PROCEEDINGS of

4th International Hong Kong Computer Society

Database Workshop on

DATABASE MANAGEMENT HORIZON

December 12-13, 1992

Organized by

Hong Kong Computer Society, Database Special Interest Group

and

Supported by

Informix System Hong Kong
Ingres System Hong Kong
Oracle
HCL Leung (Sybase)

City Polytechnic of Hong Kong
Chinese University of Hong Kong
Hong Kong Baptist College
Hong Kong Polytechnic
Hong Kong University of Science and Technology
Open Learning Institute of Hong Kong
University of Hong Kong

Edited by

Joseph Fong
City Polytechnic of Hong Kong

Professor Vincent Lum
The Chinese University of Hong Kong

Compiled by

Choi yiu kuen
City Polytechnic of Hong Kong

FORWARD

These proceedings contain the papers chosen for presentation at the 4th International Hong Kong Computer Society Database Workshop held on December 12-13. This year the Workshop has been significantly expanded in scope in several aspects. First it has been expanded to solicit papers worldwide and it was successful to have attracted papers from 4 continents and many countries. This is evident from the papers included in the proceedings. Second, it has several invited papers from very reputed and renowned professionals in the field of database. They brought to us authoritative views of the most up-to-date researches and perspectives in this field. Third, we have been successful to have vendors of database systems to present papers that will allow us to get a glimpse of their thoughts and therefore products that will be forthcoming.

Thus true to our billing, we have indeed fulfilled our promise of the theme of 'Database Management Horizon'. Because of the large number of papers, we have to run parallel sessions and streams which reflect our theme: an 'Exploratory Technology' stream that contains papers exploring the different advanced technologies to be applied in the database world, an 'Object Oriented DBMS' stream that contains papers in an area promised to be the major breakthrough since the development of the relational database systems, and the 'Application' stream that contain the works being developed into products.

Overall the Workshop presents papers for users to see the works that will be forthcoming in the future and provides a forum for users and researchers to exchange views and interact. It is hoped that this cross fertilization will richly enhance each side to further the development of database in the coming years.

On behalf of the Workshop we want to thank the Program Committee members for their hard work to referee the many papers, and to the many Organizing Committee members who have done so much to make the Workshop successful. We are especially grateful to our invited speakers who travelled far to come to speak to us at the Workshop. We hope that all of you will enjoy and benefit from the efforts of so many individuals who unselfishly contributed so much.

Joseph FONG
Workshop Chairman

Vincent LUM
Program Committee Chairman

Table of Contents

Keynote Papers

Technology Exploration

Object-Oriented Database

Algorithms for Access Relevance to Support path-Method Generation in OODBs[1]

Ashish Mehta, James Geller, Yehoshua Perl and Peter Fankauser
Institute for Integrated Systems, CIS Dept., and CMS, NJIT,
Newark, J 07102, E-mail: ashish@earth.njit.edu
Phone: (201)596-5655, fax: (201) 596-5757
GMD-IPSI, Dolivostr. 15, Darmstadt, Germany,
E-mail: fankhaus@darmstadt.gmd.de

ABSTRACT

To retrieve and to update distant information from a class in an object–oriented database (OODB) is a difficult task. Because it might require comprehensive knowledge of all the classes of the conceptual schema, but a typical user has incomplete knowledge. A *path–method* is a mechanism to access a particular distant item of information from a class. A path–method is a method which traverses from one class through a chain of connections (i.e., user–defined and generic relationships) between classes to access information at another class. Currently we are developing a tool *Path–Method Generator* (PMG) which generates path–methods automatically according to a naive users' requests. One algorithm of PMG requires access relevance between pairs of classes as a guide for the traversal of an OODB schema. In this paper we present efficient algorithms to compute *access relevance* between all pairs of classes of an OODB schema. These access relevance will be used as a guide for path–method generation.

Keywords: Object-Oriented Database, Path-Method, Access Relevance, Triangular norms, Schema Traversal

1 Introduction

The task of retrieving distant information for a class in an OODB requires complete knowledge about the structure and content of the OODB. In a large OODB this task is difficult for a typical user who has incomplete and/or inconsistent knowledge of the conceptual schema. A *path–method* is a mechanism for such a purpose. A path-method is a method which traverses from one class through a chain of connections (user–defined and generic relationships) between classes to retrieve information from a distant class. The writing of such path-methods requires

[1]This work has been partially supported by the New Jersey State Sponsored Chair in Computer Science at NJIT

traversal of the OODB schema. Path–methods [NPGT91] are discussed for example in [KKS92, LVZ92] using the terms path–expression and path, respectively. Currently, we are developing a *Path–Method Generator* (PMG), which consists of a collection of traversal algorithms to generate desired path-methods automatically. A human will traverse an OODB schema by applying his intuitive understanding of the classes and their connections. To imitate the human traversal [MGPN92], the PMG will use as a guide for the traversal precomputed *access relevance* values between all the pairs of classes in the OODB schema. The PMG can be considered as a *powerful underlying traversal tool* for *schema independent query formulation* [KN89], *query resolution from users' tokens* [M86, L85, MU83] and *dynamic derivation of personalized views* [NS88].

