualizações dele.

Para se modelar um banco de dados, utiliza-se o modelo Entidade-Relacionamento (ME-R), que é um modelo de dados conceitual popular de alto nível. Esse modelo e suas variações costumam ser utilizados para o projeto conceitual de aplicações de banco de dados, e muitas ferramentas de projeto de banco de dados empregam seus conceitos. Ao se empregar o ME-R, cria-se o Diagrama Entidade-Relacionamento (DE-R). Entretanto, a metodologia de modelagem de objetos, como a UML (Unified Modeling Language) tornam-se a cada dia mais populares para o projeto de banco de dados, onde se utiliza o diagrama de classes. (ELMASRI; NAVATHE, 2011)

O uso de modelos conceituais para projeto de banco de dados é largamente empregado. A primeira etapa do projeto de um banco de dados é o levantamento e análise de requisitos. De acordo com Sommerville (2011, p. 65), "A especificação de requisitos é o processo de escrever os requisitos de usuário e de sistema em um documento de requisitos. Idealmente, os requisitos de usuário e de sistema devem ser claros, inequívocos, de fácil compreensão, completos e consistentes".

Sommerville (2011) ainda afirma que os requisitos de usuário devem descrever os requisitos funcionais e não funcionais e que devem ser compreensíveis para os usuários do sistema que não tenham conhecimentos técnicos. O resultado desta etapa é um conjunto de requisitos dos usuários escrito de forma concisa (ELMASRI e NAVATHE, 2011), que pode ser chamado de minimundo. O documento de requisitos não deve conter detalhes de arquitetura ou projeto do sistema. Por este motivo, não utilize jargões de software, notações estruturadas ou formais. (SOMMERVILLE, 2011)

Um requisito é um aspecto que o sistema proposto deve fazer ou uma restrição no desenvolvimento do sistema. De acordo com Sommerville (2011, pp. 65-66), "os requisitos de sistema são versões expandidas dos requisitos de usuário. […] Eles acrescentam detalhes e explicam como os requisitos de usuário devem ser atendidos pelo sistema".

Os requisitos de sistema devem descrever apenas o comportamento externo do sistema e suas restrições operacionais e não devem se preocupar como o sistema será projetado ou implementado. Já os requisitos de usuário são quase sempre escritos em linguagem natural, que no caso do Brasil é em português e, se necessário, acompanhados de diagramas ou tabelas para facilitar a compreensão. (SOMMERCILLE, 2011)

De acordo com Elmasri e Navathe (2011), "Em paralelo com a especificação dos requisitos de dados, é útil determinar os conhecidos requisitos funcionais da aplicação". Assim que os requisitos estiverem todos levantados, a próxima etapa então será a criação do modelo conceitual. Os mesmos autores ainda definem que:

O esquema conceitual é uma descrição concisa dos requisitos de dados dos usuários e inclui detalhes dos tipos de entidade, relacionamentos e restrições; estes são expressos usando os conceitos fornecidos pelo modelo de dados de alto nível.

A etapa seguinte no projeto de um banco de dados é a implementação real do próprio banco de dados utilizando um SGBD. Essa etapa é chamada de modelo lógico ou mapeamento do modelo de dados. Ela tem como resultado um esquema de banco de dados no modelo de dados da implementação do SGBD. (ELMASRI; NAVATHE, 2011)

A etapa final é a fase do projeto físico, na qual todas as estruturas de armazenamento

internas, organização de arquivo, índices, caminhos de acesso e parâmetros físicos do projeto para os arquivos do banco de dados são especificados. (ELMASRI; NAVATHE, 2011)

Uma visão esquemática destas fases pode ser vista na figura 8.

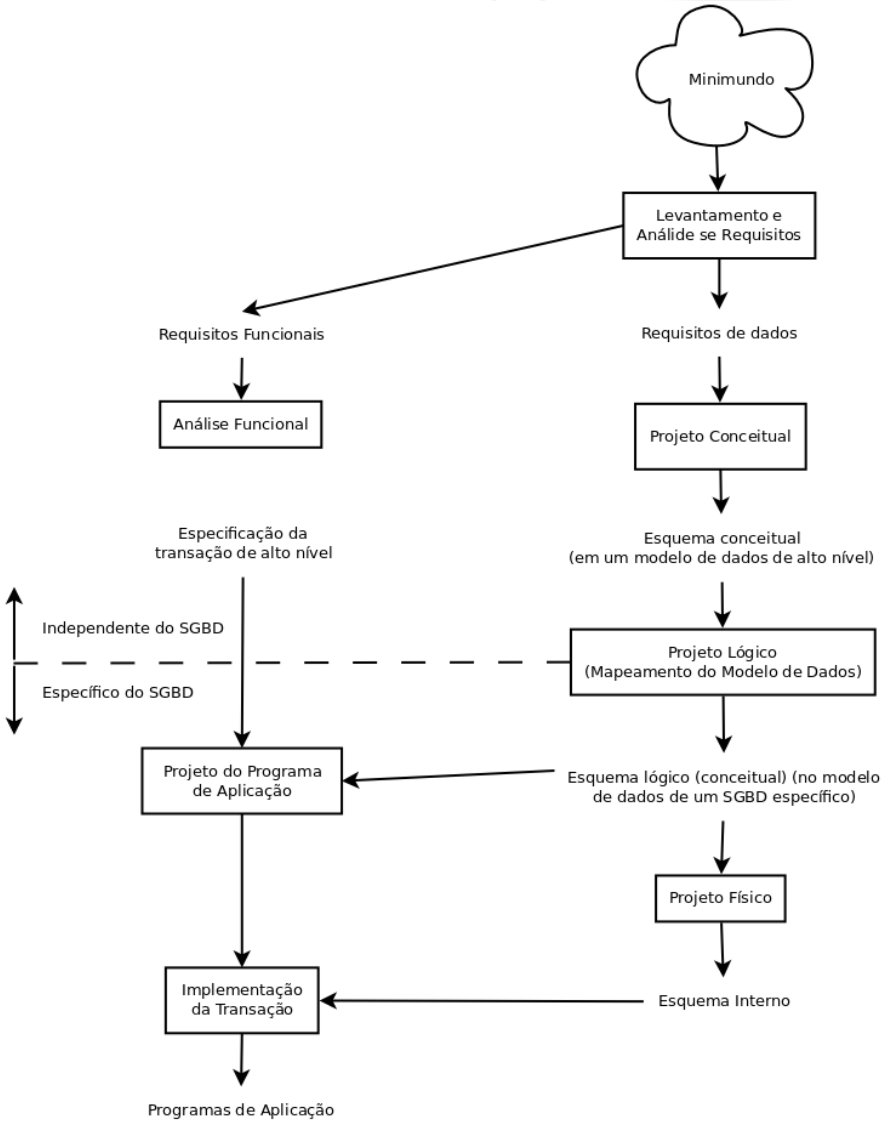


Figura 8 – Diagrama simplificado das fases do projeto de um banco de dados (ELMASRI; NAVATHE, 2011)

Capítulo 4 – Modelo Entidade-Relacionamento

4. 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)

4.1 – 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)

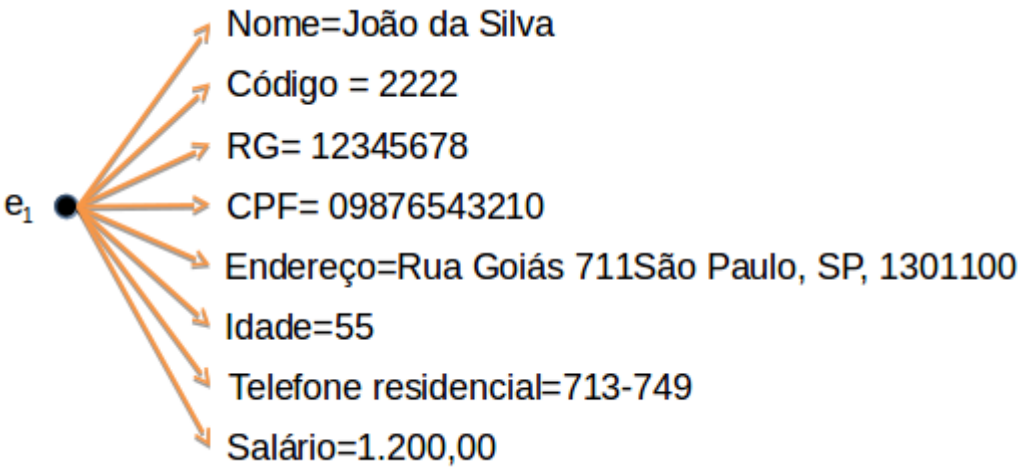


Figura 9 – Atributos de uma entidade.

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.

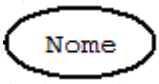


Figura 10 – Atributo simples

4.1.1 – 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 demostra o esquema do atributo composto endereço, apresentado na figura 9.

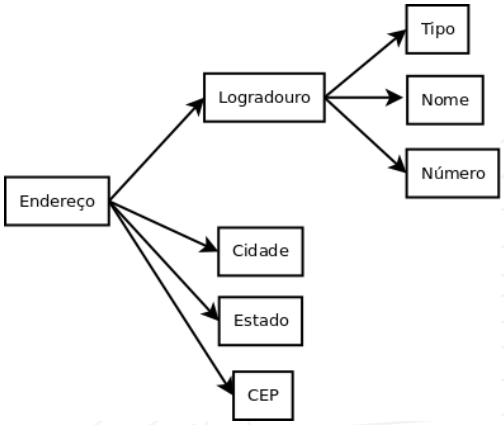


Figura 11 – Esquema de um atributo composto.

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

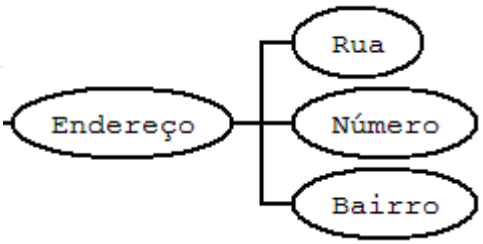


Figura 12 – Atributo Composto

4.1.2 – Atributos Multivalorados

A grande maioria dos atributos possui um único valor para uma determina 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.

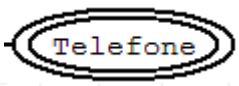


Figura 13 – Atributo Multivalorado.

Figura 14 – Carro com duas cores

Figura 15 – Cadastro de pessoa física com quatro referências comerciais

(ELMASRI; NAVATHE, 2011)

4.1.3 – 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.

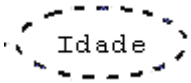


Figura 16 – Atributo Derivado

4.1.4 – 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.

4.2 – 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.

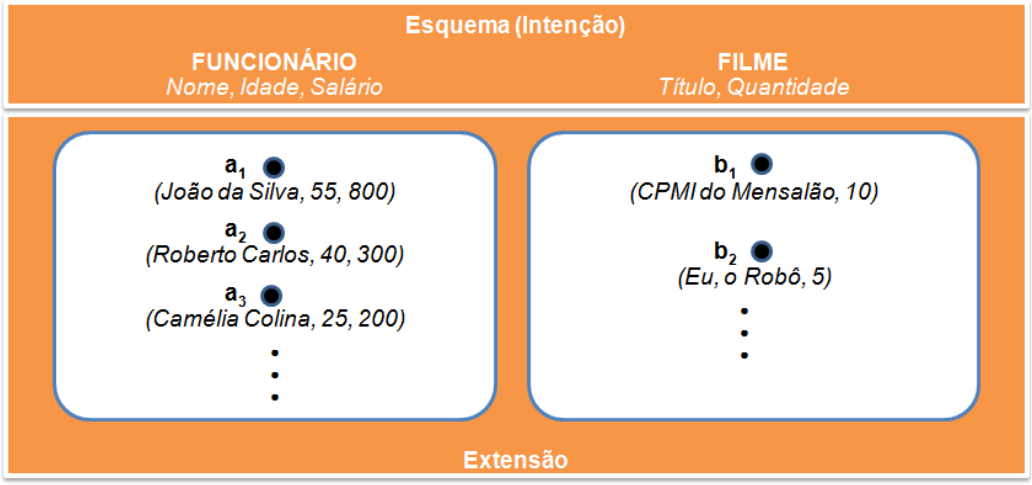


Figura 17 – Tipos de Entidades, Entidades e valores dos Atributos

4.2.1 – 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.

Figura 18 – Atributo-chave

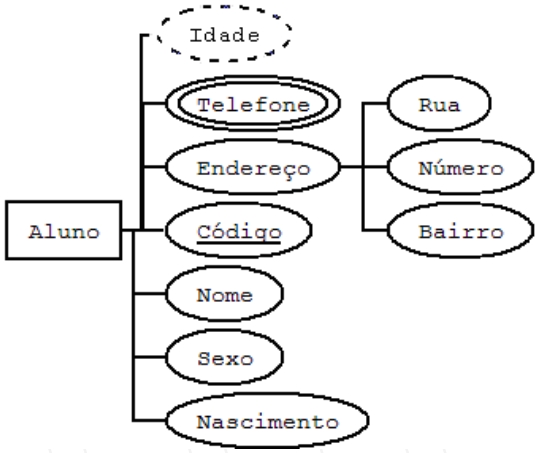


Figura 19 – Tipo de Entidade e Atributos

4.3 – 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 E_1, E_2,... E_n, 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)

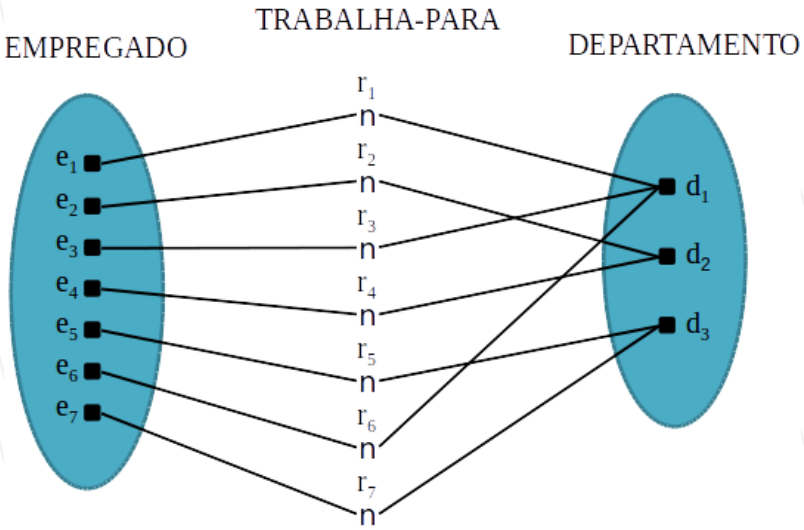


Figura 20 – Relacionamento entre dois Tipos de Entidade

"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 DER, o tipo de relacionamento é representado por um losango, como pode ser observado na figura 22.

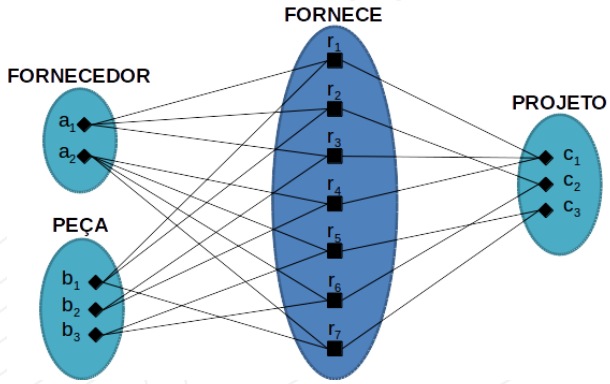


Figura 21 – Relacionamento com grau 3 (três)

Figura 22 – Tipo de Relacionamento

4.3.1 – 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)

4.3.2 – Nomes de Função e Relacionamentos Recusivos

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.

