Relationship Abstractions for an Effective Hypertext Design: Augmentation and Globalization

Yoshinori Hara [*] *Arthur M. Keller* [†] *Gio Wiederhold* [‡]

NEC Corporation and Stanford University [§]

Advanced Decision Systems and Stanford University [§]

Stanford University [§]

Abstract

Data abstractions, i.e., aggregation and generalization, are useful for representing complex objects effectively. They provide high level semantic constraints as well as extend the capabilities of entity description in the E-R model. However, corresponding concepts of relationship abstraction are not directly available, particularly in hypertext systems.

We propose two types of relationship abstractions, *augmentation* and *globalization,* aiming at the improvement of relationship design phases. The former is an abstraction which turns information held in relationships into that of attributes for existing entities. The latter is an abstraction which generates global-to-local relationship hierarchies. We show the advantages of these abstractions.

1 Introduction

Hypertext and hypermedia [YHMD88, Hala88, HaKa88, etc.] are based on a network or a graph data structure, as are semantic networks [Quil68, LeMy79] and some aspects of the structural model [Wied83]. Two advantages of graph

[*]e-mail address: hara@eclipse.stanford.edu
[†]e-mail address: ark@eclipse.stanford.edu
[‡]e-mail address: gio@cs.stanford.edu
[§]Authors' address: Department of Computer Science, Stanford University, Stanford, CA 94305-2140, U.S.A.

models are navigational access and their associative connection strategy. Navigation has the potential to support users to get target information from ill-structured information spaces. Navigation is one of the most useful retrieval clues for hypermedia, since multimedia objects are essentially composed of non-specific attributes. The connection strategy is also useful for an individual user, because he or she can easily relate objects together without the requirement of specifying a database schema. This rough but simple strategy enables users to handle heterogeneous objects homogeneously in such fields as personal, office, and educational information systems.

However, recent contributions to hypertext systems depend highly on hardware/software progress and the extension of hypertext functionalities. There are very few design strategies for hypertext systems, i.e., how to organize hypertext structures effectively [JoMa90]. Consequently, whether the hypertext structures are well organized or not depends highly on the individual designer's skill [Niel90].

As for conceptual database design, several well-formulated models have been proposed in order to provide the effective mapping into the target structures. The concept of database abstractions [SmSm77] is one of the most useful ways of thinking to capture the high level semantics Two abstraction concepts, i.e., *aggregation* and *generalization,* are widely used in the object-oriented databases [ABDD89], the semantic databases[HuKi87], etc. However, they are not concerned about relationship abstraction as much, which is more essential in hypertext or other network-based knowledge representation models.

This paper proposes the relationship abstractions to support hypertext design phases. They are *augmentation* and *globalization,* which are the extension of aggregation and generalization, respectively. We show the advantages of these abstractions by using an example of a movie database.

Section 2 describes the concept of database abstractions and its problems. In Section 3, we propose a model of relationship abstractions by extending the database abstractions. Section 4 presents a hypertext design strategy using our model. In Section 5, we discuss other possible application fields such as schema integrations and visual interfaces.

2 Database Abstractions

The concept of database abstractions is a fundamental model to incorporate real world semantics into computational forms. Similar concepts are also widely used in AI and software engineering fields as well. We describe a summary of database abstractions.

2.1 Aggregation

Aggregation refers to an abstraction in which related attributes in an object are regarded as a higher level attribute. In the relational model, it is an abstraction which integrates vertical elements in a relation, i.e., attributes. In other words, the type of an attribute is extended to a more complex structure and users can capture these attributes in any levels of abstraction.

Attributes can be organized as an aggregation hierarchy using "is-part-of" relation. In the case that an aggregated attribute is for the composition of objects, we obtain a *composite object* [WoKi87]. In the structural model [Wied83], *ownership relation* can be treated as aggregation.

Usually, an aggregation hierarchy is defined such that there is no inheritance mechanism between lower level and upper level of abstractions. However, *property inheritance,* i.e., a bottom-up inheritance, can be executed in some systems in order to provide information of a higher level based on that of lower levels.