Following, e.g., VML [KNBDF91], GemStone [BOS91], and ORION [KKS92, K90], we are modeling an OODB schema as a directed graph. Note that a directed graph of a schema may contain cycles. Classes are represented as nodes. Directly related classes are connected by a directed edge with an access weight from the range [0, 1]. We assign an access weight to each connected pair of nodes in the schema graph. The *access weight* of a connection from a class a_s to a class a_t is a measure of its significance according to the frequency of traversing this connection relative to all connections emanating from the class a_s. The access weights are assigned according to the frequencies of use of the connections accumulated during the operation of the OODB. Initially, frequency information for access weights is not available and they are assigned by the schema designer, based on his understanding of the application domain. Specific rules for assigning access weights to the connections of a schema are discussed in the next section.

The significance of a path is measured by the *access relevance* (AR) *value*. The *access relevance of a path* is obtained by applying a t–norm [FKN91, K91, Z65, KF88] over the set of access weights of the edges of the path. There exist several infinite families of t–norms and corresponding conorms [SS61]. However, in [BD86] it is empirically shown that for most practical purposes two to five different t–norms suffice. From these we have chosen PRODUCT and the more optimistic MINIMUM. For example, for the commonly used t–norm PRODUCT, the access relevance of a path is the product of the access weights of all its edges. The access relevance between non–adjacent classes a_s and a_t is a measure of the significance of the indirect connection from a_s to a_t. For performing the union for multiply indirectly related classes we use the co–norm MAXIMUM. Thus, the *access relevance from a class a_s to another class a_t* is the maximum access relevance over all paths from a_s to a_t. For these cases, we present efficient algorithms which determine the access relevance between all pairs of classes.

We now summarize our motivation for using the access relevance for the Path–Method Generator (PMG). We have introduced the notion of access weight of a connection as a measure of its significance according to the frequency of its use during the operation of the OODB. Our PMG algorithm [MGPN92] has to decide of every step on the connection to be traversed from a source class s. The underlying philosophy of our approach in this paper is that traversing

commonly used connections is preferred for path–method generation. That is, traversing the most frequently used connections will more likely generate desired path–method than choosing arbitrary connections. However, a desired path–method may contain connections which are seldomly used. For example, a best first algorithm that chooses the connection of highest frequency at every step lacks the look–ahead property helpful to create the desired path–method. The decision of our PMG algorithm is based on the access weight of the connection to a neighboring class u and the access relevance from u to the target class t, that is, on the significance of the direct connection (s, u) and the significance of the indirect connection from u to t. The same process will be repeated for the other steps of the path–method generation. Our experiments [MGPN92] show that traversal of a schema according to the above rules will generate the desired path–method more successfully and more efficiently than a uniform traversal. For example, for the PRODUCT t–norm, the PMG algorithm found 84% of the desired path–methods in the first phase and the rest in its second phase in which the user utilizes the feedback from the first phase to set some parameters. Further experiments [MGPN92] show that the results of the PMG algorithm are better than for best first search.

The presentation in this paper uses the most significant features of an abstract OODB model rather than a specific one. The reason for this choice is to present the computation of access relevance between classes in a way that can be implemented on a variety of OODB systems. This general representation emphasizes the possibility of computing access relevance in an Interoperable Multi–OODB containing OODBs of different models. Although this abstract OODB model is general enough to reflect a variety of existing OODB models, we use some of the terminology of the Dual Model [NPGT89, NPGT90, NPGT91, GPN91] of the VML system [KNBDF91], but not referring to its separation of structure and semantics.

A class description consists of the following four kinds of properties: attributes, user–defined and generic relationships, and methods. Attributes specify values of a given datatype while relationships specify pointers to other classes. Generic relationships are system supported relationships of general nature. We consider two specialization generic relationships: *categoryof* (*roleof*) for the case where the superclass and the subclass are in the same (different) context. Two other generic relationships, *memberof* and *setof* connect a set class and its member class.

The remainder of this paper is organized as follows. In Section 2, we discuss examples and formally define the problem of determining the access relevance of a pair of classes. In Sections 3 and 4, we present and validate two efficient algorithms using the PRODUCT and MINIMUM weighting functions, respectively. In Section 5, a more efficient algorithm is presented for the MINIMUM weighting function for the special case of bidirectional schemas. Section 6 contains conclusions.

