

Universidade Federal de São Carlos
Departamento de Computação

Bancos de Dados

Mauro Biajiz

Dezembro - 2001

Bancos de Dados

1. Introdução

Hoje em dia o termo banco de dados é bastante popular em diversas áreas de atuação. Com o aumento da utilização de computadores na manipulação de dados que envolvem diversas aplicações, bancos de dados estão sendo desenvolvidos e aplicados nas diferentes áreas que envolvem o comércio, a indústria e a pesquisa acadêmica. Por esses aspectos já se pode considerar a importância do assunto para profissionais da área de Informática e afins. Nesta seção são definidos os conceitos básicos envolvendo o tema.

Um **Banco de Dados** (ou *Base de Dados*) é uma coleção de dados relacionados, organizada e armazenada de forma a possibilitar fácil manipulação, incluindo alterações, inserções, remoções e consultas. Os tipos de “coleções de dados” são ilimitados, ou seja, quaisquer aplicações do mundo real que possam ser representadas através de dados computáveis, podem ser armazenadas em um banco de dados. Exemplos de coleções são: dados de um Banco Financeiro, dados de Controle de uma Universidade, dados de Controle de Estoque de Empresas, dados sobre os Genes Humanos (projeto Genoma), dados sobre Meteorologia, etc.

A manipulação desses dados armazenados é feita por um conjunto de programas computadorizados denominado **Sistema Gerenciador de Bancos de Dados (SGBDs)**. Um SGBD tem uma gama de funções pré-implementadas que gerenciam as operações de inserção, remoção, atualização e consulta dos dados armazenados.

Os *SGBDs* e os *Bancos de Dados* juntos formam um ambiente denominado **Sistema de Banco de Dados (SBD)**. Pode-se definir esse sistema como um ambiente cujo objetivo global é registrar e manter informação. Um SBD busca oferecer:

☞ **rapidez** → consultas on-line para informação;

☞ **disponibilidade total** → toda a informação contida no interior da base está disponível o tempo todo;

☞ **flexibilidade** → questões não tratadas tornam-se tratáveis, ou seja, mudanças são relativamente fáceis de se implementar.

☞ **Integridade** → a duplicação de dados é reduzida, e políticas de atualização podem ser padronizadas, resultando em consistência de dados.

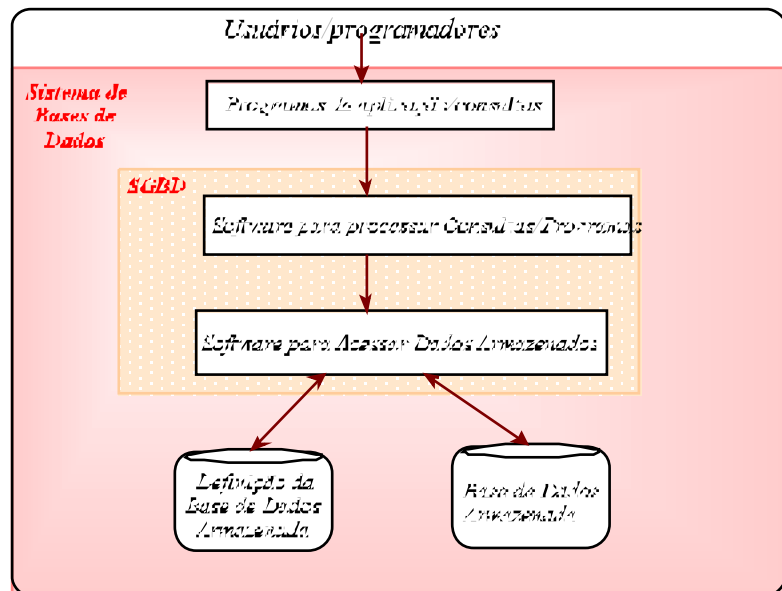


Figura 1: Arquitetura de um Sistema de Banco de Dados.

Como um todo, fazem parte de um SBD:

☞ **Dados** → valores fisicamente registrados no banco de dados;

☞ **Hardware** → memória secundária, unidades de controle, canais de comunicação, etc.

☞ **Software** → SGBD.

☞ **Usuários** → todos os usuários que estão envolvidos na definição e utilização de um banco de dados. Esses usuários podem ser divididos em três classes:

i) *programadores de aplicações* → responsáveis pela escrita de programas de aplicação que utilizem o banco de dados;

ii) *usuários finais* → utilizam uma linguagem de consulta fornecida como parte integrante do

sistema, ou podem chamar uma aplicação escrita pelo programador sob a forma de um programa (efetua operações de recuperação, criação, eliminação ou modificação);

iii) *DBA* → administrador do banco de dados, ou seja, o responsável pelo controle do “bom funcionamento” do banco de dados.

Todos esses conceitos apresentados até aqui podem ser visualizados através da representação gráfica apresentada na **Figura 1**.

A utilização de um banco de dados oferece um controle centralizado de seus dados operacionais com as seguintes vantagens:

☞ **redundância pode ser reduzida** → em sistemas de bancos de dados, a redundância deve ser controlada, isto é, o sistema deve ter conhecimento dessa redundância e assumir a responsabilidade de propagar as atualizações.

☞ **a inconsistência pode ser evitada** → através de regras muito bem definidas, os dados são estruturados garantindo-se a consistência dos dados armazenados.

☞ **concorrência entre aplicações** → os dados podem ser compartilhados por diversas aplicações ao mesmo tempo.

☞ **segurança** → a integridade pode ser mantida com a aplicação de restrições de segurança.

É usual a utilização de uma representação gráfica da “arquitetura” de um SBD.

Pode-se afirmar que quase

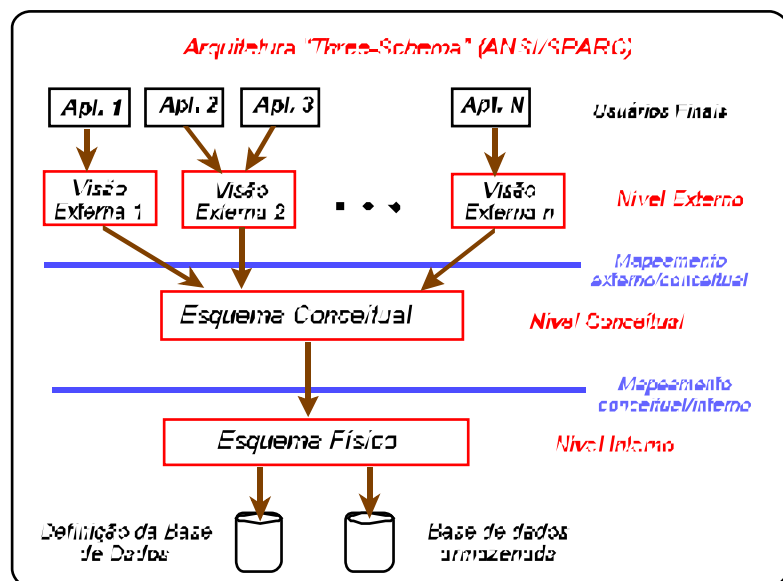


Figura 2: Arquitetura de um SGBD segundo níveis de visão.

todas as vantagens e propriedades de um banco de dados são possíveis graças à forma como são esquematizados e estruturados os seus dados. Para que um SGBD possa “entender” a estrutura como os dados do banco estão armazenados, há a necessidade que esses dados estejam *esquematizados* de forma a serem interpretados pelas funções pré-definidas no gerenciador. Esse *esquema* é elaborado segundo um **Modelo de Dados**. Um *modelo de dados* compreende uma coleção de elementos de representação com propriedades semânticas e sintáticas pré-definidas. Um elemento ou um conjunto de elementos de representação são devidamente agrupados e organizados para representar uma porção do “mundo real”, constituindo um “esquema de dados” compreensível pelo gerenciador.

Agora é apresentada uma arquitetura para banco de dados em níveis. Esta arquitetura pode ser vista na **Figura 2**, apresentando três níveis, os quais são denominados *esquemas externo, conceitual e interno*.

| O **esquema interno** descreve a estrutura de armazenamento físico do banco de dados. O esquema interno usa um modelo de dados físico e descreve os detalhes completos de armazenamento de dados e caminhos de acesso para banco de dados.

| o **esquema conceitual** descreve a estrutura do banco de dados para toda a comunidade de usuários. O esquema conceitual esconde os detalhes das estruturas de armazenamento físico e concentra-se na descrição das entidades, tipos de dados, relacionamentos, operações dos usuários, e restrições.

| o **esquema externo** inclui vários esquemas externos ou visões dos usuários. Cada esquema externo descreve a parte da base de dados que um grupo particular está interessado e oculta o resto do banco para o grupo de usuários.

Essa estrutura em níveis permite a implementação de um conceito extremamente importante, o qual é denominado **independência de dados**. A *independência de dados* pode ser definida como a capacidade de se alterar o esquema em um nível de um sistema de banco

de dados sem ter que alterar o próximo nível. A independência pode ser lógica ou física.

Independência lógica de dados - é a capacidade de se alterar o esquema conceitual sem ter que alterar o esquema externo.

Independência física de dados - é a capacidade de se alterar o esquema interno sem ter que alterar o esquema conceitual ou o esquema externo.

Esse conceito é efetivamente implementado através de mapeamentos realizados entre os três níveis.

Esquemas e Instâncias

A descrição de um banco de dados é chamada *Esquema* do banco de dados. A maioria dos modelos de dados possuem algumas convenções gráficas para diagramar o esquema.

Exemplo de um diagrama de esquema:

Estudante	Nome	Número-Estudante	Class e
-----------	------	------------------	------------

Curso	Nome	Número-Curso	Créditos	Departamento
-------	------	--------------	----------	--------------

Pré-requisito	Número-Curso	Número-pré-requisito
---------------	--------------	----------------------

Os dados que estão armazenados em um determinado momento de tempo no banco de dados constituem uma *Instância* do banco de dados. Dessa forma, os estudantes, cursos e pré-requisitos armazenados em um determinado instante constituem uma *instância*.

Na próxima seção é descrito o Modelo Entidade-Relacionamento.

2. Modelo Entidade-Relacionamento

O Modelo Entidade-Relacionamento foi apresentado por Peter Chen em 1976. O modelo baseia-se em representar os dados do “mundo real” através da definição de *conjuntos de entidades* e o *relacionamento* entre esses *conjuntos de entidades*. Um *conjunto de entidades* representa um conjunto de elementos do mundo real, como por exemplo, um conjunto

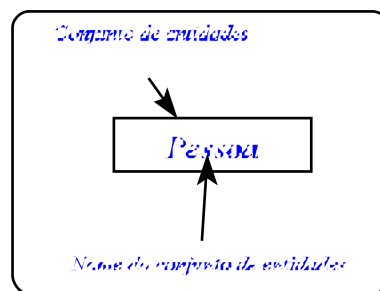


Figura 3: Conjunto de Entidades

Pessoa representando inúmeras pessoas, um conjunto Casa representando inúmeras casas, um conjunto Empresa representando inúmeras empresas, etc.

Um *conjunto de entidades* é representado por um retângulo, como ilustrado na **Figura 3**. Um elemento do *conjunto de entidades* é definido como uma *Entidade*, sendo identificado por características individuais definidas através do conceito de *atributos*. Assim, uma Pessoa pode ser caracterizada através dos atributos Nome, CIC, Sexo, Idade, Altura, etc.

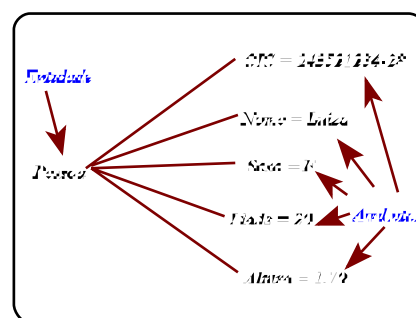


Figura 4: Representação de uma pessoa particularizada pelos valores de seus atributos.

Cada vez que são atribuídos valores para os *atributos* de um *conjunto de entidades*, tem-se a instanciação (= exemplificação) de uma ocorrência de uma *entidade* da vida real. Na **Figura 4**, pode-se ver um exemplo, onde a *entidade* Pessoa é Luiza, CIC 243521234-28, do sexo feminino, idade de 20 anos e altura de 1.70m. Cada *atributo* deve ser definido como pertencente a um domínio, e os valores desses atributos devem pertencer a esses domínios.

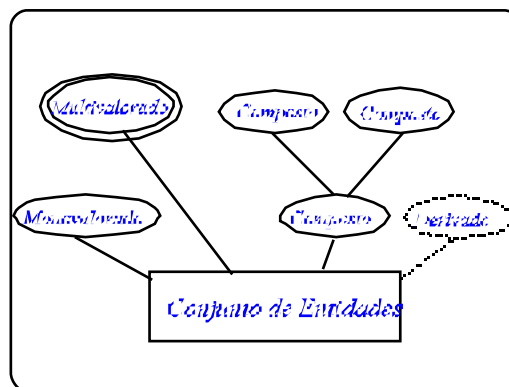


Figura 5: Representação de Atributos.

Os *atributos* desses *conjuntos de entidades* são representados através de seus nomes envoltos em elipses (círculos ou indicados por setas), as quais ficam ligadas ao conjunto de entidades que está caracterizando.

