A Comparison Analysis of Graphical Models of Object-Oriented Databases and the GOQL Model

EUCLID KERAMOPOULOS  
School of Computer Science  
Westminster University  
115 New Cavendish Street,  
London W1M 8JS, UK

PHILIPPOS POUYIOUTAS  
School of Computing  
Intercollege  
46 Makedonitissas Ave.,  
Nicosia 1700, CYPRUS

TASOS PTOHOS  
School of Computer Science  
Westminster University  
115 New Cavendish Street,  
London W1M 8JS, UK

Abstract: - In this paper we present a comparison analysis of the most important graphical models of object-oriented databases. These models are also compared with the graphical model (User’s View – UV) of GOQL - our graphical query language for object-oriented databases. The comparison is based on an example of an object-oriented database schema whose graphical representation is given in the most important graphical models that exist in the literature – including GOQL. The comparison illustrates the limitations of these models as compared with the UV of GOQL.

The paper presents the basic characteristics/concepts of an object-oriented database schema. Each graphical representation of the running example illustrates how each graphical model supports these characteristics in terms of graphical constructs. These graphical constructs of each graphical model are therefore evaluated within the context of the running example. Having evaluated these models, the paper continues and illustrates the graphical constructs of GOQL. The design of these constructs has been based on the concept of desktop metaphors. The paper presents an overview of this concept and explains the way they have been utilized in the design of the graphical constructs of the UV of GOQL. The emphasis on the design was to provide a graphical model that does not contain any of the perplexing details of an object-oriented database schema and therefore provide end-users with a user-friendly query environment for ad-hoc queries.

The UV of the language GOQL was first presented in [1]. The query mechanism of GOQL was presented in [2], whereas a more comprehensive part of the language is to appear in [3]. The work in [1, 2, 3] includes a survey of the various graphical models, an evaluation methodology of the features supported by these models and the design of the UV and query constructs of GOQL.

The purpose of this paper is to substantiate our decision for proceeding with the implementation of GOQL. The comparison presented illustrates the advantages of the UV of GOQL over the rest of the graphical models. The UV of GOQL therefore provides the basis for building the query mechanism of a graphical query language for object-oriented databases. The conclusions of course, presented herein, are based on our own views and perspective. Our near future work involves the evaluation of the graphical models presented in this paper, by a group of users of object-oriented databases so as to get a more independent set of conclusions.

Key-Words: Graphical Query Languages, Object-Oriented Databases, Desktop Metaphors

1. Introduction

A graphical representation of an object is used to provide a visual and better picture of the object, emphasizing on its most important/interesting characteristics. Graphical representations are being used in our every day life as well as in many scientific and research applications. In this paper, we examine one such application of graphical representations, namely the graphical database schemas. A graphical database schema is a graphical representation of a database schema. It can be derived either for an external database schema (addressed for naive users and being free of some perplexing technical details), or for a conceptual schema (addressed for database designers and developers). More particularly, this paper concentrates on graphical database models for object-oriented databases and shows the design of the GOQL graphical model, which utilizes the concept of metaphors.

The use of metaphors is one of the most successful types of teaching and is often used for children learning. Most of the times, in our attempt to explain something, we use examples from other subjects that we (or our audience) know better and feel more comfortable with, thus introducing metaphors. The most famous application of metaphors in software applications is the turtle in LOGO [4]. The illustration of painting by the movement of the turtle tail in the dust was very successful and helped especially the children to use LOGO more effectively.

In a similar way, metaphors can be used for database applications to make them friendlier to naive users. An office is probably the most suitable
area to look for the best metaphor for database applications. Database applications substitute the office records that are kept in folders and documents. Thus, it seems logical for an office employee to work more effectively using a database application whose environment consists of pictures from his/her working environment.

However, the use of metaphors needs careful consideration. An inappropriate metaphor can confuse users. For example, the ejecting of a disk has been illustrated in some computer systems by throwing a disk icon into a wastebasket. This metaphor is totally wrong and confuses users, since normally we throw garbage into the wastebasket and not a useful disk.

