RF-379

GENERALIZATION APPLIED TO INFORMATION ORGANIZATION IN RELATIONAL MODELS

Max Cirino de Mattos, (UFMG, Minas Gerais, Brasil) – max@cmca.srv.br
Beatriz Valadares Cendón, (UFMG, Minas Gerais, Brasil) – cendon@eci.ufmg.br

Based on the fundamental ideas of the development of relational models the paper presents the principle of generalization proposed by McGee (1959) applied to the organization of information. Original and traditionally used in the processing of information, generalization offers a number of benefits to the development of systems including ease of maintenance, software reuse and reducing in costs. After the explanation of some of the advantages of the widespread application by information processing, the application of this concept to the organization of information - specifically regarding the process of data modeling - is discussed. Advantages of this application are presented, including the flexibility and stability of the model, low maintenance over time and the consequent significant reduction in costs. Further studies of this application based on concepts covered in information science are suggested, especially in relation to the organization and use of information.

Keywords: Generalization. Computer Science. Data Modeling. Information Science. Organization of Information.

O PRINCÍPIO DA GENERALIZAÇÃO APLICADO À ORGANIZAÇÃO DA INFORMAÇÃO NOS MODELOS RELACIONAIS

Retomando ideias fundamentais do desenvolvimento de modelos relacionais, o artigo apresenta o princípio da generalização proposto por McGee (1959) aplicado à organização da informação. Original e tradicionalmente utilizada no processamento de informações, a generalização oferece uma série de benefícios para o desenvolvimento de sistemas, entre eles a facilidade de manutenção, reutilização de programas e consequente redução de custos. Após a explicitação de algumas vantagens da aplicação da generalização ao processamento de informações, é discutida a aplicação desse conceito à organização da informação – especificamente em relação à modelagem de dados. São discutidas as vantagens dessa aplicação, entre elas a flexibilidade e estabilidade do modelo, a baixa manutenção ao longo do tempo e consequente redução significativa de custos. Sugere-se o aprofundamento deste estudo a partir de conceitos tratados na Ciência da Informação, principalmente em relação à organização e uso da informação.

Palavras-chave: Generalização. Modelagem de Banco de Dados. Organização da Informação. Ciência da Computação. Ciência da Informação.

1 INTRODUÇÃO

A proliferação exacerbada de discussões e modismos em torno do tema “informação” é patente na sociedade atual. Apesar da enorme ambiguidade que cerca esse termo polissêmico – e dos diversos problemas derivados dessa situação – não é objetivo deste trabalho uma discussão epistemológica.

Buscando uma aproximação prática entre as Ciências da Computação e da Informação, discute-se a proposição de uma metodologia que integra de forma pragmática conceitos e ferramentas utilizados na primeira com alguns temas tratados na segunda. Entre esses temas destaca-se a organização da informação, que será apresentada como correlata da modelagem de dados na Ciência da Computação. No nosso entendimento, tanto um como o outro se referem à estruturação, à forma como as informações estão organizadas para representar determinada realidade – ainda que a Ciência da Computação trate de situações abstratas e não necessite de um conteúdo *a priori*, enquanto parece não ser esse o caso da Ciência da Informação.

Apesar de focar a questão da organização da informação, o presente trabalho reconhece e ressalta a relação compulsória que esse tema apresenta com o processamento, a recuperação e o uso da informação. Em outras palavras, assume-se como premissa que a forma como a informação está organizada tem impacto direto no seu processamento, recuperação e uso.

Retomando algumas ideias essenciais ao desenvolvimento dos modelos relacionais na Ciência da Computação, incialmente apresenta-se o conceito original e tradicional de generalização – aplicado ao processamento de informações – sugerido por McGee (1959). Um exemplo prático é utilizado para demonstrar passo-a-passo o processo de generalização.

Em seguida discute-se o ponto fulcral deste trabalho: a proposta de aplicação do conceito de generalização à modelagem de dados. Dessa forma, pretende-se demonstrar que é possível melhorar ainda mais a estabilidade e produtividade derivadas da generalização restrita exclusivamente ao processamento de informações.

2 METODOLOGIA

Inicialmente foram estudados dois textos clássicos que tiveram um impacto muito forte para a Ciência da Computação. O primeiro propôs uma das metodologias mais utilizadas no mundo para a modelagem de dados: o Modelo Entidade-Relacionamento (CHEN, 1976). O outro é uma pedra fundamental para o desenvolvimento dos bancos de dados relacionais – o texto clássico de Edgar Frank Codd (1969) a respeito do Modelo Relacional.