Quando um determinado atributo possui um valor para cada entidade que caracteriza, então é denominado *atributo monovalorado*. Como exemplo, os atributos CPF, Idade, Altura e Sexo possuem apenas um valor para uma *entidade* particular Pessoa. Por sua vez, quando um atributo possui mais de um valor para cada *entidade* que caracteriza, então é denominado *atributo multivalorado*. Como exemplo, um *atributo* Telefone pode ter vários valores para uma mesma entidade, e portanto é multivalorado. Um *atributo*

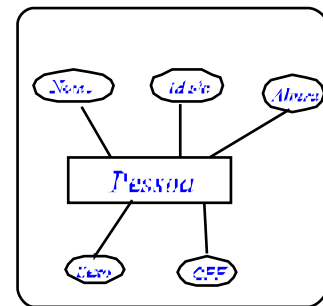


Figura 6: Conjunto de Entidades Pessoa com seus Atributos.

monovalorado ou *multivalorado* pode ter vários “subcampos”, como por exemplo Endereço, que poderia ser escrito como Rua, Número, CEP e Cidade. Nesse caso, o *atributo* é denominado *atributo composto*. Quando o valor de um *atributo* é obtido através de valores de outros *atributos*, esse atributo é denominado *atributo derivado*. Um exemplo de atributo derivado é o *atributo* Salário, que pode ser obtido pelos valores dos *atributos*

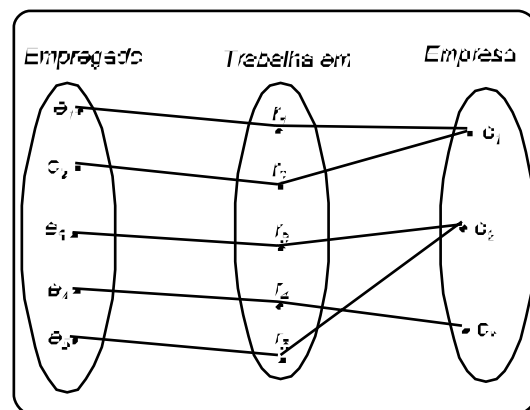


Figura 7: Relacionamento entre empregado e empresa.

Salário-Hora e Horas-Trabalhada-Mês. A simbologia desses atributos é ilustrada na **Figura 5**. Em uma forma mais completa, o conjunto de entidades Pessoa seria representado no ME-R, tal como na **Figura 6**. Um *Conjunto de Entidades* possui vários *atributos* para caracterizá-lo. Dentre esses atributos, deve-se sempre definir um *atributo*

ou um *conjunto de atributos*, que com seus valores consiga identificar uma única Entidade dentro do *Conjunto de Entidades*. Esse *atributo* ou *Conjunto de Atributos* é

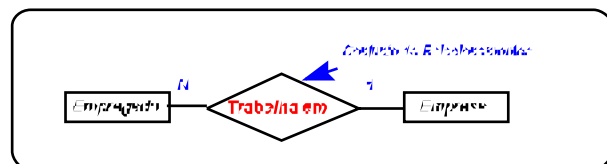


Figura 8: Exemplo de um Conjunto de Relacionamento.

denominado *Chave*. Uma *Chave* deve ser, por definição, mínima. Mínima, não no sentido de ser composta pelo menor número de atributos possível, mas no sentido de que se a chave for

composta, nenhum atributo que a compõe poderá ser retirado, e ainda sim, a composição resultante continuar sendo *Chave*. A notação adotada é a sublinhação do *atributo* ou *conjunto de atributos* em questão. Quando um *conjunto de atributos* contém uma chave, esse conjunto forma uma superchave.

Para a modelagem de uma situação real, é necessário representar-se o *relacionamento* entre os *Conjuntos de Entidades*. Esses *relacionamentos* são denominados *conjuntos de relacionamentos* e “possuem” rica semântica envolvida. Essa semântica é expressa através dos conceitos de *cardinalidade*, *multiplicidade*, *participação total* ou *parcial*, e *grau de relacionamento*. Considere os *conjuntos de entidades* Empresa e Empregado, e o *conjunto de relacionamentos* entre eles Trabalha_em, representando o fato de empregados trabalharem em empresa. A **Figura 6** ilustra essa situação. Estão ilustrados cinco empregados (e_1, e_2, e_3, e_4, e_5), três empresas (c_1, c_2, c_3) e cinco ocorrências de *relacionamento* (r_1, r_2, r_3, r_4, r_5). Pode-se inferir através da figura, que um empregado trabalha em uma única empresa, pois cada um participa de apenas uma ocorrência de *relacionamento*, enquanto uma empresa emprega vários empregados, uma vez que uma empresa pode participar de mais de uma ocorrência de *relacionamento*. Essa percepção da realidade pode ser representada através do conceito de *Cardinalidade de Relacionamento*. No modelo, um *conjunto de relacionamentos* é representado simbolicamente por um losângulo, enquanto a *cardinalidade* é representada por um número. Na **Figura 7**, está

ilustrado um *conjunto de relacionamentos* Trabalha_em, entre os *conjuntos de entidades* Empregado e Empresa. A *cardinalidade* do *conjunto de relacionamentos* é definida pelos

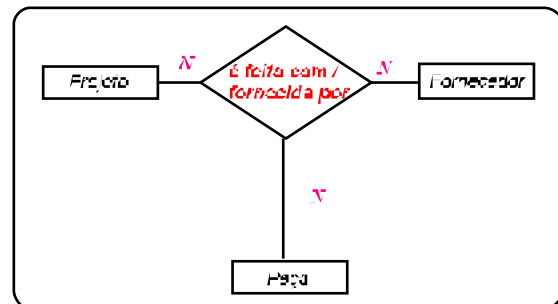


Figura 10:Relacionamento ternário.

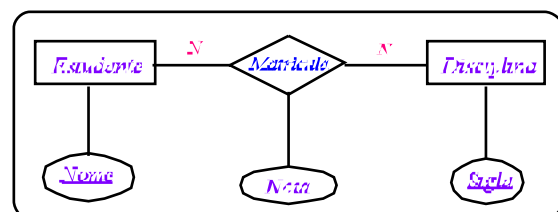


Figura 9:Atributo de Relacionamento.

números 1 e N. No caso, está representando que um Empregado pode trabalhar em apenas uma (1) empresa, enquanto uma Empresa pode empregar vários (N) empregados. As *cardinalidades* entre os *conjuntos de relacionamentos* podem ser de 1 para 1, ou de 1 para N (ou N para 1), ou de N para N. A letra “N” representa vários e pode ser substituída por qualquer outra letra, tal como, M, P, Q, etc.

O *grau do relacionamento* indica quantos *conjuntos de entidades* estão envolvidos naquele determinado *relacionamento*. No exemplo da **Figura 7**, tem-se que o *grau do relacionamento* é 2, indicando um *relacionamento binário*. Um *conjunto*

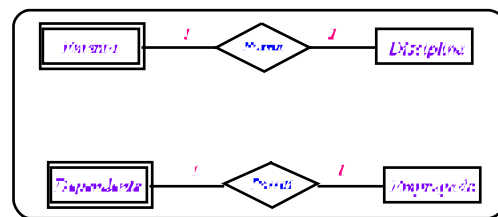


Figura 11: Conjunto de Entidades Fracas.

de relacionamentos pode ter associado vários *conjuntos de entidades*, caracterizando relacionamentos ternários (grau 3), relacionamentos quaternários (grau 4), etc. Um exemplo de relacionamento ternário, ou seja, com grau 3, está ilustrado na **Figura 10**. É importante observar que um relacionamento com grau $N > 2$ só se justifica se não puder ser decomposto em *relacionamentos* com graus menores e ainda manter a semântica desejada. *Conjuntos de relacionamentos* também podem ter *atributos*. Considere por exemplo os *conjuntos de entidade* Estudante e Disciplina, relacionando-se através do *conjunto de relacionamento* Cursa. Se deseja-se representar e "armazenar" a Nota que um determinado Estudante obteve em uma determinada Disciplina, o atributo Nota deve pertencer ao relacionamento, e não aos *conjuntos de relacionamentos*. Isso deve-se à cardinalidade envolvida no relacionamento (**Figura 10**).

Os *Conjuntos de Entidades* destacados até o momento podem ser denominados *Regulares*. Existe uma categoria particular de *Conjunto de Entidade* que envolve o conceito de *Entidade Fraca*. Um *Conjunto de Entidades Fracas* corresponde a um *conjunto* que no contexto da modelagem em que se encontra, não possui identificação própria. Um exemplo pode ser visto na **Figura 11**, onde os *conjuntos de Entidades Fracas* são denotados por um retângulo duplo. No primeiro caso, uma Ementa só tem sentido se existir uma Disciplina a ela associada. No segundo

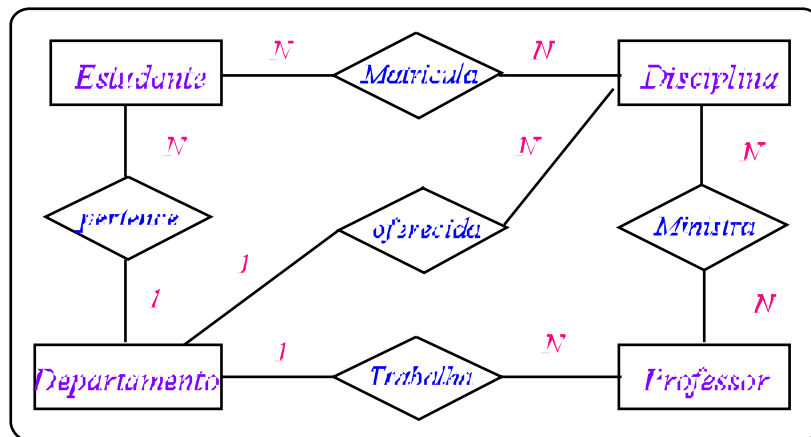


Figura 12: Exemplo de modelagem.

caso, Dependente só existe se houver um Empregado a ele associado. Note que não adianta usar um atributo identificador para um Dependente, pois só terá sentido para o ambiente que está modelado, se esse identificador estiver associado a um Empregado.

Considere um exemplo de uma Universidade na Figura 12, onde está ilustrada uma modelagem utilizando o ME-R. No exemplo são representados

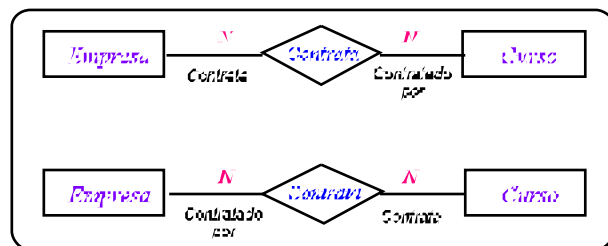


Figura 13: Exemplo do uso de papel no relacionamento.

Estudantes, Professores, Disciplinas, Departamento e o relacionamento entre eles. Os relacionamentos representados são Matrícula, entre os conjuntos de entidades Estudante e Disciplina, Pertence, entre Estudante e Departamento, Oferecida entre Departamento e Disciplina, Trabalha entre Departamento e Professor, e Ministra entre Disciplina e Professor. Note que para efeito de simplicidade, foram omitidos os atributos no desenho.

Uma característica importante é o de *Papel de Relacionamentos*. Em muitas

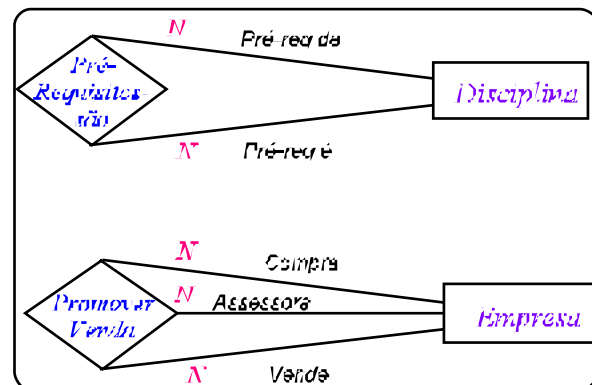


Figura 14: Exemplos de Auto-relacionamento

situações não é clara qual a participação de um determinado *Conjunto de Entidades* em um *Relacionamento*. Quando isso ocorre, anota-se o *papel* do *conjunto de entidades* no *relacionamento*. Na **Figura 13** ilustra-se um exemplo. Estão representados dois *Conjuntos de Entidades*, que são Empresa e Curso, e dois *Conjuntos de Relacionamentos* distintos, apesar de terem o mesmo nome. Em uma situação, Empresa contrata Curso, e na outra, Curso contrata Empresa caracterizando semânticas totalmente diferentes. O *Papel* também é extremamente importante quando modela-se *Conjuntos de Relacionamentos Unários*, ou seja, quando apenas um *Conjunto de Entidades* está envolvido. Na **Figura 14** Estão ilustrados dois *Conjuntos de Relacionamentos Unários*. No primeiro, o *Conjunto de Entidades* é Disciplina e o *Conjunto de Relacionamentos* é Pré-Requisitos-são. Nesse caso o *papel* indica qual Disciplina é Pré-requisito de de qual Disciplina. No segundo, o *Conjunto de Entidades* é Empresa e o *Conjunto de Relacionamentos* Promover Venda. Nesse caso o *Papel* indica qual Empresa compra, qual vende e qual assessora a venda. Note a importância semântica do papel para o entendimento do relacionamento.

O Modelo Entidade-Relacionamento apresentado até aqui é original como apresentado por Peter Chen. Muitas extensões foram definidas sobre o modelo. O ME-R com extensões foi denominado Modelo Entidade-Relacionamento Estendido (ME-RX). Essas extensões envolvem, entre outras coisas, um refinamento do conceito de *Cardinalidade*, o qual foi denominado *Multiplicidade*. Envolve também a utilização de construtores semânticos mais elaborados, tal como o da *Abstração de Generalização* e o da *Abstração de Agregação*.

A *multiplicidade* é identificada por um par de números entre parênteses, onde o primeiro indica um limite mínimo e o direito indica um limite máximo. Considere novamente a **Figura 6**. a *multiplicidade* é definida da seguinte forma: para que uma *Entidade* seja um Empregado, é necessário que trabalhe em pelo menos uma Empresa. Como está sendo definido que ele pode trabalhar em no máximo uma Empresa, tem-se a *multiplicidade*(1,1) para o *Conjunto de Entidades* Empregado. Para existir uma *Entidade* Empresa, não é necessário, a princípio, existir um

Empregado relacionado. Assim, o limite mínimo para a multiplicidade é zero. Por outro lado, uma Empresa pode ter vários Empregados, contituindo o máximo como um número M. A representação da multiplicidade pode ser vista na **Figura 15**.