2. Different graphical models
In this section, eight different models for representing graphically a database schema are critically represented. A picture of the graphical schema is given for the example database (given in Appendix) for every model presented. Our evaluation examines nine basic characteristics that a graphical schema should represent, which are:

1. classes or entities (depending on the database model)
2. attributes (properties) of classes/entities
3. methods (properties) of classes
4. relationships between classes/entities
5. class/entities hierarchies
6. recursion (type of property (attribute/method) of a class/entity type is the same class/entity)
7. complex properties (type of a property is a composite property of more than one other properties)
8. collection properties (type of a property is of set or bag or list or array type)
9. metaphors (pictures/icons are used to represent the schema characteristics in a more natural way, hiding their technicality without losing their meaning).

Two are the basic categories of the graphical schema models, the condensed and the detailed models. A condensed model represents only classes/entities, relationships and hierarchies. A detailed model represents all the elements of the database external schema. The detailed model is better than the condensed because it presents a more complete picture of the database schema.

2.1 THE O.D.M.G. 2.0 MODEL
The graphical representation of the schema in the O.D.M.G. 2.0 model [5] is given in Fig. 1. Only the classes and the relationships between them are represented. Attributes, methods, complex properties and collection types are not shown. Directed arrows are used for the representation of the relationships between the classes. A thick directed arrow represents class hierarchy which starts from the child and finish to the father. Recursion is shown as a relationship that has as source and target the same class (e.g. Paper.Is_Referenced). Finally, there is no attempt for using metaphors to represent the parts of the graphical schema. We believe that the O.D.M.G. 2.0 attempt to represent graphically the schema is targeted to expert users and not to naïve users.

Fig. 1: The O.D.M.G. 2.0 Graphical Representation

2.2 THE COAD/YOURDON MODEL
The graphical schema of the COAD/YOURDON model [6] represents the class as a rounded box which has as a heading the name of the class, as middle part the names of the attributes without their types and in the bottom part the class methods. The relationships between classes are given as undirected lines of which on the two edges are written a) “1”, b) “0,m” or c) “1,m” depending on the type of the relationship. The hierarchy of the schema is given by lines between the classes using the symbol (          ). A complex attribute (such as place or Con_date in Proceedings) is given by another box (linked to the complex attribute of the class), which has as heading the name of the complex attribute. Collection attributes are not represented and metaphors are not used. We believe again that this model does not fully satisfy end-users.

Fig. 2: The COAD/YOURDON Graphical Representation
2.3 The Extended Entity Relationship model
The graphical representation of the Extended Entity Relationship (EER) model [7] is very detailed (Fig. 3). It has two representations for classes; a rectangle with the name of the class and a double-line rectangle for a class appearing again in the diagram. Attributes are represented by ovals and are linked to the class they belong. There is no representation for methods. A relationship is represented by a line with a named rhombus - the type of the relationship is written on the edges of the line. A special symbol (a double line ending in a D in circle) is used in the hierarchy representation (e.g. Document, Proceedings and Journal). A relationship of an entity to itself represents recursion. Complex attributes are linked to the attributes (given by ovals) they consist of. Collection types are not represented and metaphors are not used.

![Fig. 3: The EER Graphical Representation](image)

2.4 The AMAZE model
The AMAZE graphical model [8] represents the classes as 3D boxes, the relationships as directed lines and recursion as relationships, which have as source and target the same class (e.g. Paper.Is_Referred). Class Hierarchy is represented by a thick arrow from the child to the father.

In the AMAZE graphical schema, there is no representation for attributes, methods, complex attributes and collection types. Also metaphors are not used thus. We believe that the 3D representation does not give any advantage to the schema representation and that the lack of metaphors makes this model usable only by expert database developers and not by end users.

![Fig. 4: The AMAZE Graphical Representation](image)

2.5 The G-LOG Model
The G-Log model [9] is detailed but confusing. In this model classes are represented by rectangles. Properties, class hierarchy, relationships and recursion are all represented with arrows, thus leading to confusion and not distinctive representation constructs. This makes it very difficult for both end-users and database experts to understand the underlying database schema since many database elements are represented with the same geometrical shape. Complex attributes are represented as classes. Finally, collection types are not represented and no metaphors are used.

