



ICININFO

Knowledge and Information in Nutrition Software Design

Nikitas N. Karanikolas *

Dept. of Informatics, TEI of Athens, Aigaleo 12210, Greece

Abstract

The intension of this paper is to point out that both Knowledge and Information can be persisted within ordinary database management systems. It is a well-known demand for codifying the knowledge versus of embedding it inside the business logic (the source code). The consequences are that knowledge can be extended, utilization of knowledge can be extended and that the application's evolution can be achieved, almost without programming. To achieve such a goal, database and system designers should distinguish what is information and what is knowledge for the system's domain of application. They should also foresee an abstract knowledge representation that will be able to hold the future knowledge in the domain. Our approach is based on an interesting domain of software technology, the nutrition software design.

© 2014 Elsevier Ltd. This is an open access article under the CC BY-NC-ND license

(<http://creativecommons.org/licenses/by-nc-nd/3.0/>).

Selection and peer-review under responsibility of the 3rd International Conference on Integrated Information.

Keywords: Knowledge Evolution; Self-Adapted Systems; Entity Attribute Value; Nutrition software;

1. Introduction*1.1. What is Data*

Data is information in raw or unorganized form. The letters of an alphabet are data. The numbers (for example decimal numbers – 123) are data. Data are used to synthesise Information.

In Computer science parlance, data are symbols or signals that constitute the input in some processing (program or process) and can be stored (kept for future use) in volatile (e.g. RAM) and non volatile (e.g. magnetic disks)

* Corresponding author. Tel.: +30-210-538-5737; fax: +30-210-5910975.

E-mail address: nnk@teiath.gr

devices. The result of computer processing of data (by some program) is communicated to the users in a usable form that is called information.

1.2. *What is Information*

Information is data that are accurate and timely. Information is specific data, organized for a purpose. Information is data presented within a context that gives them meaning and relevance. Information contributes to increase in understanding and decrease in uncertainty of some topic. For the people (Cognitive Agents), the Information is valuable because it can affect their behavior and can support them for getting decisions. Information is an influence which leads (but not necessarily) to a transformation. However, Systems theory assumes that information does not necessarily involve any conscious mind.

1.3. *What is Knowledge*

The philosopher Plato defined knowledge as "justified true belief" (JTB). According to the Plato's definition, a statement must meet three criteria in order to be considered knowledge. It must be justified, true, and believed. Someone can say that (s)he knows something under three conditions:

- (s)he believes the statement to be true,
- the statement is in fact true,
- (s)he is justified in believing the statement to be true.

Many philosophers reject the JTB formulation altogether and others think that JTB needs to be "fixed up" somehow. We remain to the Plato's definition and complement that knowledge involves concepts and abstractions in our brains. Consequently knowledge is interwoven with a conscious, living being.

According to some approach (Control-Z), there are three general categories of knowledge:

- Factual or Propositional Knowledge: it permits a conscious mind to make statements which are factually correct,
- Procedural Knowledge: it is when we know how to go about doing something (actually being able to do it),
- Knowledge of Personal Experience: knowledge drawn upon personal experiences (not available to others).

Factual and Procedural Knowledge are further explained elsewhere (Anderson, 2001).

According to another approach (Steve Denning), the human mind is capable of two kinds of knowledge:

- Rational Knowledge: must have demonstrable, provable, fact-based information to support it,
- Intuitive Knowledge: the ability to acquire knowledge without inference.

The Rational and Intuitive Knowledge are further explained elsewhere (Spencer Carr, 1978).

It is worthwhile to mention the term Tacit Knowledge, introduced by Michael Polanyi (Polanyi, 1958; Polanyi 1966). This term recognizes the inherent difficulties in transferring knowledge from one person to another.

1.4. *From Data to Wisdom*

Knowledge is not the highest-grade content existing in the human mind. Wisdom is the highest grade content. Detailed knowledge regarding the pyramid of content, in the human mind, can be found in (Gene Bellinger, et al). They discuss the Russell Ackoff's article (Ackoff, 1989) for the pyramid of content in human mind. We shortly summarize the levels of pyramid: Data (lower), Information, Knowledge, Wisdom (upper).