A *Abstração de Generalização* é utilizada em um processo de simplificação e enriquecimento semântico da modelagem. Em muitas situações, vários *Conjuntos de*

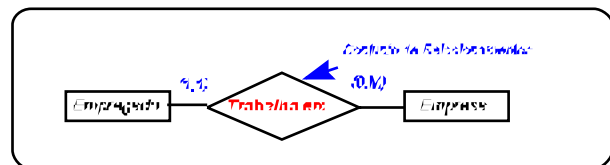


Figura 15:Exemplo do uso da Multiplicidade.

Entidades com origem em um *Conjunto de Entidades* comum (superconjunto), são modelados e utilizados independentemente. Nesses casos, modela-se o *Conjunto de Entidades* comum colocando nele todos os *atributos* que são comuns entre todos os *Conjuntos de Entidades*

(subconjuntos) nele originados, considerando-se duas propriedades importantíssimas para a implementação. Um exemplo pode ser visto na **Figura 16**.O *Conjunto de Entidades* Pessoa representa uma generalização dos *conjuntos de entidades* Aluno, Professor e Engenheiro. Os *atributos* que são comuns são modelados no *conjunto de Entidades*

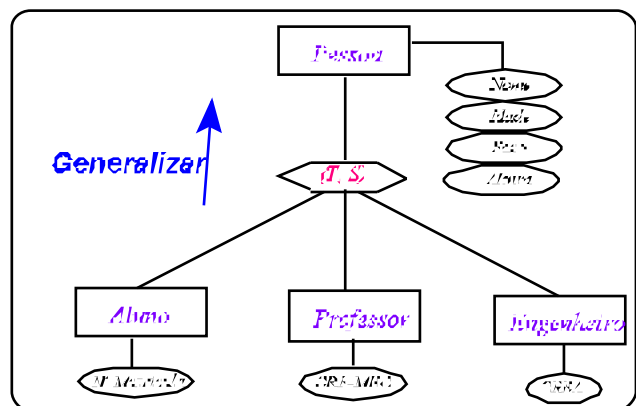


Figura 16: Abstração de Generalização

Genérico Pessoa, ficando apenas os específicos de cada um nos respectivos conjuntos. As letras colocadas no losângulo ao meio, são propriedades da generalização. A primeira propriedade pode ser T ou P. O T representa Total e significa que todas as ocorrências de *Entidades* a partir do *Conjunto de Entidades* Pessoa tem que ser de um dos tipos de *Entidades* pertencentes aos *conjuntos de entidades* especializados Aluno, Professor ou Engenheiro. Caso fosse P, significaria que poderiam ocorrer Pessoa sem ser dos tipos abaixo.

A *Abstração de Agregação* é utilizada quando há a necessidade do estabelecimento de uma “associação” entre os *Conjuntos de Entidades* envolvidos em um *Conjunto de Relacionamentos*. A situação mais comum ocorre quando há necessidade de associar um *relacionamento* a um outro *relacionamento*, o que não é possível. Nesse caso estabelece-se uma

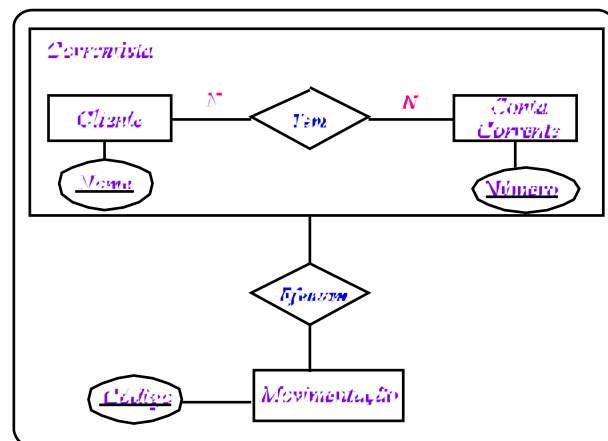


Figura 17: Abstração de Agregação.

agregação. Um exemplo pode ser visto na **Figura 17**, onde os *Conjuntos de Entidades* Cliente e Conta corrente relacionam-se, e desse *relacionamento* é gerada a ocorrência do *Conjunto de Entidades* Movimentação. Assim, o *Conjunto de Entidades* Movimentação relaciona-se com o *Conjunto de Entidades* Correntista originado de uma Agregação.

Os conceitos do ME-R aqui apresentados cobrem todo o modelo original, sendo que, outros conceitos podem ser encontrados na bibliografia citada.

Em seguida, são descritas algumas especificações que envolvem “problemas” para serem utilizados como exercícios de modelagem.

Especificações simplificadas

(1) Agência de Turismo

Deseja-se criar um banco de dados para uma agência de turismo contendo informações sobre recursos oferecidos pela cidades que fazem parte da programação de turismo da agência. As informações a serem mantidas sobre cada cidade referem-se a hotéis, restaurantes e pontos turísticos. Sobre os hotéis deseja-se guardar seu nome, endereço, categoria (5 estrelas, 4 estrelas etc), os tipos de apartamentos existentes, o valor da diária de acordo com o tipo do apartamento. Sobre cada cidade deve-se armazenar seu nome, estado e um código único para buscas. Sobre os restaurantes é de interesse guardar o nome, endereço e a categoria (de luxo, simples, etc). Nesse banco estão sendo considerados como pontos turísticos igrejas, casas de show e museus,

entre outros. No caso de igrejas deseja-se manter informações sobre o seu nome, a data de construção, a religião em que é ou foi utilizada, o endereço, uma descrição e o estilo de construção. Para casas de show devem ser guardados o seu nome, o tipo de show, o endereço, uma descrição e os dias de funcionamento e horário. Para museus deseja-se guardar seu nome, a data de fundação, o número de salas, o endereço, uma descrição e fundadores. Sobre os outros pontos turísticos deseja-se armazenar somente informações gerais, tais como, um nome, uma descrição e o endereço.

(2) Construtora EmHab

A empresa de habitação EMHAB está desenvolvendo um sistema de controle de todos os condomínios que já construiu. Cada condomínio possui um conjunto de prédios (edifícios com vários andares) que obedecem uma determinada numeração. Assim, o condomínio "X de Tal" possui, por exemplo, 36 prédios, cada prédio possuindo Y apartamentos. Deseja-se saber dados sobre os moradores de cada apartamento/prédio, incluindo nome, cic, rg, sexo, idade, e a renda média de todos os moradores de um determinado apartamento. Um apartamento deverá ser identificado, dentro de um prédio, por um número e pelo seu andar - e um prédio deverá ser identificado, em um determinado condomínio, também por um número. Podem ser colocados mais dados sobre o prédio, tais como, as cores de sua pintura, quantos andares contém, etc. Para identificar um condomínio, pode-se atribuir um nome, a cidade, o bairro em que se localiza e um número de identificação. Não se esqueça de representar as cardinalidades, os atributos (multivalorados, derivados, etc, se existirem), a participação dos conjuntos de entidades nos relacionamentos, etc.

(3) Rádio FM

Uma Rádio FM decidiu criar um sistema automatizado para atender aos seus ouvintes. O objetivo do sistema é o atendimento imediato pelo telefone, ou seja, ao atender um pedido para tocar uma música, o locutor poderá localizá-la imediatamente e tocá-la.

Para esse fim, organizou o acervo de discos da seguinte maneira: numerou as prateleiras e os discos, criou um banco de dados com todas as músicas e idealizou a busca através de índices. O sistema deverá então armazenar todos os dados referentes às prateleiras, aos discos, às músicas contidas em cada disco, aos compositores de cada música, à duração de cada música em minutos, à data de gravação, etc. Deve-se considerar que uma música pode estar em vários discos, pode ter vários interpretes, mas sempre tem os mesmos compositores. A organização das prateleiras pode ser feita de várias formas. Uma sugestão é a organização considerando os estilos das músicas em um disco.

Estando com esses dados armazenados, o sistema gerenciará os programas de música,

sendo capaz de atender imediatamente aos pedidos, gerar relatórios detalhados com os pedidos, gerar estilos de programação, etc. O sistema deverá também cadastrar os ouvintes e possibilitar o envio de mala direta aos mesmos.

(4) Agência de Modelos

O sistema a ser modelado será utilizado pelo Sindicato das Agências de Moda e Desfile, devendo guardar informações sobre as diversas Agências cadastradas no sindicato.

Uma Agência possui armazenado, em seu banco de dados, todos os dados sobre todas as pessoas com quem tem relação. Entre as pessoas armazenadas estão os modelos masculinos e femininos, os clientes (fabricantes de roupas, lojistas), e outras pessoas que simplesmente gostam de moda (pessoas comuns). Sobre modelos, ficam armazenados dados como nome completo, CPF, endereço, cor dos olhos, cor da pele, tamanho (altura, coxas, cintura, busto), peso, sexo e RG. Sobre os Clientes, ficam armazenados nome completo, RG, CPF, endereço, sexo, informação dizendo se é proprietário de loja ou fábrica, e um código único para sua identificação. Sobre outras pessoas, ficam guardados o CPF, o endereço, o nome completo, e um atributo descritivo indicando qual é o seu interesse em desfiles. Os modelos de uma determinada Agência pertencem a uma única Agência, não podendo desfilarem para outras Agências. Devem ser armazenados todos os Desfiles organizados por uma determinada Agência, guardando dados, como Nome_Desfile, a data, o Local, o Estilo_do_Desfile. Para cada Desfile, deseja-se saber quais foram os modelos que desfilaram, quais foram os clientes que o frequentaram, e quais pessoas comuns também estiveram presentes, ou seja, que assistiram ao desfile. É interessante notar que os desfiles dividem-se naturalmente entre Desfiles de Moda-Verão e Desfiles de Moda-Inverno. É de interesse também guardar informações sobre o número de pessoas que frequentou um determinado desfile, a duração em minutos de um determinado desfile e quais foram os patrocinadores de um determinado desfile.

(5) Céu

O céu é composto por moradores comuns, por anjos, por santos e, é claro, por Deus. Os anjos e santos desempenham funções particulares. Os anjos são divididos em anjos operários e anjos da guarda. Os anjos da guarda são alocados para olharem por mortais ainda na terra. Cada mortal tem dois anjos que o guardam, e cada anjo da guarda é alocado para apenas um mortal. Os anjos operários desempenham funções no céu, as quais são denominadas funções celestiais. Essas funções são alocadas para inúmeros anjos, enquanto um anjo pode ser responsável por inúmeras dessas tarefas. Os santos ficam o dia todo atendendo pedidos provenientes dos mortais. Em algumas vezes, esses atendimentos são entendidos como milagres. Os moradores comuns passam o dia orando e se purificando, e tem a função de venerar santos e Deus por uma determinada quantidade de horas por dia. Durante o restante do tempo, esses moradores descansam. Os dados que devem ser armazenados sobre os anjos são cod_anjo, cor_asas, nome e idade; sobre os santos

são: cod_santo, cor_vestes, tempo_beatificação, nome e idade; sobre os moradores comuns: cod_mor, grau_luz, nome e as horas que ora por dia. Além de tudo isso, um anjo sempre é supervisionado por outros anjos, o qual pode supervisionar vários anjos. Sobre Deus, não se sabe muita coisa, e por isso atribui-se apenas um código para leitura.

(6)Campeonato

A especificação refere-se ao controle de um campeonato de futebol. Participam do campeonato 24 equipes. Cada equipe possui um nome, nome de seu técnico, nome de seus 11 titulares, nome de seus 11 reservas, uniformes número 1 e 2 com a cor da camisa, das meias e do calção. Deve-se relacionar com cada equipe, as informações sobre a que país pertence. Cada país possui nome, continente, população, tamanho em km quadrados, renda-percapita e condição (país desenvolvido, em desenvolvimento ou subdesenvolvido). Devem ser guardadas informações sobre as partidas realizadas. Sobre cada partida deve-se guardar as equipes participantes, o placar, o nome do juiz principal, a localização do campo (cidade) e o nome do campo.

3. Modelo Relacional

O **Modelo Relacional** foi apresentado por Codd em 1970. Seus dados são vistos como armazenados em

"tabelas", tais como a da

Figura 18. Todos os conceitos definidos para o modelo estão ligados à representação de uma tabela, sendo definidos como:

NÚMERO	ORIGEM	DESTINO	PARTIDAS	CHEGADAS
83	São Paulo	Manaus	11:30	19:00
84	São Paulo	Brasília	13:00	18:00
213	Rio de Janeiro	São Paulo	10:00	10:35
214	Rio de Janeiro	São Paulo	10:30	11:05

Figura 18:Relação Vôos.

- ☞ uma *tabela* é uma *relação*;
- ☞ uma *linha* em uma relação representa uma instância com valores definidos entre um conjunto de valores, recebendo o nome de *tupla*;
- ☞ uma *tupla* é composta de *valores* os quais são designados como *atributos*;
- ☞ um *atributo* pode assumir um *valor* dentro de um *conjunto de valores possíveis*, conjunto esse denominado *domínio do atributo*.
- ☞ toda relação possui um *subconjunto de nomes de atributos* que tem a propriedade de que quaisquer *tuplas* podem ser distingüidas umas das outras, através dos valores correspondentes aos atributos do subconjunto. Esse *subconjunto* denomina-se *chave estrangeira*.
- ☞ *grau* de uma relação é o *número de atributos* que essa relação possui.

Considerando-se as definições acima, identifica-se na relação Vôos, cinco atributos que são: Número, Origem, Destino, Partidas e Chegadas. Esses atributos juntos formam uma relação de grau 5, composta de 4 tuplas. Para identificar uma tupla dentro de uma relação, é

necessário o estabelecimento de um ou mais campos como chave dessa relação. No exemplo acima, o atributo Número identifica univocamente uma tupla, e pode ser designado como chave da relação.

A conceituação apresentada acima têm suas definições baseadas na Teoria Relacional, e são apresentadas a seguir.

Um esquema de relação \mathbf{R} , denotado por $R(A_1, A_2, \dots, A_n)$ é composto por um nome de relação R e uma lista de atributos A_1, A_2, \dots, A_n . Para cada nome de atributo A_i , existe um conjunto D_i denominado domínio de A_i ou $dom(A_i)$. Um nome de atributo pode ser chamado símbolo do atributo ou simplesmente atributo.