![Fig. 5: The G-Log Graphical Representation](image)

2.6 The GOMI Model
In the GOMI model [10] the root OBJECT (Fig. 6) represents the top superclass of the schema hierarchy. That OBJECT is linked to the immediate subclasses by thick arrows (i.e. class hierarchy is represented by a thick arrow from the superclass to the subclass). Tables are used to represent classes. Attributes are listed in the tables together with their types. Methods are not supported. Two different types of arrows represent relationships. A dotted arc represents the “one to many” relationship and a dashed arrow represents the “many to many” relationship. Finally, recursion is represented as a relationship from a class attribute to the same class itself (e.g. Paper.Is_Referred).

GOMI’s model gives a nice feel for the class hierarchy but uses too many lines and arcs that make the schema confusing and sometimes look like a maze. This model again does not use...
metaphors and methods. We believe that it is very difficult for the naive user to understand this model.

![Diagram](image1.png)

**Fig. 6: The GOMI Graphical Representation**

### 2.7 The KHOSHAFIAN model

Another graphical model is the Khoshafian model [11] (Fig 7). In this model, rectangles represent classes and arrows represent properties. A dashed-oval gives the type of an attribute and a double-line oval the type of a method. Arrows represent relationships with the relationship name labelling the arrow and thick-white arrows represent hierarchies that start from the children and conclude to the parent classes. An arrow starting from a class and returning back to the class represents recursion. Dashed-rectangles represent complex attributes. A circle with star represents a set type whereas a circle with an “L” represents a collection type. Metaphors are not used. This model provides a lot of useful information to the expert database person but a naive user will have problems to understand and work using such a graphical schema.

![Diagram](image2.png)

**Fig. 7: The Khoshafian Graphical Representation**

### 3. THE GOQL MODEL

The GOQL model [1,2,3] has utilized the concept of desktop metaphors in representing graphical constructs. With the end-users in mind, GOQL hides any perplexing details and presents to users database schemas that are easy to understand. The GOQL graphical representation of the schema of the running example of the paper is given in Fig. 8. A comparison table summarizing the supporting features of each of the models presented herein is given in Fig. 9.

![Diagram](image3.png)

**Fig. 8: The GOQL Graphical Representation**

![Table](image4.png)

**Fig. 9: Comparison of the 8 Models**

In the GOQL model, metaphors have been used to represent the database schema features. More specifically the database schema constructs have been represented as follows:

1. classes are represented by tables
2. attributes are represented by cells in tables
3. methods are also represented by cells in tables (users do not distinguish between attributes and methods; they know these as properties)
4. relationships are represented by folders/briefcases (the class related to an attribute is represented by a folder/briefcase and is linked with a line to the attribute).
5. class hierarchy is incorporated into the folders/briefcases; a folder/briefcase can be opened to reveal any subclasses
6. recursion is represented by a relationship from a class to itself
7. a complex property is shown by an envelope which is placed in the right edge inside the cell of the property; a user can open the envelope to reveal the properties that constitute the complex property
8. a collection property (set, bag, list or array) is shown by a paper clip that is placed in the top right edge inside the property cell.

The design of the graphical representation of GOQL that lead to the decision to incorporate a condensed graphical representation (Fig. 10), has taken into consideration the following methods that deal with database schemas consisting of too many classes:

**Top-Down:** With this method data are grouped into *interesting* levels. The most well known method is the hierarchical where in the first level the most general data are represented and in the other levels the more detailed data are represented. For example, for the object-oriented model, the classes and the class hierarchy are shown in the top level and the other details of the database (properties, relationships, etc.) in the next levels.

**Browsing:** With this method a user can have as a starting point an element in the database schema and then ‘travel’ to the next ‘relative’ by a relationship. This method is more successful when the user has knowledge of the database structure because the change from a level of detail to another is developed slowly.

In the condensed GOQL graphical representation, all the classes are represented by closed folders. The user can select one of the folders to ‘open’ it in order to examine all the features of the class. By this selection the schema is moving one level down (top down method) for the class under consideration. By browsing through relationships, the user can open a related folder. Using the relationship browsing a user can therefore develop the part of the schema that is needed for a query (Fig. 11).