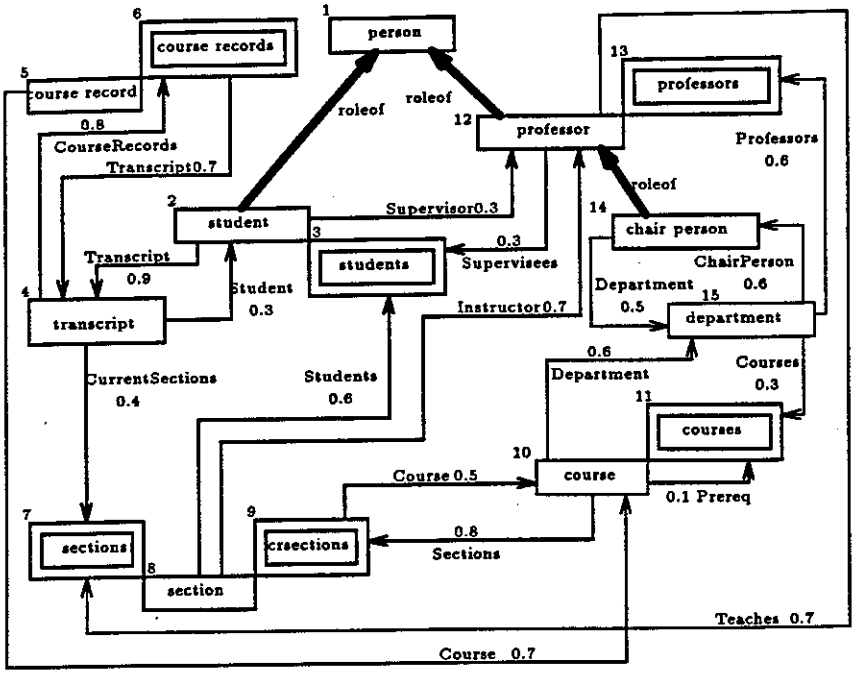

Figure 1: A subschema of a University Database

2 Examples and Formal Definition of the Problem

Let us consider a subschema of a university database which contains information regarding students, courses, etc. This subschema is shown in Figure 1. A directed graph G(V, E) corresponding to the schema of Figure 1 is shown in Figure 2. This representation is in OOdini, a graphical schema representation language and system [HGPN92]. A rectangle represents a class, and a double line rectangle represents a set class. A set class representation shares one corner with the box that represents its member class, as for example, the class **section** and the class **sections**. Note that the schema contains two different set classes for the class **section**. The class **crsections** represents a set of sections of the same course while the class **sections** represents a set of sections not necessarily of the same course, e.g., the current sections a student is registered for. A thick arrow represents specialization generic relationships such as *roleof* and *categoryof*, and a thin arrow represents a user–defined relationship between two classes.

In this paper, we use the common term "connection" for a user–defined relationship or a generic relationship. Each connection has two characteristics, a name and an access weight W, where $0 \leq W \leq 1$. For the two generic relationships *setof* and *memberof* and for user–defined relationships the access weight is assigned based on a simple rule. The rules discussed in this paper were also discussed in [MGPN92, MGPF92a, MGPF92b] and are summarized here to make this paper self–contained.

Rule 1: The sum of the weights on the outgoing connections of a class is 0.5 * (the number of outgoing connections). From this sum, each connection is assigned a weight from [0, 1], reflecting

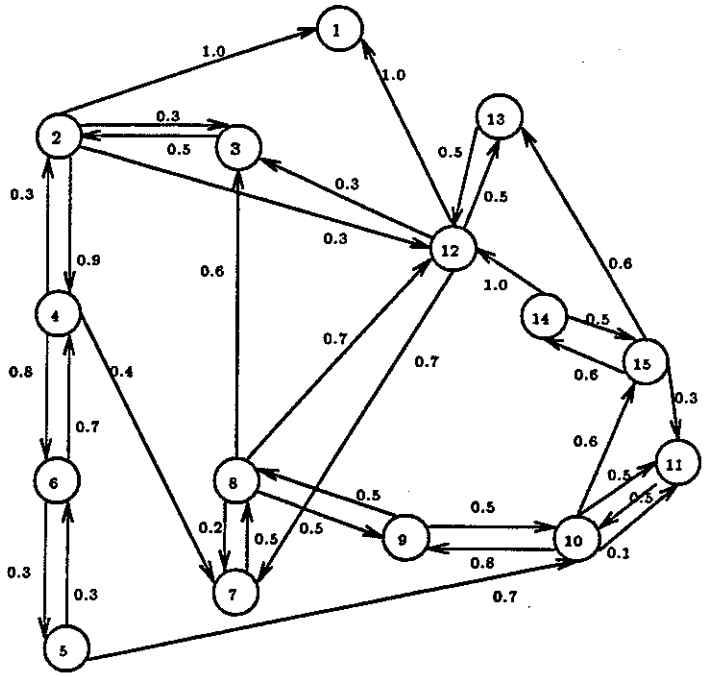

Figure 2: The subschema as a directed graph

its relative frequency of traversal.

