PS-386

THE REDUNDANCY IMPLIED IN THE TRADITIONAL RELATIONAL MODEL: A BRIEF INTRODUCTION TO ESSENTIAL MODELING

Max Cirino de Mattos (UFMG, Minas Gerais, Brasil) – max@cmca.srv.br

This paper discusses the possibility of redundancy implied of information present in several models created from traditional techniques of computer science. The classic texts of some of the main theories on relational models are revisited and some essential points to avoid this kind of redundancy are analyzed. The sequence of presentation of the theoretical foundation purposely follows a reverse chronological order because the most important point is to clarify the understanding of some of the basic ideas to create these models. It presents the Entity-Relationship Model (1975), the Relational Model (1969), the issue of data independence in relation to the computational structure (1967) and the concept of generalization (1959). According to essential modeling suggested some situations created by traditional modeling are analyzed, and it seems like there was not an analysis of the essence of the objects to be represented in the model - then the redundancy implied appears. Further studies based on concepts covered in Philosophy on the essence of things, and in Information Science especially in relation to the organization and use of information are suggested.

Keywords: Essential Modeling. Relational Model. Entity-Relationship Model. Modeling Database. Organization of Information.

A REDUNDÂNCIA IMPLÍCITA NA MODELAGEM RELACIONAL TRADICIONAL: UMA BREVE INTRODUÇÃO À MODELAGEM ESSENCIAL

Este artigo discute a possibilidade de ocorrência da redundância implícita da informação presente em diversos modelos criados a partir das técnicas tradicionais da Ciência da Computação. Os textos fundamentais de algumas das principais teorias sobre os modelos relacionais são revisitados, e alguns pontos essenciais para se evitar essa redundância são analisados. A sequência de apresentação da fundamentação teórica segue propositalmente uma ordem cronológica invertida, pois o ponto mais importante a se explicitar é a compreensão de alguns dos fundamentos da criação desses modelos. Dessa forma, serão apresentados o Modelo Entidade-Relacionamento (1975), o Modelo Relacional (1969), a questão da independência dos dados em relação à estrutura computacional (1967) e o conceito de generalização (1959). De acordo com a modelagem essencial proposta, são analisadas algumas situações criadas a partir do uso da modelagem tradicional – e defende-se que não houve uma análise da *essência* dos objetos a serem representados no modelo, gerando então a redundância implícita. Sugere-se o aprofundamento desse estudo a partir de conceitos tratados na Filosofia sobre a essência das coisas, e na Ciência da Informação principalmente em relação à organização da informação.

Palavras-chave: Modelagem Essencial. Modelo Relacional. Modelo Entidade-Relacionamento. Modelagem de Banco de Dados. Organização da Informação.

1 INTRODUÇÃO

A proliferação exarcebada 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.

Independente da forma como se nomeia a questão, fato é que essa sociedade precisa utilizar informações e quando se trata de sistemas de informação (automatizados ou não) a duplicidade e incompatibilidade de dados é um problema real.

Este artigo discute a possibilidade de ocorrência da **redundância implícita** da informação presente em diversos modelos criados a partir do uso das técnicas tradicionais da Ciência da Computação. Os textos fundamentais de algumas das principais teorias sobre os modelos relacionais são revisitados, e alguns pontos essenciais para se evitar essa redundância são analisados.

Dessa forma, discute-se aqui a proposição de uma metodologia que integra de forma pragmática conceitos e ferramentas utilizados na Ciência da Computação com temas tratados na Ciência da Informação. 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 – e pretende-se explicitá-la – 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, este artigo apresenta algumas considerações críticas ao processo de modelagem de dados que estão relacionadas à ocorrência de redundância implícita de informações no modelo gerado.

Pretende-se demonstrar que o problema de redundância – ainda que implícita no modelo de dados – afeta o processamento, a recuperação e consequentemente, o uso da informação. As considerações apresentadas aplicam-se não somente a sistemas automatizados, mas a qualquer forma de estruturação ou organização de arquivos e informações.

2 METODOLOGIA

A sequência de apresentação da fundamentação teórica deste artigo seguiu propositalmente uma ordem cronológica invertida, pois o ponto mais importante que se pretende demonstrar é a compreensão de alguns detalhes essenciais do processo de criação desses modelos.

Dois textos iniciais 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 na Ciência da Computação: o Modelo Entidade-Relacionamento (CHEN, 1976). Outra pedra fundamental para o desenvolvimento dos bancos de dados relacionais atuais é o texto clássico de Edgar Frank Codd (1969).

Também foram analisadas algumas citações usadas por esses autores, procurando-se as raízes de sua teoria que poderiam ter relação com as questões da Ciência da Informação. Por esse motivo, após a apresentação do Modelo Relacional (CODD, 1969) – de acordo com a sequência cronológica invertida proposta – realizar-se-á a discussão dos textos de Mealy (1967) e McGee (1959), referências bibliográficas citadas no trabalho inicial sobre o Modelo Relacional. Certamente outros autores da época desenvolveram temas e conceitos similares, porém não é objetivo deste trabalho uma revisão bibliográfica exaustiva. Propositalmente algumas definições foram deixadas em seu idioma original para que se preservasse sua originalidade.

Inicialmente serão destacados alguns pontos essenciais desses textos, que serão ilustrados com a apresentação de um modelo criado pelo autor deste trabalho em sua prática profissional. O modelo foi adaptado para se tornar mais simples – alguns atributos foram excluídos – porém suficiente para os propósitos aqui propostos. Esse modelo foi criado a partir da metodologia proposta neste trabalho: a modelagem essencial.

Ao final são apresentadas algumas considerações sobre os tópicos abordados.

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.

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.

A seguir, são apresentados o Modelo Entidade-Relacionamento (CHEN, 1975), o Modelo Relacional (CODD, 1969), a questão da independência dos dados em relação à estrutura computacional (MEALY, 1967) e o conceito de generalização (MCGEE, 1959).