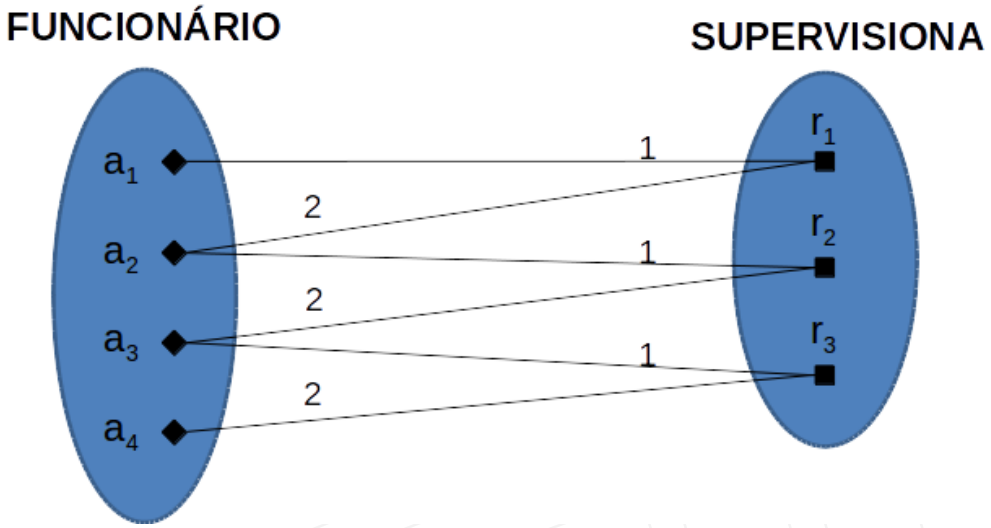


Figura 23 – Relacionamento Recursivo e Nomes de Função

4.4 – 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)

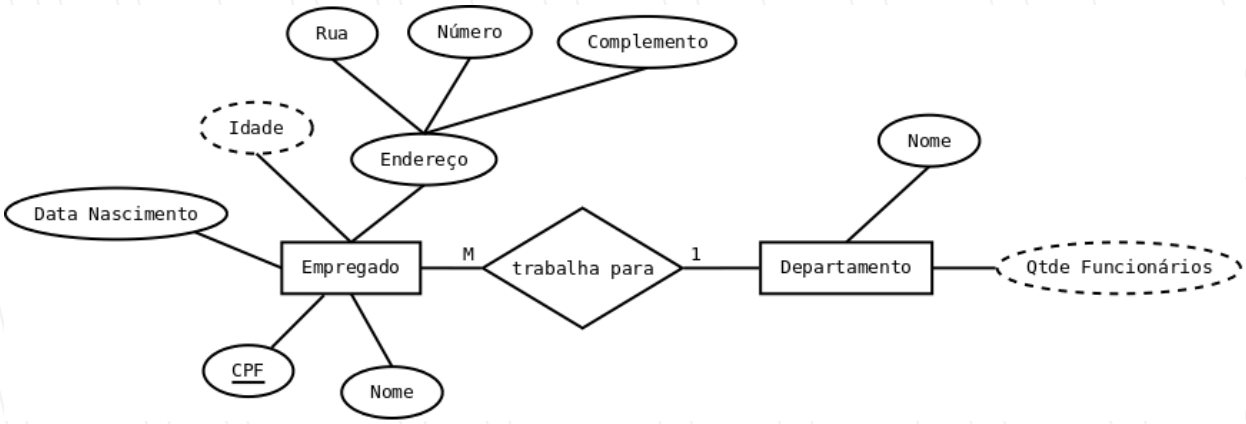


Figura 24 – Relacionamento binário com restrição de cardinalidade

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.

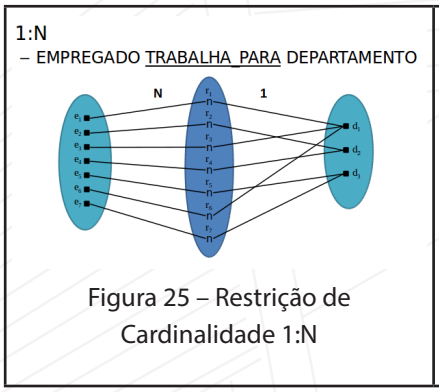


Figura 25 – Restrição de Cardinalidade 1:N

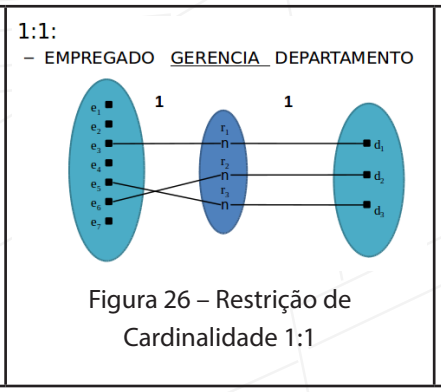


Figura 26 – Restrição de Cardinalidade 1:1

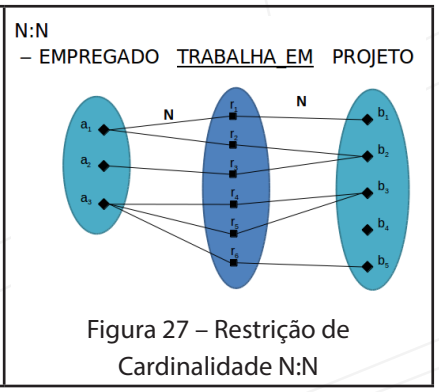


Figura 27 – Restrição de Cardinalidade N:N

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)

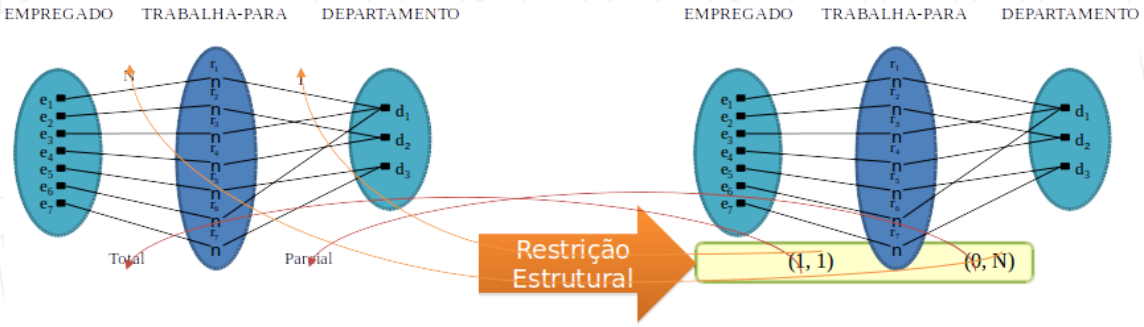


Figura 28 – Restrição Estrutural

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.

DataInício em
- EMPREGADO TRABALHA_PARA DEPARTAMENTO

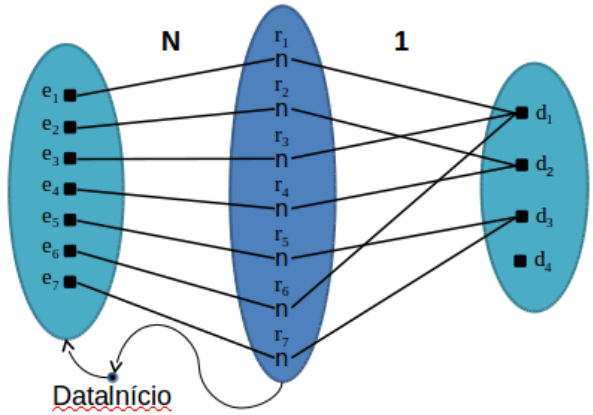


Figura 29 – Atributo de Relacionamento 1:N

4.5 – 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.

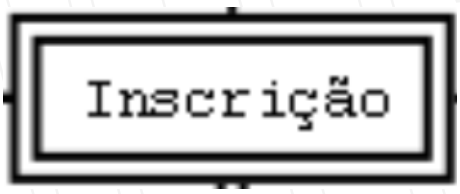


Figura 30 – Tipo de Entidade Fraca

4.6 – 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.

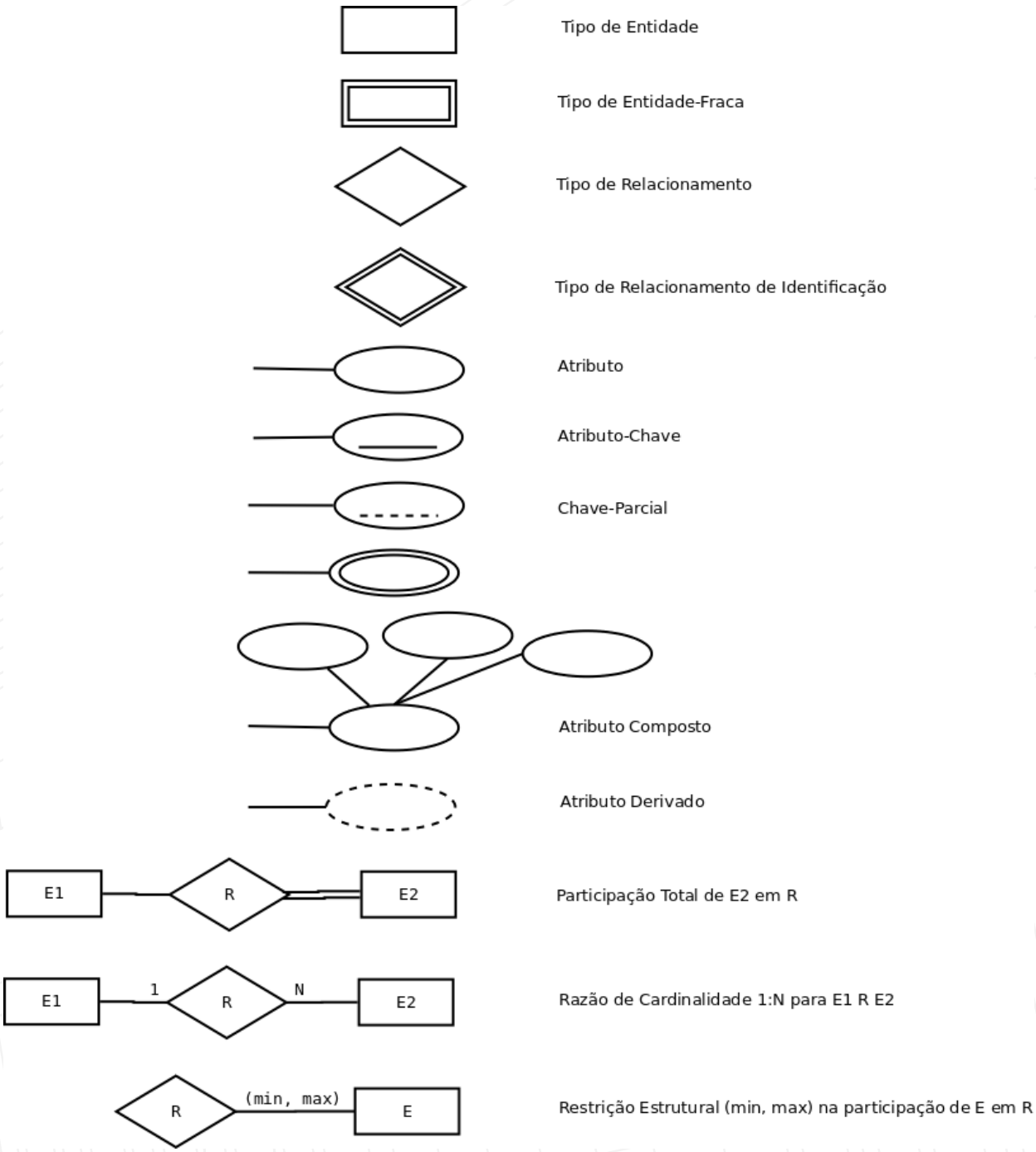


Figura 31 – Resumo da Notação para o Diagrama Entidade-Relacionamento

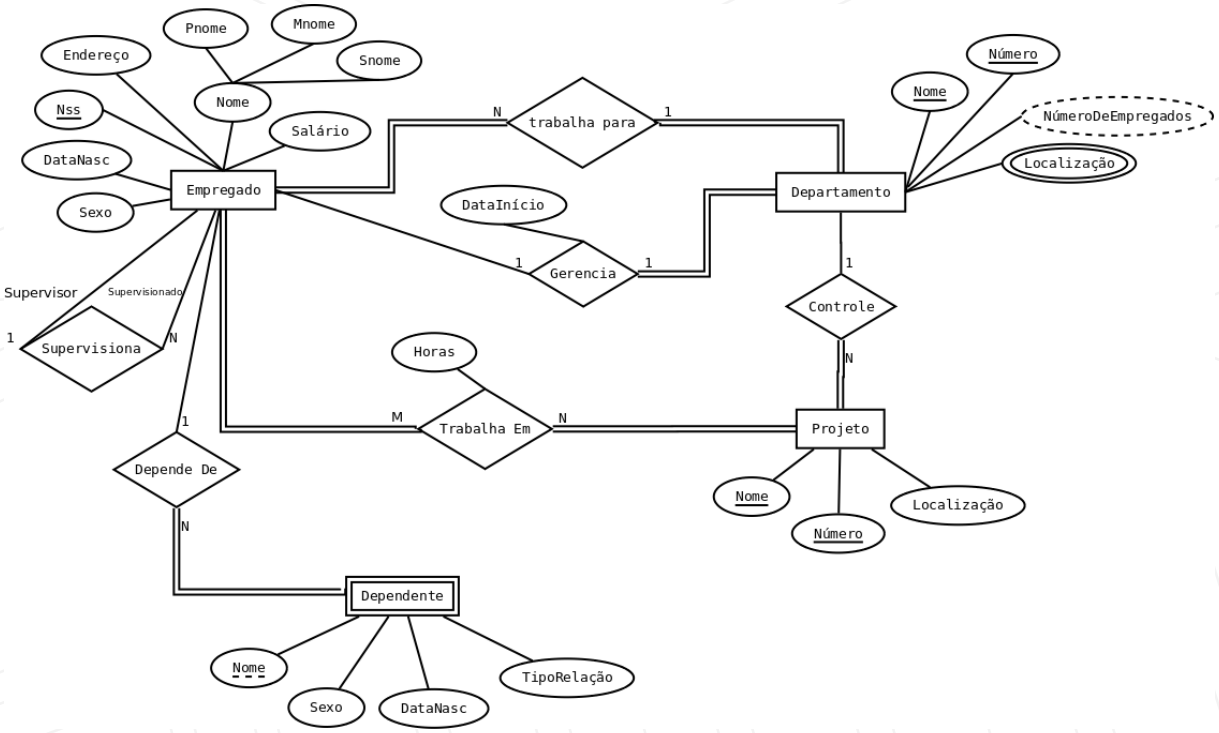


Figura 32 – DE-R para o esquema EMPRESA.

Capítulo 5 – Modelo Entidade-Relaciona-mento 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 visto na figura 33.

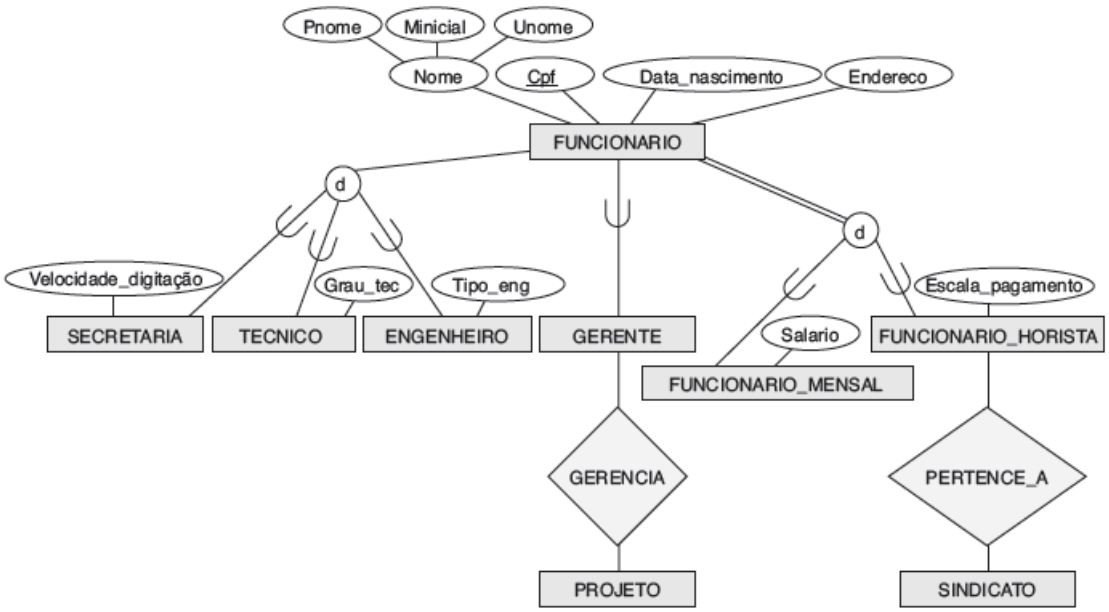


Figura 33 – Notação do EER para representar subclasse e especialização (ELMASRI; NAVATHE, 2011, p. 162)

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 EMERSOM 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)

5.1 – 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.

