Conceptual Universal Database Language: Literature Review and the Future of Database Design

Nikitas N. Karanikolas
Department of Informatics,
Technological Educational Institute (TEI) of Athens
Ag. Spyridonos street, 12210 Aigaleo, GREECE
+30-210-5385882
nnk@teiath.gr

ABSTRACT

Today, the restricted data type domains of the relational model affect Information Systems design. We claim another approach where the Information System designers would be able to portray directly the real world in a database model that provides more powerful and composite data types, as those of the real world. However, more powerful databases models need the introduction of data query and manipulation languages that reflect the features of the new composite data types. We introduce such a language and also reveal a direct consequence, which is the introduction of higher database design levels.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages – *Data description languages (DDL), Data manipulation languages (DML).*

H.2.4 [**Database Management**]: Systems – *Query processing, Relational databases.*

D.3.2 [**Programming Languages**]: Language Classifications – *Design languages, Nonprocedural languages.*

General Terms

Design, Management, Languages.

Keywords

Conceptual database design, Entity Relationship Diagrams.

1. INTRODUCTION AND MOTIVATION

The methodology that is used up to today in the design of relational databases is the localization of a set of attributes in (usually) one and universal relation and the localization of a set of functional dependencies and afterwards the decomposition of the set of attributes into smaller relations which consist of subsets of the original set of attributes, in order to eliminate update anomalies and reduce data redundancy.

A speculation of the methodologies of data analysis is that we do not know the structure of information that we were called to impress in an Information System. For this reason we begin with interviews of the persons involved in the operation of a non-computerized system and from this process we arrive in a series of fundamental informations (attributes). The right correlation and grouping of the fundamental attributes is a next stage of data analysis. This speculation and technique of data analysis has also influenced the design of relational databases.

The normalization is the process that aims to produce the best from a (from the beginning) weak data model. ("the relational model is limited with respect to semantic content (i.e., expressive power) and there are many design problems which are not naturally expressible in terms of relations" [1], "The relational model is weak when showing many-to-one relationships" [2]).

The real world that we are called to impress with an Information System (often with the use of a database) seldom incorporates repetitions of data (data redundancy). As an example in a non-computerized managed library we do not incorporate a lot of series of books (copies) and a corresponding number of bookshelves in order to place the first set of book copies sorted according to the title, the second set of book copies sorted according to the first writer, the third set of book copies sorted according to the theme category, etc. On the contrary we do not use any ordering (more precisely we use the ordering of books according to the date of import into the library) or use some of the orderings that interest us (usually thematic) and in addition we create indexes (with cards) for every one of the ordering that interest us. Each card contains the key of the classification and a reference to the natural ordering (the location of book in the bookshelves).

Thus, we claim that the Information System design should not decompose the real world (that we were called to impress in an Information System) in its fundamental characteristics and afterwards to proceed with simple compositions of characteristics that relational model allows. We claim another approach where the Information System designers would be able to portray directly the real world in a model that provides more powerful structures, as those of the real world. Section 2 provides a review of such models. Some of them could be used instead of the relational model. However, having more powerful database models but still using data manipulation languages designed only for fundamental data attributes, is a waste of time and resources. Therefore, there is a necessity for a database query and manipulation language able to manipulate directly the composite (real world) data types. This necessitation is explained, more detailed, in section 3. Section 4 gives details of the Conceptual Universal Database Language (CUDL), which satisfy the mentioned necessitation. Section 5 reveals the consequence of higher database design levels.