3.1 O MODELO ENTIDADE-RELACIONAMENTO DE PETER CHEN

De acordo com o próprio Chen (2002), "*the first ER paper [CHEN, 1976] was first presented at 1st International Conference on Very Large Databases in 1975 and subsequently published in the first issue of ACM Transactions on Database Systems in March of 1976*". O referido artigo oferece todo o arcabouço conceitual necessário ao desenvolvimento desta seção, descrevendo os conceitos essenciais do Modelo Entidade-Relacionamento, que são apresentados a seguir.

O Modelo Entidade-Relacionamento (MER) foi desenvolvido em uma época em que três outros modelos eram usados no meio acadêmico e comercial, e que, de acordo com Chen (1976), apresentavam algumas limitações:

> The **network model** provides a more natural view of data by separating entities and relationships (to a certain extent), but its capability to achieve data independence has been challenged[2]. The **relational model** is based on relational theory and can achieve a high degree of data independence, but it may lose some important semantic information about the real world[3,4,5]. The **entity set model**, which is based on set theory, also achieves a high degree of data independence, but its viewing of values such as "3" or "red" may not be natural to some people[6]. (p. 9, grifo nosso)

Para o autor, o MER apresenta características mais apropriadas para representar o mundo, inclusive como base unificadora dos três modelos citados a partir da qual eles poderiam ser derivados.

O Modelo Entidade-Relacionamento apresenta alguns conceitos básicos explicados a partir do exemplo original apresentado por Chen (2002):

---

2 CODD, E.F. A relational model of data for large shared data banks. Comm. ACM 13, 6 (June 1970), 377-387.

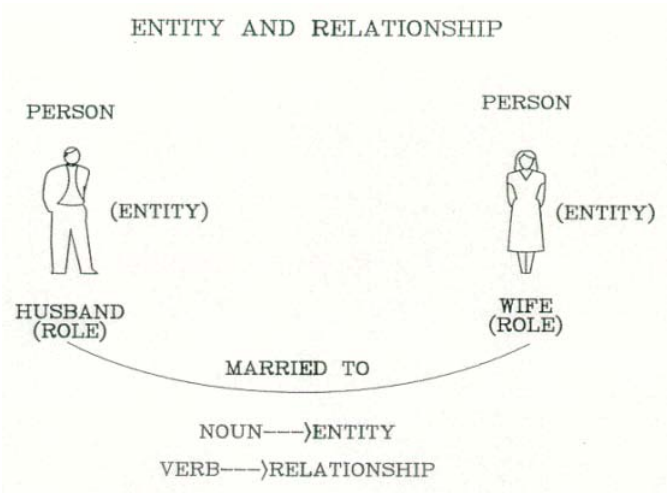
3 DEHENEFFE, C., HENNEBERT, H., AND PAULUS, W. Relational model for data base. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 1022-1025.

4 HAINAUT, J.L., AND LECHARLIER, B. An extensible semantic model of data base and its data language. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 1026-1030

5 SCHMID, H.A., AND SWENSON, J.R. On the semantics of the relational model. Proc. ACM-SIGMOD 1975, Conference, San Jose, Calif., May 1975, pp. 211-233.

6 SENKO, M.E., ALTMAN, E.B., ASTRAHAN, M.M., AND FEHDER, P.L. Data structures and accessing in data-base systems. IBM Syst. J. 12, 1 (1973), 30-93.

FIGURA 1: The Concept of Entity and Relationship



Fonte: Chen (2002, FIG. 1)

A FIG. 1 apresenta duas **entidades**, ambas do **tipo** "Pessoa". Existe um **relacionamento** entre essas duas entidades – chamado de "é-casado-com", e cada uma das entidades assume um **papel** nesse relacionamento: marido e mulher. Os **atributos** podem ser definidos como as características de uma entidade ou relacionamento.

Ao definir um relacionamento, Chen (1976) faz uma observação importante:

> **It is possible that some people may view something (e.g. marriage) as an entity while other people may view it as a relationship**. We think that this is a decision which has to be made by the enterprise administrator[7]. He should define what are entities and what are relationships so that the distinction is suitable for his environment. (p. 10, footnote 1, grifo nosso)

O autor destaca que os relacionamentos também podem conter atributos: "o conceito de atributo de relacionamento é importante para a compreensão da semântica dos dados e para a determinação das dependências funcionais entre os dados" (CHEN, 1976, p. 12). Ele também ressalta que é importante separar as informações sobre as entidades daquelas referentes aos relacionamentos, porque "esta separação é útil na identificação de dependências funcionais entre os dados". (CHEN, 1976, p. 12-3)

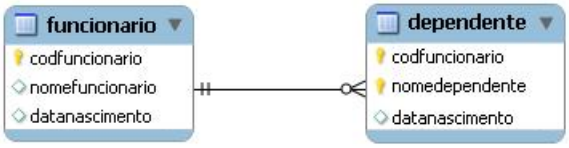
Chen (1976, p. 14) explica que as entidades, relacionamentos e valores descritos no primeiro nível de análise são objetos conceituais, e que o segundo nível cria as representações desses objetos. Outro conceito importante do Modelo é definido pelo autor como **chave primária**, que é um atributo (ou grupo de atributos) que identifica exclusivamente cada entidade.