5.1.1 - Especialização

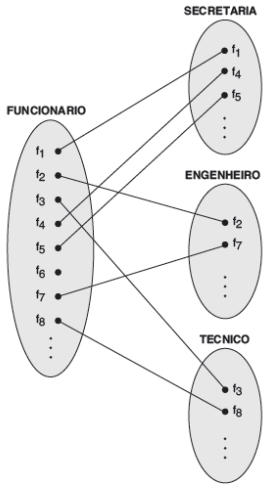


Figura 34 - Instâncias de uma especialização (ELMASRI; NAVATHE, 2011, p. 164)

"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/subclasse e especializações em um banco de dados, sendo que o primeiro motivo deve-se ao fato de que certos 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)

5.1.2 - 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 especifica, 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.

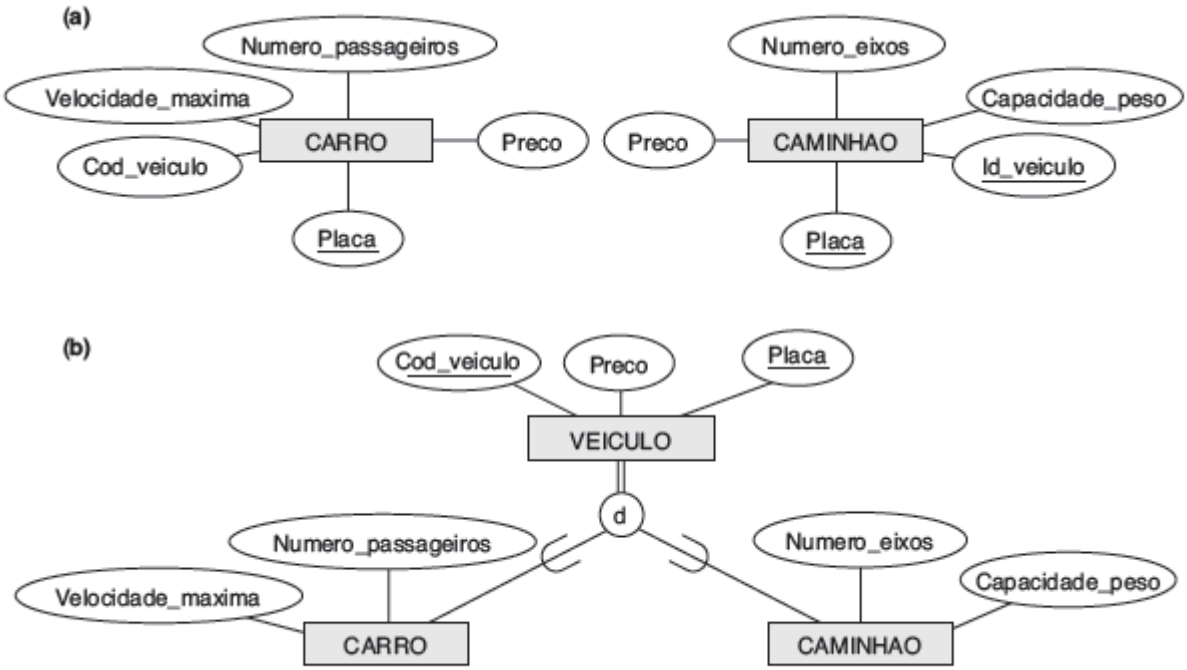


Figura 35 – Generalização de tipos de entidades específicas (ELMASRI; NAVARTHE, 2011, p. 165)

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)

5.1.3 – 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.

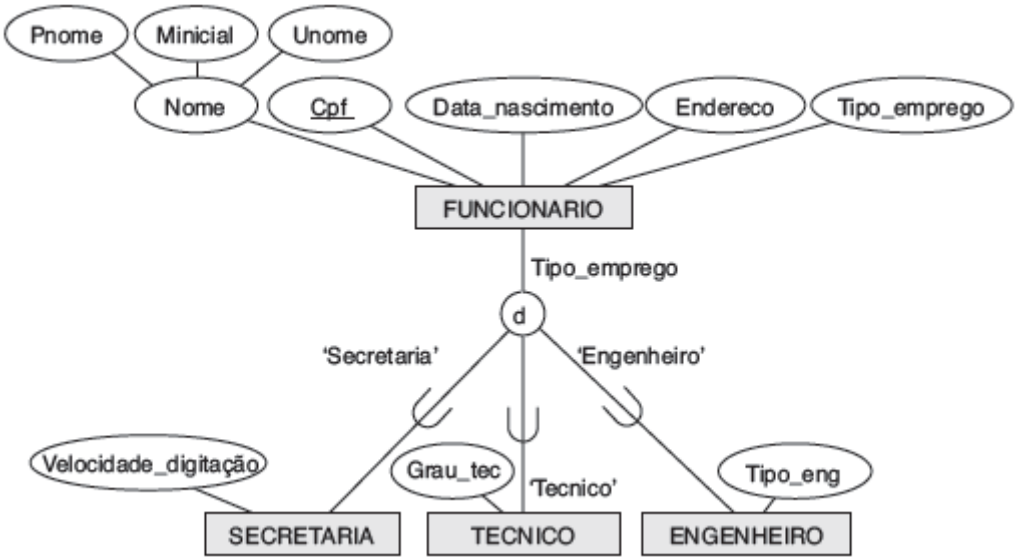


Figura 36 – Notação de para disjunção de uma especialização (ELMASRI; NAVATHE, 2011, p. 166)

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 pode uma linha simples. Um exemplo pode ser visto na figura 37. (ELMASRI; NAVATHE, 2011)

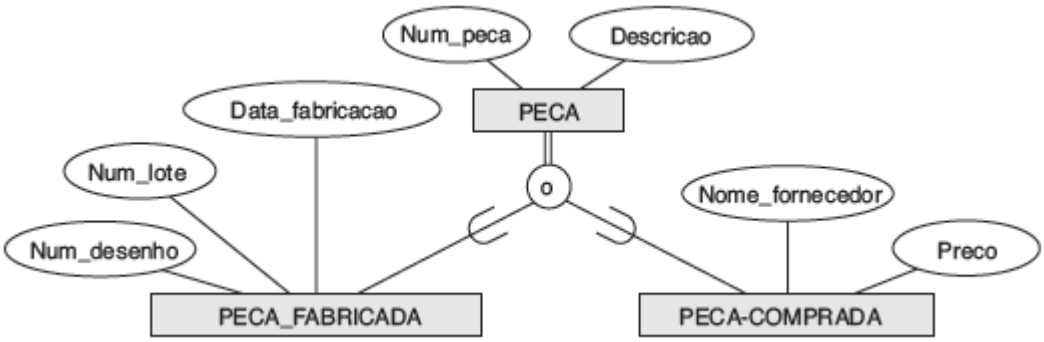


Figura 37 - Notação de para sobreposição de especialização (ELMASRI; NAVATHE, 2011, p. 167)

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.

5.2 – 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)

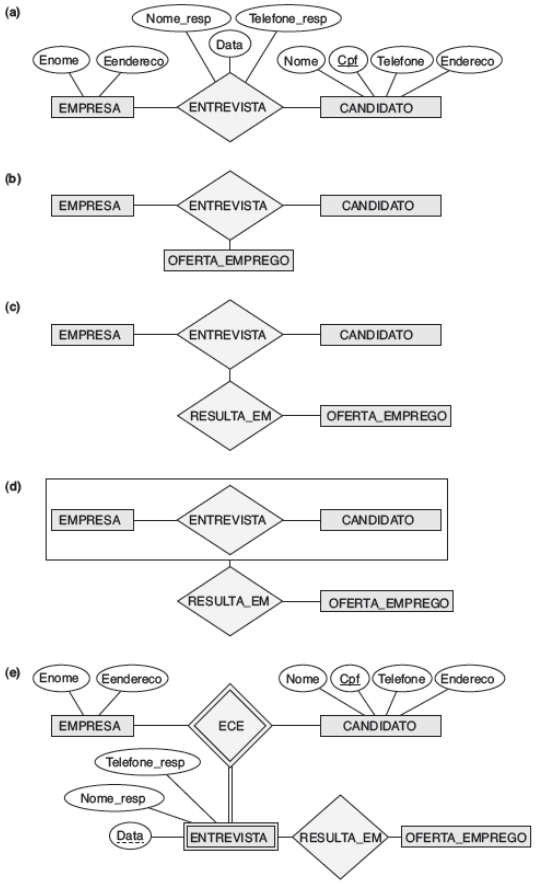


Figura 38 – Exemplo de agregação em EER
(ELMASRI; NAVATHE, 2011, p. 180)

Capítulo 6 – 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)

6.1 – 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 varia 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 tem 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)

6.1.1 – Nomes das Classes

De acordo com GÓES (2014, p. 134),

> O nome da Classes deve sempre estar no singular, em negruto 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.

Na figura 39, pode-se observar um exemplo dos compartimentos de uma classe.

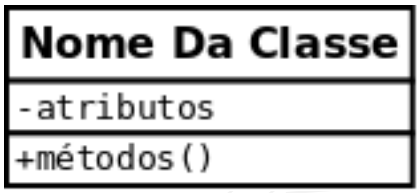


Figura 39 – Compartimentos do Diagrama de Classes

6.1.2 – 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.

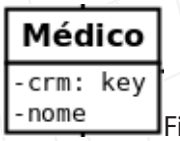


Figura 40 – Classe com atributos

6.1.3 – 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:

- publico, representado pelo símbolo "+"
- protegido, representado pelo símbolo "#"
- privado, representado pelo símbolo "-"

(GÓES, 2014)

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.

6.1.4 – Multiplicidade do atributo

No caso dos atributos, a multiplicidade determino a 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

6.1.5 – 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 e um atributo composto pode ser observado na figura 42.

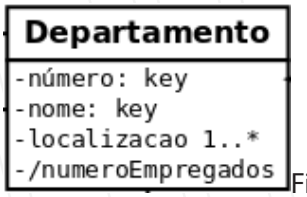


Figura 42 – Atributo derivado e atributo composto.

6.2 – 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.

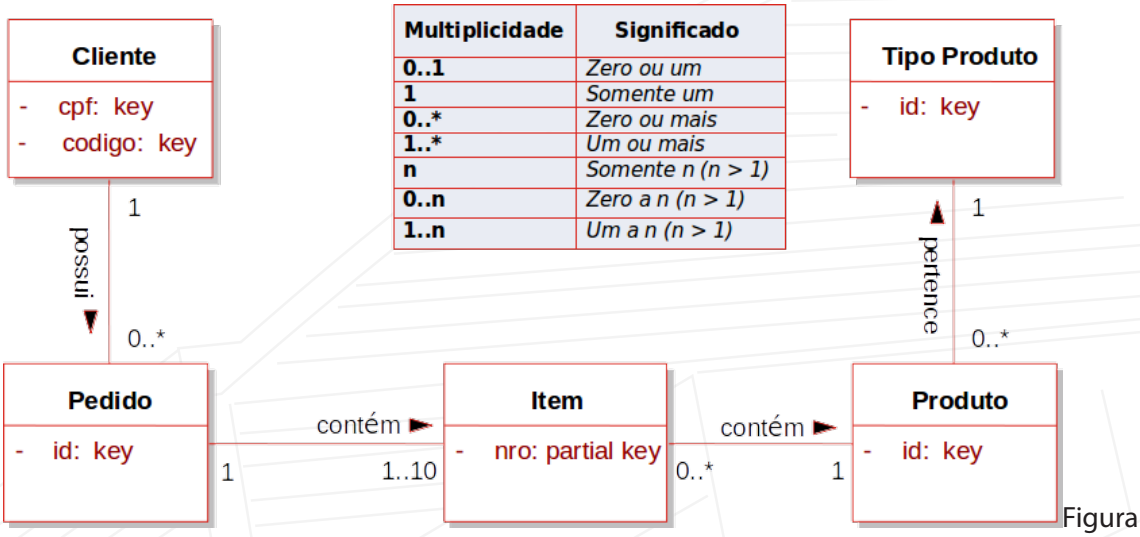


Figura 43 – Multiplicidade e Navegabilidade em Associações de Classes

6.3 – 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.

6.4 – 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.

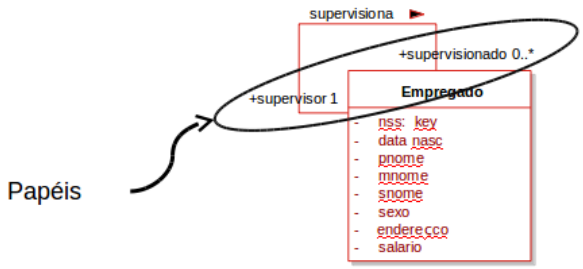


Figura 44 – Relacionamento reflexivo com indicação de papeis

6.5– 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

6.6 – 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.

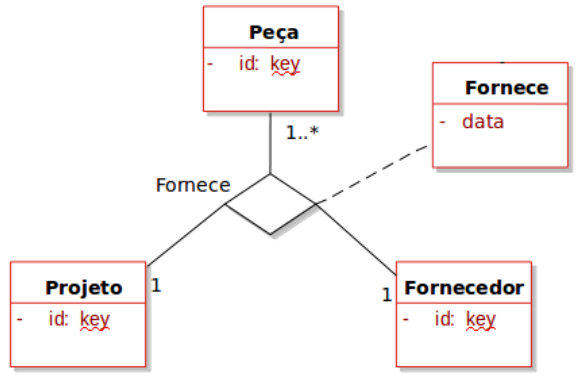


Figura 45 – Associação ternária

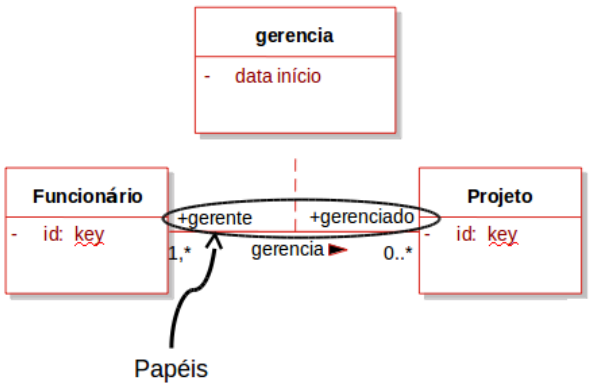


Figura 46 – Associação Binária

6.7 – 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)

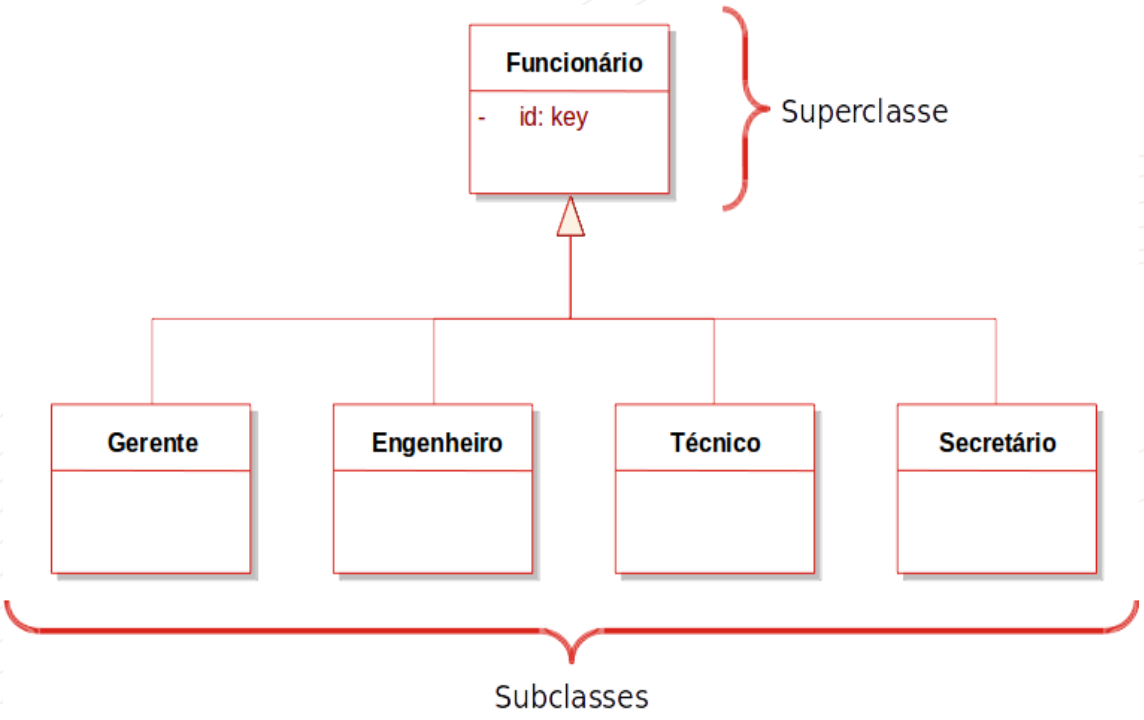


Figura 47 – Generalização / Especialização

6.7.1 – 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.