2. BACKGROUND

2.1 Generic Data Modeling

The generic data modeling [3] approach is an outcome mainly emanate from research in the Medical Informatics domain. The fact that, in case of Health Care data maintenance, the amount of information and the complexity of information lead to a huge (Daedal / mazy) conceptual schema, concerned the Medical Informatics scientists. Moreover, the fact that the direct production of a logical schema for a relational DBMS, from a given huge conceptual schema, obviously conserve this Daedal characteristic, gave raise for research for alternative data modeling approaches. Another inherent characteristic of relational logical schemata is the difficulty for supporting data evolution (changing information needs). The research results for both problems (Daedal conceptual and logical schema and difficulty for data evolution in relational data) reveal the generic data modeling approach. This approach defines two generic transformations (namely “flattening” and “relation merging”). The later transformation (“relation merging”) is also the basis for supporting data evolution. These transformations, when applied to the original conceptual schema, produce a generic logical schema consisted of a reduced number of tables. However, this process is not a very strict procedure and actually it is depended from (the personal perception and) the selections made by the person who guides the process and applies the mentioned generic transformations. The final number of tables, as the outgrowth of the transformation process, is dependent from the selections made and is not a concrete (predefined) set of tables (as happens in other cases, e.g. in the FDB data modeling).

The disadvantage of generic data modeling is that querying the resulting generic logical schema with standard SQL requires multiple statements and considerable intellectual effort, especially when the queries are intended to retrieve data for feeding data analysis tasks (e.g. feeding data mining applications). To overcome such difficulties, researchers have defined the Extended Multi-Feature (EMF) SQL extension [4] that provides simpler to understand, more compact and more efficient query constructions.

2.2 EAV Data Modeling

The Entity Attribute Value (EAV) data modeling [5,6] is also an outcome from research in the Medical Informatics domain. The motivation for the research that revealed the EAV data modeling was that, in the medical domain, the number of parameters (facts) that potentially apply to any clinical study is vastly more than the parameters that actually apply to an individual clinical study. For example, the potential number of laboratory examinations that a patient could be submitted to is a huge superset of the actually submitted examinations in a specific medical case (e.g. a patient suffering from a bilestone). Another reason, that motivated the research that revealed the EAV data modeling, was that clinical studies are subject to evolution as a result of medical research. As a consequence the number of clinical parameters related to a clinical study are always differentiated (and, in most cases, are increased). Thus, the data model should be able to host new clinical parameters for any clinical study, without the need for data (structure) reorganization. The research, motivated by the above-mentioned reasons, revealed the EAV data modeling. According to EAV design, metadata and data tables compose the logical database schema. The facts (that actually apply to a clinical study) are recorded into the data tables, as a triplet: the Entity, the Attribute and the Value. The Attribute is the recorded fact (clinical parameter) and the Entity is a composition of the relevant patient’s identifier and some timestamp. The metadata tables are used to define the data composition (which clinical parameter (Attribute) pertain to which clinical study).

There are three main versions of the EAV data modeling [7] but all of them share the same basic principle (the triplet: Entity / Object, Attribute, Value). Another interesting feature of the EAV models is that they permit mixture of EAV stored and conventionally stored data. However, the existence of this heterogeneity complicates significantly the task of data querying. We should also mention that the EAV data modeling support facts evolution (equivalent to the addition columns in a relational table without the need for any reorganization) but does not support table (entity) evolution.

2.3 FDB Data Modeling

In previous works [8,9] there has been an investigation of dynamically evolving database environments and corresponding schemata, allowing storage and manipulation of variable number of fields per record, variable length of fields, composite fields (fields having subfields), multiple values per field (either atomic or composite), etc. The ultimate goal of the research work of Yannakoudakis was to make the design and maintenance of a database a simpler task for database designers, so as that they will not have to put in a lot of effort to design the database and later they will not have to pay special attention and work for database changes. Their research proposed a new framework for the definition of a universal logical database schema that eliminates completely the need for reorganization at both logical and internal levels, even when major modifications of the database requirements have occurred. This framework was called FDB [8].

This Universal logical database schema is based on well and strictly defined set of Metadata and data tables and it does not permit any mixture with conventionally stored data. All the available to the user entities and their attributes are documented exclusively in the metadata tables and the facts concerning the instances of the entities are recorded exclusively in the data tables of the FDB Universal schema. Another noteworthy feature of FDB is that it support Schema evolution, both for facts and entities.