1.5. The classic computational approach in software design and implementation

In the current decade, systems are persisting *information* (in a wide interpretation of the term) into databases. Some decades ago, systems' designers were designing structured files (files of records) and assigned them indexes (index files). We will continue, and base our examples, with databases (for further simplification we will use only relational examples).

The interpretation of the term *information* (in the previous paragraph) is wide, because it includes Factual or Propositional Knowledge. For example the knowledge that any patient (in an application for the medical domain) is identified by his (her) social security number, (s)he has a name (first name and surname) and characterized by the attributes sex and date of birth, is realized by a relational table that hold instances of patients. Instances (table tuples) are *information* in the strict interpretation of the term.

Knowledge of Personal Experience is rather rarely occupying systems' designers. However, table cell values, are *Data* because, by itself, are in an unorganized form and they acquire usable form when combined with their adjacent cells (in the same row) to communicate to the user a concrete patient. So far, we have seen that the wide interpretation of *information* (used in the classical systems' design approach) includes Factual or Propositional Knowledge, *Information* (in its strict meaning) and *Data*.

In the classical systems' design approach, Procedural Knowledge is usually hard coded into source code of applications. For example, Systems' designers pose the functional requirement that patients can be retrieved by providing their surname (to the system) and permitting the user to select the appropriate one between the patients having the same surname. Software developers (programmers) implement this functional requirement by codifying (in source code statements) the procedural logic needed. Thus, in some later stage in the application's lifecycle, a new requirement for retrieving patients on the base of their date of birth, impose programming intervention (source code modifications and additions).

1.6. Restrictions of the classic approach

Consequently, any new functional requirement imposes programming intervention that cost in time and money.

1.7. Non-classic approaches

Good European Health Record (GEHR) was a three year project started in 1991, funded by the European Union's Advanced Informatics in Medicine (AIM) initiative. Two participants in the original GEHR project, subsequently working on the development of EHRs in Australia, proposed the concept of two levels modeling, and the use of archetypes to separate information and knowledge (Ping Yu, 2003). Archetypes are definitions and restrictions on how to order and structure principal information (information from GEHR Reference Model, and, here for simplicity, tuples from different database tables). This permits the real participation of users (or domain experts) in software development. It provides users with the authority of designing and manipulating components of the software. The intention of the archetype approach is to pass the task of archetype maintenance and evolution to domain experts. The domain experts are expected to add archetypes to the data model when new concepts arise. It is also the domain expert's responsibility to delete, revise or update an existing archetype.

1.8. The domain of our application

Our purpose is to provide some priming for designing systems able to extend their knowledge and utilize it for achieving application's evolution, without programming. In order to practically evaluate our suggestion, we apply it for the development of nutrition software, available through the web, and acting as an information resource. In section 2, we detail our methodology. Section 3 describes the result of methodology, a database schema able to hold the current knowledge and some future knowledge extensions. Section 4 gives details for the Adaptable Nutrition Application and the final section, contains our conclusions and intensions for future work.

2. Methodology

Our solution for designing systems able to extend their knowledge and utilize it for achieving application's evolution, without programming intervention, is summarized in the following algorithmic steps:

- Distinguish what is information and what is procedural knowledge for the domain of application,
- Define a schema that will host both data (information) and procedural knowledge,
- Foresee the schema changes that will allow the insertion into the database of the future procedural knowledge extensions in the domain,
- Define an application structure that will be able to adapt itself according to the persisted procedural knowledge.

These steps constitute a procedural Knowledge by itself. We hope that this knowledge (the algorithm) will be adopted by systems' designers and will allow the design and implementation of more flexible and adaptable (in their future functional requirements) systems. Such systems will be able to extend their procedural knowledge and utilize it for achieving application's evolution, without programming.

3. The Schema for a flexible Nutrition Database

The application of the first three steps of the algorithm presented in the previous methodology resulted in a Relational Schema depicted in Figure 1 and described in this section. This schema hosts current procedural knowledge and permits some future knowledge extensions.

Our nutrition database is a relational one and it is composed of eighteen tables. Due to the limited capacity, we are unable to analyze the whole logical relational schema. Hence, we are going to discuss three specific points of our logical relational schema. These are: a) the categories that a food belongs to, b) the benefits, overdose consequences, insufficiency consequences and the resistance that some nutrition can have and c) information regarding combinations of foods, nutrition and food categories.