Para exemplificar o uso da chave primária em um relacionamento, Chen (1976) apresenta o exemplo de dependentes de empregados de uma empresa. A FIG. 2 ilustra esse exemplo, usando o conjunto de entidades "Dependente" – para

[7] STEEL, T.B. Data base standardization - a status report. Proc. ACM-SIGMOD 1975, Conference, San Jose, Calif., May 1975, pp. 65-78.

representar os dependentes de funcionários de uma empresa qualquer – e "Funcionário":

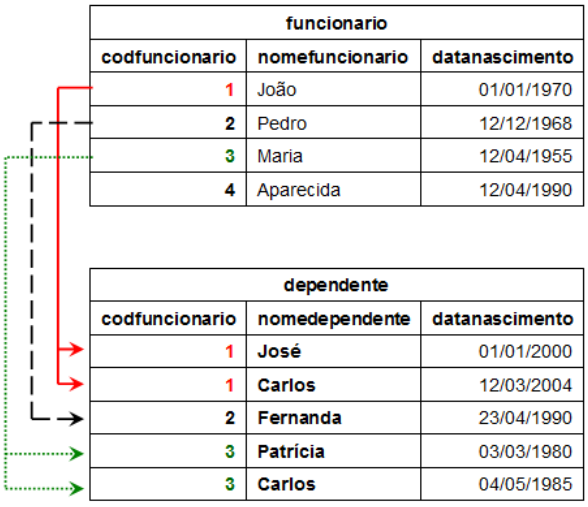
FIGURA 2: A relação explicitada entre dependentes e funcionários



Fonte: Desenvolvida pelo autor

A explicação de Chen apresenta o conjunto de entidades "Funcionário" com uma chave primária numérica "codfuncionario". É importante ressaltar que ele não explicita essa chave, afirmando apenas que ela será usada no conjunto de entidades "Dependente" para estabelecer a conexão entre seus conteúdos, mas ela foi criada nesse exemplo seguindo a orientação do próprio autor de que deve haver uma forma de identificação única para as entidades, mesmo que criada artificialmente. Para o conjunto de entidades "Dependente", o autor afirma que a chave é o "nomedependente" ("*dependents are identified by their names*") em conjunto com o "codfuncionario", o que estabelece o relacionamento entre os dois conjuntos de entidades. O que ele explicita é que, isoladamente, os diversos dependentes cadastrados não significam nada; somente quando é estabelecida a relação com um funcionário é que passam a fazer sentido, tornando-se um "dependente de um funcionário específico".

FIGURA 3: Conteúdos e relacionamentos entre dependentes e funcionários



funcionario		
codfuncionario	**nomefuncionario**	**datanascimento**
1	João	01/01/1970
2	Pedro	12/12/1968
3	Maria	12/04/1955
4	Aparecida	12/04/1990

dependente		
codfuncionario	**nomedependente**	**datanascimento**
1	**José**	01/01/2000
1	**Carlos**	12/03/2004
2	**Fernanda**	23/04/1990
3	**Patrícia**	03/03/1980
3	**Carlos**	04/05/1985

Fonte: Desenvolvida pelo autor

A partir da observação do atributo "codfuncionario" nos dois conjuntos de entidades é possível estabelecer-se a relação entre o funcionário João e seus dependentes, José e Carlos; o funcionário Pedro e sua dependente Fernanda; a funcionária Maria e seus dependentes Patrícia e Carlos. A funcionária Aparecida não possui dependentes. Cada dependente está ligado a um funcionário somente.

Esse tipo de relacionamento é definido por Chen (1976) como 1:N:

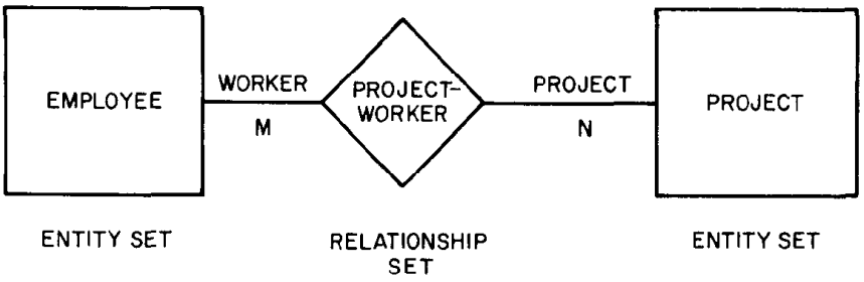
> For simplicity, we shall restrict ourselves to the use of only one kind of relationship: the binary relationships with 1:n mapping in which the existence

> of the n entities on one side of the relationship depends on the existence of one entity on the other side of the relationship. For example, one employee may have n (= 0, 1, 2, ...) dependents, and the existence of the dependents depends on the existence of the corresponding employee. This method of identification of entities by relationships with other entities can be applied recursively until the entities which can be identified by their own attribute values are reached. For example, the primary key of a department in a company may consist of the department number and the primary key of the division, which in turn consists of the division number and the name of the company. (p. 18)

O autor apresenta dois tipos de entidades: fracas e regulares. O ponto central para essa distinção é a necessidade ou não de relacionamentos para que as entidades sejam identificadas. No exemplo anterior, “Funcionário” existe de forma independente, e pode ser classificada como regular; “Dependente” somente tem sentido a partir de uma informação externa do relacionamento com “Funcionário”, e por esse motivo é classificada como “fraca”. A classificação de entidades fracas ou regulares é útil na manutenção da integridade das informações, de acordo com o autor. No exemplo anterior, se um funcionário deixasse a empresa, seus dependentes também seriam excluídos.

Outro tipo de relacionamento definido por Chen (1976) é chamado de M:N, e foi tratado pelo autor a partir da FIG. 4:

FIGURA 4: Um exemplo de relacionamento M:N de Peter Chen



Fonte: Chen (1976, FIG. 10, p.19)

Para este exemplo, ele usou uma situação em que cada entidade do conjunto “Funcionário” pode trabalhar em vários projetos, mas simultaneamente cada entidade de “Projeto” pode possuir vários funcionários. Quando ocorre esse tipo de relacionamento de “vários para vários”, o autor define o relacionamento M:N: “*The relationship set PROJECT-WORKER is an m:n mapping, that is, each project may have zero, one, or more employees assigned to it and each employee may be assigned to zero, one, or more projects*”. (p. 20)

Para a criação de bancos de dados usando o Modelo Entidade-Relacionamento, o autor propõe quatro passos:

a) Identificar os conjuntos de entidades e relacionamentos entre os conjuntos de interesse;

b) Identificar a informação semântica nos relacionamentos, como quando um determinado relacionamento é um mapeamento 1:N;

c) Definir os conjuntos de valores e atributos;

d) Organizar os dados nas entidades e seus relacionamentos e decidir as chaves primárias.