Após a análise das citações usadas por esses autores – procurando-se as raízes de sua teoria que poderiam ter relação com as questões da Ciência da Informação – foi selecionado como núcleo da fundamentação teórica deste trabalho o artigo de McGee (1959), “*Generalization: Key to Successful Electronic Data Processing*”. Certamente outros autores da época desenvolveram temas e conceitos similares, porém não é objetivo deste trabalho uma revisão bibliográfica exaustiva.

A fim de ilustrar o conceito de generalização foi utilizado um conjunto de tabelas retirado de um modelo relacional criado pelo primeiro autor deste trabalho em sua prática profissional. A partir deste conjunto foram criados alguns comandos de fácil entendimento para ilustrar passo-a-passo o processo de generalização do processamento das informações. Não se pretende detalhar todo o código de programação, mas apenas mostrar as similaridades existentes em diversas partes desse código, sem a preocupação com o uso de nenhuma linguagem de programação formal.

O processo de generalização, em seguida, foi aplicado ao mesmo conjunto de tabelas, **porém em relação à estrutura das informações**, e não ao processamento. Apresenta-se o resultado criado a partir da metodologia proposta neste trabalho.

Ao final são apresentadas algumas considerações e perspectivas de novos estudos sobre os tópicos abordados.

Apenas para simplificar a terminologia adotada – sem comprometimento do conteúdo do artigo – foram usados os termos “tabela” e “atributo” para descrever o exemplo ao longo do texto. Em relação à fundamentação teórica apresentada a seguir, esses termos corresponderiam aos conceitos de “arquivo” e “campo” definidos por McGee (1959).

Propositalmente algumas definições foram deixadas em seu idioma original para que se preservasse sua originalidade.

3 FUNDAMENTAÇÃO TEÓRICA

Os modelos relacionais são importantes para o desenvolvimento deste trabalho porque representam a forma como a informação está organizada na maioria das bibliotecas digitais e em outros sistemas que utilizam os bancos de dados relacionais como estrutura básica de armazenamento.

Apesar de existirem outros modelos para o tratamento dos relacionamentos na Ciência da Computação, a escolha do Modelo Entidade-Relacionamento de Peter Chen baseia-se na premissa de que é possível uma forma flexível de tratamento da tecnologia e este Modelo é importante, pois, de acordo com o próprio Chen (2002), “*one concept such as the ER concept can be applied to many different things across a long time horizon (for more than twenty-five years) in this fast-changing Information Technology area*”. O Modelo Entidade-Relacionamento surgiu, de acordo seu autor, como uma forma alternativa e mais apropriada de representação do mundo em relação a outros modelos vigentes à época – entre eles o Modelo Relacional de Codd.

A opção pela análise do Modelo Relacional justifica-se porque Edgar Frank Codd foi agraciado com o Prêmio Turing [1] em 1981 em função do desenvolvimento dessa teoria. Este prêmio é concedido anualmente pela *Association for Computing Machinery* (ACM) para uma pessoa selecionada por contribuições duradouras e fundamentais no campo da computação – é considerado o “Prêmio Nobel” da Computação.

[1] A denominação do prêmio é uma homenagem a Alan Mathison Turing, matemático britânico considerado um dos criadores da ciência da computação moderna.

De acordo com White (2004), em 1969 Edgard Frank Codd apresentou a uma audiência restrita da IBM seu relatório de pesquisa "*Derivability, redundancy and consistency of relations stored in large data banks*". No ano seguinte, a versão revisada foi publicada como "*A relational model of data for large shared data banks*", considerado por alguns como o texto fundador do Modelo Relacional.

O artigo "*Generalization: Key to Successful Electronic Data Processing*" (MCGEE, 1959) trata da descrição de um processo chamado pelo autor de "generalização" de programação para o reaproveitamento de códigos, simplificação da implantação de sistemas e redução de custos. O estudo foi conduzido a partir de uma experiência prática no *Hanford Atomic Products Operation* (HAPO) em 1955, local de trabalho do autor à época. A partir desse processo, efetivamente foi percebida a redução drástica de custos, de tempo de implantação e manutenção dos sistemas do HAPO.

Para desenharmos o contexto da época, e a motivação do autor para a proposta de tal metodologia, a seguir apresentamos uma síntese de suas principais considerações em termos de recursos computacionais e suas implicações.

A tecnologia utilizada consistia de cartões perfurados e uma máquina processadora, que a partir da leitura desses cartões executava determinados comandos e imprimia os resultados ou gravava-os em uma fita magnética. A capacidade dos computadores permitia a realização de operações limitadas (no exemplo utilizado pelo autor, 32 operações como somar, subtrair, multiplicar, e outras), e cada computador possuía uma forma de operação própria, o que exigia uma adaptação de programas já existentes para o padrão definido na nova máquina.