In Figure 1, the class **transcript** has three relationships, *CourseRecords* to the class **course_records**, *CurrentSections* to the class **sections** and *Student* to the class **student**. The relationship *CourseRecords* has an access weight 0.8, the relationship *CurrentSections* has an access weight 0.4, and the relationship *Student* has an access weight 0.3 based on their access frequencies. Observe that the sum of access weights is 0.8 + 0.4 + 0.3 = 1.5 = 0.5 * 3 as required by Rule 1. The justification for Rule 1 is as follows. It is not sufficient to assign access weights which add up to 1 according to traversal probabilities of a connection. This would imply that the connections of a class with few connections are more significant than the connections of a class with many connections, which is not true. Thus, Rule 1 makes the value of access weights independent of the number of connections of a class. The access weights for specialization relations are assigned by other rules and are discussed later, following some examples. Some obvious access weights are omitted in Figure 1 but are shown in Figure 2, discussed soon.

The AR value of a path can be computed using different t-norms. As mentioned before we have chosen to use 2 t–norms. For convenience we shall refer from now on to the t-norms as weighting functions.

1. **PRODUCT weighting function:** multiplies the access weights of all the edges in the path.

2. **MINIMUM weighting function:** selects the minimum access weight over all the edges in the path.

The minimum weight edge of a path is called the bottleneck edge.

Let us consider the paths from the class **professor** (12) to the class **courses** (11).

1. The path p_1 represents the class sequence (professor (12), sections (7), section (8), crsections (9), course (10), courses (11)): This class sequence can be interpreted to find all the courses being taught by a professor. The AR value = 0.0437 (0.5) for the PRODUCT (MINIMUM) weighting function.

2. The path p_2 represents the class sequence (professor (12), students (3), student (2), transcript (4), sections (7), section (8), crsections (9), course (10), courses (11)): This class sequence can be interpreted to find all the courses currently being taken by all the students supervised by a professor. The AR value = 0.0034 (0.3) for the PRODUCT (MINIMUM) weighting function.

3. The path p_3 represents the class sequence (professor (12), students (3), student (2), transcript (4), course_records (6), course_record (5), course (10), courses (11)): This class sequence can be interpreted to find all the courses which have already been taken by all the students supervised by a professor. The AR value = 0.0113 (0.3) for the PRODUCT (MINIMUM) weighting function.

4. The path p_4 represents the class sequence (professor (12), sections (7), section (8), students (3), student (2), course_records (6), course_record (5), course (10), courses (11)): This class sequence can be interpreted to find all the courses already been taken by all the students which are currently registered in the sections being taught by a professor. The AR value = 0.0079 (0.3) for the PRODUCT (MINIMUM) weighting function.

Finally, the access relevance between classes a_s and a_t is computed by applying the co–norm MAXIMUM over all directed paths from a_s to a_t. The access relevance from **professor** to **courses** will be the maximum AR value over all the paths. A path with the maximum AR value is called a *most relevant path.* For both weighting functions, p_1 is the most relevant path. Considering the interpretations of the four paths this is not surprising, since the interpretation of p_1 is the most straightforward.

A path does not necessarily maximize both weighting functions. For example, consider the paths from the class **professor** (12) to the class **student** (2).

1. The path p_5 represents the class sequences (professor (12), students (3), student (2)): This path can be interpreted to find all the students currently supervised by a professor. The AR value = 0.15 (0.3) for the PRODUCT (MINIMUM) weighting function.

2. The path p_6 represents the class sequence (professor (12), sections (7), section (8), students (3), student (2)): This path can be interpreted to find all the students, in the sections

a professor is teaching. The AR value = 0.105 (0.5) for the PRODUCT (MINIMUM) weighting function.

Here, path p_5 is the most relevant path for the PRODUCT weighting function, and path p_6 is the most relevant path for the MINIMUM weighting function. Considering the interpretations of the two paths, it is not surprising that each of them is maximizing one weighting function since both interpretations reflect straightforward connections.

Property 1: For every pair of nodes there exists a simple (i.e., no repeated nodes) most relevant path.

The property is implied from the fact for both WF used, the deletion of a cycle from a path can just increase its access relevance.

