Beyond The Relational Mode: Using Ontology for Medical Data Management

Iyad Zayour
Faculty of Science
Lebanese University
Beirut, Lebanon
email: iyad@alumni.uottawa.ca

Bilal El-Hajj-Diab
School of Computer Science and Software Engineering
University of Wollongong
Wollongong, Australia
emial: b.diab@alumni.uottawa.ca

Bassam Hussein
Department of Industrial Engineering
Lebanese International University
Beirut, Lebanon
email: bassam.hussein@liu.edu.lb

Ali Hage-Diab
Department of Biomedical Engineering
Lebanese International University
Beirut, Lebanon
email: ali.hagediab@liu.edu.lb

Hassan M. Khachfe
Center for Quality Assurance, Institutional Assessment &
Scientific Research (QAIASR)
Lebanese International University
Beirut, Lebanon
email: hassan.khachfe@liu.edu.lb

***Abstract*—Software tools that manage structured data are predominantly confined to mainly two tools: spreadsheets and databases. Investigation in the medical field shows that the experience with either tool has not been satisfactory. While a spreadsheet offers flexibility to manipulate the structure of data and can be operated by the end user, it is too limited for extensive data in term of size or complexity. Databases are suitable for moderately complex data that can be large in size, but are costly, take a long time to stabilize and can be only implemented by programmers. These limitations are due to the rigidity of the data model based on arranging data in a tabular form. In order to address this issue, we investigated the frame-based-knowledge-base data model (ontology) from the artificial intelligence literature. This model seems to be more effective and efficient in managing medical research data. To operate using this model, we used Protégé, an open source tool that offered a near database like functionality.**

Keywords-Database; Ontology; Data Management;

I. INTRODUCTION

Data management is one of the most classical areas of computing. In the last 30 years, no new data model could challenge the dominance of the relational model. But how good is this model in managing the kind of domains that are conceptually complex, small in size and with continuously changing structure—something we call personal data management? Our investigation in a hospital was that the industrial tools based on the relational model are not satisfactory in managing the kind of data that needs to be collected. Existing database systems have several critical shortcomings that prevent using them directly to process such data. They are too heavy-weight and slow, devoting much complexity to handling tasks that might be irrelevant to the healthcare domain [1].

Interestingly, we identified a more suitable model that has been there for long time in the dust of artificial intelligence (AI) arena. Frame-based-knowledge-base that date back to the early days of AI provided a superior data model for medical data management.

A. Available Tools and their Data Model

Practically, available software tools to manage structured data are predominantly confined to two classes: spread sheets and software/database applications.

1) Spreadsheets: On one hand, a spreadsheet is very flexible for creating data models and thus managing data; it is personal built with specific and exact data in mind. Any user with computer literacy can create a tabular list of attributes as columns and actual data as rows. Although a very primitive model, the enormous usability functions and features done by Microsoft on Excel have considerably stretched the limits of what this model can handle. Features like auto filter, auto repeat, dynamic forms and pivot tables touch on many of the functionality that are typically afforded by special purpose software applications.

Yet, these limits did not stretch far enough; data representing several classes/concepts have to be de-normalized to fit the tabular form of a spreadsheet. This de-normalization will increase almost exponentially the redundancy of data. For example, for a one-to-many relation between a professor and students, the complete data of the professor has to be repeated for every student (what an SQL will transparently do in inner join). The common practice and a fundamental principle of building spreadsheets has been normalization, which is a way to build a data model that

 ISBN: 978-1-61208-511-1

has the properties of simplicity, non-redundancy, and minimal maintenance [2]. Violating the principle of normalization may create confusion and is far from well-established norms and standards.

2) Databases: On the other hand, software applications based on relational database are suitable to handle moderately complex data that can be very large in size. Theoretically, with enough programming efforts, any data domain can be modeled in relational form with minimum redundancy. However, like any other software, database solutions are costly, take a long time to stabilize and can be done only An by very specialized people; the programmers. This work can be tedious and needs to be done before users can start focusing their attention on the issues that they should be really working on [3]. The end user has little control over adapting the database to the change in requirements that are inevitable in most real world settings.

Microsoft, that stretched the personal Excel into the professional database side, did also try to stretch Access to the more "personal tool" side. But experiences show that there was little success on both sides. Excel still fails when data get to certain size, and Access applications also begin to need full-fledged programmers when requirements bypass the most common scenarios.

It becomes evident that there is a gap between simple knowledge suitable for spreadsheet and costly database application. Thus, there is a need to provide a personalized tool that can manage the complex data space. In order to bridge the gap, we propose and define a personal data management tool that allows the end user to control the structure of data, to enter data and retrieve it without the need for a programmer.