Esse processo demandava muito tempo e dinheiro, um retrabalho enorme para a reprogramação e realização de novos testes – para sistemas já implantados. A capacitação dos programadores para a operação das novas máquinas também exigia muitos recursos. Essa complexidade de compreensão da forma de programação das novas máquinas é, inclusive, um dos fatores citados pelo autor para a dificuldade em compreender as necessidades informacionais – o tempo era escasso para tentar compreender bem essas necessidades uma vez que compreender o funcionamento das novas máquinas era uma prioridade e demandava muito tempo.

Apenas para ilustrar a complexidade e a limitação das máquinas da época, o autor elogia a evolução tecnológica quando aponta que "*some of these machines, like the IBM 702 and 705, even have the ability to recognize letters of the alphabet and special simbols*" (p.4). Em seguida, deixa escapar um sonho ainda atual relacionado ao processamento da linguagem natural: "*the trend toward 'anglicizing' machines is well established and will continue. However, it will be many years before the machine is marketed which will be able, literally, to converse in English with the programmer*" (p.4).

A linguagem de cada máquina era codificada, principalmente devido à baixa capacidade de armazenamento. Para evitar o trabalho dos programadores diretamente com essa linguagem – o que aumentava o tempo e custo – foram desenvolvidos "compiladores", que eram programas que permitiam a criação de

uma pseudo-linguagem que posteriormente era traduzida para a linguagem de máquina. Assim, eram criados cartões nessa pseudo-linguagem, que ao serem processados nas máquinas eram “traduzidos” em um novo conjunto de cartões na linguagem de máquina. Esses, por sua vez, eram processados junto com os arquivos de dados.

Na HAPO foram desenvolvidos dois desses compiladores, o *Scientific and Commercial Subroutine Interpreter and Program Translator* (SCRIPT) e o OMNICODE. Enquanto o primeiro permitia o uso de símbolos (que representavam uma série de comandos em linguagem de máquina), o OMNICODE permitia o uso de letras e nomes na programação. A FIG. 1 apresenta uma comparação entre as três formas de programação:

FIGURA 1: Automatic programming systems

Actual 702 Machine Code	Script Symbolic Code	Omnicode Literal Code
2 0100 Y 8483 B 0004 8 1984 0 0464 2 0902 6 0514	REC 70.00.0	Read Payroll Record
H 8496	RA 70.02.0	Take Hours
V 8506	MU 70.03.0	Mult Rate
E 0001	RO 00.01.–	
F 9372	ST 80.01.0	Is Gross Pay
⋮	⋮	⋮

In each example information punched in cards (payroll records) is to be read into the machine's memory, two quantities (hours and rate) are to be multiplied, the product is to be rounded, and the result (gross pay) is to be stored. In machine language this is accomplished with eleven instructions, SCRIPT requires five, and OMNICODE four.

Fonte: McGee (1959)

Para tentar resolver ou minimizar os problemas deste cenário de desenvolvimento de sistemas, o autor propõe a criação de uma técnica que, de forma similar ao compilador, reduza o tempo de programação. Ele parte inicialmente da ideia que existem alguns processos comuns a vários tipos de programas, como a ordenação de um arquivo. Ele afirma que esse é um processo comum a praticamente todos os programas de processamento de dados.

Usando o método tradicional de programação, em cada momento em que é necessária a ordenação de um arquivo o programador deve escrever o código e, no processamento, devem ser inseridos os respectivos cartões – o que demanda muito tempo. Além disso, a eventual mudança na forma de ordenação em determinado ponto gera a necessidade de mais tempo e dinheiro.

O autor apresenta uma forma de generalização desse procedimento, ou seja, o desenvolvimento de uma rotina única, capaz de ordenar qualquer arquivo, independente de seu conteúdo. Para que possa ser usada, essa rotina precisa receber alguns parâmetros – elementos essenciais para o processo – que são específicos de cada arquivo. Assim, uma vez implementada e testada, cada programador que necessitasse ordenar um arquivo deveria apenas realizar a chamada a essa rotina com os parâmetros específicos do arquivo a ser usado, reduzindo o tempo e o custo.

Além do processo de ordenação de um arquivo, o autor propõe também a generalização de outros procedimentos, como a manutenção do conteúdo dos arquivos e a geração de relatórios. Um dos requerimentos centrais discutidos por ele para a consecução da generalização é a centralização dos arquivos. Para compreender melhor a forma como ele desenha esses processos genéricos, são importantes algumas das definições apresentadas pelo autor, e que possuem estrita ligação com a Ciência da Informação.