Aggregation is useful because it has the capability for different users to access objects at different levels of abstractions. It contributes to raising the flexibility in a utilization phase as well as to providing the extensibility in a design phase. However, this contribution is mainly to the extension of entity description in the E-R model.

2.2 Generalization

Generalization refers to an abstraction in which objects having common attributes are regarded as a generic object. In the relational model, it is an abstraction which integrates the common attributes in horizontal elements in a relation, i.e., tuples. In the object-oriented model, this abstraction is represented as a superclass-subclass relation.

A set of tuples can be organized as a generalization hierarchy using "is-a" relation. The inverse operation of generalization is called *specialization.* In the structural model, *subset relation* can be treated as generalization.

Compared with aggregation, there is a (top-down) inheritance mechanism in the generalization hierarchy. That is, the attributes in an upper level are common, so that they inherit to its lower levels. If the attribute name is the same in both levels, the value of the attribute in the lowest level is used.

Generalization is useful because it provides a high level semantic structure and its compact representation. It also contributes to the information usability both in a utilization phase and in a design phase. However, it is mainly focused on the complex structures of entities in the E-R model.

2.3 An example

Let us consider the design of a database of information about movies in order to illustrate the example of aggregation and generalization. Figure 1 shows an example of abstractions on the relational model. We can recognize how these abstractions are organized.

Note that we are not concerned with the abstraction for relationships. If we concerned about the relationship abstraction as shown in Fig. 2, they are aggregated as one aggregated object or decomposed into two many-to-one relationships. A more precise semantic structure for relationships is not yet available.

2.4 Other types of abstractions

There are some other types of abstractions besides aggregation and generalization. They are versions, time abstractions, etc. They are useful particularly for such specific applications as CASE and CAD. However, since they depend on specific semantics, these issues are beyond the scope of this paper.

3 Relationship Abstractions

In this section, we extend the concept of abstraction for relationships in the E-R model. It is similarly applicable for *associative links* in the hypertext, i.e., self relationships in the E-R model.

An objective of relationship abstraction is to prevent excess relationships. The excess causes not only a decrease in the efficiency of navigation, but also a deterioration of the semantics of relationships. As a result, the usability of hypertext information may be decreased.

Another objective of relationship abstraction is to capture high level semantic structures in a relationship. It contributes to providing fruitful queries.

3.1 The properties of a relationship

Let us consider the properties of a relationship first. Its properties are different from those of an entity, although both of them can be represented as an equivalent form in the relational model. As shown in Fig. 3, the different properties are as follows:

(1) The key of relationship relation R_R is the combination of two foreign keys.

The relation of relationship R_R consists of three attributes. Two of them are foreign keys of corresponding entity relations R_{E_1}, R_{E_2}. The other is an attribute of the relationship or an aggregation attribute of the relationship. Due to having foreign keys, the existence of relationships is affected by the updating of corresponding entity relations.

A labeled directed graph can be equally represented except that the relevant element sets are disjoint or not. The first two attributes are input and output node IDs. They are considered to be foreign keys. The remaining is the attribute of link types.

(2) The number of attributes in a relationship relation is few.

The number of attributes in a relationship relation is relatively fewer than that in an entity relation. There are less semantic structures in the relationship relation. Therefore, it is difficult to specify useful semantic constraints such as functional dependencies.

(3) The number of tuples in a relationship relation is relatively large.

The number of tuples in a relationship relation is on the order of the product of the number of tuples in the corresponding entity relations. In other words, the number of tuples in a relationship relation turns out to be much larger than that in an ordinary entity relation, particularly when the relationship becomes complex.

3.2 Problems of previous works

Two problems arise when we apply database abstractions to relationship relations. One is that aggregation has little meaning for relationship abstraction. This is because the number of attributes in a relationship is so few that there is little need for aggregation.

The other problem is that generalization cannot be directly applied to relationship abstraction. Except in some trivial cases, generalization is mainly focused on a class of entity objects into a generic object. Even when it is applied to a relationship, this operation is only the separation of tuples in the relationship relation.

3.3 Requirements for relationship abstraction

In order to improve relationship design and maintenance phases, we set out the following three requirements for relationship abstraction. We must extend database abstractions in order to satisfy the requirements.