Uma base de dados relacional é composta pelas instâncias das suas relações, e uma relação instanciada é composta por instâncias de suas tuplas. A aplicação de uma operação sobre uma relação instanciada, tal como inclusão e exclusão, faz com que o número de tuplas instanciadas varie, criando-se uma nova relação.

Para a identificação de uma tupla, uma *chave* tem que ser definida. Antes de defini-la, é necessário definir-se o conceito de *superchave*. Uma superchave de uma relação r é um sub-conjunto $K = \{A_1, A_2, \dots, A_n\}$ de R , onde para duas tuplas

$$t_1, t_2 \in r, t_1 \neq t_2, \exists K' \subseteq K \mid t_1(K') = t_2(K') \quad , \text{ ou seja, não existem duas tuplas com os mesmos}$$

valores para todos os atributos em K .

Chave de uma relação r , é um sub-conjunto K de R , tal que,

$$\forall t_1, t_2 \in r, t_1(K) = t_2(K), \exists K' \subsetneq K \quad \text{que satisfaça essa propriedade, ou seja, uma chave não pode}$$

ter nenhum de seus atributos removidos e ainda continuar sendo uma chave. Considerando a relação *Vôos*, $\{\text{Número}\}$ é uma chave (e uma superchave), enquanto $\{\text{Número}, \text{Partidas}\}$ é uma superchave, mas não é chave.

Em geral, um esquema de relação pode ter mais do que uma chave. Neste caso, cada uma das chaves é chamada de **chave candidata (ou chave secundária)**. É comum designar uma das chaves candidatas como **chave principal (ou chave primária)** da relação. Por convenção, os atributos que formam a chave principal de um esquema da relação são sublinhados. Note que, quando um esquema da relação tem muitas chaves candidatas, a escolha de uma delas para ser chave principal é feito de forma arbitrária; entretanto, usualmente é melhor escolher a chave principal com um único atributo ou um pequeno número de atributos.

3.1. Esquemas de Banco de dados Relacional e Restrições de Integridade

Um banco de dados relacional usualmente contém muitas relações, com tuplas nas relações que estão relacionadas de várias formas.

• Um **esquema de banco de dados relacional** S é um conjunto de esquemas de relação

$S = \{R_1, R_2, \dots, R_m\}$ e um conjunto de restrições de integridade IC .

• Uma instância de banco de dados relacional DB de S é um conjunto de instâncias de relação

$DB = \{r_1, r_2, \dots, r_m\}$ tal que, cada r_i é uma instância de R_i e tal que as relações r_i satisfaçam as restrições de integridade especificadas em IC .

Restrições de integridade são regras com respeito aos valores que podem ser armazenados nas relações e que devem ser sempre satisfeitas, em quaisquer das relações do banco de dados.

Existem três restrições consideradas necessárias para um banco de dados relacional:

Restrição de Unicidade de Chave, Restrição de Integridade da Entidade e Restrição de Integridade Referencial.

(i) Restrição de Unicidade de Chave

Uma chave principal (candidata) não pode ter o mesmo valor em duas tuplas distintas da

mesma relação.

(ii) Restrição de Integridade da Entidade

A chave principal de qualquer relação não pode ter valor nulo em nenhuma tupla da relação.

(iii) Restrição de Integridade Referencial

O conceito de integridade referencial envolve duas relações e o conceito de chave estrangeira. É usada para manter a consistência entre tuplas de duas relações. A **restrição de integridade referencial** especifica que uma tupla em uma relação que refere-se a uma outra relação deve se referir a uma tupla existente naquela relação. Essa referência é realizada através da chave estrangeira.

Chave estrangeira - um conjunto de atributos **FK** no esquema de relação R_1 é uma chave estrangeira se satisfizer as seguintes condições:

- Os atributos em **FK** possuem o mesmo domínio dos atributos **PK** da chave primária de um outro esquema de relação R_2 ; diz-se que os atributos **FK referenciam** a relação R_2 .
- Um valor de **FK** em uma tupla t_1 de R_1 , ou ocorre como um valor de **PK** para alguma tupla t_2 , ou é nulo. No primeiro caso, tem-se $t_1[FK] = t_2[PK]$, e diz-se que a tupla t_1 referencia ou faz referência à tupla t_2 .

Restrições de integridade referencial tipicamente aparecem a partir de relacionamentos entre entidades. Note que uma chave estrangeira pode se referir à sua própria relação. Em um sistema relacional, é desejável que a linguagem de definição de dados (DDL) inclua provisões para especificar os vários tipos de restrições de forma que o SGBD possa automaticamente garanti-las. Existe uma outra classe de restrições gerais, algumas vezes chamadas de **restrições de integridade semântica**, que devem ser especificadas e garantidas em um banco de dados relacional. Exemplos de tais restrições são “o salário de um empregado não deve exceder o salário de seu supervisor” e “o número máximo de horas que um empregado pode trabalhar em todos os projetos, por semana, é 56”. Alguns SGBDs relacionais comerciais oferecem

mecanismos para especificação e garantia destas restrições.

3.2.A Álgebra Relacional

Uma *linguagem de consulta* é uma linguagem na qual um usuário requisita informações do banco de dados. Essas linguagens são tipicamente de mais alto nível do que as linguagens de programação comuns. Linguagens de consulta podem ser classificadas como *procedurais* e *não-procedurais*. Numa linguagem procedural, o usuário instrui o sistema a executar uma seqüência de operações no banco de dados a fim de computar o resultado desejado. Numa linguagem não-procedural, o usuário descreve a informação desejada sem fornecer um procedimento específico para obter tal informação.

A maioria dos sistemas de banco de dados relacionais comerciais oferece uma linguagem de consulta que inclui elementos de abordagens tanto procedurais como não-procedurais. A álgebra relacional é uma linguagem procedural, que possui uma coleção de operações que são utilizadas para manipular relações inteiras. O resultado de cada operação é uma nova relação, que pode ser manipulada posteriormente.

As operações da álgebra relacional são usualmente divididas em dois grupos:

- operações da Teoria de Conjuntos da matemática: *união, interseção, diferença, e produto cartesiano*.
- operações desenvolvidas especificamente para bancos de dados relacionais, que inclui *seleção, projeção e junção*, entre outras. A tabela a seguir será usada para exemplificar as operações.

Empregado

Nome	Sobren	RG	DataN	Endereço	Sexo	Salário	RG Super	NroD
João	Silva	123456789	09.01.55	Rua 15 Novembro, 731, São Carlos	M	3000	333445555	5

Francisco	Souza	333445555	08.12.45	Rua Jorge Assef, 100, S.Paulo	M	4000	888665555	5
Alice	Fernandes	999887777	18.03.75	Rua9 de Julho, 535, São Carlos	F	2500	987654321	4
Joana	Pereira	987654321	20.06.31	Rua Tiradentes, 291, Rib.Preto	F	4300	888665555	4
Rodolfo	Nogueira	666884444	15.09.52	Av.São Carlos,1005, São Carlos	M	3800	333445555	5

3.2.1.A operação *SELECT*

A operação *SELECT* é utilizada para selecionar um subconjunto de tuplas numa relação que satisfaça uma **condição de seleção**. Por exemplo, para selecionar o subconjunto de tuplas de EMPREGADO que trabalha no departamento 4 ou cujo salário é maior do que R\$ 3000,00, pode-se especificar cada uma dessas duas condições com a operação *SELECT*, como segue:

$$\sigma_{\text{NR0D}=4}(\text{EMPREGADO})$$

$$\sigma_{\text{SALARIO}>3000}(\text{EMPREGADO})$$

Em geral, a operação *SELECT* é denotada por:

$$\sigma_{\langle \text{condição da seleção} \rangle}(\text{Nome da Relação})$$

onde o símbolo σ (sigma) é usado para denotar o operador *SELECT* e a condição de seleção é uma expressão Booleana especificada nos atributos da relação. A relação resultante da operação *SELECT* tem os mesmos atributos da relação especificada em <nome da relação>. A expressão Booleana especificada em <condição de seleção> tem a forma:

$$\langle \text{Nome do Atributo} \rangle \langle \text{operador de comparação} \rangle \langle \text{valor constante} \rangle, \text{ ou}$$

$$\langle \text{Nome do Atributo} \rangle \langle \text{operador de comparação} \rangle \langle \text{Nome do Atributo} \rangle$$

onde <Nome do Atributo> é o nome de um atributo de <nome da relação>, <operador de comparação> é normalmente um dos seguintes operadores {=, <, ≤, >, >, ≠} e <valor constante> é um valor constante do domínio do atributo. As cláusulas podem ser arbitrariamente conectadas pelos operadores Booleanos AND, OR e NOT, para formar uma condição de seleção geral. Por exemplo, para selecionar as tuplas para todos os empregados que ou trabalham no departamento 4 e recebem mais de R\$ 4000,00 por ano, ou trabalham no departamento 5 e recebem mais de R\$ 3000,00, podemos especificar a seguinte operação SELECT:

$$\sigma_{(NROD=4 \text{ AND } SALARIO>4000) \text{ OR } (NROD=5 \text{ AND } SALARIO>3000)}(\text{EMPREGADO})$$

O resultado é mostrado na figura 19.

Nome	Sobre	RG	DataN	Endereço	Sexo	Salário	RG Super	NroD
Francisco	Souza	333445555	08.12.45	Rua Jorge Assef, 100, S.Paulo	M	4000	888665555	5
Joana	Pereira	987654321	20.06.31	Rua Tiradentes, 291, Rib.Preto	F	4300	888665555	4
Rodolfo	Nogueira	666884444	15.09.52	Av.São Carlos,1005, São Carlos	M	3800	333445555	5

Figura 19 - Resultado da Seleção: $\sigma_{(NROD=4 \text{ AND } SALARIO>4000) \text{ OR } (NROD=5 \text{ AND } SALARIO>3000)}$

(EMPREGADO)

Note que os operadores de comparação no conjunto {=, <, ≤, >, >, ≠} aplicam-se a atributos cujos domínios são *valores ordenados*, tais como domínios numéricos ou datas. Domínios de strings de caracteres são considerados ordenados baseados na seqüência dos caracteres. Se o domínio de um atributo é um conjunto de *valores desordenados*, então somente os operadores de comparação no conjunto {=, ≠} podem ser aplicados ao atributo. Um exemplo de um domínio desordenado é o domínio Cor={vermelho, azul, verde, branco, amarelo, ...} onde não existe qualquer ordem especificada entre as várias cores. Alguns domínios permitem tipos adicionais de operadores de comparação; por exemplo, um domínio

de strings de caracteres pode permitir o operador de comparação SUBSTRING-OF.

O grau da relação resultante de uma operação SELECT é o mesmo da relação R original na qual a operação é aplicada, pois possui os mesmos atributos de R. O número de tuplas na relação resultante é sempre menor ou igual ao número de tuplas na relação original R.

Note que a operação SELECT é **comutativa**; isto é, $\sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(R)) = \sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond1} \rangle}(R))$. Assim, uma seqüência de SELECTs podem ser aplicada em qualquer ordem. Além disso, podemos sempre combinar uma **cascata** de operações SELECT em uma única operação SELECT com uma condição conjuntiva (AND), isto é:

$$\sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\dots(\sigma_{\langle \text{condn} \rangle}(R))\dots)) = \sigma_{\langle \text{cond1} \rangle \text{ AND } \langle \text{cond2} \rangle \text{ AND } \dots \text{ AND } \langle \text{condn} \rangle}(R).$$

3.2.2.A Operação PROJECT

A operação PROJECT seleciona certas *colunas* da tabela e descarta as outras colunas.

Exemplo: Para listar o nome, o sobrenome e o salário de cada empregado, pode-se usar a operação PROJECT da seguinte forma:

$$\pi_{\text{NOME, SOBREN, SALARIO}}(\text{EMPREGADO})$$

Nome	Sobren	Salário
João	Silva	3000
Francisco	Souza	4000
Alice	Fernandes	2500
Joana	Pereira	4300
Rodolfo	Nogueira	3800

Figura 20. Resultado da operação: $\pi_{\text{NOME, SOBREN, SALARIO}}(\text{EMPREGADO})$

A forma geral da operação PROJECT é:

$$\pi_{\langle \text{lista de atributos} \rangle}(\langle \text{Nome da Relação} \rangle)$$

onde π (pi) é o símbolo usado para representar a operação PROJECT e <lista de atributos> é uma lista de atributos da relação especificada por <Nome da Relação>. A relação resultante tem somente os atributos especificados em <lista de atributos> e *na mesma ordem que aparecem na lista*. Assim, seu **grau** é igual ao número de atributos em <lista de atributos>.

A operação PROJECT *remove implicitamente qualquer tupla duplicada*, assim o resultado de uma operação PROJECT é um conjunto de tuplas e portanto uma relação válida. O número de tuplas em uma relação resultante de uma operação PROJECT é sempre menor ou igual ao número de tuplas da relação original. Se a lista de projeção inclui uma chave da relação, a relação resultante tem o mesmo número de tuplas que o original. A **comutatividade** não é mantida em PROJECT.