O autor apresenta inicialmente as definições e relações entre **arquivo** e **registro**:

> A file may be defined as a collection of information about a group of uniquely identificable people, objects, or ideas. The collection of information about any one of these people, objects, or ideas is called a record. A file, then, may be regarded as a collection of records arranged in some specified order. (p.7)

A seguir ele define os **campos** e os **formatos de campos**:

> Each record in this file is partitioned into fields, with each field holding a different piece of data. For example, the first field contains the identification number of the employee this record pertains to; the second field contains the employee's department; and so forth. The arrangement of the fields in a record is referred to as the field format of the record. In the payroll file illustrated, the field format is the same for every record in the file; it is the data which changes from one record to the next. (p.7)

E o conceito aproximado de metadados, **registros de descrição dos formatos de campos**:

> Each field format description records describes a different field in the file record. This description consists of the name of the field, its size, and similar information. Each field is also assigned a number, by means of which reference to the field may be made on file maintenance change notices. Field format description records are supplied when a file is originally created, and, because they are physically a part of the file, need not be supplied again each time the file is maintained. (p.12)

A seguir, o autor explica a facilidade para alterar-se a estrutura do arquivo a partir desses "metadados":

> Occasionally it is necessary to add fields to a record, delete fields from a Record, or alter the size of existing fields. This can be accomplished quite easily any time the file is being maintained by supplying a special field format change card. (p.13)

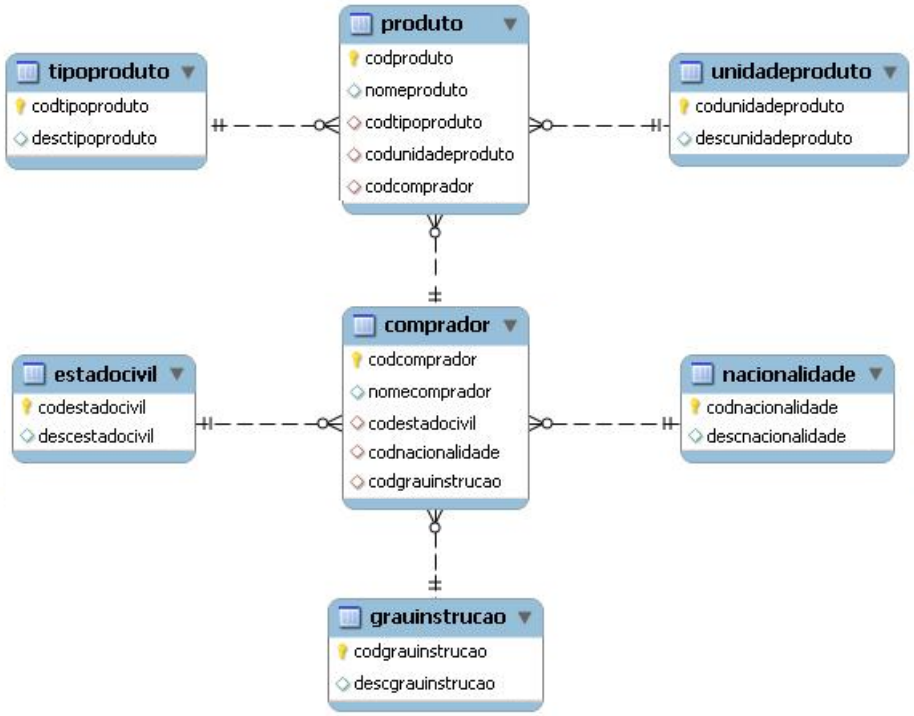
O autor destaca a importância do arquivo único como um dos requisitos essenciais para um processamento integrado. Esse arquivo deve conter toda a informação sobre determinado conteúdo. Exemplos citados são: o arquivo de pessoal, de centros de custo e organização, de equipamentos e outros. Ele aponta como vantagem do arquivo único em relação a arquivos múltiplos a inexistência de esforço de controle de redundância que eventualmente gera duplicidade e incompatibilidade de dados.

O tratamento de arquivos integrados dessa forma, entretanto, seria inviável, para ele, com as técnicas tradicionais de programação descritas no contexto do artigo. Por isso, ele justifica a adoção da generalização com uma alternativa viável para a implantação de arquivos únicos.

4 A GENERALIZAÇÃO APLICADA AO PROCESSAMENTO DE DADOS

Para ilustrar a discussão deste artigo, a FIG. 2 apresenta as tabelas que armazenam os valores possíveis para preenchimento de tipo de produto, unidade de medida de produto, estado civil, nacionalidade e grau de instrução, e as tabelas que armazenam os dados de produtos e compradores, sendo que essas últimas aparecem apenas com alguns de seus atributos.