Moreover the FDB model allocates ways of imprinting strictly connected (Hardly related) information with innate (inherent/native) mechanisms. In contrast to the relational model that compels the creation of artifact structures (tables) to represent strictly connected (Hardly related) data. As an example, the relational model requires the creation of new table to store data that relate of the form one to many (the addresses or the telephones of customer). In contradiction to the relational model, the FDB model can maintain the same information with a field that is accommodated in the side of the one and accepts multiple values. Even more complex forms of strictly connected (Hardly related) information, are impressed, in the FDB model, with innate (inherent/native) mechanisms. For example a correlation of information with a form many to many (as are the DVDs that have been rented to a member of a Video Club) is maintained in one of the two connected sides without requiring the creation of new table to correlate the information. That is to say in the correlations many to many we follow a mechanism that emanates from the real world (in the example of the Video Club we maintain inside the card of a customer a table with his/her renting of DVDs).

The most important fact in the FDB model is that it organizes information without any repetition of values. In order to be more precise not only it does not proceed in repetitions but it ensures that these cannot be created. In the example of the addresses (or alternative the telephones) of a customer the basic data of a customer (let us say name, surname and code) are stored once and in the field addresses are stored the all different addresses that the customer may have. That is to say the use of a single big (universal) table to store/repeat as many times the basic attributes of a customer (name, surname and code) as the number of his/hers addresses is avoided naturally (without any effort). Thus the FDB model provides as an inherent feature the no redundancy property.

2.4 Not First Normal Form (NF^2) or Nested Relational Data Modeling

The motivation for inventing the Nested Relational data model was that: The Relational model has difficulties of modeling the real world; It is also inconvenient for handling even simple data structures commonly used in IR. To overcome these problems, Researchers has proposed a relational model where Non First Normal Form (NF^2) relations are allowed [10,11]. This extension encompass the classical 1st Normal Form (1NF) model and adds, to the relational algebra, two basic operators (namely “nest” and “unnest”). Based on the “nest” operator, the proposal allows sets (as the result of “one-attribute” nest operation) and sets of sets (as the result of “multi-attribute” nest operation) as attribute values. NF^2 sets are equivalent to simple FDB fields with repetitions and NF^2 sets of sets are equivalent to composite FDB fields with repetitions. The researchers have also proposed a query language extension for NF^2 table definition and manipulation. However, the NF^2 present some weak points:

- Does not support Entity or Fact (Attribute) evolution
- Does not have Universal logical schema
- The proposed query language extension only undertakes (be engaged in) Retrieval statements
- This Retrieval statements are rather suggestions or hypothetical statements and are not parts of a mature language that handles relational tables with non-atomic (sets and sets of sets as) attribute values
- The notion that governs the whole idea, which has passed through and is reflected by the proposed query language extension, is that the subfields (of composite fields) are not directly accessed by the user.
- Related to the previous point is that the proposed query language uses Nested Select statements whenever a restriction over a subfield should be applied

Possibly, these weak point has the consequence that, after 26 years, Non First Normal Form does not seem to be implemented as a DBMS.

2.5 Object Oriented and Object Relational Databases

The weakness of the relational model to manage complex, highly interrelated information motivated the research for Object Databases (ODB) and Object Relational Databases (ORDB). Both models are also described in textbooks [12].

The portability and interoperability of ODBs is ensured by the Object Model suggested by the Object Database Management Group (ODMG). The ODMG Object Model provide also the definition for an Object Definition Language (ODL) and an Object Query Language (OQL). The ODL statements seem to (or are influenced by) the Java language statements used for class definitions, while the OQL statements seems to (or are influenced by) the SQL statements used for data retrieval. At mid of the 1990’s decade there were a notable number of ODB solutions but today only half of those remain active. Our personal opinion is that the data management professionals are not like to bother themselves with strange programming constructions of classes, inheritance and so on, and consequently they do not decide easily to use an ODB.