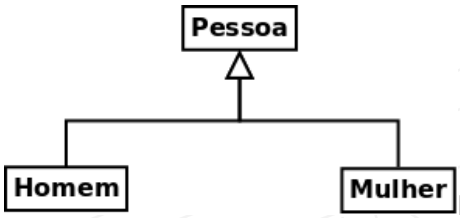


Figura 48 – Generalização Normal

6.7.2 – 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)

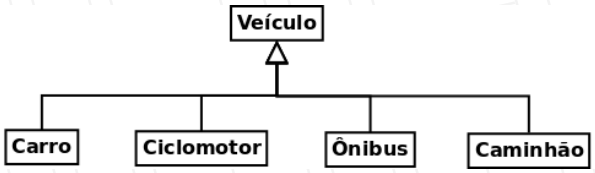


Figura 49 – Restrição disjuntiva

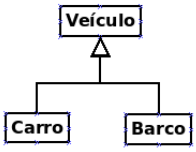


Figura 50 – Restrição de sobreposição


Figura 51 – Amphicar como carro


Figura 52 – Amphicar como barco

6.7.3 – 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 pertencer 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.

(Rangel et al, 2014)

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.

6.8 – 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.

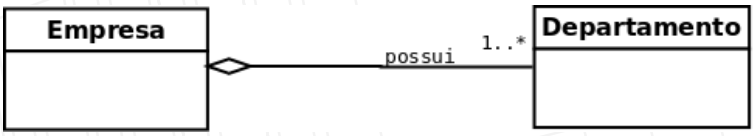


Figura 53 – Agregação de Departamentos de um Empresa

6.9 – 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".

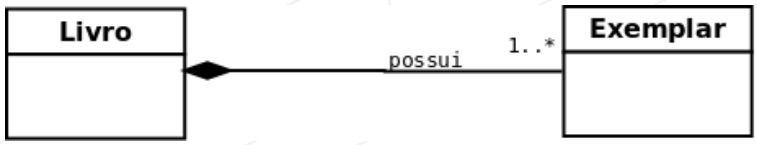


Figura 54 – Composição de Exemplares de um Livro

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

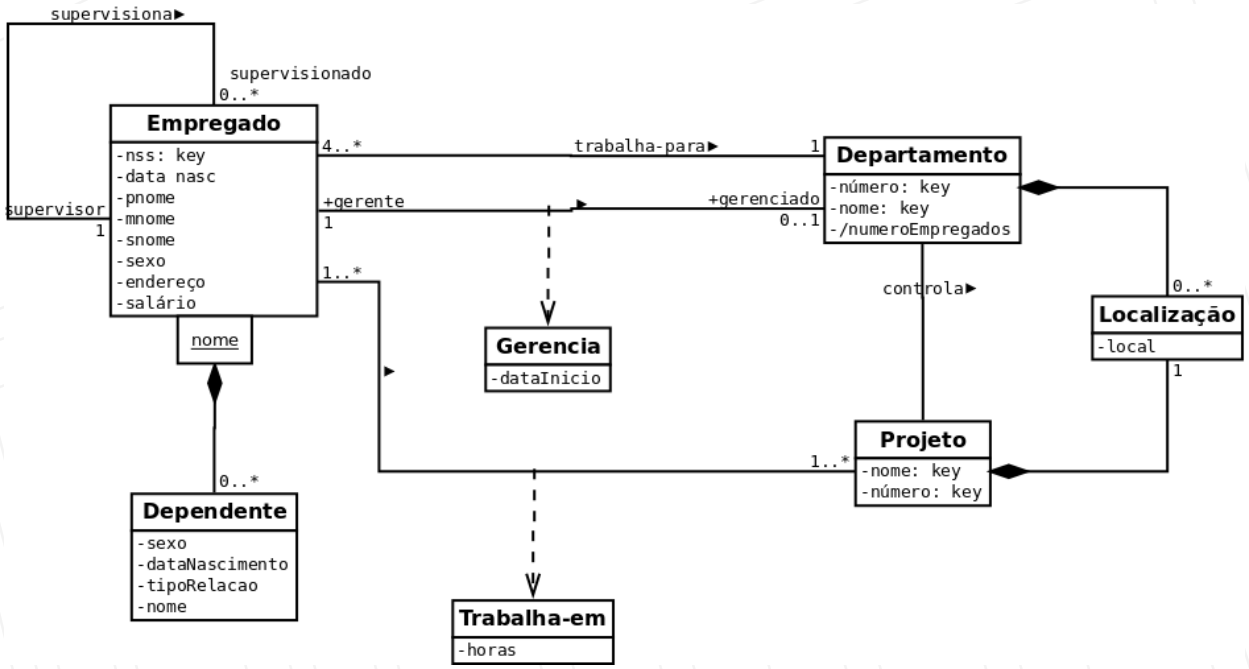


Figura 55 – Diagrama de Classes para o esquema EMPRESA. (ELMASRI; NAVATHE, 2011)

Capítulo 7 – Modelo Relacional

> O modelo relacional é um modelo de dados representativo (ou de implementação), adequado a ser o modelo subjacente de um Sistema Gerenciador de Banco de Dados (SGBD), que se baseia no princípio de que todos os dados estão armazenados em tabelas (ou, matematicamente falando, relações). Toda sua definição é teórica e baseada na lógica de predicados e na teoria dos conjuntos.
>
> (WIKIPEDIA, 2004)

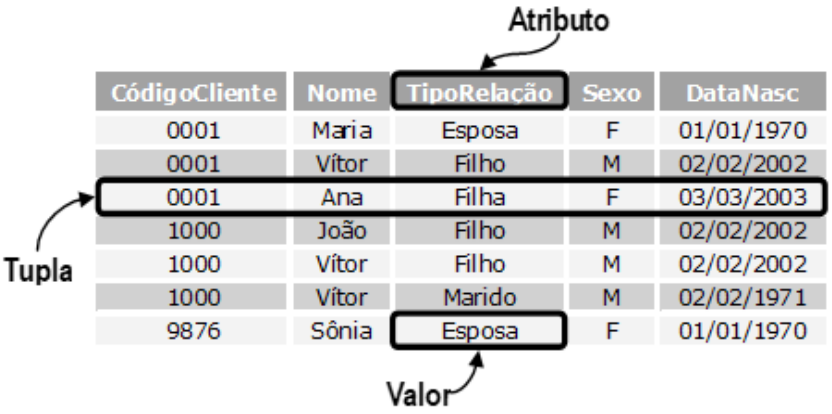
O conceito do modelo relacional foi criado por Tedd Codd, em 1970, sendo este baseado em lógica e na teoria dos conjuntos. Neste modelo, os dados são armazenados em relações que também armazenam os relacionamentos entre estas relações.

Os primeiros SGBDs relacionais comerciais surgiram no início da década de 1980, através da IBM e da Oracle. Dentre os SGBDs relacionais mais populares atualmente podemos destacar o Oracle, o DB2 da IBM e o SQL Server da Microsoft, dentre os que são comerciais e o Postgree, o MariaDB e o Firebird dentre aqueles que são livres ou de código aberto. (ELMASRI; NAVATHE, 2011)

7.1 – Conceito do Modelo Relacional

O banco de dados no modelo relacional é representado por meio de relações. De maneira informal, cada relação é semelhante a uma tabela ou um arquivo de dados. Quando uma relação é considerada uma tabela de valores, cada linha nela representa uma coleção de valores relacionados, representando assim, cada linha uma entidade do mundo real. Os nomes utilizados na tabela e nas colunas ajudam a interpretar o significado de cada uma das linhas. (ELMASRI; NAVATHE, 2011)

Tupla é o nome formal dado a cada linha da relação enquanto o cabeçalho das colunas recebe o nome de atributo. Cada tipo de dado que descreve os tipos de valores que uma coluna pode receber é representado por um domínio de valores possíveis, como pode ser observado na figura 56. (ELMASRI; NAVATHE, 2011)



CódigoCliente	Nome	TipoRelação	Sexo	DataNasc
0001	Maria	Esposa	F	01/01/1970
0001	Vítor	Filho	M	02/02/2002
0001	Ana	Filha	F	03/03/2003
1000	João	Filho	M	02/02/2002
1000	Vítor	Filho	M	02/02/2002
1000	Vítor	Marido	M	02/02/1971
9876	Sônia	Esposa	F	01/01/1970

Figura 56 – relação DEPENDENTE.

No modelo relacional os atributos devem ser atômicos, ou seja, indivisíveis e cada atributo pode ter somente um domínio. O conjunto de atributos de uma relação recebe o nome de relação esquema. O grau de uma relação é determinado pelo número de atributos que esta relação possua. "Um domínio D é um conjunto de valores atômicos. Com atômico queremos dizer que cada valor no domínio é indivisível em se tratando do modelo relacional formal". (ELMASRI; NAVATHE, 2011, p. 39). Um exemplo de uma relação e dos domínios possíveis nela pode ser visto na figura 57.

DEPENDENTE (CódigoCliente, Nome, TipoRelação, Sexo, DataNasc)

- É a **relação esquema**.
- **DEPEDENTE** é o nome da relação.
- O **Grau da Relação** é 5.
- Os **Domínios** dos Atributos são:
  - dom(CódigoCliente) = 4 dígitos que representam o Código do Cliente.
  - dom(Nome) = Caracteres que representam nomes dos dependentes.
  - dom(TipoRelação) = Tipo da Relação (filho, esposa, pai, mãe e outras) do dependente em relação do seu cliente .
  - dom(Sexo) = Caractere: (M: Masculino, F: Feminino) do dependente.
  - dom(DataNasc) = Datas de Nascimento do dependente.

Figura 57 – Detalhes da relação DEPENDENTE

7.2 – Notação relacional

Um esquema relacional R, indicado por $R(A_1, A_2, ..., A_n)$, é composto de um nome de relação R e por uma lista de atributos $A_1, A_2, ..., A_n$. Cada atributo A_i é o nome de um papel desempenhado por algum domínio D no esquema de relação R. D é chamado de domínio

de Ai, e indicado por dom(Ai). Um esquema de relação é usado para descrever uma relação; R é o nome dessa relação. O grau (ou aridade) é o número de atributos n desse esquema de relação. (ELMASRI; NAVATHE, 2011, p. 40)

Um resumo da notação relacional pode ser visto na figura 58 e um exemplo da notação pode ser visto na figura 59.

- A relação esquema R de grau n:
  - $R(A_1, A_2, ..., A_n)$.
- A tupla t em uma relação r(R) :
  - $t = <v_1, v_2, ..., v_n>$,

  v_i é o valor do atributos A_i.
- $t[A_i]$ indica o valor v_i em t para o atributo A_i.
- $t[A_u, A_w, ..., A_z]$ indica o conjunto de valores $<v_u, v_w, ..., v_z>$ de t correspondentes aos atributos $A_u, A_w, ..., A_z$ de R.

Figura 58 – Notação Relacional

A figura apresenta a Relação DEPENDENTE:

- t = <0001, Ana, Filha, F, 03/03/2003> é uma tupla
- t[CódigoCliente] = 0001
- t[Nome, Sexo] = <Ana, F>

Figura 59 – Exemplo de Notação Relacional

7.3 – Restrições em modelo relacional e esquemas de bancos de dados relacionais

Uma restrição é uma condição restritiva ou imposição de limite. Pode ser entendida também como uma limitação ou condição imposta ao livre exercício de um direito ou de uma atividade. (HOUAISS; VILLAR, 2009)

As restrições baseadas em esquema incluem restrições de domínio, restrições de chave, restrições sobre NULLs, restrições de integridade de

entidade e restrições de integridade referencial. (ELMASRI; NAVATHE, 2011)

7.3.1 – Restrições de domínio

As restrições de domínio especificam que, dentro de cada tupla, o valor de cada atributo deve ter um valor indivisível, ou seja, o projetista deve definir quais são os domínios possíveis em cada atributo, como, por exemplo, aceitar F ou M, ambos maiúsculos para o atributo SEXO. (ELMASRI; NAVATHE, 2011)

7.3.2 – Restrições de domínio e Chaves

No modelo relacional, uma relação é um conjunto de tuplas, que por definição são todos distintos, ou seja, cada tupla é diferente das demais e que não podem ter a mesma combinação de valores para todos os seus atributos. Além disso, existem outros subconjuntos que podem identificar de forma única uma tupla da tabela e estes subconjuntos são chamados de superchave. Assim, uma superchave de uma determinada tupla deve ser diferente da superchave em outra tupla, representados por t1[SC] ≠ t2[SC]. Cada relação tem pelo menos uma superchave padrão: o conjunto de todos os atributos dela. Como uma superchave pode conter atributos redundantes, é mais útil o conceito de chave. Uma chave é uma superchave mínima, ou seja, uma superchave que não pode mais ter nenhum de seus atributos removidos e ainda assim identificar de forma única uma tupla da relação, como pode ser observado na figura 60.(ELMASRI; NAVATHE, 2011)

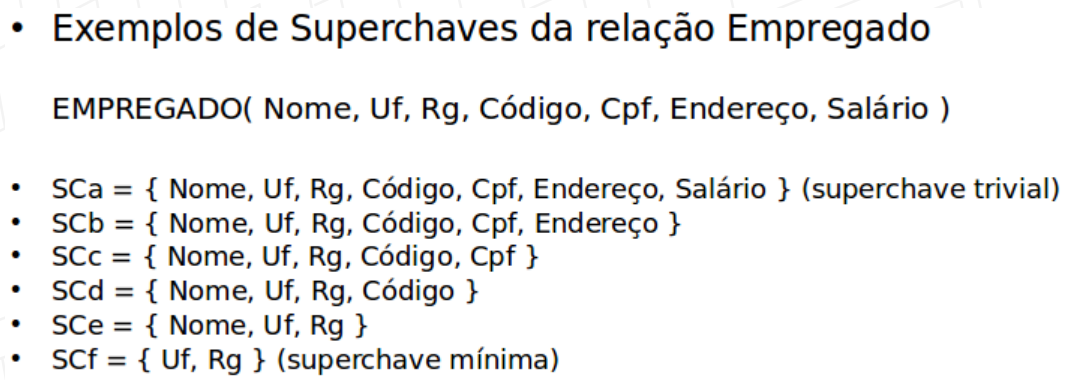


Figura 60 – Exemplos de Superchaves de uma relação

Como pode ser observado na figura 60, a SCf (UF e RG) são uma superchave mínima pois nenhum dos dois atributos pode ser removido sem que se perca a capacidade de identificar de forma única uma tupla da relação.

Assim, uma chave satisfaz a duas propriedades. Na primeira, duas tuplas não podem ter valores idênticos para os atributos de uma chave, propriedade que também se aplica às superchaves e, na segunda, ela é uma superchave mínima, da qual não é mais possível remover nenhum dos seus

atributos. Quando uma relação tem mais de uma chave ela é denominada como chave candidata. Mas, candidata a que? Candidata a chave primária! Sim, do rol de chaves candidatas escolhe-se uma delas para chave primária da relação. Embora não seja uma regra, um bom critério é o de escolher a menor chave, ou seja, a que utiliza o menor número de bytes ou caracteres, uma vez que este(s) atributo(s) será(ão) utilizado(s) nos relacionamentos com outras relações, como pode ser observado nas figuras 61 e 62. (ELMASRI; NAVATHE, 2011)

Como exemplo de chave candidata, pode-se usar a relação EMPREGADO(Nome, Uf, Rg, Código, Cpf, Endereço, Salário), conforme listado abaixo:

- CC1 = { Uf, Rg } (Superchave mínima, Chave e Chave-Candidata)
- CC2 = { Código } (Superchave mínima, Chave e Chave-Candidata)
- CC3 = { Cpf } (Superchave mínima, Chave e Chave-Candidata)

Estas chaves são candidatas à chave primária já que todas elas identificam uma tupla da relação de forma única. Conforme citado anteriormente, um bom critério é escolher a menor chave, uma vez que a chave primária pode ser chave estrangeira em outra relação e assim utiliza-se menos espaço no banco de dados e, desta forma, por este critério, chave escolhida seria a CC2 (Código) que utiliza no máximo 4 bytes por ser um número inteiro.