De acordo com Chen (2002), além de contribuir para o desenvolvimento de padrões, metodologias e ferramentas da computação, o Modelo Entidade-Relacionamento também apresenta uma relação direta com o processo de *datamining*:

> In our view, it [datamining] is a discovery of "hidden relationships" between data entities. The relationships exist already, and we need to discover them and then take advantage of them. This is different from conventional database design in which the database designers identify the relationships. In data mining, algorithms instead of humans are used to discover the hidden relationships.

O autor ainda expõe algumas descobertas relacionadas ao Modelo Entidade-Relacionamento e sua influência sobre o desenvolvimento dos padrões da recomendação XML – *eXtensible Markup Language* – do *World Wide Web Consortium* (W3C), entre elas os diversos artigos publicados sobre o Modelo Entidade-Relacionamento e o padrão RDF (*Resource Definition Framework*) para a descrição de metadados na internet e a citação explícita de que *"... RDF can be viewed as a member of the Entity-Relationship model family...*".

3.2 O MODELO RELACIONAL DE EDGAR FRANK CODD

Chen (2002) apresenta o seguinte comentário sobre o Modelo Relacional:

> In 1970, the relational model was proposed, and it generated considerable interest in the academic community. It is correct to say that in the early 70's, most people in the academic world worked on relational model instead of other models. One of the main reasons is that many professors had a difficult time to understand the long and dry manuals of commercial database management systems, and Codd's relational model paper was written in a much more concise and scientific style. For his contributions in the development of the relational model, Codd received ACM Turing Award in 1981.

O suporte matemático e lógico do Modelo Relacional foi muito influenciado pela teoria dos conjuntos, e de acordo com Codd (1969, p.1) "o termo relação é usado aqui no sentido matemático". No ano seguinte, Codd (1970) explicita ainda mais essa influência e seu caráter inovador ao afirmar que

> this paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs[8], the principal application of relations to data systems has been the deductive question-answering systems. (p.377)

Childs (1968b), destacado por Codd como um texto pioneiro sobre a aplicação da teoria dos conjuntos para a forma de armazenamento de dados, apresenta o conceito de "complexo", "que apresenta um recurso adicional que permite uma extensão natural das propriedades das relações binárias às propriedades das relações gerais". Seu artigo apresenta uma série de 54 definições e 15 teoremas com as respectivas provas matemáticas, além de diversos exemplos práticos. Entretanto, o próprio autor afirma que "estas são apenas algumas

[8] CHILDS, David L. Feasibility of a set-theoretic data structure: a general structure based on a reconstituted definition of relation. Proc. IFIP Cong., 1968, North Holland Pub. Co., Amsterdam, p. 162-172

das possíveis definições para complexos; construções mais imaginativas podem ser desenvolvidas em caso de necessidade". (p.31, tradução do autor)

O foco de Childs (1968a, b) relacionava-se à independência da representação dos dados em relação à estrutura computacional utilizada:

> The overall goal, of which this paper is a part, is the development of a machine-independent data structure allowing rapid processing of data related by arbitrary assignment such as: the contents of a telephone book, library files, census reports, family lineage, networks, etc. data which are non-intrinsically related have to be expressed (stored) in such a way as to define the way in which they are related before any data structure is applicable. Since any relation can be expressed in set theory as a set of ordered pairs and since set theory provides a wealth of operations for dealing with relations, a set-theoretic data structure appears worth investigation. (1968b, p.1)

Importantes contribuições de Childs (1968a) que influenciaram Codd estão relacionadas à ideia central de um STDS – *Set-Theoretic Data Structure* – que, de acordo com o autor,

> is a storage representation of sets and set operations such that: given any family of sets and any collection S of set operations an STDS is any storage representation which is isomorphic to n with S. The language used with an STDS may contain any set-theoretic expression capable of construction from n and S. Every stored representation of a set must preserve all the properties of that set and every representation of a particular set must behave identically under set operations. (p.557)
>
> The purpose of an STDS is to provide a storage representation for arbitrarily related data allowing quick access, minimal storage, generality, and extreme flexibility. With the definition of a complex, a predefined well-ordering, and the operations of set theory, such a storage representation can be realized. (conclusion)

Apesar de a construção teórica proposta por Childs (1968b) apresentar-se em grande parte a partir de representações matemáticas, procurar-se-á trabalhar apenas com os conceitos centrais relacionados à Ciência da Informação, de forma discursiva, sem a discussão de modelos lógicos matemáticos. Nesse sentido, Codd (1969) destaca a importância da compreensão da proposta do modelo, independentemente de sua representação matemática.

Preocupado com a consistência e o controle da redundância nos grandes bancos de dados, Codd (1969) propôs o Modelo Relacional como uma forma de organização e gerenciamento de grandes volumes de dados:

> The large, integrated data banks of the future will contain many relations of various degrees in stored form. It will not be unusual for this set of stored relations to be redundant. Two types of redundancy are defined and discussed. One type may be employed to improve accessibility of certain kinds of information which happen to be in great demand. When either type of redundancy exists, those responsible for control of the data bank should know about it and have some means of detecting any 'logical' inconsistencies in the total set of stored relations. Consistency checking might be helpful in tracking down unauthorized (and possibly fraudulent) changes in the data bank contents. (abstract)

Apesar do desenvolvimento de estruturas de tabelas, que auxiliavam na questão da independência dos dados[9], Codd (1970) afirma que ainda restavam obstáculos a serem vencidos, e os três principais estavam relacionados basicamente a problemas de armazenamento dos dados e restringiam as possibilidades de ordenar, indexar e acessar o conjunto de dados independentemente da forma como eles estavam armazenados nos computadores.