3.2.3. Seqüências de Operações e Renomeação de Atributos

As operações podem ser aplicadas de forma aninhada, usando uma única expressão da álgebra relacional, como a seguir, cuja expressão recupera o primeiro nome, o último nome, e o salário de todos os empregados que trabalham no departamento cujo código é 5:

$$\pi_{\text{NOME, SOBREN, SALARIO}} (\sigma_{\text{NR0D}=5} (\text{EMPREGADO}))$$

Nome	Sobren	Salário
João	Silva	3000
Francisco	Souza	4000
Rodolfo	Nogueira	3800

Figura 21 Resultado da operação: $\pi_{\text{Nome, Sobren, Salário}} (\sigma_{\text{DN0}=5} (\text{EMPREGADO}))$

ou podemos criar relações resultantes intermediárias, dando um nome para cada relação intermediária:

$$\text{DEP}_5\text{EMPS} \leftarrow \sigma_{\text{NROD}=5}(\text{EMPREGADO})$$

$$\text{RESULT} \leftarrow \pi_{\text{NOME, SOBREN, SALARIO}}(\text{DEP}_5\text{EMPS})$$

Podemos também utilizar esta técnica para renomear os atributos nas relações intermediárias e resultantes, como a seguir. Isso pode ser útil na conexão com operações mais complexas tais como UNION e JOIN.

$$\text{TEMP} \leftarrow \sigma_{\text{NROD}=5}(\text{EMPREGADO})$$

$$\text{R}(\text{Nome, sobrenome, Salário}) \leftarrow \pi_{\text{NOME, SOBREN, SALARIO}}(\text{TEMP})$$

Temp

Nome	Sobren	SSN	DataN	Endereço	Sexo	Salário	RG Super	NroD
João	Silva	123456789	09.01.55	Rua 15 Novembro, 731, São Carlos	M	3000	333445555	5
Francisco	Souza	333445555	08.12.45	Rua Jorge Assef, 100, S.Paulo	M	4000	888665555	5
Rodolfo	Nogueira	666884444	15.09.52	Av. São Carlos, 1005, São Carlos	M	3800	333445555	5

Figura 22 - Resultado da Seleção: $\text{Temp} \leftarrow \sigma_{\text{NROD}=5}(\text{EMPREGADO})$

R

Nome	Sobrenome	Salário
João	Silva	3000
Francisco	Souza	4000
Rodolfo	Nogueira	3800

Figura 23 - Resultado da Projeção: $\text{R}(\text{Nome, Sobrenome, Salário}) \leftarrow \pi_{\text{NOME, SOBREN, SALARIO}}(\text{Temp})$

3.2.4. Operações da Teoria de Conjuntos

O próximo grupo de operações da álgebra relacional são operações matemáticas padrões em conjuntos. Elas se aplicam ao modelo relacional, pois uma relação é definida como um conjunto de tuplas e pode ser usada para processar as tuplas em duas relações como conjuntos. Por exemplo, para recuperar os RGs de todos os empregados que trabalham no departamento 5 ou supervisionam diretamente um empregado que trabalha no departamento 5, podemos utilizar a operação UNION:

```

DEP5EMPS ← σNR0D=5(EMPREGADO)
RESULT1 ← πRG(DEP5EMPS)
RESULT2 ← πRGSUPER(DEP5EMPS)
RESULT ← RESULT1 ∪ RESULT2

```

A relação RESULT1 tem os RGs de todos os empregados que trabalham no departamento 5, enquanto que RESULT2 tem os RGs de todos os empregados que supervisionam diretamente um empregado que trabalha no departamento 5. A operação UNION produz as tuplas que ou estão em RESULT1 ou estão em RESULT2 ou estão em ambas.

RESULT1

RG
123456789
333445555
66884444

RESULT2

RG
333445555
888665555

RESULT

RG
123456789
333445555
666884444
888665555

Figura 24 Resultado da Operação: RESULT ← RESULT1 ∪ RESULT2

Para a utilização dessas operações em bancos de dados relacionais devemos garantir que o resultado da aplicação dessas operações em duas relações gere uma terceira relação também

válida. Para isso, as duas relações envolvidas devem possuir o mesmo tipo de tuplas; esta condição é chamada *compatibilidade de união*.

Duas relações $R(A_1, A_2, A_3, \dots, A_n)$ e $S(B_1, B_2, B_3, \dots, B_n)$ são ditas **união-compatíveis** se possuem o **mesmo grau n**, e se $\text{dom}(A_i) = \text{dom}(B_i)$ para $1 \leq i \leq n$. Isto significa que as duas relações possuem o mesmo número de atributos e que cada par de atributos correspondentes possuem o mesmo domínio.

Estudante	
N	S
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang

Instrutor	
Nome	Sobrenome
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

Figura 25 Duas relações União-Compatíveis

Podemos definir as operações UNION, INTERSECTION e DIFFERENCE em duas relações união-compatíveis R e S, como se segue:

UNION: O resultado desta operação, denotada por $R \cup S$, é uma relação que inclui todas as tuplas que estão em R ou em S ou em ambas R e S. Tuplas duplicadas são eliminadas.

N	S
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne

Figura 26 Estudante \cup Instrutor

INTERSECTION: O resultado desta operação, denotada por $R \cap S$, é uma relação que inclui todas as tuplas que estão em ambas R e S.

N	S
Susan	Yao
Ramesh	Shah

Figura 27 Estudante \cap Instrutor

Obs.: ambas as operações UNION e INTERSECTION :

- são comutativas, ou seja, $R \cup S = S \cup R$ e $R \cap S = S \cap R$
- são aplicadas a qualquer número de relações
- são associativas, ou seja, $R \cup (S \cup T) = (R \cup S) \cup T$ e $R \cap (S \cap T) = (R \cap S) \cap T$

DIFFERENCE: O resultado dessa operação, denotada por $R - S$, é uma relação que inclui todas as tuplas que estão em R mas não estão em S.

(a)	(b)																				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">N</th> <th style="width: 50%;">S</th> </tr> </thead> <tbody> <tr><td>Johnny</td><td>Kohler</td></tr> <tr><td>Barbara</td><td>Jones</td></tr> <tr><td>Amy</td><td>Ford</td></tr> <tr><td>Jimmy</td><td>Wang</td></tr> <tr><td>Ernest</td><td>Gilbert</td></tr> </tbody> </table>	N	S	Johnny	Kohler	Barbara	Jones	Amy	Ford	Jimmy	Wang	Ernest	Gilbert	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Nome</th> <th style="width: 50%;">Sobrenome</th> </tr> </thead> <tbody> <tr><td>John</td><td>Smith</td></tr> <tr><td>Ricardo</td><td>Browne</td></tr> <tr><td>Francis</td><td>Johnson</td></tr> </tbody> </table>	Nome	Sobrenome	John	Smith	Ricardo	Browne	Francis	Johnson
N	S																				
Johnny	Kohler																				
Barbara	Jones																				
Amy	Ford																				
Jimmy	Wang																				
Ernest	Gilbert																				
Nome	Sobrenome																				
John	Smith																				
Ricardo	Browne																				
Francis	Johnson																				

Figura 28 (a) ESTUDANTE - INSTRUTOR (b) INSTRUTOR - ESTUDANTE

A operação DIFFERENCE não é comutativa, pois em geral, $R-S \neq S-R$. Note que a convenção adotada é que a relação resultante tem os mesmos nomes de atributos da primeira relação R.

Produto Cartesiano (X)

Combina tuplas de duas relações R e S, resultando em uma relação que tem uma tupla para cada combinação de tuplas - uma de R e outra de S:

$$Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m) \leftarrow R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$$

Exemplo:

R1	R2												
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Nome</th> <th style="width: 50%;">Sobren</th> </tr> </thead> <tbody> <tr><td>João</td><td>Silva</td></tr> <tr><td>Francisco</td><td>Souza</td></tr> </tbody> </table>	Nome	Sobren	João	Silva	Francisco	Souza	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">NroD</th> <th style="width: 50%;">NomeD</th> </tr> </thead> <tbody> <tr><td>1</td><td>Pesquisa</td></tr> <tr><td>2</td><td>Administração</td></tr> </tbody> </table>	NroD	NomeD	1	Pesquisa	2	Administração
Nome	Sobren												
João	Silva												
Francisco	Souza												
NroD	NomeD												
1	Pesquisa												
2	Administração												

R1 X R2

Nome	Sobren	NroD	NomeD
João	Silva	1	Pesquisa
João	Silva	2	Administração
Francisco	Souza	1	Pesquisa
Francisco	Souza	2	Administração

JOIN - Operação Junção (\bowtie)

Denotada por \bowtie , é utilizada para combinar tuplas de duas relações, através de um ou mais atributos comuns às duas relações. A tabela resultante contém as colunas das duas tabelas que participaram da junção:

$$Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m) \leftarrow R(A_1, A_2, \dots, A_n) \bowtie S(B_1, B_2, \dots, B_m)$$

Esta operação é muito importante para qualquer banco de dados relacional com mais do que uma relação, pois nos permite processar relacionamento entre relações.

A forma geral de uma operação **JOIN** em 2 relações $R(A_1, A_2, \dots, A_n)$ e $S(B_1, B_2, \dots, B_m)$ é:

$$R \bowtie_{\langle \text{condição de junção} \rangle} S$$

Uma condição de junção é da forma: $\langle \text{condição} \rangle$ AND $\langle \text{condição} \rangle$ AND ... AND $\langle \text{condição} \rangle$

onde a condição é da forma $A_i \theta B_j$, A e B são atributos com o mesmo domínio, e θ é um dos operadores de comparação $\{=, <, \leq, >, >, \neq\}$.

Uma operação **JOIN** com uma condição de junção geral é chamada **THETA JOIN**. A operação de junção mais comumente usada envolve na condição de **JOIN** somente comparações de igualdade. Nesse caso ela é chamada **EQUIJOIN** (notação \bowtie). Essa operação conserva pares

de atributos com valores idênticos em cada tupla. A operação **Natural Join** (notação $*$) conserva apenas uma das colunas de valores repetidos. O exemplo a seguir usa a **EQUIJOIN**.

Exemplo: Considere as relações:

Empregado

Nome	Sobren	RG	NroD
João	Silva	123456789	5
Francisco	Souza	333445555	5
Alice	Fernandes	999887777	4
Joana	Pereira	987654321	4
Rodolfo	Nogueira	666884444	5

Departamento

NDepto	NomeD	RGGerente
1	Pesquisa	333445555
2	Administração	987654321
3	Finanças	666884444

Suponha que queiramos recuperar o nome do gerente de cada departamento. Para obter o nome do gerente, precisamos combinar cada tupla de Departamento com a tupla de Empregado cujo valor de RG seja igual ao valor de RGGerente na tupla de Departamento.

$$\text{DEPT_GER} \leftarrow \text{DEPARTMENT} \bowtie_{\text{RGGERENTE}=\text{RG}} \text{EMPLOYEE}$$

$$\text{RESULT} \leftarrow \pi_{\text{NDEPTO, NOME, SOBREN}}(\text{DEPT_GER})$$

Dept_Ger

Ndepto	NomeD	RGGerente	Nome	Sobren	RG	NroD
1	Pesquisa	333445555	Francisco	Souza	333445555	5
2	Administração	987654321	Joana	Pereira	987654321	4
3	Finanças	666884444	Rodolfo	Nogueira	666884444	5

Result

Ndepto	Nome	Sobren
1	Francisco	Souza
2	Joana	Pereira
3	Rodolfo	Nogueira

A Operação Divisão (division)

A operação **DIVISION** é útil para um tipo especial de consulta que algumas vezes ocorrem em aplicações de banco de dados. Considere o exemplo : "Recupere os números dos empregados que trabalham em todos os projetos".

Projeto

NomeP	CodProj	NroD
PROJETO A	P1	1
PROJETO B	P2	3
PROJETO C	P3	1

Trabalha

NroE	NroProj	Horas
1	P1	10
1	P3	15
2	P1	10
2	P2	20
2	P3	15
3	P2	30
4	P1	10
4	P2	10
4	P3	20

$$R1 \leftarrow \pi_{\text{CodProj}}(\text{PROJETO})$$

CodProj
P1
P2
P3

$$R2 \leftarrow \pi_{\text{NroE, NroProj}}(\text{TRABALHA})$$

NroE	NroProj
1	P1
1	P3
2	P1
2	P2
2	P3
3	P2
4	P1
4	P2
4	P3

$$\text{RESULT} \leftarrow R2 \div R1$$

NroE
2
4

Em geral a operação **DIVISION** é aplicada a duas relações $R(Z) \div S(X)$, onde $X \subseteq Z$.

Tomemos $Y=Z-X$; temos em Y o conjunto de atributos de R que não são atributos de S . O resultado da divisão é uma relação $T(Y)$ que inclui a tupla t se uma tupla t_r cujo $t_r[Y] = t$ aparece em R , com $t_r[X] = t_s$ para toda tupla t_s em S . Isso significa que, para uma tupla t aparecer no resultado T da divisão, os valores em t devem aparecer em R na combinação com toda tupla em S .

O operador DIVISION pode ser expresso como uma seqüência de operações π , \bowtie e $-$:

$$T_1 \leftarrow \pi_y(R)$$

$$T_2 \leftarrow \pi_y((S \bowtie T_1) - R)$$

$$T \leftarrow T_1 - T_2$$

Considere como exemplo as relações R e S . A divisão entre as tabelas R e S resulta em uma tabela

$$T = R \div S.$$

R

A	B
a ₁	b ₁
a ₂	b ₁
a ₃	b ₁
a ₄	b ₁
a ₁	b ₂
a ₃	b ₂
a ₂	b ₃
a ₃	b ₃
a ₄	b ₃
a ₁	b ₄
a ₂	b ₄
a ₃	b ₄

S

A
a ₁
a ₂
a ₃

T

B
b ₁
b ₄

Exercícios:

Faça a expressão na Álgebra Relacional, das seguintes consultas:

1. Busque os nomes de empregados que trabalham em todos os projetos controlados pelo departamento nro 5.
2. Faça uma lista de números de projetos para projetos que envolvem um empregado cujo sobrenome é 'Silva', ou como trabalhador ou como gerente do departamento que controla o projeto.
3. Liste o nome de todos os empregados que não possuem dependentes.
4. Liste os nomes de gerentes que possuam no mínimo um dependente.

OBS: Instancie a relação dependente cujo esquema de relação é:

DEPENDENTE(RG, Nome-Dependente, Sexo, DataNasc, Grau_Parentesco)+

4. Mapeamento entre Esquemas

Modelo Entidade-Relacionamento → Modelo Relacional

O ME-R é o mais usado para a modelagem de esquemas de bancos de dados. Isso ocorre pelo fato do modelo ser simples, possuir alto grau de capacidade de representação semântica e apresentar os dados em nível lógico. Após a modelagem ser idealizada através do ME-R, ela é “traduzida” para a representação do Modelo Relacional. Esse procedimento é adotado porque os gerenciadores no mercado são em sua maioria relacionais.

O mapeamento pode facilmente ser dividido em passos “algorítmicos”, o que possibilita uma tradução direta e normalizada dos dados.