B. Data models

In our opinion, the problem is not in the usability of the tools but rather in the inherent limitations of the data model used in both of spreadsheet and relational databases. This model, based on arranging data in tabular form where all the knowledge space has to be squeezed in between columns and rows, has limited modeling expressiveness all often creating semantic gap with the domain it is modeling. Usually this gap is alleviated either by redundancy as in the spreadsheet or by complex programming as in the database. In this paper, we investigate the little-known frame-based-knowledge-base data model as an alternative approach to commonly used data models. This is an attempt to have a more consolidated base for scattered data; something that would be a must in the near future as elaborated by Tim Berners-Lee (2005): "It is about the data which currently is in relational databases, XML documents, spreadsheets, and proprietary format data files, and all of which would be useful to have access to as one huge database." [4]

Section II outlines the data experience including spreadsheet and database experience. Section III presents a review of the modeling process adapted in this paper. Section IV introduces the concept of ontology. Section V discusses the experience with Protégé. Section VI concludes the paper and provides recommendations for future work.

II. THE DATA EXPERIENCE

The data requirement for medical research domain has been for long recognized to be demanding [5]. Our recent experience in a hospital was a perfect example of hitting the relational model limits. The type of medical data that needs to be collected in a hospital is very versatile. Data need to be collected about different objects each with specific attributes in addition to other common attributes. An example of such data and objects includes neuro-imaging parameters (e.g., regional neuro-anatomical and metabolic variations), neuropsychological testing, clinical features, laboratory measures, and genetic studies. Managing this sort of information has been found to be challenging. Next, we describe how the hospital in question tried to manage this data and what kind of challenges it faced.

A. Spreadsheet experience

To save and retrieve data, spreadsheets were initially used in the hospital that we studied. Experience with spreadsheets was satisfying on a smaller size data space. New attributes can be added instantaneously (new columns), data manipulation (add, edit, delete) were trivial and there were no complex space to worry about (e.g., no table relations), they just added columns and rows. Querying was particularly satisfactory using the auto-filter and pivot tables that answered almost all the queries.

Almost any new data requirement could be met by adding new columns, but this could not continue forever. In one spreadsheet that tracks medical imaging information and activities, we reached the "FV" vertical cells (176 cells). At this scale, the spread sheet became poorly useable and prone to error. Navigation activities, such as finding the value of an attribute, understanding the meaning of a column and identifying valid values became increasingly demanding and challenging tasks.

Furthermore, redundancy became a major concern due to the lack of normalization. To add the data of a new image (about 10 new attributes), all other attributes related to the subject need to be repeated. Moreover, as the same column needs to be used for different classes of images having different attributes, it was confusing to identify which attributes (columns) are relevant to the current images and which are not applicable. This issue is related to the lack of inheritance support.

B. Database experience

After the spreadsheet reached its limit, we moved the data from spreadsheet to custom software based on databases. The database approach did not yield any more satisfactory experience. From a development perspective, the suitability issue of relational database models and normalization for complex domains is well-known (e.g., producing a plethora of tables) [6].

More importantly, from the user perspective, the main concern was due to the logistics of database development. This development requires that changes should be contracted to programmers who are not familiars with the data domain. This necessitated a time consuming knowledge transfer task to these programmers.

 ISBN: 978-1-61208-511-1

In our setting, requirements keep evolving as new ideas and observations are brought in. Continuously, there were attributes that are added or removed and values that were reengineered.

Following a rigorous change management process, the change requests created by the evolution in data requirement were more frequent than what the programmers could handle and the time needed in a typical software change was more than users could tolerate; so much so that they eventually, users regressed to spreadsheets.

III. REVIEWING MODELING

As we came to the conclusion that the tabular (spreadsheet) and the relational models (database) are unsuitable for our data management requirements, we began looking for alternative models.

In general, the quality of any modeling technique is related to how much it can faithfully represents the interesting knowledge of the domain it is modeling. The notion of semantic gap [7] has been used to describe this sort of quality. A semantic gap, desired to be minimal, refers to the difference between the way knowledge to be encoded is naturally specified and the syntax of the computer representation used to do the encoding. A minimal semantic gap means that the knowledge of a domain can be "paralleled" in a computer representation without applying additional constrains.

In the database literature, the schema design is called the logical design, which is preceded by a more permissible and flexible form of design called the conceptual design. This latter design exhibits the highest fidelity to the real world domain (minimum semantic gaps). To bridge the impedance mismatch between conceptual and relational (logical) model, some relational constrains need to be applied on the conceptual model to produce the logical model.