Codd (1969) afirma que sua proposta "fornece uma base para uma linguagem de alto nível de recuperação de dados que permitirá obter a independência máxima entre os programas, por um lado, e apresentação e organização de dados nos computadores, de outro" (p.1, tradução do autor). Ele também afirma (1970) que ela "fornece um meio de descrever os dados com a sua estrutura natural apenas - isto é, sem sobrepor qualquer estrutura adicional para efeitos de representação no computador" (p.377, tradução do autor).

Para Codd (1969), a adoção do Modelo Relacional poderia permitir o desenvolvimento de uma linguagem universal de recuperação de dados, resolvendo assim as barreiras de dependência em relação à forma de armazenamento dos mesmos.

Algum tempo após a apresentação do Modelo Relacional, Donald D. Chamberlin e Raymond Boyce, pesquisadores da IBM, apresentavam sua proposta para uma linguagem para bancos de dados relacionais baseada na álgebra relacional: o SEQUEL (CHAMBERLIN; BOYCE, 1974). Essa linguagem utilizava os conceitos de DDL (*Data Definition Language*) e DML (*Data Manipulation Language*), que serão discutidos no próximo tópico, e serviu de base para a linguagem SQL[10], hoje padrão dominante do mercado de bancos de dados relacionais.

3.3 A INDEPENDÊNCIA DOS DADOS EM RELAÇÃO À ESTRUTURA COMPUTACIONAL

Mealy (1967) baseia-se em um estudo sobre informações genealógicas[11] para analisar a natureza da teoria das relações e propor alguns conceitos básicos importantes:

> Our plan of attack is to indicate the nature of the theory of relations, based on the example of genealogical data. This will lead immediately to formulation of our notions about data in general, including rather precise definitions of concepts such as data structure, list processing, and representation. These notions are used in the second part of the paper as the basis for some remarks and suggestions concerning language and system design. (p.525)

---

9 "The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs is still quite limited*. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items)" (p.377)

10 Existem variações para os diversos bancos de dados comerciais no Mercado, mas geralmente são adotados os padrões mínimos definidos pela ANSI – *American National Standards Institute* – em 1986 e pela ISO - *International Organization for Standardization* em 1987.

11 G. L. DAVENPORT; E. O. DAVENPORT. The genealogies of the families of Cohasset. Massachusetts: Stanhope Press Boston, 1909

O autor afirma que relações são generalizações dos relacionamentos familiares:

> A data base never records all of the facts about a group of entities; a fact may be recorded with complete or lesser accuracy; and non-facts may be recorded with equal facility. It was, no doubt, the study of genealogical data that led to the invention of the theory of relations, which will lead in turn to our notions about data in general. Informally, relations are simply a generalization of family relationships, and genealogical data is one of the older instances of recorded data. We start with a set of individuals (or any other type of entity) and a second set, which mayor may not be the same set as the first. A relation is a correspondence between members of the two sets. For instance, son-of, children-of, father-of, ancestor-of, sib-of, and the like are all relations, as are birth-date-of, occupation-of, age-of, residence-of, marital-status-of, etc. (p.526)

A terminologia apresentada pelo autor engloba algumas definições iniciais, a partir da noção de que um modelo "é um sistema de conjuntos de entidades, valores, mapas de dados e mapas de procedimento" (p.533, tradução do autor). Ele ainda reforça a explicação sobre os principais termos abordados:

a) **Entidades**: correspondem aos objetos no mundo real sobre os quais dados são gravados ou computados;

b) **Mapas de dados**: atribuem valores aos atributos das entidades; estes mapas são considerados como conjuntos de pares ordenados de entidades e valores, ou itens de dados;

c) **Dados estruturais**: tipo especial de mapas de dados, nos quais o valor definido é o próprio conjunto de entidades;

d) **Procedimentos**: operações sobre os mapas de dados, produzindo novos (ou redefinindo) mapas de dados.

Para definir o conceito de **representação**, o autor identifica três sistemas presentes em qualquer situação, uma vez que computadores não podem lidar com mapas abstratos: o primeiro representa, ao menos a partir de uma visao platônica, uma parte do mundo real; o segundo é a teoria construída sobre o primeiro; o terceiro é uma representação do segundo para o computador. Dessa forma, ele afirma que a representação é definida como "**um mapa estabelecendo a correspondência entre dois desses sistemas**". (p.529, tradução do autor, grifo nosso)

Outro conceito importante apresentado pelo autor é o de **organização dos dados**, e sua relação com o acesso aos dados:

> In the case of a machine representation, we have an abstract system and a representation map mapping it into the machine system. […] On the other hand, there is the physical storage system, and the machine representation of the data and procedure maps must be mapped into the physical storage. The structural part of this mapping - that is, the correspondence between the structure of the data and the structure of storage, we call the **data organization**. While this enables data access, it is not access. (p.530, grifo nosso)

Para o autor, o **acesso** "é uma característica do processamento dos dados, não dos dados em si ou como eles são representados; procedimentos diferentes, em geral, querem acessar os mesmos

dados de maneiras e em ordens diferentes" (p.530, tradução do autor). Ele também estabelece uma comparação entre os procedimentos de acesso e a organização dos dados:

> **Access functions** are maps whose values are entities; they are used by procedures to get access to the entities, and hence to the data. Access functions may involve use of structural data, but are principally a feature of the individual procedure. **Data organization**, on the other hand, is the way the structure of the data is mapped into the structure of the storage media. (p.533)