Passo 1: para cada entidade regular E em um esquema $E-R$, cria-se uma relação R que inclui todos os atributos simples de E . Para um atributo composto inclui-se somente os componentes simples do atributo. É escolhido um dos atributos chave de E , como chave primária de R . Se a chave de E é composta, então o conjunto de atributos simples, juntos formarão a chave de R .

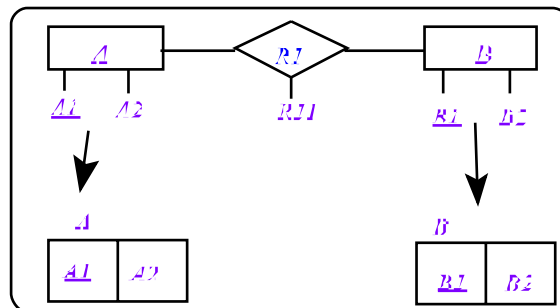


Figura 29: Mapeando entidades regulares.

Passo 2: para cada entidade fraca W no esquema $E-R$ com seu próprio tipo de entidade E , cria-se uma relação R e inclui-se todos atributos simples (ou componentes simples dos atributos compostos) de W como atributos de R . Em adição, nós incluímos como atributos de chave estrangeira de R os atributos da chave primária da relação que corresponde à entidade proprietária do tipo E ; a chave primária de R é a combinação da chave primária do proprietário e a chave

parcial da entidade fraca do tipo W .

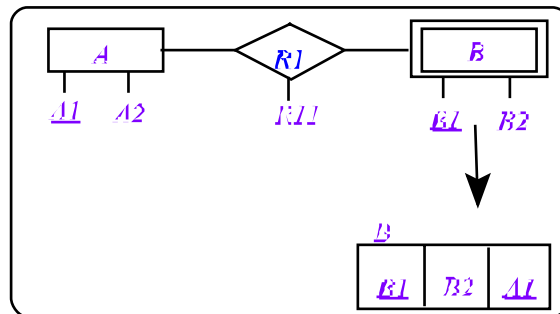


Figura 30: Mapeamento de Entidade Fraca

Passo 3: para cada tipo de relacionamento binário 1:1 R no esquema $E-R$, nós identificamos as relações S e T que correspondem aos tipos participantes. Escolhemos uma das relações, digamos S , e incluímos como chave estrangeira de S a chave primária de T . É melhor escolher uma entidade com participação total em R no papel de S . Nós incluímos todos os atributos simples (ou componentes simples de um atributo composto) do relacionamento R 1:1 como atributo de S .

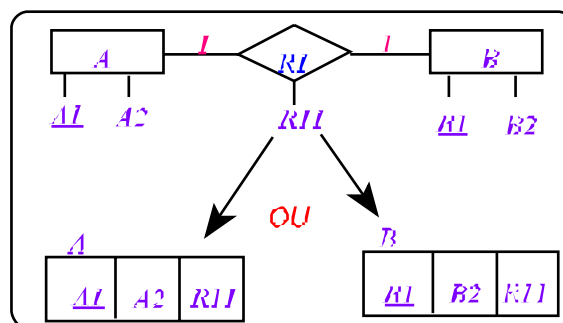


Figura 31: Mapeamento de relacionamento com cardinalidade 1 para 1.

Passo 4: para cada relacionamento R (regular) binário 1:N, nós identificamos as relações S que representam o tipo de entidade participante no Lado N do tipo de relacionamento. Nós incluímos como chave estrangeira em S a chave primária da relação T que representa o outro tipo de entidade

participante em R ; isto porque cada instância de entidade no lado N é relacionada a uma instância de entidade no lado 1 do tipo de relacionamento.

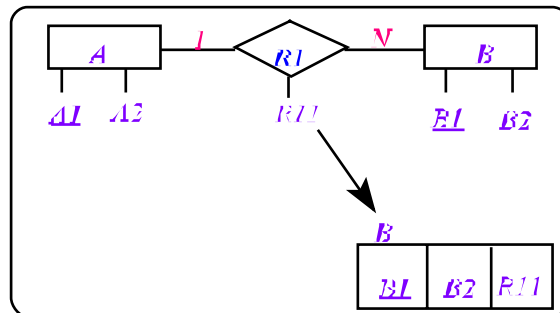


Figura 32: Mapeamento de relacionamento com cardinalidade 1 para N (ou N para 1).

Passo 5: para cada relacionamento R binário $M:N$, nós criamos uma nova relação S para representar R . Nós incluímos como atributos de chaves estrangeiras em S as chaves primárias das relações que representam os tipos de entidades participantes; sua combinação formará a chave primária de S . Nós também incluímos os atributos simples do tipo de relacionamento $M:N$ (ou os componentes simples dos atributos compostos) como atributos de S . Nós não podemos representar um relacionamento $M:N$ por um atributo chave estrangeira singular nas relações participantes, por causa da cardinalidade $M:N$.

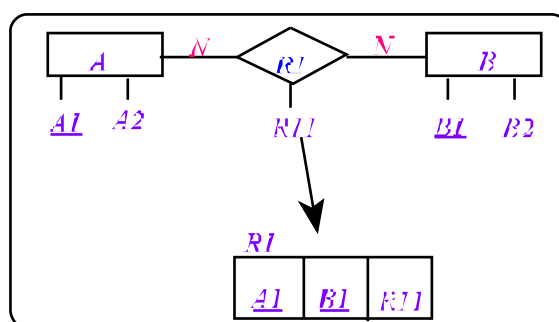


Figura 33: Mapeamento de relacionamento com cardinalidade N para N.

Passo 6: para cada atributo multivalorado A , nós criamos uma nova relação R que inclui um atributo correspondendo a A mais o atributo K da chave primária da relação que representa a entidade ou relacionamento que tem A como um atributo. A chave primária de R é então a combinação de A e K . Se o atributo multivalorado é composto, nós incluímos seus componentes simples.

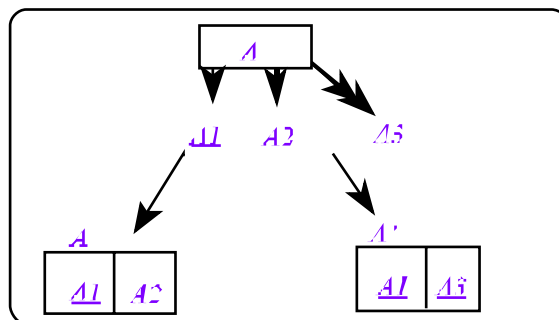


Figura 34: Mapeamento de atributo multivalorado.

Passo 7: para cada relacionamento n -ário R , $n > 2$, nós criamos uma nova relação S para representar R . Nós incluímos como atributos chave estrangeira em S as chaves primárias das relações que representam os tipos de entidades participantes. Nós também incluímos atributos simples do relacionamento n -ário (ou componentes simples dos atributos compostos) como atributos de S . A chave primária de S é usualmente uma combinação de todas as chaves estrangeiras que referenciam as relações representando os tipos de entidades participantes. No entanto, se a restrição de participação (min, max) de um dos tipos de entidades participantes E em R tem no $\max=1$, então a chave primária de S pode ser o atributo singular da chave estrangeira que referencia a relação E' correspondente a E ; isto porque neste caso cada entidade e em E participará em ao menos uma instância de relacionamento R e pode identificar univocamente aquela instância de relacionamento.

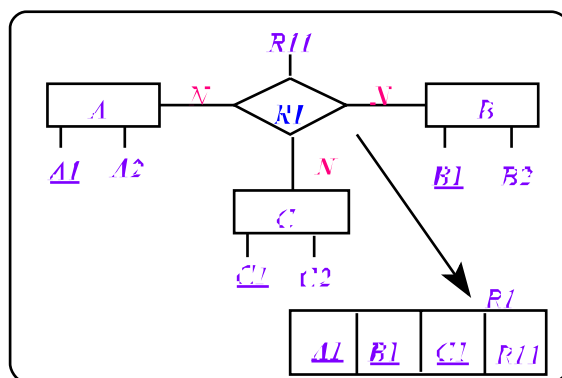


Figura 35: Mapeamento de relacionamento com grau $N > 2$.

Passo 8: converter cada especialização com m subclasses $\{S_1, S_2, \dots, S_m\}$ e superclasse C (generalizada), onde os atributos de C são $\{k, a_1, \dots, a_n\}$ e k é a chave primária, em relações esquema usando uma das quatro opções seguintes:

Opção 8A: criar uma relação L para C com atributos $Atr(L) = \{k, a_1, \dots, a_n\}$ e $PK(L) = k$. Criar uma relação L_i para cada subclasse S_i , $1 \leq i \leq m$, com os atributos $Atr(L_i) = \{k\} \cup \{\text{atributos de } S_i\}$ e $PK(L_i) = k$.

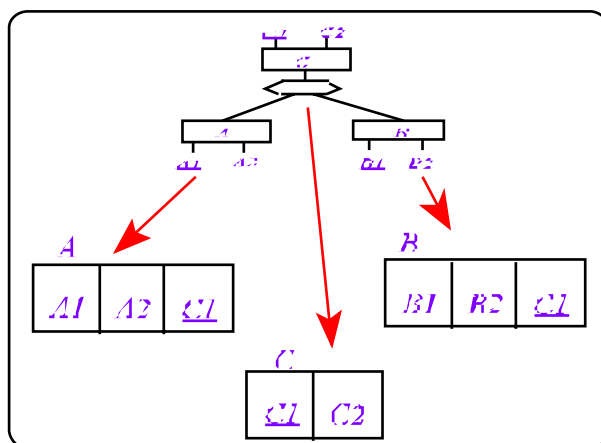


Figura 36: Mapeamento de Generalização

Opção 8B: Criar uma relação L_i para cada subclasse S_i , $1 \leq i \leq m$, com os atributos $Atr(L_i) =$

$\{k, a_1, \dots, a_n\} \cup \{\text{atributos de } S_i\}$ e $PK(L_i) = k$.

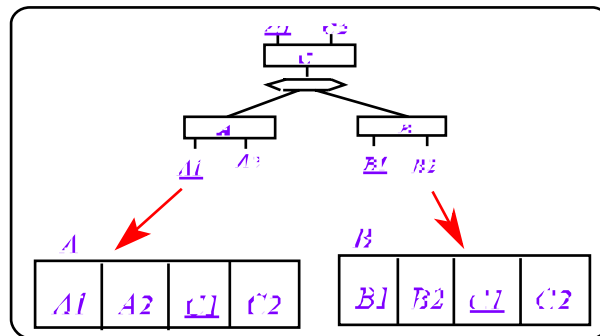


Figura 37: Mapeamento de Generalização

Opção 8C: criar uma relação única L com atributos $Atr(L) = \{k, a_1, \dots, a_n\} \cup \{\text{atributos de } S_i\} \cup \{t\}$ e $PK(L) = k$. Esta opção é para uma especialização cujas subclasses são disjuntas, e t é um tipo de atributo que indica a subclasse para a qual cada tupla pertence, se existir. Esta opção pode gerar um grande número de valores **nulos**.

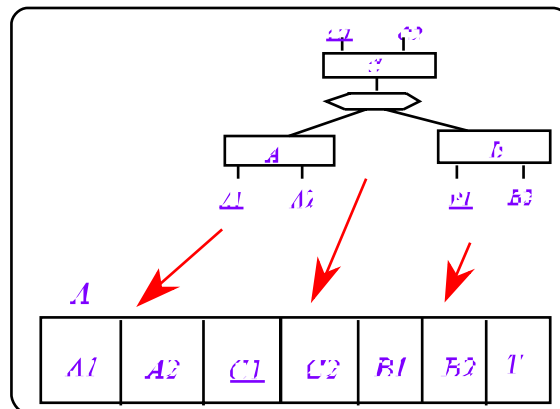


Figura 38: Mapeamento de Generalização

Opção 8D: criar uma relação única L com atributos $Atr(L) = \{k, a_1, \dots, a_n\} \cup \{\text{atributos de } S_i\} \cup \{\text{atributos de } S_m\} \cup \{t_1, \dots, t_m\}$ e $PK(L) = k$. Esta opção é para uma especialização cujas subclasses possuem sobreposição (não disjuntas), e cada t_i , $1 \leq i \leq m$, é um tipo de atributo booleano indicando se a tupla pertence a subclasse S_i .

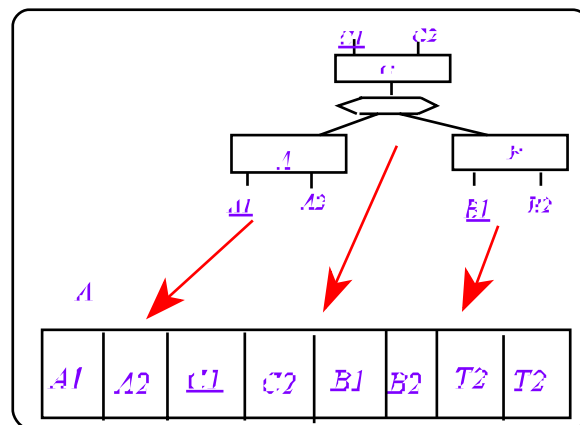


Figura 39: Mapeamento de Generalização

O resultado do mapeamento é um esquema relacional. Seguindo corretamente as regras de mapeamento, o esquema resultante pode ser “implementado” utilizando-se um SGBD Relacional. Entretanto, algumas características podem comprometer o esquema, tais como, complexidades que geram redundâncias. Na próxima seção é apresentada a Teoria da Normalização de Relações, a qual deve ser aplicada para gerar um esquema que possa ser utilizado sem problemas.

5. Normalização

Um dos objetivos principais do gerenciamento de Bancos de Dados é manter a consistência dos dados nele armazenados, e para esse fim, algumas regras precisam ser consideradas. Algumas dessas regras são garantidas pelo próprio gerenciador, tais como *unicidade da chave*, ligação entre relações através da *chave estrangeira*, etc. Outras regras são definidas nos programas de aplicação que ficam responsáveis por mantê-las. Para que essa manutenção seja válida, é necessário que as relações sejam bem fundamentadas, no sentido de evitar redundâncias que possam gerar, entre outros problemas, inconsistência de dados. Para procurar garantir esse aspecto, foi desenvolvida uma técnica chamada *Normalização*.