The assignment of access weights to the specialization generic relationships *roleof* and *categoryof* needs to reflect inheritance, i.e., each property of a superclass is available at the subclass at no extra cost. We consider two possible rules to take inheritance into account.

Rule 2a: Assign an access weight of 1.0 to each *roleof* and *categoryof* connection (see Figure 2).

For both WFs such a value implies that the properties of the superclass are available at the subclass without decreasing the access relevance. However, Rule 2a has the following disadvantage. It enables the traversal of a specialization connection as a regular connection rather than just to support inheritance. That is, it enables traversal which stops at the superclass as a target rather than continuing to use one of its properties. But there is no reason for such traversal since it does not lead to find any meaningful information not available at the subclass. For example, consider the access relevance from the class **course** (10) to **professor** (12). One path is p_7 which represents the class sequence (course (10), crsections (9), section (8), professor (12)). This class sequence can be interpreted to find all the professors which teach the sections of a given course. Another path is p_8 which represents the class sequence: (course (10), department (15), chair_person (14), professor (12)). This path can be interpreted to find the instance of the **chair_person**, of the department of the given course, as a professor. That is, it finds the internal i.d. of the chair–person in the class professor. This path which is enabled by traversing the *roleof* connection from **chair_person** to **professor** as its last connection is inacceptable since its last connection provides information which is not relevant to the user. There is no meaning to traversing the *roleof* connection unless it is utilized to inherit a property of a professor for the chair_person such as the sections s/he teaches, in which case the traversal does not stop at the superclass. By the PRODUCT t–norm p_7 has an AR value = 0.28 and p_8 has an AR value = 0.36. However, p_8 is possible only due to the traversal of the *roleof* connection. Thus, we would like to block traversals through specialization connections while still having the inheritance properties. But an access weight of 1.0 enables such a traversal and furthermore gives it high priority.

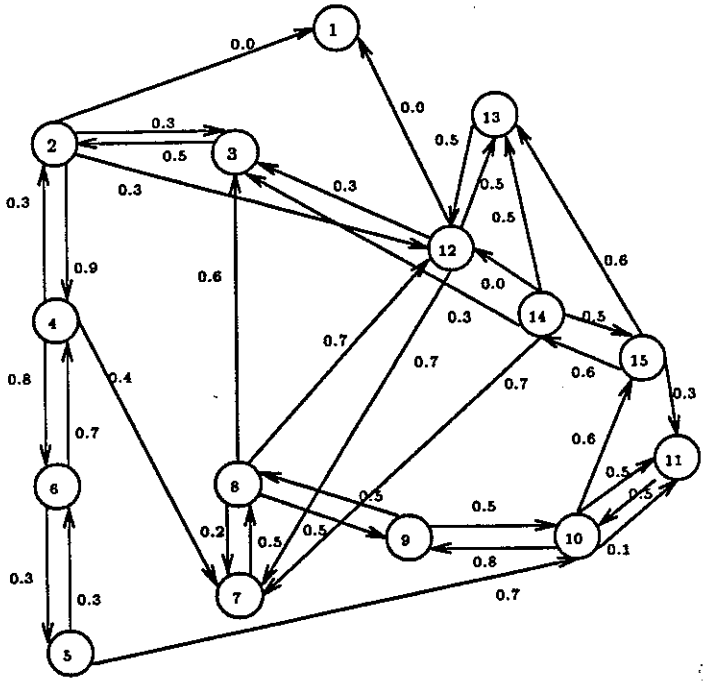

Figure 3: The directed graph using Rule 2b

Rule 2b: Assign an access weight of 0.0 to *categoryof* and *roleof* connections. Copy all the properties of the superclass to the subclass in the schema's underlying graph to allow inheritance.

Rule 2b allows computation of access relevance exactly as discussed before. But it practically disallows unwanted traversal of specialization connections. One disadvantage is that the schema graph becomes more dense. Figure 3 shows the directed graph for the computation of access relevance using Rule 2b for a schema of two OODBs. Typically, this increases the running time of the algorithm by a low order function of the number of classes. The schema visible to the user as well as the algorithms for computing the access relevance are not changed. We computed access relevance for a subschema of a university OODB. Our experiments with the PMG algorithm [MGPN92] show slightly better results for Rule 2b than Rule 2a for generating path–methods based on precomputed access relevance values. Thus, the choice between the Rule 2a and Rule 2b is a case of tradeoff between accuracy and complexity where the difference is small in both dimensions.