FIGURA 2: Modelo de dados inicial



Fonte: Desenvolvida pelo primeiro autor

A tabela “comprador” está relacionada às tabelas “estadocivil”, “nacionalidade” e “grauinstrucao”. Ao armazenar os respectivos códigos – ao invés de texto livre – nos campos homônimos em “comprador”, os conteúdos ficam estruturados e padronizados, facilitando futuras pesquisas e análises. O mesmo ocorre com a tabela “produto”, que possui campos para armazenar o “tipoproduto” e a “unidadeproduto”. Ela também permite a associação de um “comprador” para o “produto” ao armazenar o “codcomprador”. A FIG. 3 apresenta alguns conteúdos fictícios para as tabelas citadas.

FIGURA 3: Conteúdo fictício das tabelas

tipoproduto	
codtipoproduto	desctipoproduto
1	Bebidas
2	Achocolatados
3	Frios
4	Carnes
5	Biscoitos

unidadeproduto	
codunidadeproduto	descunidadeproduto
1	Kg
2	litro
3	peça
4	unidade

estadocivil	
codestadocivil	descestadocivil
1	Solteiro
2	Casado
3	Viúvo
4	Divorciado

produto				
codproduto	nomeproduto	codtipoproduto	codunidadeproduto	codcomprador
1	Refrigerante	1	2	1
2	Cerveja	1	2	1
3	Queijo	3	1	2
4	Presunto	3	1	3
5	Mortadela	3	1	4

nacionalidade	
codnacionalidade	descnacionalidade
1	Brasileira
2	Americana
3	Italiana
4	Alemã

comprador				
codcomprador	nomecomprador	codestadocivil	codnacionalidade	codgrauinstrucao
1	Roberto	2	1	3
2	Natállia	1	3	3
3	Luccas	1	3	3
4	Sibelle	2	1	3
5	Angela	2	4	3

grauinstrucao	
codgrauinstrucao	descgrauinstrucao
1	Ensino fundamental
2	Ensino médio
3	Ensino superior

Fonte: Desenvolvida pelo primeiro autor

O foco da discussão sobre a generalização do processamento recai sobre as tabelas “tipoproduto”, “unidadeproduto”, “estadocivil”, “nacionalidade” e “grauinstrucao”, que serão nomeadas deste ponto em diante de *tabelas classificatórias*. Cada uma dessas tabelas possui dois atributos: um código único e uma descrição, também única.

Para gerenciar o conteúdo dessas tabelas, é preciso definir funcionalidades que permitam a inclusão, alteração e exclusão de informações em cada uma delas. A princípio, devem existir comandos para cada uma dessas situações, para cada uma das tabelas listadas. Por exemplo, deve haver uma função que receba um código de tipo de produto e exclua essa informação da tabela:

```
Função ExcluirTipoProduto ($codtipoproduto) {
        delete from tipoproduto where codtipoproduto = $codtipoproduto;
}
```

Assim, caso se execute o comando ExcluirTipoProduto(2), o tipo de produto “Achocolatados” será excluído da tabela. Da mesma forma, devem existir funções que excluam informações das outras tabelas classificatórias:

```
Função ExcluirUnidadeProduto ($codunidadeproduto) {
        delete from unidadeproduto where codunidadeproduto = $codunidadeproduto;
}

Função ExcluirEstadoCivil ($codestadocivil) {
        delete from estadocivil where codestadocivil = $codestadocivil;
}

Função ExcluirNacionalidade ($codnacionalidade) {
        delete from nacionalidade where codnacionalidade = $codnacionalidade;
}

Função ExcluirGrauInstrucao ($codgrauinstrucao) {
        delete from grauinstrucao where codgrauinstrucao = $codgrauinstrucao;
}
```

De forma análoga, devem existir comandos para incluir e alterar o conteúdo de cada uma das tabelas classificatórias. Caso o programador deseje incluir uma mensagem “Operação concluída com sucesso!” para todas as operações de inclusão, alteração ou exclusão, deverá incorporar os respectivos comandos de programação em cada função citada – o que demandará mais tempo e aumentará o custo.

A generalização do processamento de dados pode ser aplicada a este exemplo evitando que sejam criadas diversas funções de inclusão, alteração e exclusão. Para ilustrar esse procedimento, vamos trabalhar com as funções de exclusão citadas anteriormente.

É possível perceber alguma similaridade nos comandos utilizados:

delete from ***tipoproduto***	where **cod*tipoproduto***	= **$codtipoproduto**;
delete from ***unidadeproduto***	where **cod*unidadeproduto***	= **$codunidadeproduto**;
delete from ***estadocivil***	where **cod*estadocivil***	= **$codestadocivil**;
delete from ***nacionalidade***	where **cod*nacionalidade***	= **$codnacionalidade**;
delete from ***grauinstrucao***	where **cod*grauinstrucao***	= **$codgrauinstrucao**;

O comando genérico pode ser expresso da seguinte forma:

delete from **TABELA**	where **ATRIBUTO**	= **CÓDIGO**;