Uma importante observação destacada por ele é a de que "a ordem em que itens de dados são obtidos e armazenados é (ou deveria ser) independente da organização de dados" (p.530, tradução do autor). Em seguida, o autor apresenta sua concepção para a **descrição dos dados**:

> **Data description** describes machine data systems, representations, and organizations, rather than abstract data itself. That is, it is a specification of the maps, usually in terms of procedures which will accomplish the mapping, and the salient characteristics of the entity and value sets. To describe a data aggregate - that is, a file or data set - we supply this information together with information concerning the aggregate as a whole. […] We are forced to conclude that a **data type** is a fragment of data description and, as such, describes a portion of a system and its associated representation and organization maps. It is an attribute of entities. (p.530-1, grifo nosso)

Esses conceitos, de acordo com Mealy (1967), são básicos para a questão da independência e flexibilidade na escolha dos computadores, a qual ele denomina **independência de representação**:

> On the basis of the point of view about data advocated earlier, more light can be shed on several issues of current interest. In some sense, these issues are all related to the possibilities of flexibility in choice of machine (machine independence), choice of data representation (ability to define new data types), or ability to strike a balance between compilation and interpretation (variable binding time). […] Independent of one's personal degree of confidence in fulfillment of such an objective (and most of us believe that the objective is a Good Thing), I would urge adoption of the term "representation independence" as being more appropriate than the term "machine independence." In espousing such an objective, one's philosophical point of view might be that data processing takes place in the abstract realm in which our theory of data is formulated. (p.531)

A seguir ele propõe que a descrição de dados deveria ser explicitamente armazenada para que o usuário tivesse liberdade de manipulação de acordo com suas necessidades:

> We should, I believe, seriously investigate the possibility of system designs that allow the individual user to make his own judgment of the proper tradeoff. This would necessitate explicit storage of data descriptions. […] However, language design has tended to suppress the notion of representation to the extent that the programmer frequently cannot talk about it. I regard this as being a mistaken approach toward our practical goals; on the contrary, significant progress toward representation independence of results can only be made by making the notion of representation much more explicit in language designs than it is at present. (p.533)

Ao final, ele afirma que a "padronização dos métodos de descrição de dados pode vir a revelar-se muito mais importante do que a padronização dos métodos de representação de dados e especificação de procedimentos" (p.534, tradução do autor)

A busca pela independência dos dados armazenados em relação à estrutura computacional era também a preocupação de uma das associações importantes da época. Orientado para a definição de padrões de linguagem para essa independência, o CODASYL (*Conference on Data Systems Languages*) era

> an informal and voluntary organization of interested individuals, supported by their institutions, who contribute their efforts and expenses toward the end of designing and developing techniques and languages for data systems, analysis, design, and implementation. (HARE JR., 1971, 1971, p.1)

Em seu relatório (HARE JR., 1971) foram apresentados alguns conceitos padronizados, entre eles dois que são importantes para o desenvolvimento desta tese:

a) **Linguagem de Descrição de Dados / Linguagem de Definição de Dados**:

> Data Description Language (DDL). A language for defining records, sets of records (and their relationships) and areas of the storage space. Divided into the DDL schema, which defines the 'universal' data base itself, and various DDL sub-schema, one to describe the data known to each application program, which may involve different host languages. (p.11)

b) Linguagem de Manipulação de Dados:

> Data Manipulation Language (DML). A language for specifying the storage and retrieval actions desired by an application program. Used to control the data base and interface it via the DDL schema and sub-schema to the host language used in the application programs. (p.11)

O objetivo da criação dessas linguagens consistia na explicitação da criação da estrutura dos dados a partir de uma linguagem (DDL) acessível ao programador e independente dos parâmetros técnicos[12] que interferiam na forma como a descrição dos dados era utilizada (Mealy, 1967, p.533). A manipulação dos dados a partir da DML também tornou o acesso aos dados independente da estrutura computacional.

3.4 A GENERALIZAÇÃO DO PROCESSAMENTO DE DADOS

O artigo "*Generalization: Key to Successful Electronic Data Processing*" de 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

[12] "How much of the data description is stored explicitly, and where, is partially a matter of taste. It is largely determined by the **language**, **language processor**, and **operating system** one is working with at present. Classically, we have tended to use the data description at compile time and then throw it away. Moreover, much of the data description has been implicit in the compiler's structure; the programmer has had little explicit control. (p.533, grifo nosso)"

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 breve descrição apresentada por ele 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.

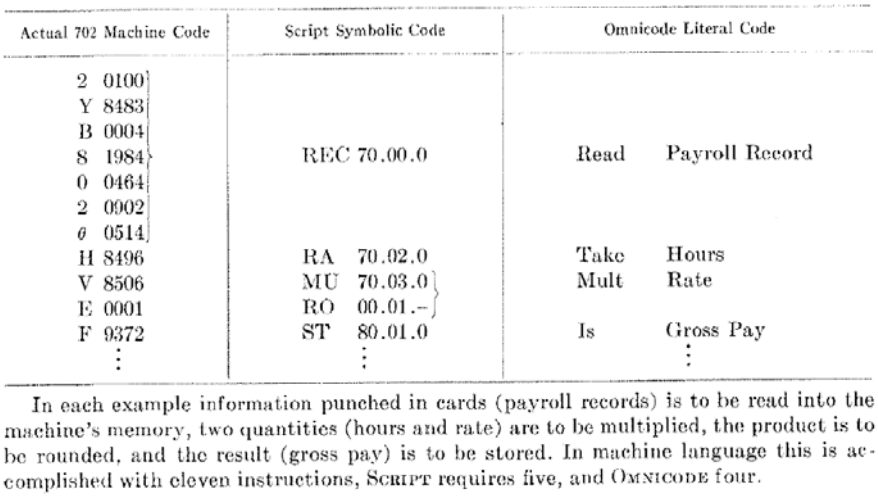
Esse processo demandava muito tempo e dinheiro, um retrabalho enorme para a reprogramação e realização de novos testes – para sistemas já implantados. A capacitação dos programadores para a operação das novas máquinas também exigia muitos recursos. Essa complexidade de compreensão da forma de programação 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. 5 apresenta uma comparação entre as três formas de programação:

FIGURA 5: 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 0 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 identifiable 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 wich 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 FRAGILIDADE DA MODELAGEM DE DADOS: A REDUNDÂNCIA IMPLÍCITA

A partir da análise da fundamentação teórica levantada para este trabalho, é importante destacar alguns pontos que tem impacto direto na questão da redundância implícita na modelagem de dados.