The above problem of computing access relevance can be formally defined as follows: Let a_s and a_t be classes of an OODB. There may be many directed paths from a_s to a_t. For a weighting function WF (i.e., PRODUCT or MINIMUM), for each path $P(a_s,a_t) = a_s(= a_{i_1}), a_{i_2}, a_{i_3}, \ldots, a_{i_k}(= a_t)$ we have an access relevance value (ARV) defined as

$$ARV(P) = WF_{(1 \le r < k)} W(a_{i_r}, a_{i_{r+1}})$$

The access relevance from a_s to a_t is defined as the maximization of the access relevance ARV(P)

over all paths P (a_s, a_t). That is

$$AR(a_s, a_t) = \max_{P(a_s,a_t)} ARV(P) = \max_{P(a_s,a_t)} (WF_{((a_{i_r},a_{i_{r+1}})\in P(a_s,a_t))} W(a_{i_r}, a_{i_{r+1}}))$$

Maximizing the MINIMUM weighting function finds a path with the heaviest bottleneck edge, while maximizing the PRODUCT weighting function finds a path with the highest product of access weights of all the edges. We note that sometimes the user may be interested in the access relevance between a class a_s and an attribute atr_{a_t} of another class a_t. Our definition can be extended to handle this case by representing the connection between a class and its attributes by an edge with a given access weight.

For a weighting function WF we define

$$AR(a_s, atr_{a_t}) = WF\ AR(a_s, a_t), W(a_t, atr_{a_t})$$

Note that if W(a_t, atr_{a_t}) = 1 then for both weighting functions AR(a_s, atr_{a_t}) = AR(a_s, a_t).

3 An Algorithm for the PRODUCT weighting function

We propose an algorithm *PRODUCT_AR* for the PRODUCT weighting function which computes access relevance from a source class to all the classes in the schema. This algorithm is a variation of the well-known nearest neighbor greedy algorithm of Dijkstra (e.g., [AHU83]). The algorithm of Dijkstra solves the single source shortest path problem. In order to find the all pairs access relevance in a schema of n classes we need to apply the algorithm, *PRODUCT_AR*, n times, once for each class as a source class.

The *PRODUCT_AR* algorithm finds the access relevance AR[v] from a source class represented by node s to every other class v. The algorithm is described in terms of the graph representation of the schema. It assumes that V = {1, 2, . . .,n}. The algorithm works by maintaining a set S of nodes whose maximum access relevance from the source is already computed. Initially, S contains only the source node {s}. At each step, we add to S a node $u \in$ V–S of maximum access relevance. A path from s to a node v is called *special* if all its nodes (except possibly v itself) belong to S. At each step of the algorithm, we use an array AR to record the maximum access relevance value of a special path to each node. In each step, after u is chosen to be inserted into S, a special path may contain u. Hence, we update AR[v] for each node $v \in$ V–S as follows. AR[v] is the maximum of two values: (1) The old AR[v] containing the access relevance of a special path not containing u; and (2) AR[u] * W[u, v] representing the access relevance of a special path containing u as the last node before v. Once S includes all nodes, all paths are "special," so AR[v] will hold the maximum access relevance from the source to each node $v \in$ V. W is a two-dimensional array, where W[i, j] is the access weight of the edge (i, j). If there is no edge (i, j), then we assume W[i, j] = 0.

```
Procedure PRODUCT_AR (IN s: node, OUT AR: array[1..n] of REAL)
begin
(1)     S := {s};
(2)     for each node v other than s do
(3)         AR[v] := W[s, v];
(4)     for i := 1 to n–1 do begin
(5)         choose a node u in V – S such that
                AR[u] is a maximum;
(6)         add u to S;
(7)         for each node v in V – S do
(8)             AR[v] := max (AR[v], AR[u] * W[u, v])
        end
end;
```

The algorithm will work for an undirected graph as well.

Let us apply *PRODUCT_AR* to the directed graph of Figure 2. The source is 12 (**professor**). In steps (2)–(3), S = {12}, AR[3] = 0.3, AR[1] = 0.0, AR[7] = 0.7, AR[13] = 0.5, and for the rest of the entries of the array, AR = 0. Note that the value of AR[1] is 0.0 using Rule 2b, though it is shown 1.0 (using Rule 2a). In the first iteration of the for-loop of lines (4)–(8), u = 7 is selected as the node with the maximum AR value. Then we set AR[8] = max(0, 0.7 * 0.5) = 0.35. Other values of the array AR do not change. The sequence of the AR values after each iteration of the for–loop is shown in Table 1.