Para o desenvolvimento da *Teoria Formal de Normalização*, foram estabelecidas *dependências funcionais* (DF)¹, sobre as quais está fundamentada a teoria. Uma DF, é uma "formulação" de uma restrição sobre a semântica dos atributos que compõem uma relação.

Dependência Funcional: seja r uma relação sobre o esquema R , com X e Y sendo subconjuntos de R e t_1 e t_2 duas tuplas em r . A relação r satisfaz a DF $X \rightarrow Y$, se para todos valores de X , em que $t_1(X) = t_2(X)$, tem-se sempre que $t_1(Y) = t_2(Y)$.

Em uma DF $X \rightarrow Y$, X é denominado *lado esquerdo* da dependência, enquanto Y é denominado *lado direito* da dependência. Para ilustrar a definição, considere a **Figura 40**. Os valores do atributo RG são diferentes para todas as tuplas, e definem Nome e Curso. Assim, com o valor de um RG, como por exemplo, 13.245.522, define-se os valores de Curso e de Nome, que são

Nome	Curso	RG
Mário	Computação	12.122.421
Paulo	Elétrica	13.245.522
Almir	Fisioterapia	12.322.122
Marta	Computação	32.787.771
Vanía	Elétrica	14.722.098

Figura 40: Exemplo de Dependências

¹ No decorrer do texto, uma dependência funcional será escrita abreviadamente por DF.

respectivamente, Elétrica e Paulo. Dessa forma, o atributo RG é uma chave candidata e possui as dependências funcionais $RG \rightarrow Nome$ e $RG \rightarrow Curso$. Essa característica é verdadeira porque o “projetista” definiu a semântica de que um valor de RG sempre definirá um único valor para Nome e um único valor para Curso. Em outro contexto ou aplicação, o “projetista” poderia definir uma semântica diferente para esses atributos, caracterizando outras dependências funcionais.

Naturalmente, as Dependências Funcionais que envolvem uma relação são determinadas pelo projetista do banco de dados. Entretanto, foram definidos alguns axiomas de inferência, os quais são construídos sobre o conceito de DFs. Um *Axioma de inferência* é uma regra que estabelece que, se uma relação satisfaz certas DFs, ela também precisa satisfazer outras DFs, que são logicamente inferidas através dessa regra.

Para uma relação $r(R)$, em qualquer momento existe uma família de DFs F , que são verdadeiras em r . Um estado de uma relação pode satisfazer uma certa DF, enquanto outro estado não. Deseja-se então encontrar uma família de DFs F , tal que em todos estados permissíveis de r , essas DFs sejam satisfeitas.

O número de dependências funcionais que podem ser aplicadas em uma relação $r(R)$ é finito, uma vez que há apenas um número finito de subconjuntos de R . Assim, é sempre possível encontrar todas as DFs que são verdadeiras em r . Conhecendo alguns membros de F , é frequentemente possível deduzir outros membros de F . Um conjunto F de DFs, denotado por $F = X \rightarrow Y$, implica na existência da DF $X \rightarrow Y$, se toda relação que satisfaz todas as DFs em F também satisfaz $X \rightarrow Y$.

São introduzidos agora, seis Axiomas para DFs. Na declaração das regras, r é uma relação sobre R , e W, X, Y , e Z são sub-conjuntos de R .

Axiomas

F1. Reflexiva - Se $\exists X \supset Y$, então $X \rightarrow Y$.

F2. Aumentativa - Se $X \rightarrow Y$, e $\exists Z \in R$, então $XZ \rightarrow YZ$.

F3. Transitiva - Se r satisfaz $X \rightarrow Y$ e $Y \rightarrow Z$, r satisfaz $X \rightarrow Z$.

F4. Decomposição/Projetiva - Se r satisfaz $X \rightarrow YZ$, então, r satisfaz $X \rightarrow Y$ e $X \rightarrow Z$.

F5. União/Aditiva - Se r satisfaz $X \rightarrow Y$ e $X \rightarrow Z$, então, r satisfaz $X \rightarrow YZ$.

F6. Pseudotransitiva - Se r satisfaz as DFs $X \rightarrow Y$ e $WY \rightarrow Z$, então r satisfaz $WX \rightarrow Z$.

Considere como exemplo a relação r na **Figura 40**:

1. A relação r satisfaz a DF $A \rightarrow B$. Pelo Axioma F2, $AB \rightarrow A$, $AC \rightarrow B$, $AD \rightarrow B$, $ABC \rightarrow B$, $ABD \rightarrow B$, $ACD \rightarrow B$, e $ABCD \rightarrow B$.

2. A relação r satisfaz as DF $A \rightarrow B$ e $A \rightarrow C$. Pelo Axioma F5, $A \rightarrow BC$.

3. A relação r satisfaz a DF $A \rightarrow BC$. Pelo Axioma F4, $A \rightarrow B$, $A \rightarrow C$.

Considere agora a relação r' na **Figura 42**:

4. A relação r satisfaz a DF $A \rightarrow B$ e $B \rightarrow C$. Pelo Axioma F3, $A \rightarrow C$.

Deve-se notar que, através dos Axiomas F1 até F6, é possível se derivar outras regras de inferência para DFs. Por exemplo: Seja r uma relação sobre R com X e Y subconjuntos de R . O Axioma F1 afirma que r satisfaz $Y \rightarrow Y$. Aplicando o Axioma F2, tem-se que r satisfaz $XY \rightarrow Y$. Uma outra maneira de estabelecer

esta regra, é que para $\exists Z \in R$, r satisfaz $X \rightarrow Y$.

A	B	C	D
a ₁	b ₁	c ₁	d ₁
a ₂	b ₂	c ₁	d ₁
a ₁	b ₁	c ₁	d ₂
a ₃	b ₃	c ₂	d ₃

Figura 41: Relação r .

O conceito de “atributos chave”, tal como descrito em seções anteriores, é essencial para a Teoria de Normalização. Esse conceito será rerepresentado a seguir.

Chave - Dada uma relação esquema R, uma chave em R é um subconjunto de atributos K de R, tal que para qualquer possível relação $r(R)$, não há duas tuplas distintas t_1 e t_2 em r , tal que $t_1(K) = t_2(K)$, e nenhum subconjunto próprio K' de K tem esta propriedade. Em outras palavras, quando são atribuídos valores para os atributos que compõem uma *chave*, estes valores identificam univocamente uma tupla, ou seja, somente uma tupla dentro da relação.

A	B	C	D
a ₁	b ₁	c ₂	d ₁
a ₂	b ₂	c ₁	d ₁
a ₃	b ₁	c ₂	d ₂
a ₄	b ₁	c ₂	d ₃

Figura 42: Relação r' .

Em uma mesma relação podem existir vários subconjuntos de atributos atendendo à definição acima. Logo, escolhe-se um desses subconjuntos como *Chave*, o qual é denominado *Chave primária* (ou *principal*); os outros subconjuntos possíveis são denominados *Chaves Secundárias* (ou *Candidatas*) (candidatas porque também poderiam ter sido escolhidos como a *Chave Principal*). Note pela definição, que nenhum subconjunto próprio dos atributos que compõem uma *Chave* continua contendo a propriedade de identificação.

Para a Teoria de Normalização, também é muito importante o conceito de *Superchaves*. Uma *Superchave* é qualquer conjunto de atributos contendo uma *chave*, seja ela principal ou candidata. Note que *superchaves* podem conter subconjuntos próprios que ainda identificam uma tupla, pois sua definição exige apenas que contenha uma *chave* em seu conjunto de atributos. Dessa forma pode-se dizer que uma *chave* (*principal* ou *secundária*) é uma *superchave minimal*, ou seja, aquela *superchave* da qual não se pode retirar qualquer atributo e ainda manter a propriedade de identificação.

Em termos de DFs, uma *chave* para um esquema R é um subconjunto K de R, tal que para qualquer relação possível $r(R)$, a DF $K \rightarrow R$ é satisfeita, mas nenhum subconjunto próprio de K tem esta propriedade. Se K é uma *chave* para r , então não há tuplas distintas t_1 e t_2 em r , tal que t_1 e t_2 tenham os mesmos valores em K. Logo, enquanto há a DF $K \rightarrow R$, se $t_1(K) = t_2(K)$, tem-se

que ter $t_1 = t_2$, a qual é denotada $t_1(R) = t_2(R)$.

Será assumido de agora em diante que uma relação *esquema* R é composta de duas partes, S e \mathbf{K} , onde S é um conjunto de atributos e \mathbf{K} é um conjunto de chaves. O *esquema* R será escrito como $R = (S, \mathbf{K})$. Antes da definição das formas normais, alguns conceitos são necessários.

•Seja U um conjunto de atributos, cada um com um domínio associado. Um *esquema* R sobre U , de uma **base de dados relacional**, é uma coleção de relações esquema $\{R_1, R_2, \dots, R_p\}$, onde $R_i = (S_i, \mathbf{K}_i)$, $1 \leq i \leq p$, $\bigcup S_i = U$, $1 \leq i \leq p$, e $S_i \cap S_j = \emptyset$, $i \neq j$. Uma base de dados d sobre o esquema

\mathbf{R} , é uma coleção de relações $\{r_1, r_2, \dots, r_p\}$, tal que para cada relação esquema $R = (S, \mathbf{K})$ em \mathbf{R} , existe uma relação r em d tal que r é uma relação sobre S que satisfaz toda chave em \mathbf{K} .

•Dada uma relação R , um atributo A em R , e um conjunto de DFs F sobre R , um atributo A é **primo** em R com respeito a F , se A está contido em alguma chave candidata de R . De outra maneira, A é **não primo** em R .

•Dado um conjunto de DFs F e uma DF $X \rightarrow Y$, Y é **totalmente dependente** de X , se não há um subconjunto próprio X' de X , tal que $X' \rightarrow Y$.

•Dada uma relação esquema R , um subconjunto X de R , um atributo A em R , e um conjunto de DFs F , A é **transitivamente dependente** sobre X em R , se há um subconjunto Y de R com $X \rightarrow Y$, Y não determina X e $Y \rightarrow A$ sobre F e $A \notin X$.

Observação: Na literatura pode-se encontrar a definição da Segunda Forma Normal e da Terceira Forma Normal considerando apenas *chaves primárias*. Nesta apostila, as definições consideram todas as *Chaves Candidatas*. As definições considerando as *chaves candidatas* são denominadas *Formas Normais generalizadas*.

Formas Normais

Primeira Forma Normal (1FN)

Uma relação esquema R está na Primeira Forma Normal (1FN), se todos valores no $\text{dom}(A)$ são atômicos para todo atributo A em R . Isto é, os valores no domínio não são listas, ou conjuntos de valores, ou valores compostos. Um esquema de bases de dados \mathbf{R} está na 1FN, se toda relação esquema em \mathbf{R} está na 1FN.

Um valor que é atômico em uma aplicação, pode ser não atômico em outra aplicação. Um valor é não atômico se a aplicação lida com somente uma parte do valor.

Para exemplificar, considere a relação *Nascimento* apresentada na **Figura 43** (A). Se a “aplicação” possui interesse somente no *dia* e *mês*

(A)

Nome	Aniversário
Alberto	7 de junho de 1987
Luiz	12 de dezembro de 1987

(B)

Nome	Dia	Mês	Ano
Alberto	7	junho	1987
Luiz	12	dezembro	1987

Figura 43: Relação Nascimento.

que uma pessoa nasceu, a relação *Nascimento* não está na 1FN, uma vez que está lidando com apenas parte do valor do atributo *aniversário*. Para a relação *Nascimento* estar na primeira forma normal, ela deve ser quebrada como em (B). Assim, todos os valores da relação se tornam atômicos, e a relação está na 1FN. Note que para a análise da 1FN deve ser considerado todo o banco de dados, e não apenas uma relação desse banco. Assim, verifica-se a semântica de um atributo ser atômico no banco de dados como um todo, e não apenas na relação a que o atributo pertence. Ainda considerando a mesma relação, é necessário verificar se todos os atributos são monovalorados. No caso são, pois para um determinado valor para o atributo *Nome*, sempre haverá apenas um valor correspondente para o atributo *Aniversário*, determinando assim, que o atributo é monovalorado. É importante ressaltar que para o exemplo, considerou-se que não existem dois valores iguais para o atributo *Nome* na relação *Nascimento*. Em uma aplicação real, essa restrição certamente não seria verdadeira.

Segunda Forma Normal (2FN)

Uma relação esquema R está na segunda forma normal (2FN), com respeito a um conjunto de DFs F , se ela está na 1FN, e todo atributo não primo é totalmente dependente de todas as chaves candidatas de R . Um esquema de bases de dados \mathbf{R} , está na segunda forma normal com respeito a F , se toda relação esquema R em \mathbf{R} está na 2FN com respeito a F .

Considere a **Figura 44** (A) contendo as dependências funcionais $Sigla \rightarrow Número-Horas$ e $Número, Sigla \rightarrow Horário, Número-Horas$. Os atributos $Número$ e $Sigla$ juntos formam uma chave candidata. Como não há nenhuma outra possível, ela é definida como chave principal. Tem-se então

- $\{ Número, Sigla \}$ como chave principal e superchave;
- $\{ Número, Sigla, Horário \}$, $\{ Número, Sigla, Número-Horas \}$ e $\{ Número, Sigla, Horário, Número-Horas \}$ como superchave.