A food can belong to more than one category (e.g. spaghetti with mince meat belongs to both spaghetti and meat categories) and many foods can belong to the same category. This is a simple many to many relation, implemented in our schema in three tables: Food, Category and FoodCategory. The latter table implements the many to many (M:N) relationship between the other two tables. This is a very well known technique and it cannot be characterized as advanced database design. However, the table Category has its own interest. More specifically, a category can be primary or can be a refinement of a primary category (e.g. alkaline fruits and sour fruits are refinements of fruits, shell are refinement of sea food, etc), a category can be a refinement of a refinement of a primary category and so on. Here we have a hierarchy of categories. This hierarchy is a hierarchy of nodes with single predecessors (single parents). In general, there are hierarchies with single and multiple parents. The latter (with multiple parents) are also known as tangled hierarchies. Hierarchies (single parent or multiple parents) could be represented with ontologies, but also with ordinary relational tables. Ontologies are not inherent characteristics of databases. However, Ontologies permit knowledge representation and consequently can support knowledge evolution (in our case, the system can start with a single parent hierarchy and evolve, if the future requirements demand it, to a multiple parent hierarchy supporting also inference). A systems' designer, given the current situation where the requirement is to host only single-parent food taxonomy, could decide to stick to a simple relational implementation of it. This is the case of the relational table Category, depicted in figure 1. More details about this solution are given in a recent paper (Karanikolas et al, 2012).

In case that future requirement, during the lifecycle of the nutrition software, imposes the use of a multiple-parent hierarchy, the database designers can intervene and give a more sophisticated relational solution. (Karanikolas et al, 2012) present such a relational solution, but this solution does not have an inherent inference mechanism. Inference could be a next future requirement that would demand another cycle of interventions by database designers and software programmers. Thus, the ontology solution seems to be the solution for extending the system's knowledge and achieving system's evolution, without programming intervention.