Iteration	S	u	new value of AR
Initial	{12}	–	AR[3] = 0.3, AR[1] = 0.0, AR[7] = 0.7, AR[13] = 0.5
1	{12, 7}	7	AR[8] = 0.35
2	{12, 7, 13}	13	–
3	{12, 7, 13, 8}	8	AR[9] = 0.175
4	{12, 7, 13, 8, 3}	3	AR[2] = 0.150
5	{12, 7, 13, 8, 3, 9}	9	AR[10] = 0.088
6	{12, 7, 13, 8, 3, 9, 2}	2	AR[4] = 0.135
7	{12, 7, 13, 8, 3, 9, 2, 4}	4	AR[6] = 0.108
8	{12, 7, 13, 8, 3, 9, 2, 4, 6}	6	AR[5] = 0.032
9	{12, 7, 13, 8, 3, 9, 2, 4, 6, 10}	10	AR[11] = 0.044, AR[15] = 0.053
10	{12, 7, 13, 8, 3, 9, 2, 4, 6, 10, 15}	15	AR[14] = 0.032
11	{12, 7, 13, 8, 3, 9, 2, 4, 6, 10, 15, 11}	11	–
12	{12, 7, 13, 8, 3, 9, 2, 4, 6, 10, 15, 11, 5}	5	–
13	{12, 7, 13, 8, 3, 9, 2, 4, 6, 10, 15, 11, 5, 14}	14	–
14	{12, 7, 13, 8, 3, 9, 2, 4, 6, 10, 15, 11, 5, 14, 1}	1	–

Table 1: Computation of PRODUCT_AR on graph of Figure 2

The results of this application of *PRODUCT_AR* appear in line 5 of Table 2, showing the access relevance for each pair of nodes. It is necessary to prove the validity of the *PRODUCT_AR* algorithm.

Lemma 1: At all times AR[v] contains the highest access relevance of a special path from node s to node v. Due to space limitation the proof is omitted. It appears in [MGPF92a].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.3	0.9	0.216	0.72	0.36	0.18	0.121	0.151	0.076	0.3	0.15	0.054	0.091
3	0.0	0.5	0.0	0.45	0.108	0.36	0.18	0.09	0.061	0.076	0.038	0.15	0.075	0.027	0.045
4	0.0	0.3	0.12	0.0	0.24	0.8	0.4	0.2	0.151	0.134	0.084	0.14	0.07	0.061	0.01
5	0.0	0.084	0.168	0.21	0.0	0.3	0.176	0.28	0.63	0.7	0.35	0.196	0.252	0.252	0.42
6	0.0	0.21	0.084	0.7	0.3	0.0	0.28	0.14	0.168	0.21	0.105	0.098	0.076	0.076	0.126
7	0.0	0.15	0.3	0.135	0.032	0.108	0.0	0.5	0.25	0.125	0.063	0.35	0.175	0.045	0.075
8	0.0	0.3	0.6	0.27	0.065	0.216	0.49	0.0	0.5	0.25	0.125	0.7	0.35	0.09	0.15
9	0.0	0.15	0.3	0.135	0.032	0.108	0.245	0.5	0.0	0.5	0.25	0.35	0.18	0.18	0.3
10	0.0	0.12	0.24	0.108	0.026	0.086	0.252	0.4	0.8	0.0	0.5	0.28	0.36	0.36	0.6
11	0.0	0.06	0.12	0.054	0.013	0.043	0.126	0.2	0.4	0.5	0.0	0.14	0.18	0.18	0.3
12	0.0	0.15	0.3	0.135	0.032	0.108	0.7	0.35	0.175	0.088	0.044	0.0	0.5	0.032	0.053
13	0.0	0.075	0.15	0.068	0.016	0.054	0.35	0.175	0.088	0.044	0.022	0.5	0.0	0.016	0.026
14	0.0	0.15	0.3	0.135	0.032	0.108	0.7	0.35	0.175	0.088	0.15	0.25	0.5	0.0	0.5
15	0.0	0.09	0.18	0.081	0.019	0.065	0.42	0.21	0.12	0.15	0.3	0.3	0.6	0.6	0.0

Table 2: All-pair PRODUCT access relevance for graph of Figure 2

The following theorem completes the validity proof of *PRODUCT_AR*.

Theorem 1: At all times AR[v] contains the maximum access relevance of a path for all nodes v of S. (Ommited proof appears in [MGPF92a].)

When the operation of the algorithm is complete S = V. Hence, Theorem 1 implies that AR[v] is the highest access relevance of a path to v when the algorithm is completed.

The running time of *PRODUCT_AR* algorithm is O(n^2). If $e = |E|$ is much less than n^2, we might do better by using an adjacency list representation of the directed graph and using a priority queue implemented as a heap [AHU83] to organize the nodes in V–S. Choosing and deleting a maximum access relevance node from S in lines (5) and (6) takes O(lg n) time. This operation is repeated n times yielding O(n lg n) time. The loop of lines (7) and (8) can then be implemented by going down the adjacency list for u and updating the access relevance values in the priority queue. At most a total of e updates will be made, each at a cost of O(lg n), so the total time spent in lines (7) and (8) is now O(e lg n), rather than O(n^2). Thus the total time spent on this implementation of *PRODUCT_AR* algorithm is bounded by O(e lg n). This

running time is considerably better than $O(n^2)$ if $e << n^2$, as it is for a typical OODB schema whose graph representation is a sparse graph.