The conceptual design corresponds mainly to OO-UML (Object Oriented Unified Modeling Language) class diagrams (or entity-relationship diagrams) and it is sometimes called domain modeling or analysis modeling [8]. It models a domain faithfully regardless of the technology that will utilize the model (a UML class diagram can be the input for schema as much as for OO software). Such a design is typically extracted from a conceptual data model, which is a representation of organizational data. The purpose of a conceptual data model is to show as many rules about the meaning and interrelationships among data as possible, independent of any database management system or other implementation considerations [2].

A direct data tool that can parallel a conceptual design should satisfy our modeling requirements. What we needed exactly to satisfy our data management requirement is a personalized tool that can faithfully model our complex data space by paralleling its conceptual model (especially inheritance) and still be personal enough to avoid to the need for costly programming cycles. Note that by personal we mean the ability of the user to modify the data structure himself (like it is in spreadsheets) without the need for a programmer help (like in databases).

IV. FRAME BASED KNOWLEDGE BASE (ONTOLOGY)

It was interesting to find that one of the oldest technologies that appeared in the early days of computing as part of the artificial intelligence literature was the most satisfying technology. The frame-based-knowledge-base, also called ontology, offers a very semantically rich data and knowledge representation by acting as the semantic mediator for heterogeneous databases [9].

Using one of the many available ontology editors, ontology could represent knowledge and data in a way that accurately parallel the conceptual model. In fact, an ontology editor with sufficient graphic capabilities can replace any conceptual design tool.

Like a class diagram, ontology permits the user to define classes (called concepts in the ontology jargon) and attributes (called slots), even methods in a class diagram can have "daemon" counterparts. An inheritance hierarchy can be created in between classes by means of a binary relation over entities called ISA relation, and even attributes can be inherited to create sub attributes [10].

With an ontology editor, the user defines an ontology for the data space being modeled. However, unlike other forms of design, the implementation of the design is instantaneously available. By allowing the creation of instances of classes, an ontology editor allows the user to enter the actual data in a simple interactive manner. In some ontology editor like Protégé [11] that we choose to use after evaluating several other editors, the UI input forms are dynamically generated by the editor to represents a controlled interface for the data corresponding to the class attributes.

For example, to create a data structure to manage information about a person, we can use Protégé to create a "person" class. Then we can add attributes to concept person such as name, gender, and married status. In return, Protégé will automatically generates a UI form that permits the user to start creating person instances and that contain UI controls to enter data for the person's attributes: a text box for the name, a combo to select "male" or "female" (these are the possible values that were assigned during gender attribute definition) and a check box for the Boolean married attribute.

Creating ontology, thus, was an interwoven and iterative activity of creating conceptual models and entering the data instances for the concepts. The end product is the ontology; an artifact that describes both the abstract conceptual knowledge and the actual data instances. This conceptual annotation and organization of data around the conceptual model has much facilitated the manual navigation and exploration of data. Data instances are explicitly connected to their actual level of abstractions - the concepts (classes). This provides logical data independence, using high-level abstractions to shield the user from the complexity of the underlying hardware and software platforms [1].

V. EXPERIENCE WITH PROTÉGÉ

Although Protégé was satisfactory as a technology platform, we note that several issues arise when users that are used to simple data model have to work with something

 ISBN: 978-1-61208-511-1

as complicated as an ontology. Next, we highlight and discuss the advantages and disadvantages of such approach.

A. On The Intellectual Experience in Ontology Editing

Conceptual models were never trivial to do, many design decisions are conceptually perplexing such as grouping the decomposition into classes and the location of attributes; unlike the typical simple columns adding in spreadsheets. Not any Excel user would be an ontology editor, as the sophistication of the ontology model requires equal sophistication in users. Classically, this task has been assigned to a specific kind of users commonly called the "knowledge engineer".

Yet, despite the sophistication required in users that reduces the level of personalization (as many user may requires professional helps or training compared to Excel), using Protégé is still very advantageous over the database programming model.

First, any modification of the conceptual model would be of minimum cost. In Protégé, most modifications can be done by simple drag and drop within the integrity constrains (e.g., moving an attribute to super or sub classes).

Second, the utilization of the conceptual design is instant and iterative. The user can be entering new instances just after a class and at least one attribute is created. The cycle between conceptual design and instance creations has been observed to be frequent initially but stabilizes after a while thus reducing wasted time. This iteration is of minimum cost compared to the programming models where any modification will jeopardize the stability of the whole system and incur high costs due to extensive system validation and verification (quality assurance).

B. Protégé Evaluation

We found Protégé to be an adequate solution of our current medical data management requirements. Data navigation and exploration has improved drastically. By providing the inheritance capabilities, Protégé solved the issue with the horizontal growth of data in term of attributes that represent the conceptual complexity in data as depicted in Fig. 1. Common attributes are stored in the high level classes and only specialized one are associated with sub-classes.