Nesse caso, TABELA corresponde ao nome de qualquer uma das tabelas classificatórias; ATRIBUTO é o nome do código único da TABELA; CÓDIGO é um número que representa o código único a ser excluído. Se o nome da TABELA e do ATRIBUTO forem fornecidos como parâmetros para a função, será possível criar apenas uma função para exclusão de informações de todas as tabelas classificatórias:

```
Função Excluir ($tabela, $atributo, $codigo) {
        delete from $tabela where $atributo = $codigo;
}
```

Para excluir o tipo de produto “Achocolatados”, do exemplo anterior, usar-se-ia o comando Excluir(“tipoproduto”, “codtipoproduto”, 2), o que corresponderia a “excluir da tabela tipoproduto a informação que possui o valor 2 do atributo codtipoproduto”. Nesse caso, se o programador precisasse fazer qualquer alteração, bastaria alterar a função ‘Excluir” – o que seria mais simples, rápido e barato do que a forma anterior.

No caso específico do modelo da FIG. 2 foi utilizada uma padronização para a nomenclatura dos atributos, sendo que o código único sempre é nomeado a partir da concatenação de “cod” com o nome da tabela. Por esse motivo, a partir do nome da tabela é possível deduzir-se o nome do código único: tabela “produto”, código único “codproduto”; tabela “nacionalidade”, código único “codnacionalidade”. Dessa forma é possível simplificar um pouco mais a função, eliminando-se a necessidade de passar como parâmetro o nome do código único, uma vez que o nome da tabela já é passado para a função:

```
Função Excluir ($tabela, $codigo) {
        $atributo = “cod” + $tabela;
        delete from $tabela where $atributo = $codigo;
}
```

Nesse novo formato, a exclusão do tipo de produto “Achocolatados” seria realizada a partir do comando Excluir(“tipoproduto”, 2).

Aplicando o mesmo raciocínio às funções de inclusão e alteração, com as devidas adequações dos parâmetros, podemos concluir que apenas uma função de inclusão e uma de alteração são necessárias para todas as tabelas classificatórias.

É possível deduzir-se que, dependendo do número de tabelas classificatórias existentes em um modelo, o impacto da aplicação dessa técnica é muito expressivo. A generalização, dessa forma, pode ser usada com o objetivo de facilitar a manutenção, permitir a reutilização de programas e consequentemente reduzir custos, o que corrobora as ideias apresentadas por McGee (1959) na fundamentação teórica.

5 A GENERALIZAÇÃO APLICADA À ORGANIZAÇÃO DA INFORMAÇÃO

O objetivo principal deste artigo é apresentar a aplicação da generalização à organização da informação, especificamente em relação à modelagem de dados. Para isso serão usadas as tabelas classificatórias do tópico anterior para exemplificar nossa proposta.

O foco de nossa análise é a **estrutura** das tabelas classificatórias – uma vez que estamos tratando da organização da informação – apresentada a seguir:

tipoproduto	: **cod***tipoproduto*,	**desc***tipoproduto*;
unidadeproduto	: **cod***unidadeproduto*,	**desc***unidadeproduto*;
estadocivil	: **cod***estadocivil*,	**desc***estadocivil*;
nacionalidade	: **cod***nacionalidade*,	**desc***nacionalidade*;
grauinstrucao	: **cod***grauinstrucao*,	**desc***grauinstrucao*;

Genericamente, cada tabela possui um código e uma descrição – é essa a sua estrutura básica. A pergunta essencial que motivou a idealização da metodologia proposta foi: se elas possuem a mesma estrutura, não poderiam ser representadas em uma única tabela?

Este artigo apresenta a primeira das duas soluções idealizadas para responder ao questionamento acima. Essas soluções partiram da seguinte premissa: se as tabelas classificatórias são na verdade várias ocorrências com uma mesma estrutura composta por código e descrição, é possível representar essas ocorrências em uma única tabela que possua esses atributos.

A primeira solução idealizada consiste de duas tabelas, permitindo a simples concatenação dos conteúdos de todas as tabelas classificatórias a partir da inclusão de uma referência à estrutura de origem, conforme a FIG. 4:

FIGURA 4: Solução inicial da generalização aplicada às tabelas classificatórias

estrutura	
codestrutura	descestrutura
1	tipoproduto
2	unidadeproduto
3	estadocivil
4	nacionlidade
5	grauinstrucao