4. Conclusions
In this paper we have illustrated through an object-oriented database schema the various graphical models that support a graphical environment for databases, including our GOQL model. Through this illustration, a comparison of these models has been made based on the various features of object-oriented databases. The paper has also presented the design of the GOQL model based on the concept of desktop metaphors. The advantages of the GOQL model have also been addressed. We believe that the conclusions drawn from this comparison reinforce our decision to proceed with the full design and implementation of the GOQL model. Our near future work involves the evaluation of the graphical models presented in this paper, by a group of users of object-oriented databases. This will give us a more independent set of conclusions.

Fig 10: The Condensed GOQL Graphical Representation

![Fig 10: The Condensed GOQL Graphical Representation](image)

Fig. 11: Schema Developing in the GOQL Model

![Fig. 11: Schema Developing in the GOQL Model](image)
References:


APPENDIX – The Running Database Example

ISBN : alphanumeric, Title : alphanumeric, Pubs_Name : alphanumeric, Price : numeric, Editors_Name : alphanumeric, Papers_Title : alphanumeric, Pages : numeric, Year : numeric, PRIMARY KEY (ISBN), FOREIGN KEY (Pubs_Name) REFERENCES Publisher, FOREIGN KEY (Editors_Name) REFERENCES Editor, FOREIGN KEY (Papers_Title) REFERENCES Paper.

Journal ISBN : alphanumeric, Title : alphanumeric, Pubs_Name : alphanumeric, Price : numeric, Editors_Name : alphanumeric, Papers_Title : alphanumeric, Pages : numeric, Year : numeric, Volume : numeric, Number : numeric, PRIMARY KEY (ISBN), FOREIGN KEY (Pubs_Name) REFERENCES Publisher, FOREIGN KEY (Editors_Name) REFERENCES Editor, FOREIGN KEY (Papers_Title) REFERENCES Paper.

Proceedings ISBN : alphanumeric, Title : alphanumeric, Pubs_Name : alphanumeric, Price : numeric, Editors_Name : alphanumeric, Papers_Title : alphanumeric, Pages : numeric, Year : numeric, PRIMARY KEY (ISBN), FOREIGN KEY (Pubs_Name) REFERENCES Publisher, FOREIGN KEY (Editors_Name) REFERENCES Editor, FOREIGN KEY (Papers_Title) REFERENCES Paper.

Person Name : alphanumeric, Sex : alphanumeric, Date_of_birth : date, Email : alphanumeric, Web_page : alphanumeric.

Publisher Name : alphanumeric, Address : alphanumeric, Tel_no : alphanumeric, Fax : alphanumeric, Web_page : alphanumeric, Publish_ISDN : alphanumeric PRIMARY KEY (Name), FOREIGN KEY (Publish_ISDN) REFERENCES Document.

Editor Name : alphanumeric, Sex : alphanumeric, Date_of_birth : date, Email : alphanumeric, Web_page : alphanumeric, Docs_ISDN : alphanumeric PRIMARY KEY (Name), FOREIGN KEY (Docs_ISDN) REFERENCES Document.

Author Name : alphanumeric, Sex : alphanumeric, Date_of_birth : date, Email : alphanumeric, Web_page : alphanumeric, Papers_title : alphanumeric PRIMARY KEY (Name), FOREIGN KEY (Papers_title) REFERENCES Paper.

Paper Title : alphanumeric, Authors_Name : alphanumeric, Pub_ISDN : alphanumeric PRIMARY KEY (Title), FOREIGN KEY (Authors_Name) REFERENCES Author PRIMARY KEY (Title), FOREIGN KEY (Pub_ISDN) REFERENCES Document.

Con_Date PRIMARY KEY (Start_date, End_date), FOREIGN KEY (Start_date) REFERENCES Con_Date, FOREIGN KEY (End_date) REFERENCES Con_Date.

Place City : alphanumeric, Country : alphanumeric PRIMARY KEY (City).