De acordo com Chen (1976), existem dois tipos de entidade: fracas, que dependem de relacionamentos para existir, ou regulares, que existem sem essa necessidade. Este é o ponto fulcral para a nomeação da metodologia proposta: modelagem essencial (que não guarda relação com a metodologia de análise essencial) – o ponto-chave é a *existência em si*, a *essência* da entidade.

Nesse sentido, aquilo que *existe em si*, independente de qualquer relacionamento, é uma entidade regular, de acordo com a distinção proposta por

Chen (1976). Já o que existe em função de um relacionamento é uma entidade fraca. Outro conceito apresentado pelo autor é o de papel que uma entidade pode assumir em relação a outra, como no exemplo de “marido-mulher”.

Chen (1976) apresenta em uma nota de rodapé – e consideramos esse comentário importante demais para não merecer maior destaque – a possibilidade de uma pessoa considerar alguma coisa como uma entidade, e outra pessoa tratar esta mesma coisa como relacionamento. O questionamento inicial que nos levou a propor a modelagem essencial foi: como uma coisa pode ser entidade ou relacionamento, sem considerar-se sua essência? Ou ela é uma coisa, ou outra.

Nesse sentido, esse ponto se apresenta como uma fragilidade intrínseca ao processo de modelagem de dados, e potencialmente geradora da redundância implícita em diversos sistemas.

Em paralelo à generalização, Mealy (1959) reforça a necessidade de arquivos únicos – e temos ciência de que a realidade à época era diferente da atual em termos de estruturas de dados – para evitar-se o esforço de controle da redundância. Para a modelagem essencial aqui proposta, é importante a busca por arquivos únicos, no sentido de representação padronizada da realidade a partir de uma reflexão mais acurada sobre a *essência* dos objetos representados.

Se na época de Mealy (1959) era mais importante a padronização da forma de descrição dos dados – o que pode ser representado hoje com os metadados nos bancos de dados relacionais, por exemplo – para a proposição desta metodologia consideramos mais importante a padronização das formas de representação.

O próprio exemplo de Chen (1976) apresentado na FIG. 2 contém a redundância implícita: os dados de um funcionário estão representados na respectiva entidade, enquanto os dados de seus dependentes existem em “dependente”. Mas não é possível que um dos dependentes de um funcionário também trabalhe na mesma empresa? Nesse caso, suas informações estariam redundantes nas duas tabelas distintas.

Diversas situações similares são encontradas nos modelos de dados de muitos sistemas e exemplos didáticos (STAIR, 1998, p. 114; TURBAN, RAINER e POTTER, 2005, p. 139-41). É comum observarmos a existência de tabelas para armazenar dados de professores, alunos e funcionários em sistemas de gestão de instituições de ensino. Mas um funcionário não pode ser professor – e mesmo aluno – na mesma instituição? E nos sistemas comerciais, não é comum a situação em que um fornecedor é também cliente da mesma empresa?

De acordo com a modelagem essencial proposta, para as situações citadas não houve uma análise da *essência* dos objetos a serem representados no modelo. Dessa forma, professor, aluno, funcionário, cliente e fornecedor são todos **papéis** assumidos por pessoas – físicas ou jurídicas – e que na verdade são relacionamentos entre essas pessoas. “Cliente” não tem existência por si só. Essencialmente, todas essas são situações em que uma **pessoa** assumiu determinados papéis.

Para termos uma ideia da dimensão da redundância para esses casos, basta imaginar que cada “professor” ou “cliente” pode possuir muitas informações

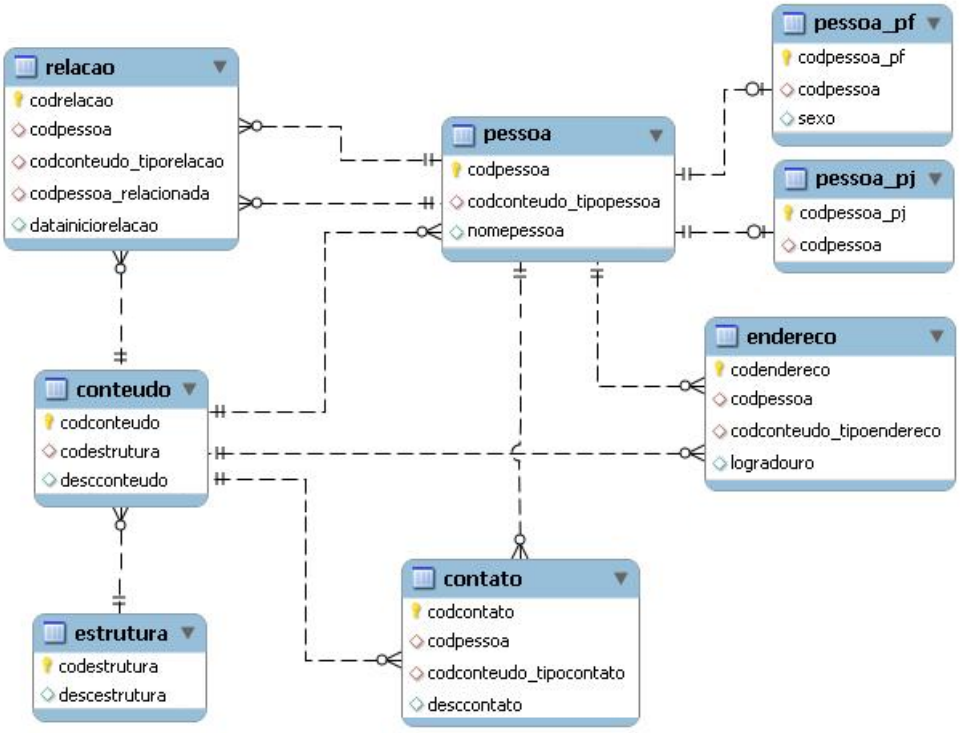
associadas a ele, como “endereço”, “telefone” e “dependente”, entre outras. A duplicação dessas informações quando ele assume outro papel certamente resultará em comprometimento da qualidade dos dados armazenados – comprometendo seriamente a gestão dessas informações.