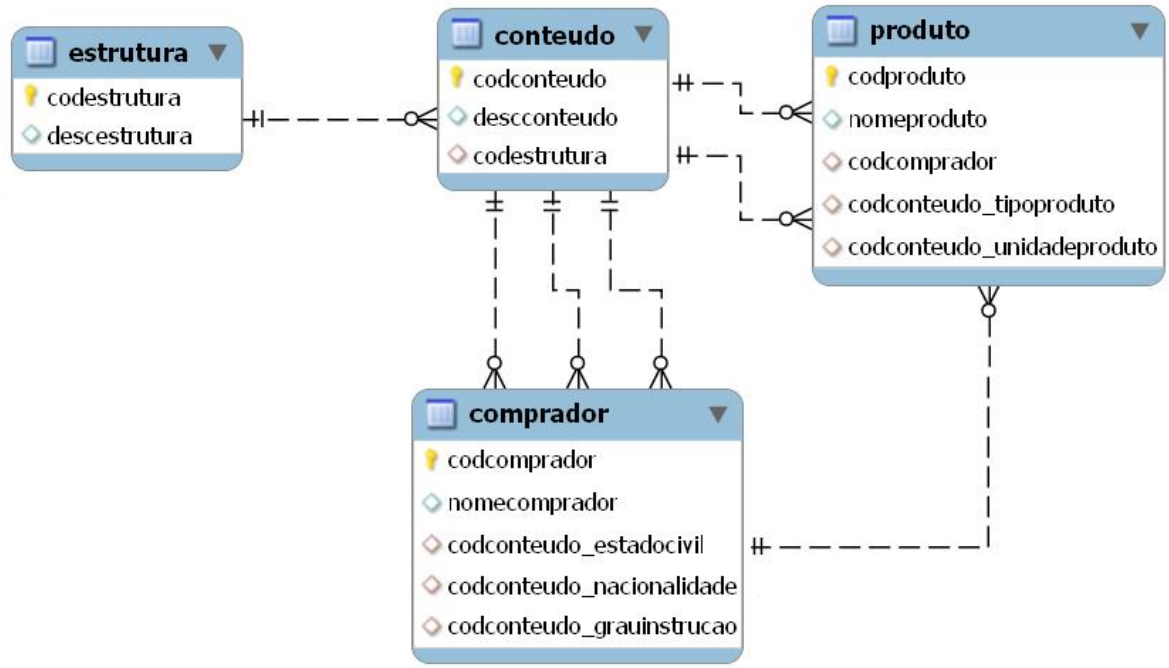
conteudo		
codconteudo	codestrutura	descconteudo
1	1	Bebidas
2	1	Achocolatados
3	1	Frios
4	1	Carnes
5	1	Biscoitos
6	2	Kg
7	2	litro
8	2	peça
9	2	unidade
10	3	Solteiro
11	3	Casado
12	3	Viúvo
13	3	Divorciado
14	4	Brasileira
15	4	Americana
16	4	Italiana
17	4	Alemã
18	5	Ensino fundamental
19	5	Ensino médio
20	5	Ensino superior

Fonte: Desenvolvida pelo primeiro autor

A tabela “estrutura” contém os nomes de todas as tabelas classificatórias existentes no modelo inicial da FIG. 2, e o código único de cada estrutura é usado na tabela “conteudo” para distinguir os conteúdos de cada estrutura.

A adaptação ao modelo inicial, para essa primeira solução, é apresentada na FIG. 5. Opcionalmente, foi realizada uma mudança na denominação dos atributos relacionados à tabela “conteudo”, que passaram a ser formados com a concatenação de “codconteudo_” com o nome da estrutura representada. Assim, o antigo “codnacionalidade” passou a se chamar “codconteudo_nacionalidade”; “codtipoproduto” passou para “codconteudo_tipoproduto”, e assim sucessivamente.

FIGURA 5: Modelo de dados revisado após a generalização



Fonte: Desenvolvida pelo primeiro autor

A FIG. 6 apresenta os conteúdos das tabelas iniciais adaptados ao novo modelo:

FIGURA 6: Conteúdo das tabelas revisado após a generalização do modelo

estrutura

codestrutura	descestrutura
1	tipoproduto
2	unidadeproduto
3	estadocivil
4	nacionlidade
5	grauinstrucao

conteudo

codconteudo	codestrutura	descconteudo
1	1	Bebidas
2	1	Achocolatados
3	1	Frios
4	1	Carnes
5	1	Biscoitos
6	2	Kg
7	2	litro
8	2	peça
9	2	unidade
10	3	Solteiro
11	3	Casado
12	3	Viúvo
13	3	Divorciado
14	4	Brasileira
15	4	Americana
16	4	Italiana
17	4	Alemã
18	5	Ensino fundamental
19	5	Ensino médio
20	5	Ensino superior

produto

codproduto	nomeproduto	codconteudo_tipoproduto	codconteudo_unidadeproduto	codcomprador
1	Refrigerante	1	7	1
2	Cervoja	1	7	1
3	Queijo	3	6	2
4	Presunto	3	6	3
5	Mortadela	3	6	4

comprador

codcomprador	nomecomprador	codconteudo_estadocivil	codconteudo_nacionalidade	codconteudo_grauinstrucao
1	Roberto	11	14	20
2	Natállia	10	16	20
3	Luccas	10	16	20
4	Sibelle	11	14	20
5	Angela	11	17	20

Fonte: Desenvolvida pelo primeiro autor

É importante ressaltar a flexibilidade e produtividade conferida ao modelo, uma vez que a criação de novas tabelas classificatórias não é necessária – e as funções de inclusão, alteração e exclusão para elas já estão prontas.