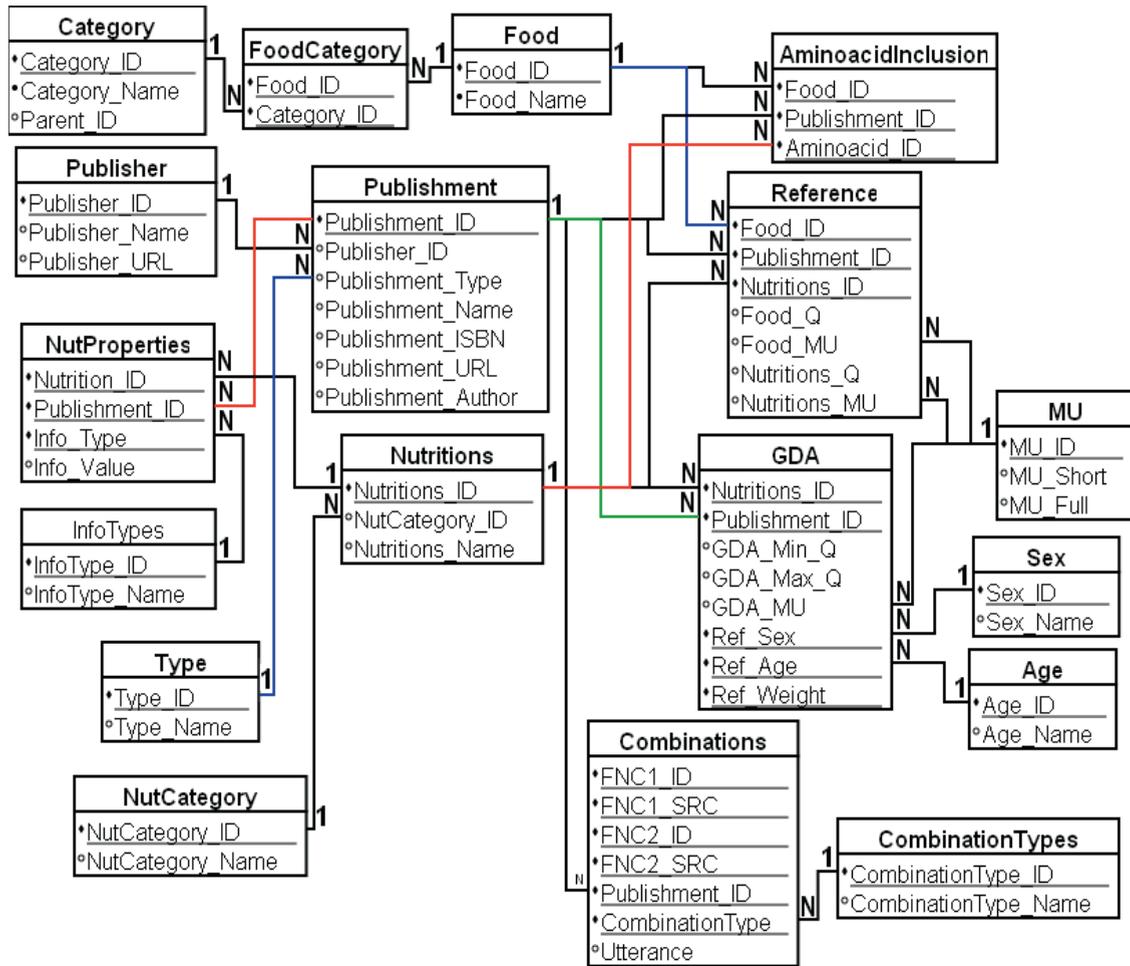


Fig. 1. The relational schema for the adaptable nutrition database.

The second case that our methodology can influence the system’s designer, arise during the design of the system’s feature to retain benefits, overdose consequences, insufficiency consequences and resistance of nutrition elements to various facts. At the time where the nutrition software was designed, there were four such potential features (listed above). A naive relational solution could use a table with six attributes (nutrition identifier, source identifier and four more attributes, one for each property). This solution suffers from a lot of null values, since the actual number of properties coming from the same source (the same book, article, etc) is almost always one or in the best case, two. Another drawback of this solution is that it merges the procedural knowledge of handling these (independent between each other) features of nutrition elements. Since the nutritional science is a modern science, it is almost certain that the number of features/properties will increase in the near future. This will demand the intervention of database designers and software programmers for holding (into the database) and handling (through the application) the new features/properties.

Our solution here is influenced by the EAV data modelling (Nadkarni, 2000, 2002). Our (logical relational) sub-schema is based on three tables. The table Nutritions defines each nutrition (every row has Nutrition_ID, Nutrition_Name, etc), the table InfoTypes defines the available properties (every row has InfoType_ID and InfoType_Name and so far, there are four rows, one row for each of benefits, overdose consequences, insufficiency

consequences and resistance). The table NutProperties combines the above two, as well as the table Publishment for declaring one, every time, of the mentioned properties. For this reason, the table NutProperties has four attributes (Nutrition_ID, Publishment_ID, Info_Type and Info_Value). The Attribute (A in EAV) is defined by the table InfoTypes, the Entity (E in EAV) is defined by table Nutritions and the Value (V in EAV) is recorded in the table NutProperties.

The idea behind our solution is: a) to remove hard coded (into source code) procedural knowledge which handle (similarly but independently) the mentioned features (a different, part of source code for each feature), b) incorporate this knowledge (actually the differentiation aspect of the similarly handled features) into the database (the table InfoTypes contains the four features), and, c) hard code, into the application, only an upper-level of procedural knowledge that handles all these features.

Another EAV influenced sub-schema in our nutrition database, which is an outcome of our methodology, is the way used for storing information known about food, nutrition and food category combinations. (Karanikolas et al, 2012) describes this sub-schema, in detail.

4. The Adaptable Nutrition Application

4.1. Self Application's Adaptation according to the current procedural knowledge

Current functional requirements include:

- Depicted attributes and Sort Orders provided to the users,
- Predefined Query templates (QBE) that can be used for retrieving relevant information.

These functional requirements embed (or imply) procedural knowledge for handling them. Thus, systems' designers could permit the direct implementation of them (hard code procedural knowledge into source code). However, some of these procedures can have similar behaviour and can be abstracted in some more general procedure and the differentiating aspects can become values in some table of parameters. Thus a small set of general procedures can be hard coded and adapt their selves according to the user selected behaviour preference (some tuple in the parameter's table). For example, Figure 2 is an interface that determine which fields be presented in the lower grid and what is the order (which one is the primary ordering field, which one is the secondary ordering field, and so on) of the presented results, according to user selected item from the list box above the grid.