Firstly, one of the essential potentials of abstraction is to translate information into a more compact representation. This involves more than the separation of information. Secondly, regardless of which operations we apply, the decomposed structures have the same information as its original information. On the other hand, the reorganized structure has no redundancy nor inconsistency compared with its original information. Thirdly, each portion of an abstracted structure can be named. That is, the abstracted relationship structure can be connected to the real world semantics.

3.4 Augmentation

There are some *visible* and *hidden* meanings in relationship attributes or link types. Let us consider a simple example of a relationship between actors/actresses and the movies they cast. This relationship is a many-to-many connection, and its relationship name is simply "casts in." However, there are several meanings behind it. The reason why an actor or an actress casts in a movie may be caused by his or her affiliation, the conductor's character of the movie, etc. These attributes often make a relationship between entities.

Augmentation proposed here is a relationship abstraction which turns information held in relationships into attributes in the corresponding entities, as illustrated in Fig. 4. It is a decomposition or an aggregation of attributes, i.e., a vertical operation which is similar to aggregation described in Section 2. However, since such a primitive attribute is related to a possible entity attribute, we can substitute the relationship information for the entity attributes and the relationship among them. This substitution often produces a more compact representation of relationship, since a real world relationship is correlated.

Several methods for augmentation can be applicable. *ACE clustering* [HaKW91] is one of them. It is a clustering method considering exceptions. A relationship relation can be turned into new attributes of entities given by the clustering. The number of the attribute values is smaller than that of original relationships.

Once we get the attributes, the original relationships can be calculated by join operation or the function between the attributes. Therefore, we recognize the reason caused by the entity attributes that makes the relationships between the entities.

Following is a summary of possible patterns for augmentation.

(1) Translated attribute types

One possible translation is that the information of relationships can be turned to some existing entity attributes, called *visible attributes.* When there is a strong correlation between relationships and some of the existing attributes, we can set them out as visible attributes.

Otherwise, we have to consider another reason that makes relationships be organized. Namely, the information of relationships can be turned to some non-existing entity attributes, called *hidden attributes.* They can be extracted by clustering the elements of entities.

(2) Mapping patterns

There are three types of mapping patterns between relationships and entity attributes. One is *overlapping mapping,* which is a many-to-many mapping between relationship elements and target attribute values. Another is *non-overlapping mapping with exceptions,* which is a many-to-one mapping from a relationship element to a target attribute value, e.g., clustering number, allowing some exceptions. The other is *recursive non-overlapping mapping,* which is plural many-to-many mappings between relationships and entity attributes. It is organized as the same attribute hierarchy as aggregation.

The most practical mapping is the second case, i.e., the non-overlapping mapping with exceptions. ACE clustering is an efficient method to organize this mapping.

(3) The relationship among attribute values

Two possible relationships among attribute values can be set out as an alternative representation for the original relationship information. The same attribute value means the simplest representation of relationship. The second simplest one is that there can be set out a relationship between two attribute values. In other cases, such translations are useless, because they turn out to be a more complex representation of relationships.

3.5 Globalization

The second relationship abstraction is defined by the degree of globality in a relationship relation. Some relationships are treated as global structures, and some are only effective within local structures. A local structure is, in other words, represented as the compensated information of its global structure. Therefore, we can organize a global-to-local relationship hierarchy, which is similar to the generalization-specialization hierarchy described in Section 2.

Globalization is a relationship abstraction which generates the global-to-local relationship hierarchy, as shown in Fig. 5. It is a grouping operation for tuples in a relationship relation, i.e., a horizontal operation similar to generalization. Since relationships in an upper level can inherit to its lower levels, relationships in the lower levels obey those in the upper level unless there is a concrete description of relationships in the lower levels.

Globalization is different from generalization by following two points. One is that the attribute in globalization is invariant and only its attribute values may change. In this meaning, the global-to-local relationship hierarchy is similar to the exception relationship hierarchy. The other point is that the instance of globalization can appear in any levels. On the other hand, the instance of generalization appears only in the lowest level.