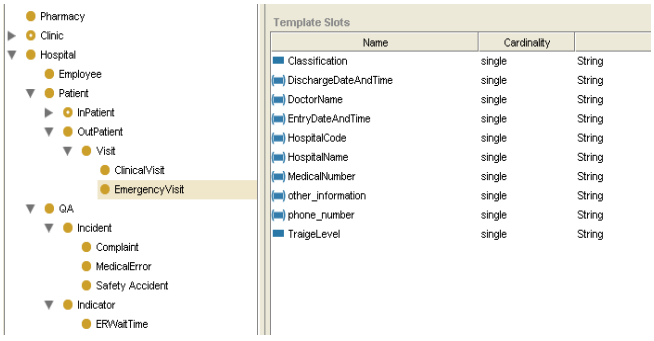

Figure 1. Instance hierarchy and data navigation capability visually created and managed.

Yet, the size of our data is still within the limit of what can be explored and navigated visually. We are not sure how much this approach can handle the vertical growth of data (increase in size) where querying and not navigation would be needed.

In particular, we note that Protégé query mechanism is confined to inter-class conditions (query the instances of one class). This does not represent a shortcoming of the model as much as for the tool. Ontology languages such as RDF, offers full-fledged intra class querying capabilities (compose a condition that utilizes the attribute of more than a class). Although RDF is supported by protégé as a plug-in, the only way to utilize it is by writing queries whose complexity is beyond most end users expectations.

VI. Conclusion And Future Work

As frame-based-knowledge-base existed for long in the research community, there is an increasing need to transform such a rugged research idea to a tool that meets the everyday requirements and needs of a data user. It is not surprising that frame-based-knowledge-base and ontology mean the same thing since the research community largely failed to envision any utilization of this technology outside the classical artificial intelligence domains and utilizations. A quick look at the knowledge base projects performed with Protégé [12], the leading ontology editor, shows how classical and purely research-oriented established foundations are still dominant for this kind of work.

While excellent data models exist at the basic level of technology, such as XML (Extensible Markup Language) and RDF (Resource Description Framework), which is an XML based ontology language; user-friendly tools for the common tasks in the industry like personal data management are still widely missing.

In our domain (medical research) like many others, data is very valuable and expensive; it is paramount to make the best use of this asset. Rich knowledge representation can help capitalizing on this data by offering models that facilitate finding best conclusions out of this data.

Encouraged by the success we encountered with Protégé in meeting our data requirements, we started to develop a tool that have the advantages of protégé and solve some of its disadvantages. The Personal Knowledge Manager (PKM) is a tool that permits the user to define his data schema and instantly use it to enter data in an Excel like simplicity. It supports inheritance and different table/concept yet it provides a very similar experience to Excel users.

References

[1] M. Balazinska et al., "Data management in the worldwide sensor web." IEEE Pervasive Computing, pp. 30-40, 2007.

[2] J. S. Valacich, F. G. Joey, and J. A. Hoffer, Essentials of systems analysis and design. Upper Saddle River, New Jersey, Pearson Education, 2015.

[3] J. Alcala-Fdez et al., "KEEL: a software tool to assess evolutionary algorithms for data mining problems." Soft Computing, pp. 307-318, 2009.

[4] T. Berners-Lee et al., "Tabulator: Exploring and analyzing linked data on the semantic web." In Proceedings of the 3rd

international semantic web user interaction workshop, p. 159, 2006.

[5] H. Andrade and J. H. Saltz, “Towards a Knowledge Base Management System (KBMS): An Ontology-Aware Database Management System (DBMS)”. SBBD 1999: 27-39

[6] K. Stoffel et al., A graphical tool for ad hoc query generation. In AMIA’98. American Medical Informatics Association, p. 503, 1998.

[7] D. Merritt, “Best Practices for Rule-Based Application Development Application Development”, Microsoft Architect Journal, January 2004.

[8] E. Malinowski and E. Zimnyi “Hierarchies in a multidimensional model: from conceptual modeling to logical representation“, Data & Knowledge Engineering, pp. 348–377, 2006.

[9] H. Chen et al., "Towards a semantic web of relational databases: a practical semantic toolkit and an in-use case from traditional Chinese medicine." In The Semantic Web-ISWC, pp. 750-763. Springer Berlin Heidelberg, 2006.

[10] T. Catarci and M. Lenzerini, "Representing and using interschema knowledge in cooperative information systems." International Journal of Intelligent and Cooperative Information Systems, pp. 375-398, 1993.

[11] Stanford University Website. Protégé User Guide. Available online at http://protege.stanford.edu/. Last accessed on June 16th, 2016.

[12] CIM Engineering Website. Highly Effective Distributed Collaboration. Available online at www.cim3.net. Last accessed on June 16th, 2016.

 ISBN: 978-1-61208-511-1