The other direction, the Object Relational Databases, aim to provide solutions for complex and highly interrelated information management, without imposing complicated programming constructions. For this reason, it provide Black Box Complex data types for various purposes (management of time series, geographic point manipulations, face recognition, content-based retrieval of digital audio, image watermarking, image search, full-text search), Opaque types for extending the repertoire of Black Box Complex data types, User Defined Complex data types. Black Box Complex data types are named as Data Blades in Informix Universal Server and are named as Cartridges in Oracle. The User Defined Complex data types have similar characteristics to the ODMG Objects. The composition of User Defined Complex data types is based on simpler structures (namely: the Collection types and the Row types). The SQL3 standard provides an extension to the previous SQL standards, for handling the most of the characteristics added with the ORDBs.

3. THE MISSING PUZZLE ITEM

It is obvious from the plethora of models (presented in the background section) that there is a need for DBMS able to provide more powerful data types, as those of the real world complex data. The first four discussed models (namely: Generic, EAV, FDB and NF^2) provide composite data types using meta-models, on top of relational databases. The problem with these approaches is that the handling of the composite data requires very good acquaintance of the underlying meta-model structures and the internal organisation of both metadata and data. The user (programmer) should combine the business logic requirements with the retrieval of the metadata that explain the composition of the requested composite data types, the retrieval of the underlying simple data and the reverse composition of the requested composite data (three steps). In some case of the first four discussed models, there is provided an SQL query language extension that uses Nested Select statements whenever a restriction over a subfield should be applied. However this approach, except that it is not mature, complicates the deliverance (expression) of data maintenance statements for real applications. Thus, the ultimate requirement should be to provide a data manipulation language able to manipulate directly the composite data types (without bothering the programmer with the mentioned three steps), while permitting the direct expression of restrictions over subfields.

In regard to the ideas presented in the Object Relational model, the new data manipulation language should provide Black Box Complex data types and Opaque types for extending the repertoire of Black Box Complex data types. However the Object data types of the Object Relational model can be excluded from the new data manipulation language, since the provided composite data types covers satisfactory the needs for complex data types.

4. THE CONCEPTUAL UNIVERSAL DATABASE LANGUAGE (CUDL)

In our approach we have adopted the Frame Database Model as the underlying model for implementing our goal for a data manipulation language able to manipulate directly composite data types. We preferred the FDB model, since it is more compact and well defined than the other models and also supports schema evolution [15]. The logical schema of FDB is based on the following tables:

Name	**Structure**	**Use**
Languages	(language_id, lang_name)	AM
Datatypes	(datatype_id, datatype_name)	AM
Messages	(message_id, language, message)	AM
Entities	(frame_entity_id, title)	PM
Tag _attributes	(entity, tag, title, occurrence, repetition, authority, language, datatype, length)	PM
Subfield _attributes	(entity, tag, subfield, title, occurrence, repetition, language, datatype, length)	PM
Catalogue	(entity, frame_object_number, frame_object_label, temp_stamp)	D
Tag_data	(entity, frame_object, tag, repetition, chunk, tdata)	D
Authority _links	(from_entity, from_tag, from_subfield, to_entity, to_tag, to_subfield, relationship_type)	R
Subfield _data	(entity, frame_object, tag, tag_repetition, subfield, subfld_repetition, chunk, sdata)	D

Notes:
AM: Auxiliary Metadata; **PM**: Primary Metadata;
D: Data; **R**: Relationships; Primary keys are underlined.

Let us explain this universal schema. Only three tables (sets in the FDB terminology) of the schema are used to host data, namely: Catalogue, Tag_data and Subfield_data. The rest sets host metadata information. Three of the metadata sets, namely the Entities, Tag_attributes and Subfield_attributes, are used to define every abstract entity and its constituents. One of the metadata sets, the Authority_links, is used to define data relationships. The rest three are auxiliary metadata sets. We must state that this universal schema is able to carry into effect 1:M and M:N relationships without the need for intermediate entities. 1:M relationships are carried into effect very easy with tags that accept repetitions. M:N relationships are also carried into effect with repetitions. Tags that accept repetitions means the possibility of adding a list of values in the place of a single field. Also, a tag can entertain subfields. From the combination of the two above, results the ability of placing an entire table in the place of a single field. In addition to this, there is the ability that each of the cells that comprise the table can accept repetitions (list of values).