Συστατικό Α	Τύπος Συνδυασμού	Συστ. Β	Λεκτικό
Ασβέστιο	Left decreases utilization of Right	Σίδηρος	Το ασβέστιο εμποδίζει την απορρόφηση του σιδήρου
Βιταμίνη C (Ασκορβικό Οξύ)	Left increases utilization of Right	Σίδηρος	Η βιταμίνη C βοηθάει στην απορρόφηση του σιδήρου

Fig. 2. An interface that adapt itself (Depicted attributes and Sort Orders) according to the user's preferences

In a similar way some Predefined Query templates (Queries By Example) can be substituted by a single upper level query, which is adapted according to some user's selection.

4.2. Self Application's Adaptation according to future procedural knowledge

More than having a small number of hard coded general procedures, this approach permits system's evolution by simply insertion of new rows in some parameter's table. Thus new functional requirements (future, procedural knowledge) can be adopted by the system as another one adaptation (another one tuple) of some general (hard coded) procedure. For example, we will have a system with extensible set of Predefined Queries.

Obviously, except the eighteen tables depicted in figure 1, there are some other parameters' tables that are used for persisting knowledge needed for application's adaptation and evolution.

5. Conclusions and Future Work

The Procedural Knowledge documented in this article (a way for designing systems able to extend their knowledge and utilize it for achieving application's evolution, without programming) could be (or it be) an Intuitive Knowledge used subconsciously by some experienced and clever systems' designers. Here, we tried to (and we hope that we did) gain distinction of this knowledge and consequently it will be adopted by more systems' designers. Ultimately, this knowledge could be taught in graduate, or even undergraduate, university degrees.

Our future plans include the accommodation of some ontology for handling food hierarchies and the evaluation of our methodology with more demanding functional requirements. Such requirements could be system's adaptation to support automatic measurement units' translations and an NLP interface that will permit ad hoc queries without standardized (form based) interface.

Acknowledgements

This research has been co-funded by the European Union (Social Fund) and Greek national resources under the framework of the "Archimedes III: Funding of Research Groups in TEI of Athens" project of the "Education & Lifelong Learning" Operational Programme.

References

- Ackoff, Russell (1989). From Data to Wisdom. *Journal of Applied Systems Analysis*, 16, pp. 3–9.
- Anderson, Lorin and Krathwohl, David (2001). *A Taxonomy For Learning, Teaching and Assessing*. New York, Longman.
- Bellinger, Gene, Castro Durval and Mills Anthony. Data, Information, Knowledge and Wisdom, <http://www.systems-thinking.org/dikw/dikw.htm>
- Carr, Spencer (1978). Spinoza's Distinction Between Rational and Intuitive Knowledge. *The Philosophical Review*, 87(2), pp. 241-252.
- Control-Z. What is Knowledge? Epistemology - Justified True Belief - Contextualism - Rationalism - Empiricism, http://control-z.com/czp/pgs/what_is_knowledge.html
- Denning, Steve. What is knowledge? Definitions of Knowledge, <http://stevedenning.com/Knowledge-Management/what-is-knowledge.aspx>
- Karanikolas Nikitas N., Anastasia Tritaki and Oleg Karavasileiadis (2012). Advanced Database design and Modern Web technologies used in Nutrition Software design. PCI'2012: 16th Panhellenic Conference on Informatics, October 5-7, 2012, Piraeus, Greece. IEEE CPS.
- Nadkarni P. (2000). Clinical Patient Record Systems Architecture: An Overview. *Journal of Postgraduate Medicine*, 46(3), pp. 199-204.
- Nadkarni P. (2002). An introduction to entity-attribute-value design for generic clinical study data management systems. Presentation in: National GCRC Meeting. Baltimore, MD.
- Polanyi Michael (1958). *Personal Knowledge: Towards a Post-Critical Philosophy*. University of Chicago Press, ISBN 0-226-67288-3.
- Polanyi Michael (1966). *The Tacit Dimension*. London, Routledge.
- Yu Ping (2003). Archetypes, GEHR, openEHR and Electronic Health Records. *The Journal on Information Technology in Healthcare*, 1(5), pp. 369–380.