3.6 An example: A movie database

Let us consider the design of a movie database in order to illustrate the example of augmentation and globalization. Figure 6 shows an example of abstractions on the relational model. The original relationships can be translated into the attributes of each entity, and its subclasses.

We are evaluating such clusterings for a movie database with 6250 movies, 5300 actors/actresses, 22000 cast entries, and 1450 directors. Some of the "casts in" relationships can be substituted by the directors and the movie categories such as horror, drama, etc.

4 A Strategy for Making Effective Hypertext Structures

In this section, we discuss a hypertext design strategy using the relationship abstractions. Hypertext systems are categorized by several viewpoints, such as scope of the user target (single or collaborative), browsing versus authoring, and task specificity [Hala88]. Here we discuss the improvement of design phases for single hypertext systems, which may be the basic component for other systems.

4.1 A hypertext design process

We have been developed several hypertext/hypermedia prototypes, such as an electronic filing system [KaHa86], an electronic encyclopedia for fine arts [HaKa88], and the PENGUIN system [BaCW89], which aims to combine database facilities with a hypertext interface. From the experiences gained with these systems, a design process for making effective hypertext structures can be set out as following five steps.

(1) Macro framework design and the focus of materials

Initially, a hypertext designer specifies a macro framework of the target system. It consists of global node structures, the relationships among them, and the focus of materials. The framework is, as it were, an implicit semantic constraint imposed on individual nodes and links.

It is important to specify the focus in the structure such as key nodes and key path sequences. They represent the author's intention, and readers are guided to read the materials according to the guidance. Otherwise, a hypertext structure is only like a complicated web.

(2) Node design

Secondly, individual nodes are specified. In this phase, the concept of database abstraction is helpful to organize nodes, as described in Section 2. A hierarchy of "is-a" relationship is also useful when combining the internal data structure with the objects seen at the human interface. When dealing with hypermedia, we also have to consider the medium to represent nodes for target objects and their access clues.

(3) Link design : a bottom-up organization

Then, relationships among the nodes are specified. There are several link types, such as *static link, dynamic link,* and *virtual link* [HaKa90]. Almost all of them are based on a bottom-up organization. That is, links are locally organized and the whole hypertext structure is formulated as a result. This linking strategy is an essential point to create a hypertext system and to maintain it.

(4) Evaluation of global consistency

An important point for good design is a feedback evaluation in any phases. In particular, the comparison between the macro framework and the link design is the most important to improve the usability of hypertext structures. However, it depends highly on the designer's skill, since there is no effective method to specify the consistency.

Of course, there are some practical rules to prevent excess links and to exclude useless links. One is to eliminate the indirect relationships from the structure, which are calculated as a transitive closure. Another method is to set out an attribute of link priority according to the user's interest. However, there is not an effective method to analyze the link design.

(5) Restructuring

The last one is a restructuring phase in order to incorporate the node/link design with the macro framework. By restructuring, a hidden but important structure may be clear, as well as the author's intention is more emphasized. When a hypertext system is larger, the evaluation and the restructuring phases are much necessary.

4.2 A design strategy using relationship abstractions

Relationship abstractions are helpful for the evaluation phase. They show an easier way to compare the link design with the macro framework. In some cases, they can automatically detect the differences between them and provide an alternative design.

Relationship abstractions are useful when the designer cannot specify the constraints in a relationship at the designing phase. The designer can recognize the constraints and improve the structure by analyzing the data. This bottom-up organization will contribute to providing an effective hypertext structure.

5 Other Applications for Relationship Abstractions

In this section, we briefly discuss other applications for relationship abstractions.

Firstly, since relationships can be organized as a higher level structure, it is possible to apply for schema integration in a collaborative environment. Usually, an individual user has his or her own view and it is difficult to integrate the whole structure. However, some common attributes are extracted by the abstractions, so that these attributes may be teated as a basis for schema integration.

Secondly, relationship abstractions can apply to improve the visual interface for recognizing large information spaces. That is, the potential exists for creating overview diagrams based on the abstractions, which are graphical browsers using aggregation. They are very important when the information space is too large for every node and link to be shown on a single map [Niel90].