O tópico a seguir apresentará um modelo genérico para tratar as situações apontadas até aqui, elaborado a partir da modelagem essencial.

5 A APLICAÇÃO DA MODELAGEM ESSENCIAL

Para ilustrar a discussão deste artigo, apresentamos a seguir parte do modelo resultante da aplicação das premissas da modelagem essencial, contendo um conjunto de tabelas usadas em sistemas desenvolvidos pelo autor deste trabalho.

FIGURA 6: Modelo de dados gerado a partir da modelagem essencial



Fonte: Desenvolvida pelo autor

Existe um cadastro único de pessoas, com os atributos comuns a pessoas físicas e jurídicas – os atributos específicos são registrados nas tabelas “pessoa_pf” e “pessoa_pj”. Cada pessoa possui endereços e contatos, registrados nas respectivas tabelas. Separar as pessoas físicas das jurídicas em duas tabelas distintas não foi uma opção adotada, porque existem algumas formas de pessoa jurídica, como o empreendedor individual, que fazem com que uma pessoa física possa ser, simultaneamente, pessoa jurídica. No modelo da FIG. 6, ela possuiria informações registradas nas duas tabelas – “pessoa_pf” e “pessoa_pj”. Se as informações sobre pessoas físicas e jurídicas fossem separadas em duas tabelas distintas, seus dados de endereço, por exemplo, ficariam duplicados.

Os papéis assumidos por essas pessoas são registrados em “relacao”. Para a modelagem essencial, “cliente” e “fornecedor” são essencialmente relações

entre pessoas. Essa relação pode assumir vários tipos, que podem inclusive ser criados sem a necessidade de intervenção no modelo, apenas a partir do cadastramento da nova forma de relacionamento na tabela “estrutura” e de seus respectivos valores na tabela “conteudo” – essas duas tabelas são explicadas em outro trabalho ainda não publicado.

O modelo da FIG. 6 apresenta, assim, uma forma isenta da redundância implícita observada em grande parte dos sistemas modelados sem as premissas previstas na modelagem essencial.

6 CONCLUSÕES E RECOMENDAÇÕES

A análise dos textos integrantes da fundamentação teórica deste artigo mostrou alguns pontos críticos em relação a alguns fundamentos da modelagem de dados.

A partir da revisão desses pontos, foi proposta a metodologia de *modelagem essencial*, que considera uma reflexão mais acurada sobre a essência dos objetos a serem representados para a confecção do modelo.

A adoção desta metodologia elimina a possibilidade de redundância implícita, uma vez que existem tabelas únicas para armazenamento das informações – ou arquivos únicos, na visão de McGee (1959).

A existência de tabelas únicas permite melhor controle sobre o conteúdo das informações registradas – o que interfere positivamente na qualidade das informações armazenadas.

O modelo apresenta flexibilidade para a incorporação de novos tipos de relação, reduzindo a necessidade de manutenção ao longo do tempo, o que representa uma redução significativa de custos.

A aplicação desta metodologia não se restringe a sistemas automatizados, podendo ser aplicada também a arquivos manuais.

Sugere-se o aprofundamento desse estudo a partir de algumas abordagens filosóficas que tratem da “essência em si” das coisas e a avaliação de conceitos ou técnicas de outras ciências, como a Ciência da Informação.

REFERÊNCIAS

CHAMBERLIN, D. Donald; BOYCE, Raymond F. SEQUEL: A Structured English Query Language. Proc. ACM SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Michigan (May 1974) pages 249-264

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.

CHILDS, David L. Description of a set-theoretic data structure. **Proceedings of the December 9-11, 1968, fall joint computer conference, part I**. San Francisco, California: ACM, 1968a. p. 557-564.

CHILDS, David L. **Feasibility of a set-theoretic data structure**: a general structure based on a reconstituted definition of relation. Washington D. C. Advanced Research Projects Agency, 1968b.

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.

HARE JR., Van Court. 1971. **A special report on the SIGBDP Forum "The new Data Base Task Group Report"**. Atlantic City, New Jersey. CODASYL (Conference on Data Systems Languages), 11p. (Relatório técnico)

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

MEALY, George. H. Another look at data. **Proceedings of the November 14-16, 1967, fall joint computer conference**. Anaheim, California: ACM, 1967. p. 525-534.

SCHMID, H. A.; SWENSON, J. R. On the semantics of the relational data model. **Proceedings of the 1975 ACM SIGMOD international conference on Management of data**. San Jose, California: ACM, 1975. p. 211-223.

STAIR, Ralph M. **Princípios de Sistemas de Informação**: uma abordagem gerencial. Rio de Janeiro: LTC, 1998.

TURBAN, E.; RAINER JR., R. K.; POTTER, R. E. **Administração de tecnologia da informação**: teoria e prática. Rio de Janeiro: Elsevier, 2005.

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