Karanikolas et al. [13], introduced the syntax and semantics of the CUDL language. There they focused mainly in presenting and analysing the statement of value retrieval (in the schema and the data). Karanikolas et al. [15], focused mainly in presenting and analysing the CUDL statement of value modifications in the schema (schema changes) and the data. Karanikolas et al. [14], focused mainly with the need for relationship declarations. This need becomes more significant for the FDB-CUDL model because the relationships between entities, in most cases, are implemented without the introduction of new tables. Without having methods to declare relationships, the user would face a refuting stage where the model is self-explained (the user can consult only tag_attributes and subfield_attributes and carry off the data model) but the data relationships are totally undocumented. To cope with this need, the FDB model introduced the Authority_links set. They also use the Authority_links set to declare authority controls and reduce variability of expressions used for the same instance of an identity. All of these (relationship declarations and authority control declaratios) are provided through CUDL statements.

In order to give an indicative example of the CUDL language, we suppose that some application undertakes the administration of the projects implemented by a company. In such an application there is an entity, named Projects, that contains all projects that the company services. The following are instances of the Projects entity.

Project_code	Proj066
Title	Hermes
Budget	455,000
Actions	

Employee	**Action**	**Deadline**
Yannis	Software analysis	17/10/2007
Vangelis	Software requirements	22/01/2008
Dimitris Panos	Program code	23/04/2008

Project_code	Proj055
Title	Athena
Budget	250,000
Actions	

Employee	**Action**	**Deadline**
Yannis	grubbing	20/3/2009
Giorgos Maria	pruning	25/3/2009
Nikos	watering	30/4/2009

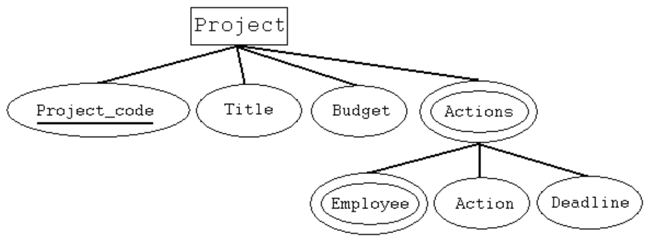
The following is a data retrieval statement, expressed in CUDL:

```
#   Find data when entity = 'Projects' and
    tag = 'title' restr data = 'Hermes' and
    subfield = 'Action' restr data = 'program code' and
    subfield = 'Employee'
```

With this CUDL statement we declare that the tag 'title' will be projected and concurrently will function as a restriction for the selection of instances, the subfield 'Action' will be projected and concurrently will function as a restriction for the selection of instances and finally the subfield 'Employee' will only be projected.

5. HIGHER DATABASE DESIGN LEVELS

As we have mentioned earlier, the ultimate goal was to define a data manipulation language able to manipulate directly composite data types, while permitting the direct expression of restrictions over subfields. CUDL is the outcome of such an effort. However, there is another interesting outgrowth of the introduction of CUDL. With CUDL, the application programmer / designer can model the structures of its application with composite data types, closer to the ER diagrams and sometimes without any decomposition of the ER entities into simpler ones. For example the following ER diagram:



is directly supported by the CUDL composite data types (see the previous instances of Projects). On the other hand the logical database level of any CUDL based application is the underlying FDB model. Thus, instead of transforming from ER to simple relational tables to provide a logical model for manipulation through SQL, we are able to transform from ER to CUDL Abstraction Level (CAL) entities (namely, CUDL data sets with composite data types). In other words, the classic database design triplet (ER, logical and physical level) is replaced by the quadruple: ER, CAL, logical and physical design, with a fixed logical design (the FDB model).