A DF $Sigla \rightarrow Número-Horas$ “fere” a 2FN, porque o atributo $Número-Horas$ (não primo) depende do atributo $Sigla$ (que é parte da chave), ou seja, não é dependente da chave toda. Esse fato torna possível a introdução de inconsistências. Note que como $Sigla$ define $Número-Horas$, para todo valor repetido para o atributo $Sigla$, o valor deveria ser repetido para o atributo $Número-Horas$. Entretanto, no exemplo isso não ocorre, porque para o valor $DC189$ tem-se em uma tupla o valor 3 e na outra o valor 4, o que é uma *inconsistência*. Como o atributo $Sigla$ sozinho não é chave candidata (nem principal), o gerenciador não “indicará” o erro na digitação, permitindo assim, que o banco de dados entre em estado inconsistente. A solução é a divisão da

relação em outras duas, tal como, na **Figura 44** (B) e (C). A primeira poderia ser chamada de Sigla_NH e a segunda de relação Sigla_1. Note que a DF que estava “ferindo” a 2FN está presente na relação Sigla_NH, agora sem ‘ferir’ a 2FN. As relações Sigla_NH e Sigla_1 estão na 2FN. Deve-se observar que não há perda de dependências funcionais durante o processo de normalização. Outro ponto que deve ser observado, é que a relação inicial original deixa de existir, ou seja, passam a existir apenas as duas originadas a partir dela.

(A)

Número	Sigla	Horário	Número-Horas
1	DC122	10:00	2
2	DC122	14:00	2
1	DC189	8:00	3
2	DC189	15:00	4
1	DC131	16:00	2

(B)

Sigla	Número-Horas
DC122	2
DC131	2
DC189	3

(C)

Número	Sigla	Horário
1	DC122	10:00
2	DC122	14:00
1	DC189	8:00
2	DC189	15:00
1	DC131	16:00

Figura 44: Relações normalizadas na 2FN

Terceira Forma Normal (3FN)

Uma relação esquema R está na terceira forma normal (3FN), com respeito a um conjunto de DFs F , se ela está na 1FN, e nenhum atributo não primo em R é transitivamente dependente sobre alguma chave de R . Ou seja, para toda DF $X \rightarrow A$, ou X é uma superchave, ou A é um atributo primo. Um esquema de bases de dados \mathbf{R} , está na terceira forma normal com respeito a F , se toda relação esquema R em \mathbf{R} está na 3FN com respeito a F .

Considere agora a relação ilustrada na **Figura 45** contendo as dependências funcionais $Número, Sigla \rightarrow Sala, Prédio$ e $Sala \rightarrow Prédio$. Os atributos $Número$ e $Sigla$ juntos formam uma chave candidata. Como não há nenhuma outra possível, ela é definida como chave principal. Tem-se então:

- $\{Número, Sigla\}$ como chave principal e superchave;

- {Número, Sigla, Sala},
{Número, Sigla, Prédio} e
{Número, Sigla, Sala, Prédio}
como superchaves.

A DF $Sala \rightarrow Prédio$
“fere” a 3FN, porque o
atributo *Prédio* (não primo)
depende do atributo *Sala* (que
também não é primo),

estabelecendo uma
dependência transitiva. Essa

característica possibilita a introdução de inconsistências na base de dados. Note que como *Sala* define *Prédio*, para todo valor repetido para o atributo *Sala*, o valor deveria ser repetido para o atributo *Prédio*. Entretanto, no exemplo, para *Sala* com o valor 4, existem dois valores para *Prédio*, no caso, *E1* e *C2*. Isso é uma inconsistência grave, que também não é “supervisionada” pelo gerenciador. A solução é a divisão da relação em outras duas, tal como, na **Figura 45**. Mais uma vez, note que a DF $Sala \rightarrow Prédio$ não foi perdida, e sim, passou para a relação *Sala_Prédio*, aonde não “fere” a 3FN. A relação original *Sala* deixa de existir. As relações resultantes *Sala_1* e *Sala_Prédio* estão na 3FN.

Existem outras formas normais que foram desenvolvidas, as quais progressivamente “retiram” complexidades (simplificam) das relações. Deve-se observar que o nível de normalização de uma relação é uma questão semântica, e não dos valores dos atributos que aparecem naquela relação em um dado instante.

Uma forma normal mais restrita que a 3FN foi definida como Forma Normal de Boyce-Codd.

Figura 45 mostra a decomposição de uma relação em 3FN. A relação original, denominada 'Sala', possui os seguintes dados:

Número	Sigla	Sala	Prédio
1	DC102	3	E1
2	DC102	4	E1
1	DC104	12	C2
1	DC155	4	C2
2	DC155	12	C2

Essa relação é decomposta em duas relações resultantes:

Sala_1

Número	Sigla	Sala
1	DC102	3
2	DC102	4
1	DC104	12
1	DC155	4
2	DC155	12

Sala_Prédio

Sala	Prédio
3	E1
4	E1
12	C2

Figura 45: Relações Normalizadas na 3FN

Forma Normal de Boyce-Codd (FNBC)

Uma relação esquema R está na forma normal de Boyce-Codd (FNBC), com respeito a um conjunto de DFs F , se para toda DF $X \rightarrow A$, X é uma superchave.

Considere as relações ilustradas na **Figura 46** contendo as DFs:

$Id_Propried \rightarrow Nome_Região$;

$Id_Propried, Nome_Região \rightarrow Lote$;

$Id_Propried, Nome_Região \rightarrow Área$;

$Lote, Nome_Região \rightarrow Id_Propried$;

$Lote, Nome_Região \rightarrow Área$;

$Área \rightarrow Nome_Região$.

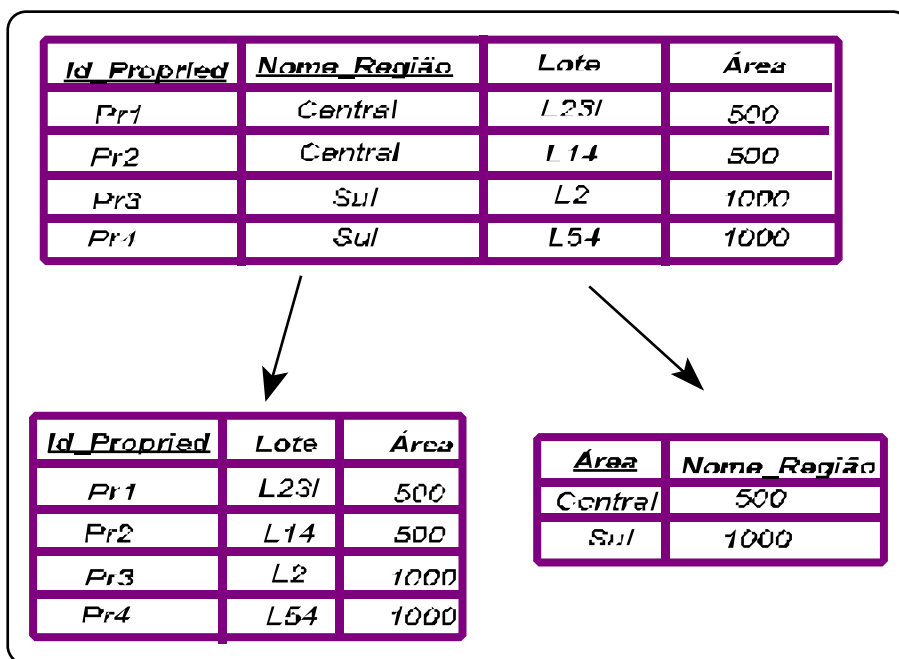


Figura 46: Exemplo de Boyce-Codd

Pelas dependências funcionais tem-se:

- {Id_Propriedade, Nome_Região} como chave candidata e superchave (obs.: também foi escolhida como chave principal);
- {Lote, Nome_Região} como chave candidata e superchave;
- {Id_Propriedade, Nome_Região, Lote}, {Id_Propriedade, Nome_Região, Área} e {Id_Propriedade, Nome_Região, Lote, Área}, {Nome_Região, Lote, Área} como superchaves.

A DF $Área \rightarrow Nome_Região$ “fere” a FNBC, pois $Área$ não é superchave. Dessa forma, a solução é a divisão das relações em outras duas, levando em consideração a DF citada. Note que nesse caso, há perda de DF, pois $Nome_Região, Lote \rightarrow Id_Propriedade$; $Nome_Região, Lote \rightarrow Área$ são perdidas. Pode-se decidir não normalizar com respeito à FNBC para manter desempenho em consultas, por exemplo. Entretanto, essa decisão deve ser devidamente documentada.

As formas normais apresentadas até aqui referem-se à utilização de DFs que envolvem atributos monovalorados. Em algumas situações, relações com atributos multivalorados exigem outra categoria de dependências que serão chamadas Dependências Funcionais Multivaloradas.

Dependência Funcional Multivalorada

Uma Dependência Funcional Multivalorada (DFM) $X \Rightarrow Y$ especificada sobre uma relação esquema R , onde X e Y são subconjuntos de R , determina que para cada valor de X , existe um conjunto possível de valores para Y .

Considere a **Figura 47**. Note que para cada valor para o atributo *Nome*, existem vários valores para o atributo *Matéria* e *Orientando*. Assim, para o valor *Alzira*, tem-se os valores *BD* e *ES* para *Matéria*, e *Paulo* e *Sônia* para *Orientando*. Pode-se afirmar

Nome	Matéria	Orientando
Carlos	SC	Márcia
Carlos	SC	Ana
Alzira	BD	Paulo
Alzira	BD	Sônia
Alzira	ES	Paulo
Alzira	ES	Sônia

Figura 47: Relação Prof_Aula_Orientação

nesse caso que tem-se as dependências multivaloradas $Nome \Rightarrow Matéria$ e $Nome \Rightarrow Orientando$.

Assim como para as Dependências Funcionais Monovaloradas, as Dependências Funcionais Multivaloradas que envolvem uma relação são determinadas pelo projetista do banco de dados. Da mesma forma, foram definidos alguns axiomas de inferência para encontrar outras DFM's inferidas. Na realidade, esses axiomas complementam os já apresentados.

Axiomas

Para os Axiomas considere que r é uma relação sobre R , e W, X, Y , e Z são sub-conjuntos de R .

1. Reflexiva - Se $X \supset Y$, então $X \rightarrow Y$.
2. Aumentativa - $\{X \rightarrow Y\}$ então $XZ \rightarrow YZ$
3. Transitiva - $\{X \rightarrow Y, Y \rightarrow Z\}$ então $X \rightarrow Z$
4. Complementação DFM's - $\{X \Rightarrow Y\}$ então $\{X \Rightarrow (R - (X \cup Y))\}$
5. Aumentativa FDM - Se $X \Rightarrow Y$ e $W \supset Z$, então $WX \Rightarrow YZ$
6. Transitiva FDM - $\{X \Rightarrow Y$ e $Y \Rightarrow Z\}$, então $X \Rightarrow (Z - Y)$
7. Replicação DFM - Se $\{X \rightarrow Y\}$ então $X \Rightarrow Y$.
8. Coalescência DFM - Se $X \Rightarrow Y$ e existe W com as propriedades que:
 - a) $W \cap Y$ é vazio; b) $W \rightarrow Z$, e c) $Y \supset Z$. Então $X \rightarrow Z$

Voltando à **Figura 47**, note que a chave da relação é formada pela composição dos atributos *Nome*, *Matéria* e *Orientando*. A incidência de dependências multivaloradas em uma relação pode trazer algumas complexidades, gerando a definição de outras duas formas normais que foram batizadas por *Quarta Forma Normal* e *Quinta Forma Normal*. Considerando ainda a **Figura 47**, se for adicionada uma nova *Matéria* para o *Carlos*, por exemplo, deverão existir duas novas tuplas combinando os valores da nova *Matéria* com os valores dos *Orientados* de *Carlos*. Essa característica estabelece que a relação deve crescer com muitos dados repetidos, exigindo

esforço para um procedimento de inserção correta e de utilização da relação, tal como, em consultas. Baseando-se nessa característica foi definida a Quarta Forma Normal.

Para definir as próximas formas normais é necessária a definição de Dependência Funcional Multivalorada Trivial, que é dada a seguir:

DFM Trivial: Uma DFM $X \Rightarrow Y$ é trivial se Y é um subconjunto de X , ou se $X \cup Y = R$, onde R é a relação envolvida.

Quarta Forma Normal (4FN)

Uma relação esquema R está na quarta forma normal (4FN), com respeito a um conjunto de DFs F , se para toda DF $X \Rightarrow A$, X for uma superchave, ou a dependência $X \Rightarrow A$ for trivial. Um esquema de bases de dados \mathbf{R} , está na quarta forma normal com respeito a F , se toda relação esquema R em \mathbf{R} está na 4FN com respeito a F .

A relação da **Figura 47** não está na 4FN. As DMF são $Nome \Rightarrow Matéria$ e $Nome \Rightarrow Orientando$. Ela não está na 4FN porque $Nome$ não é superchave e porque as duas dependências não são triviais. Para colocá-las na 4FN, a solução é dividi-la em duas relações, as quais

Prof_Mat	
Nome	Matéria
Carlos	SO
Carlos	SO
Alma	EL
Alma	EL
Alma	EL
Alma	EL
Alma	EL
Alma	EL

Prof_Orient	
Nome	Orientando
Carlos	Maria
Carlos	Ana
Alma	Paula
Alma	Sandra
Alma	Paulo
Alma	Sandra

Figura 48: Relações na 4FN.

podem ser vistas na **Figura 48**. As relações resultantes Prof_Mat e Prof_Orient estão na 4FN, pois as DFM $Nome \Rightarrow Matéria$ e $Nome \Rightarrow Orientando$ são triviais.

Existem outras formas normais, as quais não serão tratadas aqui e podem ser encontradas na bibliografia proposta.

Essa apostila traz um resumo da Teoria Básica sobre Banco de Dados, cobrindo os modelos Entidade-Relacionamento e Relacional, além de apresentar as regras de Mapeamento

entre esquemas desse modelo e a Teoria de Normalização de Esquemas Relacionais. Deve-se considerar que são necessárias leituras adicionais, as quais estão indicadas na bibliografia.

Em seguida é apresentada a bibliografia recomendada.

6. Bibliografia

Elmasri, R. e Navathe, S.B.; Fundamentals of Data Base Systems

Editora Adison Wesley, 3ª edição - 2000.

Sivlerschatz, A.; Korth, H.F. e Sudarshan, S.; Sistemas de Bancos de Dados

Editora MAKRON Books, 3ª edição - 2000.

Raghu Ramakrishman; Database Management Systems

Editora Addison Wesley, 1ª edição - 1997.