Por exemplo, para criar o atributo “proficienciaingles” que registra se o comprador tem um nível básico, intermediário ou avançado de conversação em inglês, basta adicionar o atributo “codconteudo_proficienciaingles” na tabela “comprador”. Na tabela “estrutura” é preciso apenas incluir a informação

"proficienciaingles", e na tabela "conteudo" os valores "básico", "intermediário" e "avançado". O resultado final é apresentado na FIG. 7:

FIGURA 7: Inclusão de nova estrutura no modelo

estrutura

codestrutura	descestrutura
1	tipoproduto
2	unidadeproduto
3	estadocivil
4	nacionlidade
5	grauinstrucao
6	proficienciaingles

conteudo

codconteudo	codestrutura	descconteudo
1	1	Bebidas
2	1	Achocolatados
3	1	Frios
4	1	Carnes
5	1	Biscoitos
...	...	...
21	6	Básico
22	6	Intermediário
23	6	Avançado

comprador

codcomprador	nomecomprador	...	codconteudo_proficienciaingles
1	Roberto	...	22
2	Natállia	...	23
3	Luccas	...	23
4	Sibelle	...	21
5	Angela	...	22

Fonte: Desenvolvida pelo primeiro autor

A criação das tabelas "estrutura" e "conteúdo" e das respectivas funcionalidades para o gerenciamento de suas informações (inclusão, alteração e exclusão) permite alta produtividade no desenvolvimento de sistemas, mesmo na fase inicial, quando o levantamento do conteúdo das tradicionais *tabelas classificatórias* pode ser documentado já nas tabelas reais do sistema: "estrutura" e "conteúdo". Também é possível utilizar essa ferramenta em outros sistemas, aproveitando inclusive, em alguns casos, o seu conteúdo já cadastrado – nesse sentido, além do reaproveitamento de código há também o reaproveitamento de conteúdo.

A segunda forma de solução evoluiu a partir da primeira, reforçando ainda mais o cerne do questionamento – se a tabela "estrutura" também possui os dois atributos código único e descrição, não seria possível unificar tudo isso em uma única tabela? Essa solução, entretanto, envolve uma complexidade um pouco maior em termos de modelagem de dados, e será tratada em outra oportunidade.

6 CONCLUSÕES E RECOMENDAÇÕES

A partir do exemplo utilizado neste trabalho foi demonstrada a eficiência da generalização do processamento de dados, pois ela é uma técnica que permite o reaproveitamento de código e a redução de custos, corroborando assim a teoria proposta por McGee (1959).

Quando aplicada à organização da informação, especificamente na modelagem de dados – proposta deste trabalho – a generalização permite aumentar ainda mais a produtividade e a flexibilidade do desenvolvimento e da manutenção de sistemas, além de garantir a sua estabilidade ao longo do tempo. Esses fatores sem dúvida representam uma significativa redução de custos, proporcional ao número de tabelas classificatórias em um determinado sistema.

Uma vez que a metodologia proposta neste artigo consiste de uma integração superficial de conceitos das Ciências da Computação e da Informação,

sugere-se o aprofundamento do estudo dessa aplicação em relação às teorias de organização e uso da informação.

REFERÊNCIAS

CHEN, Peter. Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned. In: **Software Pioneers**: Contributions to Software Engineering, Broy M. and Denert, E. (eds.), Springer-Verlag, Berlin, Lecturing Notes in Computer Sciences, June 2002, pp. 100-114

CHEN, Peter. The Entity-Relationship Model-Toward a Unified View of Data. **ACM Transactions on Database Systems**, Vol. 1, No. 1. March 1976, Pages 9-36.

CODD, Edgar Frank. A relational model of data for large shared data banks. **Commun. ACM [S.I.]**, v. 13, n. 6, p. 377-387, 1970.

CODD, Edgar Frank. **Derivability, redundancy and consistency of relations stored in large data banks**. IBM Research Report, 1969.

MCGEE, William. C. Generalization: Key to Successful Electronic Data Processing. **J. ACM [S.I.]**, v. 6, n. 1, p. 1-23, 1959.

WHITE, Colin. In the Beginning*: An RDBMS History*. **Teradata Magazine Online**. September 2004 edition.