- Superchave:
  - Subconjunto de atributos de uma relação cujos valores são distintos:
  - t1[SC] ≠ t2[SC]
- Chave:
  - É uma Superchave mínima
- Chave-Candidata:
  - Chaves de uma relação
- Chave-Primária:
  - Uma das Chaves escolhidas entre as Chaves-Candidatas de uma relação.

Figura 61 – Superchave e Chaves

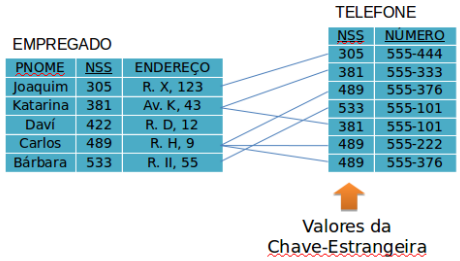


Figura 62 – Relacionamento e Chave Estrangeira

Na figura 62 pode-se observar o atributo NSS que é a chave primária da relação EMPREGADO. Como ela é a chave primária de outra relação, na relação TELEFONE este atributo passa a ser chamado de Chave Estrangeira.

7.3.3 – Restrições de integridade

Restrição de Integridade são regras que restrigem os valores que podem ser armazenados em relações. Um SGBDR (Sistema Gerenciador de Banco de Dados Relacional) deve garantir:

- Restrição de Chave: os valores das chaves-candidatas devem ser únicos em todas as tuplas de uma relação.
- Restrição de Entidade: chaves-primárias não podem ter valores nulos.
  - Observando o exemplo da figura 62, caso seja inserida uma tupla na relação EMPREGADO com o valor do atributo NSS o SGBDR deve recusar a inserção e emitir mensagem de erro.
- Restrição de Integridade Referencial: Usada para manter a consistência entre tuplas. Estabelece que um valor de atributo, que faz referência a uma outra tupla, deve-se referir a uma tupla existente.
  - Como pode ser visto na figura 62, esta restrição impede que seja inserido uma tupla na relação TELEFONE com um valor para o atributo NSS que não esteja previamente cadastrado na relação EMPREGADO

(ELMASRI; NAVATHE, 2011)

Capítulo 8 – Mapeamento do Modelo Entidade-Relacionamento para o Modelo Relacional

Ao se projetar um banco de dados, normalmente se utiliza um modelo conceitual e, o modelo conceitual mais utilizado é o Modelo Entidade-Relacionamento. Entretanto, os SGBDR (Sistemas Gerenciadores de Bancos de Dados Relacionais) utilizam o modelo relacional, retratado no capítulo 7 para representar os dados no banco de dados e, assim, é necessário que se faça uma conversão de modelos, ou seja, que se "interprete" o DE-R (Diagrama Entidade-Relacionamento) para o modelo relacional.

Para simplificar este trabalho, Elmasri e Navathe (2011) propõe 7 passos ou regras para que se chegue ao modelo relacional, como pode ser observado nos próximos tópicos. Como exemplo, será utilizado um modelo para um banco de dados de um instituição de ensino que pode ser visto na figura 63.

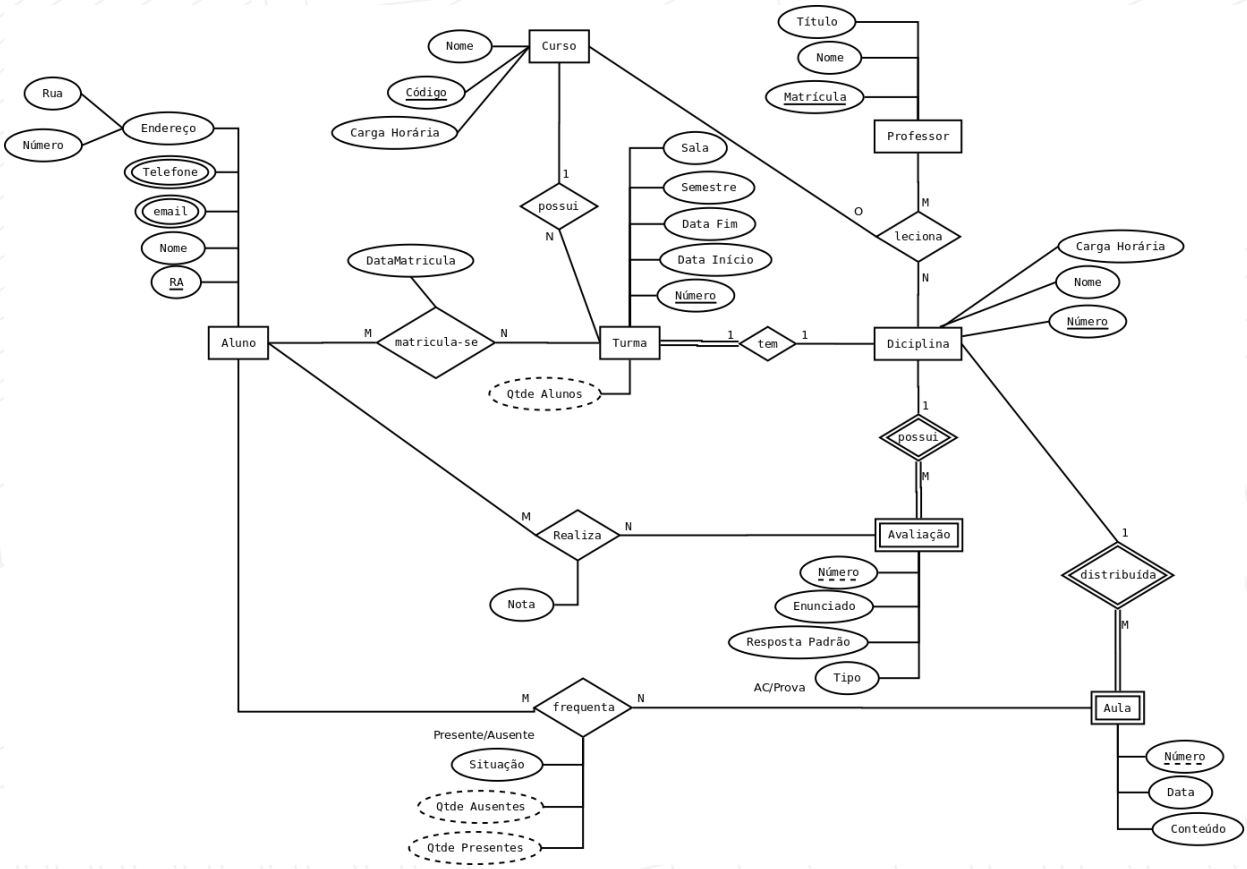


Figura 63 – DE-R para um banco de dados escolar

O produto da aplicação destas regras é o esquema do banco de dados e, para o DE-R da figura 63, será o esquema de banco de dados exibido na figura 64.

Aluno (RA, Nome, Rua, Número)
Curso (CodCurso, Nome, Carga_Horaria)
Professor (Matrícula, Nome, Título)
Disciplina (NúmeroDisciplina, Nome, Carga_Horária)
Turma (NúmeroTurma, Data_Inicio, Data_Fim, Semestre, Sala, NumeroDisciplina-CE, CodCurso-CE)

Avaliação (NumeroDisciplina-CE, NúmeroAvaliação-CE, Enunciado, RespostaPadrao, TipoAvaliação)
Aula (NumeroDisciplina-CE, NúmeroAula-CE, Data, Conteúdo)

Matriculase (NumeroTurma-CE, RA-CE, DataMatricula)

Realiza (NumeroDisciplina-CE, NumeroAvaliação-CE, RA-CE, Nota)

Frequenta (NumeroDisciplina-CE, NumeroAula-CE, RA-CE, Situação)

TelefoneCliente (CodigoCliente-CE, Telefone)

EmailCliente (CodigoCliente-CE, Email)

TelefoneFornecedor (CodigoFornecedor-CE, Telefone)

EmailFornecedor (CodigoFornecedor-CE, Email)

Leciona (CodigoCurso-CE, NumeroDisciplina-CE, Matricula-CE)

Figura 64 – Projeto Lógico do Banco de Dados

8.1 – Primeiro Passo

Para cada tipo de entidade normal E no DER, cria-se uma relação R que inclua todos os atributos simples de E. Inclua também os atributos simples dos atributos compostos. Escolha um dos atributos-chave de E como a chave-primária de R. Se a chave escolhida é composta, então o conjunto de atributos simples que o compõem formarão a chave-primária de R.

No DE-R da figura 63, os tipos de entidades normais são:

- Aluno
- Curso
- Turma
- Professor
- Disciplina

O resultado da aplicação do primeiro passo no DE-R da figura 63 é:

- Aluno (RA, Nome, Rua, Número)
- Curso (CodCurso, Nome, Carga_Horaria)
- Professor (Matricula, Nome, Título)
- Disciplina (NúmeroDisciplina, Nome, Carga_Horária)
- Turma (NúmeroTurma, Data_Inicio, Data_Fim, Semestre, Sala)

8.2 – Segundo Passo

Para cada tipo de entidade fraca W do DER com o tipo de relacionamento de identificação E, cria-se uma relação R e inclua todos os atributos simples, como também os atributos simples de atributos compostos de W como atributos de R.

Além disso, inclua como a chave-estrangeira de R a chave-primária da relação que corresponde ao tipo de entidade proprietário da identificação.

A chave-primária de R é a combinação da chave-primária do tipo de entidade proprietário da identificação e a chave-parcial do tipo de entidade fraca W.

No DE-R da figura 63, os tipos de entidades fracas são:

- Avaliação
- Aula

O resultado da aplicação do segundo passo no DE-R da figura 63 é:

- Avaliação (NúmeroDisciplina-CE, NúmeroAvaliação-CE, Enunciado, RespostaPadrao, TipoAvaliação)
- Aula (NúmeroDisciplina-CE, NúmeroAula-CE, Data, Conteúdo)

8.3 – Terceiro Passo

Para cada tipo de relacionamento binário 1:1, R, do DER, identifique as relações S e T que correspondem aos tipos de entidade que participam de R.

Deve-se escolher uma das relações, por exemplo S, e incluir-se como chave-estrangeira de S a chave-primária de T. Entretanto, é melhor escolher o tipo de entidade com participação total em R como sendo a relação S.

Inclui-se também todos os atributos simples e os atributos simples de atributos compostos do tipo de relacionamento 1:1, R, como atributos de S.

No DE-R da figura 63, os relacionamentos com cardinalidade 1:1 são:

- Tem

O resultado da aplicação do terceiro passo no DE-R da figura 63 é:

- Turma (NúmeroTurma, Data_Inicio, Data_Fim, Semestre, Sala, NumeroDisciplina-CE)

8.4 – Quarto Passo

Para cada tipo de relacionamento binário regular 1:N, excluindo-se o relacionamento do tipo de entidade proprietário com o tipo de entidade fraca; R, identificar a relação S que representa o tipo de entidade que participa do lado N de R.

Inclua como chave-estrangeira de S a chave-primária de T que representa o outro tipo de entidade que participa em R; isto porque cada entidade do lado 1 está relacionada a mais de uma entidade no lado N.

Inclui-se também quaisquer atributos simples, bem como atributos simples de atributos compostos do tipo de relacionamento 1:N, como atributos de S.

No DE-R da figura 63, os tipos de entidades vinculadas aos relacionamentos com cardinalidade 1:N são:

- Curso e Turma para o relacionamento possui.

O resultado da aplicação do quarto passo no DE-R da figura 63 é:

- Turma (NúmeroTurma, Data_Inicio, Data_Fim, Semestre, Sala, Numero-Disciplina-CE , CodCurso-CE)

8.5 – Quinto Passo

Para cada tipo de relacionamento binário M:N, R, crie uma nova relação S para representar R. Inclui-se como chave-estrangeira de S as chaves-primárias das relações que representam os tipos de entidade participantes; sua combinação irá formar a chave-primária de S.

Inclua-se também qualquer atributo simples do tipo de relacionamento M:N (ou atributos simples dos atributos compostos) como atributos de S. É importante frisar que não se pode representar um tipo de relacionamento M:N como uma simples chave-estrangeira em uma das relações participantes - como foi feito para os tipos de relacionamentos 1:1 e 1:N. Isso ocorre porque o MR não permite a representação de atributos multivalorados.

No DE-R da figura 63, os tipos de relacionamentos com cardinalidade M:N são:

- Matriculase.
- Realiza
- Frequenta

O resultado da aplicação do quinto passo no DE-R da figura 63 é:

- Matriculase (NumeroTurma-CE, RA-CE, DataMatricula)
- Realiza (NumeroDisciplina-CE, NumeroAvaliação-CE, RA-CE, Nota)
- Frequenta (NumeroDisciplina-CE, NumeroAula-CE, RA-CE, Situação)

8.6 – Sexto Passo

Para cada atributo A multivalorado, crie-se uma nova relação R que inclua o atributo A e a chave-primária, K, da relação que representa o tipo de entidade ou o tipo de relacionamento que tem A como atributo. A chave-primária de R é a combinação de A e K. Se o atributo multivalorado é composto inclua os atributos simples que o compõem.

No DE-R da figura 63, os atributos multivalorados, bem como os tipos de entidade onde este está vinculado são:

- Telefone e Email do tipo de entidade Cliente.
- Telefone e Email do tipo de entidade Fornecedor.

O resultado da aplicação do sexto passo no DE-R da figura 63 é:

- TelefoneCliente (CodigoCliente-CE, Telefone)
- EmailCliente (CodigoCliente-CE, Email)
- TelefoneFornecedor (CodigoFornecedor-CE, Telefone)
- EmailFornecedor (CodigoFornecedor-CE, Email)

8.7 – Sétimo Passo

Para cada tipo de relacionamento n-ário, R, n>2 (ou seja, um relacionamento com mais de 2 tipos de entidade), cria-se uma nova relação S para re-

presentar R. Nesta relação, devem ser incluídos como chave-estrangeira em S as chaves-primárias das relações que representam os tipos de entidades participantes. Além disso, inclua-se também qualquer atributo simples do tipo de relacionamento n-ário (ou atributos simples dos atributos compostos) como atributo de S.

A chave-primária de S é normalmente a combinação de todas as chaves -estrangeiras que referenciam as relações que representam os tipos de entidades participantes. Porém, se a restrição estrutural (min, max) de um dos tipos de entidades E que participa em R, tiver max=1, então a chave-primária de, S, pode ser a chave-estrangeira que referencia a relação E; isto porque cada entidade e em E irá participar em apenas uma instância em R e, portanto, pode identificar univocamente esta instância de relacionamento.

No DE-R da figura 63, há somente um tipo de relacionamento com n>2, um relacionamento ternário, que é:

- Leciona

O resultado da aplicação do sétimo passo no DE-R da figura 63 é:

- Leciona (CodigoCurso-CE, NumeroDisciplina-CE, Matricula-CE)

8.8 – Projeto lógico do banco de dados

Após a aplicação de cada um dos passos citados nos itens 8.1 até 8.7 gera o projeto lógico do banco de dados. Para o exemplo abordado na figura 63, o resultado após a aplicação dos 7 passos pode ser visto na figura 64 e aparece listado a seguir:

- Aluno (RA, Nome, Rua, Número)
- Curso (CodCurso, Nome, Carga_Horaria)
- Professor (Matricula, Nome, Título)
- Disciplina (NúmeroDisciplina, Nome, Carga_Horária)
- Turma (NúmeroTurma, Data_Inicio, Data_Fim, Semestre, Sala, NumeroDisciplina-CE , CodCurso-CE)
- Avaliação (NúmeroDisciplina-CE, NúmeroAvaliação-CE, Enunciado, RespostaPadrao, TipoAvaliação)
- Aula (NúmeroDisciplina-CE, NúmeroAula-CE, Data, Conteúdo)
- Matriculase (NumeroTurma-CE, RA-CE, DataMatricula)
- Realiza (NumeroDisciplina-CE, NumeroAvaliação-CE, RA-CE, Nota)
- Frequenta (NumeroDisciplina-CE, NumeroAula-CE, RA-CE, Situação)
- TelefoneCliente (CodigoCliente-CE, Telefone)
- EmailCliente (CodigoCliente-CE, Email)
- TelefoneFornecedor (CodigoFornecedor-CE, Telefone)
- EmailFornecedor (CodigoFornecedor-CE, Email)
- Leciona (CodigoCurso-CE, NumeroDisciplina-CE, Matricula-CE)

8.9 – Mapeando construções do modelo EER para relações

As estruturas do modelo EER por serem específicas tem também passos específicos a serem seguidos para se fazer o mapeamento para o modelo relacional. Para o mapeamento de

especializações e generalizações existem várias opções. (ELMASRI; NAVATHE, 2011)

Para esta parte específica de modelagem, será utilizado o DE-R (Diagrama Entidade-Relacionamento) mostrado na figura 65.