6 Conclusion

We have presented the two types of relationship abstractions to improve relationship design phases. Augmentation is an abstraction which turns information held in relationship into that of attributes for existing entities. Intuitively, it is a translation of individual relationships into more stable states, since entities are likely to exist longer than relationships. Globalization is an abstraction which generates global-to-local relationship hierarchies. Local relationships provide compensated information of their global relationships. The experimental results show the effectiveness of these abstractions. They are helpful for the evaluation phase of hypertext design processes.

Acknowledgements

We would like to thank Dimitris Karagiannis, Jack Milton, Jim Diederich, Donovan Hsieh, Peter Rathmann, and Shaibal Roy for their helpful comments.

References

[ABDD89] Atkinson, M., Bancilhon, F., et al. "The Object-Oriented Database System Manifesto," *DOOD'89,* 1989, pp. 40-57.

[BaCW89] Barsalou, T., Chavez, R. M., and Wiederhold, G. "Hypertext Interfaces for Decision-support Systems: A Case Study," *MEDINFO'89,* 1989, pp. 126-130.

[Conk87] Conklin, J. "Hypertext: An Introduction and Survey," *IEEE Computer,* Vol. 20, No. 9, 1987, pp. 17-41.

[Hala88] Halasz, F. G. "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems," *Communications of the ACM,* Vol. 31, No. 7, 1988, pp. 836-852.

[HaKa88] Hara, Y., and Kaneko, A. "A New Multimedia Electronic Book and Its Functional Capabilities," *User-oriented, Content-based, Text and Image Handling (RIAO),* 1988, pp. 114-123.

[HaKa90] Hara, Y., and Kasahara Y. "A Set-to-Set Linking Strategy for Hypertext Systems," *ACM Conference on Office Information Systems,* 1990, pp. 131-135.

[HaKW91] Hara, Y., Keller, A. M., and Wiederhold, G. "Implementing Hypertext Database Relationships through Aggregations and Exceptions," *Stanford Technical Report (to appear),* 1991.

[HuKi87] Hull, R., and King, R. "Semantic Database Modeling: Survey, Applications, and Research Issues," *ACM Computing Surveys,* Vol. 19, No. 3, 1987, pp. 201-260.

[JoMa90] Jonassen, D., and Mandl, H. (eds.) "Designing Hypermedia for Learning," *Springer-Verlag (Proc. of the NATO Advanced Research Workshop 1989),* 1990.

[KiBG89] Kim, W., Bertino E., and Garza, J.F. "Composite Objects Revisited," *ACM SIGMOD'89,* 1989, pp. 337-347.

[LeMy79] Levesque, H., and Mylopoulos, J. "A Procedural Semantics for Semantic Networks, Associative Networks," *Academic Press,* 1979.

[Niel90] Nielsen, J. "Hypertext and Hypermedia," *Academic Press,* 1990.

[Quil68] Quillian, M. R. "Semantic Memory," *Semantic Information Processing, MIT Press,* 1968.

[SmSm77] Smith, J. M., and Smith, D. C. P. "Database abstractions: Aggregation and Generalization," *ACM Trans. on Database Systems,* Vol. 2, No. 2, 1977, pp. 105-133.

[TWBK89] Teorey, T. J., Wei, G., et al. "ER Model Clustering as an Aid for User Communication and Documentation in Database Design," *Communications of the ACM,* Vol. 32, No. 8, 1989, pp. 975-987.

[Wied83] Wiederhold, G. "Database Design," *McGraw-Hill,* 1983.

[Wied87] Wiederhold, G. "File Organization for Database Design," *McGraw-Hill,* 1987.

[WoKi87] Woelk, D., and Kim, W. "Multimedia Information Management in an Object-Oriented Database System," *Proc. of the 13th VLDB Conference,* 1987, pp. 319-329.

[YHMD88] Yankelovich, N., Haan, B. J., Meyrowitz, N. K., et al. "Intermedia: The concept and the construction of a seamless information environment," *IEEE Computer,* Vol. 20, No. 1, 1988, pp. 81-96.