6. CONCLUSIONS

So far, the design of an application having a relational data repository required the decomposition of the real world structures into very simple attributes, the composition of a logical schema with naive relational structures and the implementation of query and manipulation (SQL) statements based on the logical schema. We have claimed that some of the existing higher data models (offering composite / complex data types), in conjunction with the Conceptual Universal Database Language (CUDL), can help us to ignore the logical schema and map directly from the ER diagrams into more sophisticated (higher) database entities. We have also provided arguments and evidence about our claims. Thus, with CUDL, it is permitted the direct manipulation of higher database entities and the designers and developers can be concentrated with the business logic of their applications, instead of wasting time for the expression of statements that manipulate naive database structures.

7. REFERENCES

[1] Worboys, M., Hearnshaw, H., and Maguire, D. 1990. Object-Oriented Data Modelling for Spatial Databases. International Journal of Geographical Information Systems 4, 4, 369–383.

[2] Pavković, N., Štorga, M., and Pavlić, D. 2001. Two Examples of Database Structures in Management of Engineering Data. In Proceedings of the 12th International Conference on Design Tools and Methods in Industrial Engineering (Bologna, 2000). ADM-Associazione Nazionale Disegno di Macchine, 89-90.

[3] Johnson, S.B. 1996. Generic data modeling for clinical repositories. Journal of American Medical Informatics Association 3, 5, 328-339.

[4] Johnson S. B., and Chatziantoniou D. 1999. Extended SQL for manipulating clinical warehouse data. In proceedings of American Medical Informatics Association Symposium. AMIA 1999, 819-823.

[5] Nadkarni P.M. 2000. Clinical Patient Record Systems Architecture: An Overview. Journal of Postgraduate Medicine 46, 3, 199-204.

[6] Nadkarni P. 2002. An introduction to entity-attribute-value design for generic clinical study data management systems. Presentation in National GCRC Meeting (Baltimore, MD, April 13, 2002).

[7] Anhøj, J. 2003. Generic Design of Web-Based Clinical Databases. Journal Medical Internet Research 4. http://www.jmir.org/2003/4/e27/

[8] Yannakoudakis E.J., Tsionos C.X., and Kapetis C.A. 1999. A new framework for dynamically evolving database environments. Journal of Documentation 55, 2, 144-158.

[9] Yannakoudakis E.J., and Diamantis I.K. 2001. Further improvements of the Framework for Dynamic Evolving of Database environments. In Proceeding of the 5th Hellenic – European Conference on Computer Mathematics and its Applications (Athens, Greece, 2001).

[10] Schek H.J., Pistor P. 1982. Data Structures for an Integrated Data Base Management and Information Retrieval System. In proceedings of the 8th International Conference on Very Large DataBases. VLDB 1982, 197-207.

[11] Fischer P. C., and Van Gucht D. 1985. Determining when a Structure is a Nested Relation. In proceedings of the 11th International Conference on Very Large DataBases. VLDB 1985, 171-180.

[12] Elmasri R., and Navathe, S. 2000. Fundamentals of Database Systems, 3rd Edition, Addison Wesley Publishing Company.

[13] Karanikolas, N.N., Nitsiou, M., Yannakoudakis, E.J., and Skourlas, C. 2007. CUDL language semantics, liven up the FDB data model. In Proceedings of Eleventh East-European Conference on Advances in Databases and Information Systems (Varna, Bulgaria, September 29 - October 03, 2007). ADBIS 2007 local proceedings, 1-16.

[14] Karanikolas, N.N., Nitsiou, M. and Yannakoudakis, E.J. 2008. CUDL Language Semantics: Authority Links. In Proceedings of the Twelfth East-European Conference on Advances in Databases and Information Systems (Pori, Finland, September 5-9, 2008). ADBIS 2008, 123-139.

[15] Karanikolas, N.N., Nitsiou, M., Yannakoudakis, E.J., and Skourlas, C. 2009. CUDL Language Semantics: Updating Data. Journal of Systems and Software 82, 6, 947-962.