8.9.1 – Passo 8

Converta cada especialização com n subclasses {S_1, S_2, … S_n} e a superclasse C, em que os atributos de C são {ch, a_1, …, a_n} e ch é a chave primária para os esquemas da relação usando a seguinte opção, que funciona para qualquer tipos de especialização:

- Opção 8A: Múltiplas relações – superclasse e subclasses
  - Cria-se uma relação L para C com atributos Atrs(L) = {ch = a_1, …, a_n} e ChP(L_1) = ch. Cria-se uma relação L_1 para a subclasse S_i, 1 <= i <= m, com os atributos Atrs(L) = {ch} U {atributos de S_i} e ChP(L_1) = ch. Essa opção funciona para qualquer especialização, parcial ou total; disjunta ou sobreposta.
    - No caso do DE-R da figura 65, o resultado para a especialização será:
      - Livro (CodigoLiteratura-CE, ISBN, AnoPublicação)
      - AutorLivro (CodigoLiteratura-CE, ISBN-CE, Autor)
      - Artigo (CodigoLiteratura-CE, CodigoArtigo, PagInicial, PagFinal)
      - TCC (CodigoLiteratura-CE, NivelTCC, TipoTCC, AnoPublicacao, CodigoInstituicao-CE)

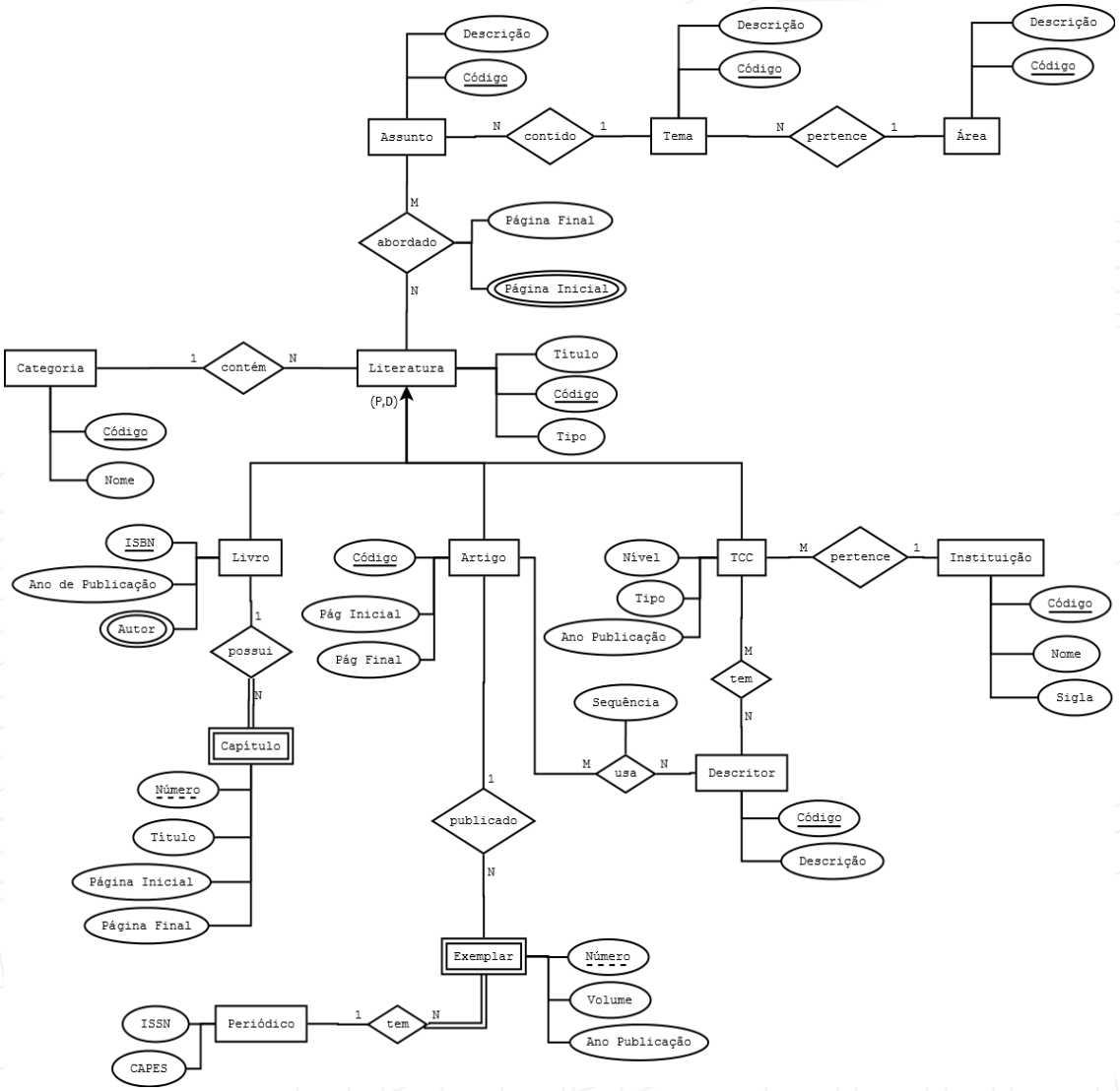


Figura 65 – DE-R com Especialização/Generalização

Capítulo 9 – Normalização

"Cada esquema de relação consiste em uma série de atributos, e o esquema de banco de dados relacional consiste em uma série de esquemas de relação". (ELMASRI; NAVATHE, 2011, p. 337)

> Normalização de banco de dados é um conjunto de regras que visa, principalmente, a organização de um projeto de banco de dados para reduzir a redundância de dados, aumentar a integridade de dados e o desempenho. Para normalizar o banco de dados, deve-se examinar as colunas (atributos) de uma entidade e as relações entre entidades (tabelas), com o objetivo de se evitar anomalias observadas na inclusão, exclusão e alteração de registros.
>
> (WIKIPEDIA, 2018)

"Obtido o esquema relacional correspondente ao documento, passa-se ao processo de normalização. Este processo baseia-se no conceito de forma normal". (HOUSER, 2009)

Existem 6 formas normais, 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

9.1 – Abordagens de Projeto de Banco de Dados

Um projeto pode ser abordado de modo Top-Down ou Botton-Up.

O método Top-Down inicia pelo agrupamento dos atributos obtidos a partir do projeto conceitual de mapeamento. Isso é chamado de projeto por análise. Já o método Bottom-UP considera os relacionamentos entre atributos, para construir as relações. Isso é chamado projeto pela síntese. Neste material será abordado o método Top-Down.

9.2 – Anomalias

Um projeto não normalizado pode proporcionar a ocorrências de anomalias. Segundo Ramakrishnan e Gehrke (2008, p. 507), "O armazenamento das mesmas informações de forma redundante, isto é, em mais de um lugar dento de um banco de dados, pode acarretar vários problemas". Estas anomalias podem ser da ordem de Inserção, de Remoção ou de Alteração. Um exemplo de um banco de dados não normalizado pode ser visto na figura 66.

CódForn	Nome	Status	Cidade	CódPeça	Preço	Qtde	Valor
F1	Fornecedor 1	20	Londres	P1	R$ 10,00	300	R$ 3.000,00
F1	Fornecedor 1	20	Londres	P2	R$ 20,00	200	R$ 4.000,00
F1	Fornecedor 1	20	Londres	P3	R$ 15,00	400	R$ 6.000,00
F1	Fornecedor 1	20	Londres	P4	R$ 25,00	200	R$ 5.000,00
F1	Fornecedor 1	20	Londres	P5	R$ 12,00	100	R$ 1.200,00
F1	Fornecedor 1	20	Londres	P6	R$ 5,00	100	R$ 500,00
F2	Fornecedor 2	10	Paris	P1	R$ 10,00	300	R$ 3.000,00
F2	Fornecedor 2	10	Paris	P2	R$ 20,00	400	R$ 8.000,00
F3	Fornecedor 3	10	Paris	P2	R$ 20,00	200	R$ 4.000,00
F4	Fornecedor 4	20	Londres	P2	R$ 20,00	200	R$ 4.000,00
F4	Fornecedor 4	20	Londres	P4	R$ 25,00	300	R$ 7.500,00
F4	Fornecedor 4	20	Londres	P5	R$ 12,00	400	R$ 4.800,00

Figura 66 – Banco de Dados de Pedidos de Compras

9.2.1 – Anomalias de Atualização

São os problemas causados durante uma inserção, atualização ou exclusão em uma base de dados não normalizada. Baseando-se no exemplo mostrado na figura 66, pode-se perceber que os dados do fornecedor, do fornecimento e da localização do fornecedor estão todos "misturados" na mesma relação. Pode-se ver ainda que não existem tabelas auxiliares, uma vez que todas as informações estão na mesma relação. Outro fator importante é que nenhum campo pode ser nulo ou faltarão informações. Por fim, a chave primária é composta pelos campos CodForn e CodPeça

Neste tipo de anomalia, considerando-se o exemplo da figura 66, nenhuma localização de fornecedor pode ser inserida até que este fornecedor forneça, pelo menos, uma peça, o que inviabiliza a participação de um fornecedor em qualquer compra, já que ele não estará previamente cadastrado. Ainda há o problema de que sempre que uma peça for fornecida será preciso repetir os dados do fornecedor. Neste caso, existe a possibilidade de que o usuário erre ou deliberadamente modifique os textos a serem inseridos, como, por exemplo, o nome da cidade.

Supondo-se a existência de um fornecedor que venha a fornecer 100 peças para a empresa em questão e que este fornecedor esteja situado na cidade de Santo Antônio do Aracanguá / SP. Pode-se encontrar, após a inserção de 100 registro variantes como Sto Antônio Aracanguá até S. A. Aracanguá, fazendo com exista perda de informação. (RAMAKRISHNAN; GEHRKE, 2008)

9.2.2 – Anomalias de Alteração

Analisando-se o exemplo da figura 66, o valor do campo CIDADE aparece repetido várias vezes na tabela. Isso aumenta a probabilidade de erros. Se o fornecedor 1 mudar para Amsterdã, será necessária a atualização de várias tuplas!

Se cuidados não forem tomados, pode haver uma ocorrência do Fornecedor 1 em Londres e outra em Amsterdã! O mesmo problema ocorre com os atributos NOME e STATUS. Referindo-se ao exemplo de inserção citado no item 9.2.1.1, caso o fornecedor do exemplo mudar-se do município de Santo Antônio do Aracanguá / SP para o município de São João do Pau d'Alho no mesmo estado, propiciará a possibilidade erro ou até mesmo a não atualização de todos os registros.

O atributo valor apresenta outra anomalia. Ele é calculado pela multiplicação de PREÇO e QTDE. Isso acarreta que, se o preço unitário for modificado, será preciso atualizar também o atributo VALOR, um passo a mais que pode ser esquecido! (RAMAKRISHNAN; GEHRKE, 2008)

9.2.3 – Anomalias de Exclusão

Com base no banco de dados de exemplo da figura 66, se todos os registros de um fornecedor precisarem ser apagados, não apenas serão excluídas suas relações com as peças, mas também, a informação de que o fornecedor está localizado em uma determinada cidade, acarretando portanto, perda de informação. (RAMAKRISHNAN; GEHRKE, 2008)

9.3 – Dependências Funcionais

"Uma dependência funcional é uma restrição entre dois conjuntos de atributos do banco de dados". (ELMASRI; NAVATHE, 2011, p. 346)

Dependências funcionais (DFs) são usadas para medir formalmente a qualidade do projeto relacional. As Dependências Funcionais e as chaves são usadas para definir formas normais de relações. As DFs são restrições que são derivadas do significado dos atributos e do seus inter-relacionamentos.

De acordo com Elmasri e Navathe (2011, p. 346) Uma dependência funcional:

> É indicada por X→Y, entre dois conjuntos de atributos X e Y que são subconjuntos de R, especifica uma restrição sobre possíveis tuplas que podem formar um estado de relação r de R.

Sempre que um atributo X identifica um atributo Y, dizemos que entre eles há uma dependência funcional, de tal forma que X é o determinante e Y é o dependente. A representação é X ◦ Y. (ELMASRI; NAVATHE, 2011)

Desta forma, se X◦Y, e se as duas tuplas tiverem o mesmo valor para X, elas devem ter o mesmo valor para Y. Ou seja, se X◦Y então, para quaisquer tuplas t1 e t2 de r(R), assim, se t1[X] = t2[X], então t1[Y] = t2[Y].

Assim, pode-se constatar que se K é uma chave de R, então K determina funcionalmente todos os atributos de R, isso porque, nunca teremos duas tuplas distintas com t1[K]=t2[K].

É importante frisar que X◦Y especifica uma restrição sobre todas as instâncias de R. As DFs são derivadas das restrições do mundo real e não de uma extensão específica da relação R.

9.3.1 – Exemplos de Dependências Funcionais

Um exemplo de dependência funcional pode ser o número do seguro social que determina o nome do empregado NSS◦ENOME. Um outro pode ser o número do projeto que determina o nome do projeto e a sua localização, PNUMERO◦{ PNOME, PLOCALIZACAO }. Por outro lado, O NSS de empregado e o número do projeto (PNUMERO) determinam as horas semanais (HORAS) que o empregado trabalha no projeto { NSS, PNUMERO }◦HORAS .

Ainda, pode-se citar mais um exemplo, que é o relacionamento entre cidade, estado e país. Neste caso, pode afirmar que CIDADE◦ESTADO e que ESTADO◦PAIS, onde tem-se que:

- Estado é funcionalmente dependente de cidade

- País é funcionalmente dependente de estado.
- Resumindo, cidade determina estado que determina país.

Trivialidade

A dependência funcional trivial indica que um determinante com mais de um atributo pode determinar seus próprios membros quando isolados.

Exemplo:

{banco, agência} ◦ banco

(DF trivial, pois banco é parte do determinante)

{banco, agência} ◦ agência

(DF trivial, pois agência é parte do determinante)

Quando um determinante indica outro atributo qualquer, temos uma dependência funcional não trivial.

Exemplo:

{banco, agência} ◦ cidade

(DF não trivial, pois cidade não faz parte do determinante)

(RAMAKRISHNAN; GEHRKE, 2008)

Dependência funcional irredutível à esquerda

Diz-se que o lado esquerdo de uma dependência funcional é irredutível quando o determinante está em sua forma mínima, que é quando não é mais possível reduzir a quantidade de atributos determinantes sem perder a dependência funcional.

Exemplo:

{cidade, estado} ◦ país

DF não está na forma irredutível à esquerda

estado ◦ país

Obs: Não significa que deverá sempre haver um único atributo à esquerda!

(RAMAKRISHNAN; GEHRKE, 2008)

9.3.2 – Regras para encontrar Dependências Funcionais

Existem algumas regras para que se encontre as Dependências Funcionais em um esquema de banco de dados relacional.

Regras de inferência de Armstrong:

RI1. (Reflexiva) Se Y ◦ X (é subconjunto de), então X◦ Y

(Isso também é válido quando X=Y)

RI2. (Aumentativa) Se X◦ Y, então XZ◦ YZ

(Notação: XZ significa X◦ U◦ Z)

RI3. (Transitiva) Se X◦ Y e Y◦ Z, então X◦ Z

RI1, RI2 e RI3 formam um conjunto completo de regras de inferência.

Regra de Separação

A◦ BC então A◦ B e A◦ C

Exemplo:

CPF◦ nome, endereço então CPF◦ nome e CPF◦ endereço

Leia-se o exemplo acima da seguinte maneira:

Se com um número de CPF encontra-se o nome e o endereço de uma pessoa, então com este mesmo número é possível encontrar apenas o nome e com este mesmo número pode-se encontrar apenas o endereço.

Regra de Acumulação (Aditiva)

A◦ B então AC◦ B

Exemplo:

CPF◦ Endereço então CPF, Idade◦ endereço

Leia-se o exemplo acima da seguinte maneira:

Se com um número de CPF encontra-se o endereço de uma pessoa, então com este mesmo número mais a idade da pessoa pode-se encontrar o endereço também.

Regra de Transitividade

A◦ B e B◦ C então A◦ C

Exemplo:

CPF◦ código-cidade e código-cidade◦ nome-cidade então CPF◦ nome-cidade.

Leia-se o exemplo acima da seguinte maneira:

Se com um número de CPF encontra-se o código da cidade de uma pessoa, e com o código da cidade é possível encontrar o nome da cidade, então com o número do CPF pode-se encontrar o nome da cidade.

Um outro exemplo pode ser:

cidade ◦ estado

estado ◦ país

cidade ◦ país (cidade determina país de forma transitiva)

Regra de Pseudo-Transitividade

A◦ B e BC◦ D então AC◦ D

Exemplo:

CPF◦ código-funcionário e código-funcionário, mês◦ salário-funcionário então CPF, mês◦ salário-funcionário

Leia-se o exemplo acima da seguinte maneira:

Se com um número de CPF encontra-se o código do funcionário, e com o código do funcionário mais um certo mês é possível encontrar o salário que ele recebeu naquele mês, então com o número do CPF mais um certo mês pode-se encontrar o salário que ele recebeu naquele mês.

9.4 – Formas Normais

> Intuitivamente, a redundância surge quando um esquema relacional impõe uma associação entre atributos que não é natural. As dependências funcionais (e, quanto a isso, outras relações) podem ser usadas para identificar tais situações e sugerir refinamentos para o esquema.
>
> A ideia básica é que muitos problemas provenientes da redundância podem ser resolvidos substituindo-se uma relação por uma coleção de relações "menores".
>
> Uma decomposição de um esquema de relação R consiste na substituição do esquema de relação por dois (ou mais) esquemas de relação, cada um contendo um subconjunto dos atributos de R e, juntos, incluindo todos os atributos presentes em R.
>
> (RAMAKRISHNAN; GEHRKE, 2008, p. 510)

A decomposição de um esquema de relação pode criar mais problemas do que resolvê-los e por isso necessita de cuidados. Desta forma, duas perguntas devem ser feitas repetidamente:

- É preciso decompor uma relação?
  - Várias formas normais foram proposta para relações e se um esquema de relação esta em uma delas sabe-se que alguns problemas podem surgir.
- Quais problemas (se houver) determinada decomposição causa
  - Duas são as propriedades principais, a propriedade da junção sem perda nos permite recuperar qualquer instância da relação decomposta e a propriedade da preservação da dependência que permite impor qualquer restrição na relação original simplesmente impondo algumas restrições em cada uma das relações menores.

Do ponto de vista do desempenho, consultas na relação original podem exigir que se juntem as relações decompostas.

(RAMAKRISHNAN; GEHRKE, 2008)

Veja o exemplo mostrado na figura 67, com a relação FUNCIONÁRIO.

Funcionário	Sexo	Salário
Carlos	M	R$ 5.000,00
Marcos	M	R$ 4.500,00

Figura 67 – Relação de Funcionário

É preciso que se façam decomposições sem perdas. Deste modo, a decomposição deve ser reversível para que nenhuma informação seja perdida. Os conceitos de decomposição sem perdas e dependência funcional são conceitos intimamente relacionados. A figura 68 mostra a decomposição da relação da figura 67 nos exemplos A e B, onde o modelo A mostra a decomposição sem perdas da relação Funcionário, uma vez que no exemplo B não há mais como de obter a relação original.

A

Funcionário	Sexo
Carlos	M
Marcos	M

Funcionário	Salário
Carlos	R$ 5.000,00
Marcos	R$ 4.500,00

B

Funcionário	Sexo
Carlos	M
Marcos	M

Sexo	Salário
M	R$ 5.000,00
M	R$ 4.500,00

Figura 68 – Exemplos de Decomposição

(RAMAKRISHNAN; GEHRKE, 2008)

9.4.1 – Primeira Forma Normal – 1FN

Proíbe que relações tenham atributos compostos, atributos multivalorados ou relações aninhadas, ou seja, permite apenas atributos que sejam atômicos. Considerado como parte da definição de relação.

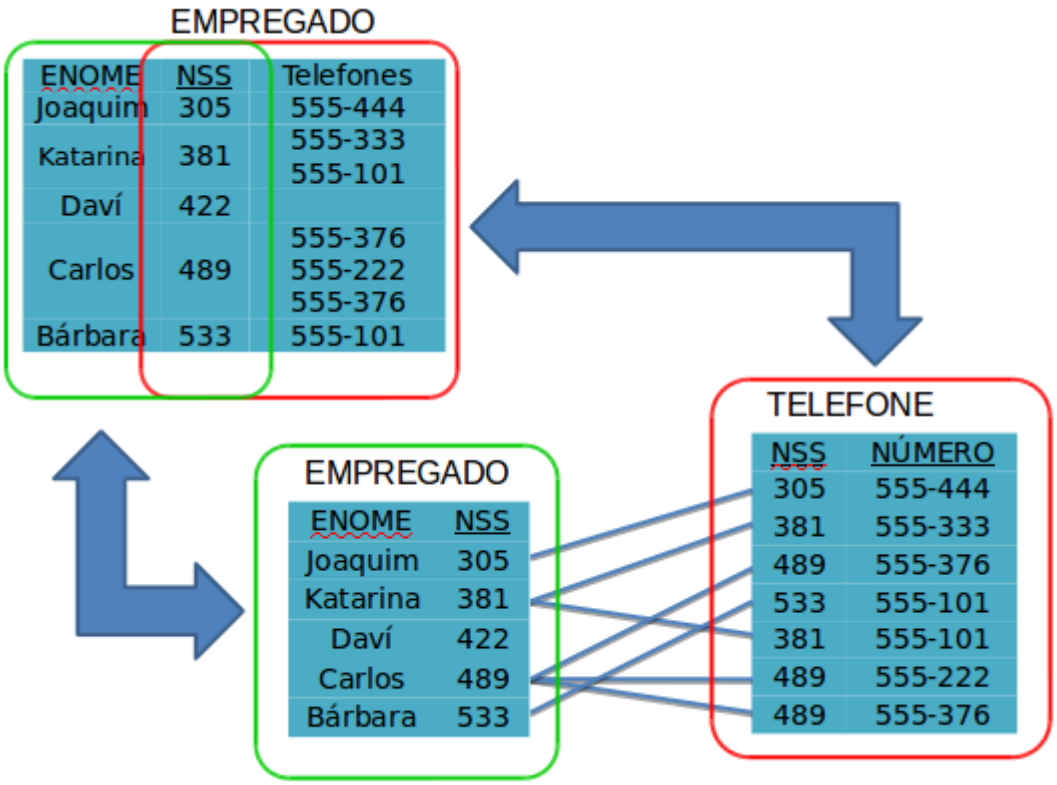


Figura 69 – Normalização para a Primeira Forma Normal

9.4.2 – Segunda Forma Normal – 2FN

Para entender a 2FN precisamos entender os conceitos de Dependência Funcional, Chave-primária, Atributo Não-Primo e Dependência funcional total.

Uma DF, Y◦ Z, onde a remoção de qualquer atributo de Y invalida a DF. Exemplos:

{ NSS, PNUMERO } ◦ HORAS é dependente totalmente de

{ NSS, PNUMERO }, uma vez que

NSS não determina HORAS e nem PNUMERO determina HORAS

{ NSS, PNUMERO } ◦ ENOME não é dependente totalmente de

{ NSS, PNUMERO }; ENOME é dependente parcialmente de

{ NSS, PNUMERO }, pois NSS ◦ ENOME

Uma relação esquema R está na 2FN se estiver na 1FN e todos os atributos não-primos A de R forem totalmente dependentes da chave-primária. A relação esquema R pode ser decomposto em relações que estejam na 2 FN através do processo de normalização.

A Segunda Forma Normal visa a diminuição da redundância e o desagrupamento de informações. Na 2FN, uma tabela representa um quantidade menor de entidades.

Uma tabela está na 2FN se:

- Estiver na 1FN;
- Todo atributo não-chave for determinado por todos os campos da chave primária. É preciso eliminar as dependências funcionais parciais;

Para fazer a normalização para segunda forma normal, deve-se mover os campos não enquadrados na 2FN para uma nova relação, que também deverá ser normalizada.

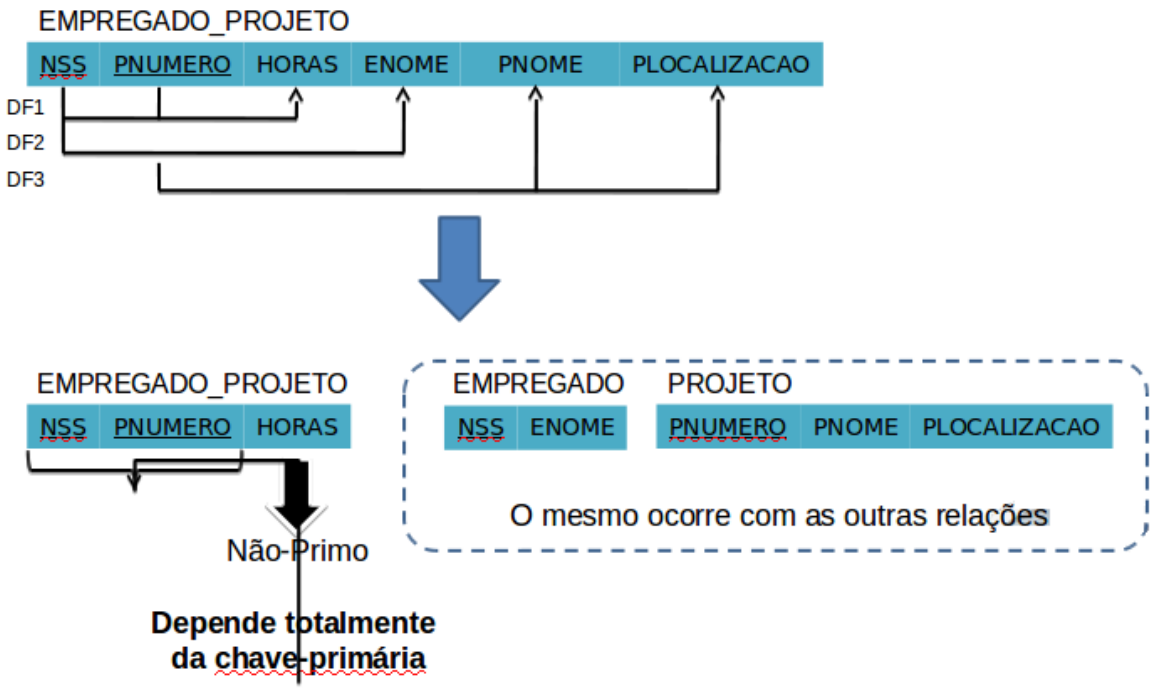


Figura 70 – Normalização para 2FN

9.4.3 – Terceira Forma Normal – 3FN

Para entender a 3FN é preciso entender:

- A 2FN – Segunda Forma Normal
- Atributo Não-Primo
- Dependência funcional transitiva
  - Se X◦ Y e Y◦ Z então X◦ Z

Desta forma, uma relação esquema R está na 3FN se ela estiver na 2FN e nenhum atributo não-primo, A, for transitivamente dependente da chave-primária. O esquema de relação R pode ser decomposto em relações que estejam na 3FN via o

processo de normalização.

- Nota:
  - Em X◦ Y e Y◦ Z, sendo X a chave-primária, pode ser considerado um problema se, e somente se, Y não for uma chave-candidata. Quando Y é uma chave-candidata, não existe problema com a dependência transitiva

Por exemplo, considerando o esquema de relação EMP (NSS, Emp#, Salario). Aqui, NSS ◦ Emp# ◦ Salario e Emp# é uma chave-candidata

A 3Fn – Terceira Forma Normal – tem o mesmo objetivo da 2FN que é o de identificar e eliminar a transitividade. Um esquema de relação estará na 3FN se:

- Estiver na 2FN
- Todo atributo não chave deve ser determinado de forma não transitiva pela chave primária ou, todo atributo não chave deve ser determinado apenas pela chave primária.

Para normalizar para a 3FN, é preciso mover os campos não enquadrados para outra tabela e normalizá-la também.

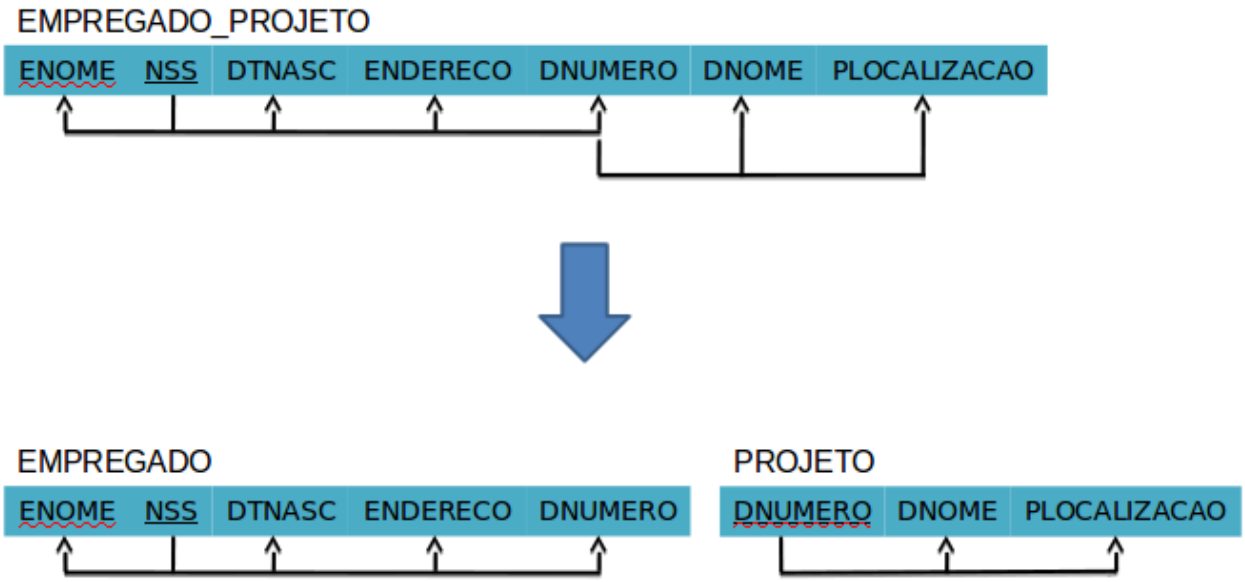


Figura 71 – Normalização para a 3FN

9.4.4 – Forma Normal de Boyce Codd – FNBC

Uma relação esquema R está na BCNF se, sempre que houver uma DF X◦ A em R, então X é uma superchave de R.

Perceba que cada forma normal engloba a forma normal anterior:

- Toda relação em 2FN está na 1FN
- Toda relação em 3FN está na 2FN
- Toda relação em BCNF está na 3FN
- Existem relações que estão na 3FN mas não em BCNF

A meta, portanto, é alcançar a BCNF (Forma Normal de Boyce Codd) ou 3FN (Terceira Forma Normal) em todas as relações. Desta forma, Uma relação esquema R está na 3FN se, sempre que houver uma DF X◦ A, então:

- X é uma superchave de R ou
- A é atributo primo de R.

Uma relação esquema R está na BCNF se, sempre que houver uma DF X◦A, então:

- X é uma superchave de R

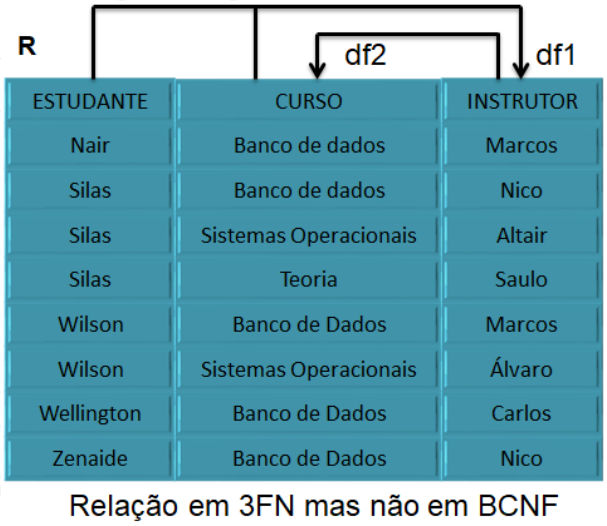


Figura 72 – 3FN e BCNF

Existem duas Definições Funcionais em relação:

- df1: { estudante, curso } ◦ instrutor
- df2: instrutor ◦ curso

Se a relação tivesse apenas df1, a relação estaria na BCNF mas em df2, instrutor não é uma superchave, e, portanto, viola a BCNF, mas não a 3FN, pois curso é primo.

Uma relação que não esteja na BCNF deve ser decomposta para atender a esta propriedade, mas abdica da preservação das dependências funcionais nas relações decompostas.

Três possíveis decomposições para relação:

- R1(estudante, instrutor) e R2(estudante, curso)
- R1(curso, instrutor) e R2(curso, estudante)
- R1(instrutor, curso) e R2(instrutor, estudante)

Assim, todas as três decomposições perdem a df1. É preciso conviver com este sacrifício, mas não podemos sacrificar a propriedade não-aditiva após a decomposição. Das três, apenas a terceira decomposição não gera tuplas espúrias após a junção (join), e, assim, mantém a propriedade não-aditiva.

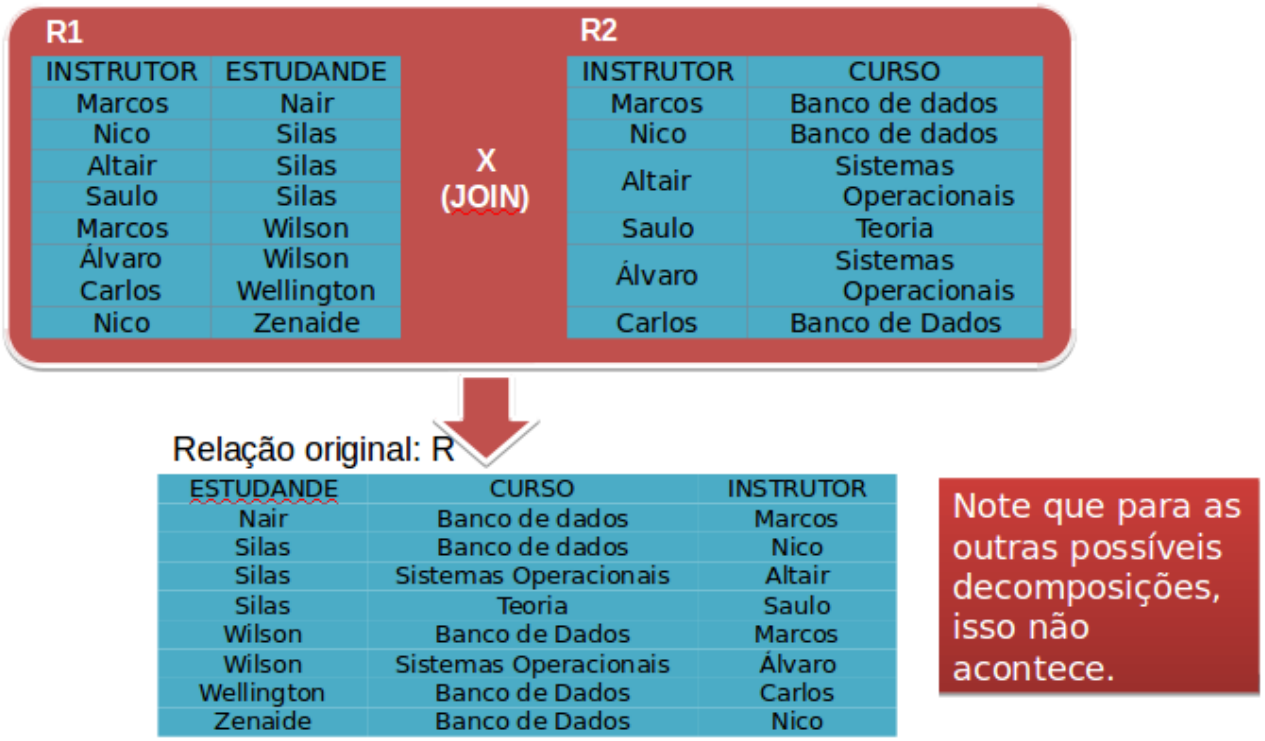


Figura 73 – Alcançando a BCNF pela Decomposição

9.4.5 – Dependências Multivaloradas e a Quarta Forma Normal – 4FN

As dependências multivaloradas são consequência da 1FN, a qual não aceita atributos multivalorados. Desta forma, sempre que X àà Y ocorrer, dizemos que X multidetermina Y. Devido a simetria da definição, sempre que X àà Y ocorrer em R, também ocorre X àà Z. Por isso, X àà Y implica X àà Z; por isso, às vezes é escrito como XààY | Z.

Considere a relação ACERVO, mostrada na figura 74.

ISBN	AUTOR	CÓPIAS
85-7323-169-6	Dantas	1, 2
0-13031-995-3	Molina, Ulman, Widom	1, 2

Figura 74 – Relação ACERVO

Considerando a regra da 4FN, na relação ACERVO da figura 74, tem-se:

- ISBN àà AUTOR | CÓPIAS

"As dependências multivaloradas são uma consequência da primeira forma normal (1FN) […], que desaprova um atributo em uma tupla para ter um conjunto de valores, e o processo correspondente de conversão de uma relação não normalizada para 1FN". (ELMASRI; NAVATHE, 2011, p. 357) Ao normalizar-se a relação ACERVO para 1FN tem-se a relação exibida na figura 75.

ISBN	AUTOR	CÓPIAS
85-7323-169-6	Dantas	1
85-7323-169-6	Dantas	2
0-13031-995-3	Molina	1
0-13031-995-3	Molina	2
0-13031-995-3	Ulman	1
0-13031-995-3	Ulman	2
0-13031-995-3	Widom	1
0-13031-995-3	Widom	2

Figura 75 – Relação ACERVO normalizada para 1FN

Pode-se observar, entretanto, que ainda existem redundâncias na relação ACERVO, mesmo depois de normalizada mas, por que? Porque restaram as dependências multivaloradas, ISBN ◦◦ AUTOR e ISBN ◦◦ CÓPIAS. Assim, o objetivo da 4FN (Quarta Forma Normal) é eliminar as redundâncias provocadas pelas dependências multivaloradas (MVD). Uma relação está na 4FN se estiver na 3FN e não contiver mais de uma dependência multivalorada MVD. (ELMASRI; NAVATHE, 2011)

É possível questionar, o porque é tão ruim ter uma tabela com múltiplas dependências multivaloradas? Ao observar-se a relação ACERVO exibida na figura 75, para inserir mais uma cópia do ISBN 0-13031-995-3, será necessário inserir 3 tuplas, uma para cada autor. Alterações e remoções sofrerão do mesmo problema, então, qual é a solução definitiva? Decompor a relação ACERVO em duas, de acordo com as dependências multivaloradas (MVD) ISBN àà AUTOR e ISBN àà CÓPIAS, como pode ser visto nas figuras 76 e 77.

ISBN	AUTOR
85-7323-169-6	Dantas
0-13031-995-3	Molina
0-13031-995-3	Ulman
0-13031-995-3	Widom

Figura 76 - ISBN àà AUTOR

ISBN	CÓPIAS
85-7323-169-6	1
85-7323-169-6	2
0-13031-995-3	1
0-13031-995-3	2

Figura 77 - ISBN àà CÓPIAS

Assim, a dependência multivalorada desejável é aquela cujo determinante é superchave da relação. Entretanto, há um caso especial. Se R tiver as seguintes MVDs (dependências multivaloradas) A àà B e B àà C, neste caso, R estará na 4FN se, e somente se, B e C são dependentes um do outro! (ELMASRI; NAVATHE, 2011)

Um outro exemplo pode ser a relação LECIONA, que possui as MVDs PROFESSOR ◦◦ DISCIPLINA e PROFESSOR ◦◦ TITULO, como pode ser visto na figura 78. Já na figura 79, pode-se observar a relação LECIONA decomposta em duas: PROFESSOR_DISCIPLINA e PROFESSOR_TÍTULO.

Professor	Disciplina	Título
Antonio	Linguagem de Programação I	Especialista
Antonio	Linguagem de Programação I	Mestre
Antonio	Linguagem de Programação I	Doutor
João	Banco de Dados	Especialista
João	Banco de Dados	Mestre
João	Banco de Dados	Doutor
João	Lógica de Programação	Especialista
João	Lógica de Programação	Mestre
João	Lógica de Programação	Doutor
Maria	Lógica de Programação	Doutora
Joaquim	Linguagem de Programação I	Mestre

Figura 78 - PROFESSOR àà DISCIPLINA

Professor	Disciplina
Antonio	Linguagem de Programação I
João	Banco de Dados
João	Lógica de Programação
Joaquim	Linguagem de Programação I
Maria	Lógica de Programação

Professor	Título
Antonio	Especialista
Antonio	Mestre
Antonio	Doutor
João	Especialista
João	Mestre
João	Doutor
Maria	Doutora
Maria	Mestre

Figura 79 - PROFESSOR àà TÍTULO

9.4.6 – Dependências de Junção e a Quinta Forma Normal – 5FN

Algumas vezes, uma relação não pode ser decomposta sem perdas em duas relações, mas pode ser decomposta em três ou mais. Se uma relação é decomposta em várias relações e a reconstrução não é possível pela junção das outras relações, dizemos que existe uma dependência de junção. A quinta forma normal (5FN) A 5FN capta a ideia de que uma relação esquema deve ter alguma decomposição sem perda (dependência de junção). Segundo Elmasri e Navathe (2011, p. 360), "É importante observar que tal dependência é uma restrição semântica muito peculiar, que é muito difícil de detectar na prática. Portanto, a normalização para a 5FN raramente é feita nestes termos". Um exemplo pode ser observado na figura 80.

AGENTE	EMPRESA	PRODUTO
Smith	Ford	Carro
Smith	Ford	Caminhão
Smith	GM	Carro
Smith	GM	Caminhão
Jones	Ford	Carro

Figura 80 – Relação AEP

Regra:

Se um AGENTE vende um certo PRODUTO e este AGENTE representa uma EMPRESA que faz este PRODUTO

então:

O AGENTE deve vender o PRODUTO para a EMPRESA.

Desta forma:

AGENTES representam EMPRESAS

EMPRESAS fazem PRODUTOS

AGENTES vendem PRODUTOS

Neste exemplo, ao normalizar a relação AEP para a 5FN, serão obtidas 3 novas relações, conforme pode ser visto na figura 81.

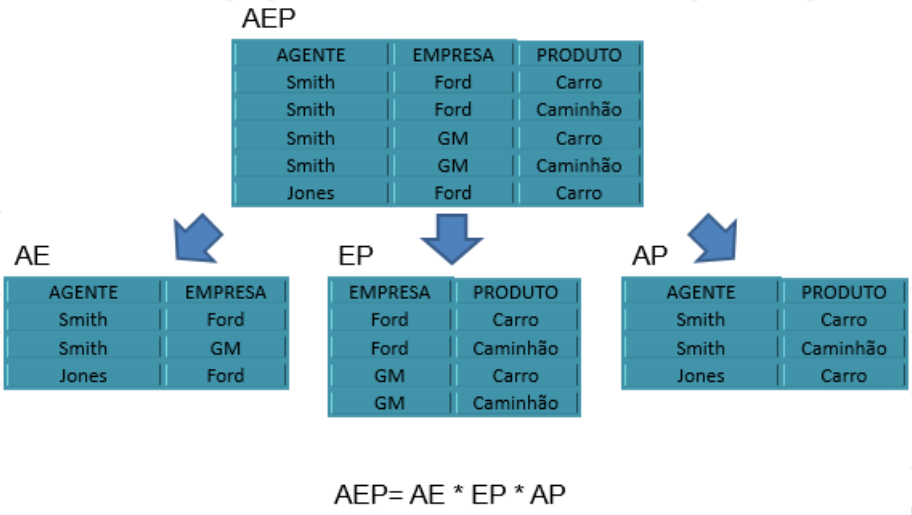


Figura 81 – Relação AEP na 5FN

Segundo Elmasri e Navathe (2011), uma relação R satisfaz a dependência de junção:

- JD (R1, R2, ..., Rn) se
  - R = R1* R2 * ...* Rn
- de forma que R1, R2, ..., Rn são subconjuntos dos atributos de R.

É possível notar que uma dependência multivalorada é um caso especial de dependência de junção (n=2). Assim, uma relação R está na 5FN se, e somente se, ela estiver na 4FN e todas as suas dependências de junção forem determinadas pelas chaves candidatas.

A descoberta de Dependências de Junção em bancos de dados reais com centenas de atributos é praticamente impossível. Isso poderá ser feito apenas contando com um grande grau de intuição sobre os dados por parte do projetista. Por isso, a prática atual de projeto de banco de dados dá pouca atenção a elas. (ELMASRI; NAVATHE, 2011)

9.4.7 – Demais Formas Normais

Existem outras formas normais, porém elas estão fora do escopo desta disciplina, pois são formas pouco utilizadas em projetos de banco de dados devido a sua dificuldade de aplicação prática.

Referências

DATE, C. J.. Introdução a sistemas de bancos de dados. Elsevier: Rio de Janeiro, 2003.

ELMASRI, Ramez; NAVATHE, Shamkant B.. Sistemas de banco de dados. 6. ed. São Paulo: Pearson Education, 2011. 788 p.

FANDERUFF, Damaris. Dominando o Oracle 9i: Modelagem e Desenvolvimento. São Paulo: Pearson Education do Brasil, 2003. 398 p.

FARIAS, Anderson Rodrigo. História da Oracle. Para Devmedia. Disponível em: <https://www.devmedia.com.br/historia-da-oracle/4685>. Acesso em: 21 mar. 2018.

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 6. ed. São Paulo: Bookman, 2009. 282 p.

HOUAISS, Antônio. Dicionário Houaiss da Língua Portuguesa. Rio de Janeiro: Objetiva, 2009. 1986 p.

METRO. Mapa do Metrô de São Paulo. Disponível em: <https://pt.saopaulomap360.com/mapa-metro-sao-paulo#.WrjiveYh3rc>. Acesso em: 26 mar. 2018.

RANGEL, Alexandre Leite et al (Ed.). BANCO DE DADOS. Batatais: Claretiano, 2015. 254 p.

SANCHES, Andre Rodrigo. Fundamentos de Armazenamento e Manipulação de Dados. 2005. Disponível em: <https://www.ime.usp.br/~andrers/aulas/bd2005-1/aula2>. Acesso em: 21 mar. 2018.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. Sistema de Banco de Dados. Rio de Janeiro: Elsevier, 2008. Tradução da 5ª Edição.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. SÃo Paulo: Pearson Education, 2011. 529 p.

TRENDS24.Trends 24: Brasil. Disponível em: <https://trends24.in/brazil/>. Acesso em: 20 mar. 2018.

TUTORIALSPOINT. COBOL: Manipulação de Arquivo Verbos. Disponível em: <https://www.tutorialspoint.com/pg/cobol/cobol_file_handling_verbs.htm>. Acesso em: 22 mar. 2018.

WIKIPEDIA. Banco de dados. 2010. Disponível em: <https://pt.wikipedia.org/wiki/Banco_de_dados>. Acesso em: 21 mar. 2018.

WIKIPEDIA. COBOL. 2004. Disponível em: <https://pt.wikipedia.org/wiki/COBOL>. Acesso em: 21 mar. 2018.

WIKIPEDIA. Concatenação. 2008. Disponível em: <https://pt.wikipedia.org/wiki/Concatenação>. Acesso em: 28 mar. 2018.

WIKIPEDIA. Modelagem de dados. 2004. Disponível em: <https://pt.wikipedia.org/wiki/Modelagem_de_dados>. Acesso em: 26 mar. 2018.