4 An Algorithm for the MINIMUM weighting function

We propose an algorithm *MINIMUM_AR* for the MINIMUM weighting function which computes access relevance from a source s to all other classes in the schema. This algorithm is similar to the previous algorithm *PRODUCT_AR*.

This algorithm begins with a set S initialized to source $\{s\}$. At each step the algorithm chooses a node $u \in$ V–S maximizing AR[u]. The main difference from the previous algorithm is in the mechanism for updating AR[v] for all $v \in$ V–S. For each neighbor v of u, after u is added to S, we compare the value of the access relevance of u with the access weight of the edge (u, v). The minimum of these two values is compared to the current access relevance of v. If this minimum value is higher than the current AR[v], then AR[v] is set to this value. As for the *PRODUCT_AR* algorithm if there is no edge $(i, j) \in$ E then we define W[i, j] = 0.

```
     Procedure MINIMUM_AR (IN s: node, OUT AR: array[1..n] of REAL)
     begin
(1)      S := {s};
(2)      for each node v other than s do
(3)           AR[v] := W[s, v];
(4)      for i := 1 to n–1 do begin
(5)           choose a node u in V – S such that
                   AR[u] is a maximum;
(6)           add u to S;
(7)           for each node v in V – S do
(8)                AR[v] := max (AR[v], min(AR[u], W[u, v]))
         end
     end;
```

Let us apply *MINIMUM_AR* algorithm to the graph shown in the Figure 2. The results of this algorithm, for source node 2, are shown in Table 3. Note that in step 4 the access relevance value of 3, AR[3] is updated. By applying this algorithm to each node in the schema, we compute all-pair access relevance (similar to Table 2 for *PRODUCT_AR*). The complexity and proof of this algorithm are similar to that of PRODUCT_AR and are not discussed here.

5 A more Efficient Algorithm for Bidirected Schemas

The graph representation of an OOdini OODB schema is a general directed graph. Examples of other OODB systems with directed relationships are VML [KNBDF91], GemStone [BOS91], and ORION [K90]. By a directed schema we mean that a connection does not guarantee an

inverse connection. In addition, if a relationship has an inverse relationship it may have a different access weight. (See, e.g., Figure 1). The reason is that the weight of a relationship is determined by its relative traversal frequency for the source class. Thus, two directed opposite relationships may have different access weights due to the different relative frequencies of the connections for each of the classes.

Iteration	S	u	new value of AR
Initial	{2}	–	AR[3] = 0.3, AR[1]=0.0, AR[4]=0.9, AR[12] = 0.3
1	{2, 4}	4	AR[6] = 0.8, AR[7] = 0.4
2	{2, 4, 6}	6	AR[5] = 0.3
3	{2, 4, 6, 7}	7	AR[8] = 0.4
4	{2, 4, 6, 7, 8}	8	AR[9] = 0.4, AR[3] = 0.4, AR[12] = 0.4
5	{2, 4, 6, 7, 8, 9}	9	AR[10] = 0.4
6	{2, 4, 6, 7, 8, 9, 10}	10	AR[11] = 0.4, AR[15] = 0.4
7	{2, 4, 6, 7, 8, 9, 10, 11}	11	–
8	{2, 4, 6, 7, 8, 9, 10, 11, 15}	15	AR[13] = 0.4, AR[14] = 0.4
9	{2, 4, 6, 7, 8, 9, 10, 11, 15, 12,}	12	–
10	{2, 4, 6, 7, 8, 9, 10, 11, 15, 12, 13}	13	–
11	{2, 4, 6, 7, 8, 9, 10, 11, 15, 12, 13, 14}	14	–
12	{2, 4, 6, 7, 8, 9, 10, 11, 15, 12, 13, 14, 3}	3	–
13	{2, 4, 6, 7, 8, 9, 10, 11, 15, 12, 13, 14, 3, 5}	5	–
14	{2, 4, 6, 7, 8, 9, 10, 11, 15, 12, 13, 14, 3, 5, 1}	1	–

Table 3: Computation of *MINIMUM_AR* on graph of Figure 2

However, there are several object-oriented database systems such as ObjectStore [LLOW91] and ONTOS [M91] which model each connection as bidirectional. Examples of knowledge representation systems which use bidirectional connections are Knowledge Craft [KC86] and Knowledge Explorer [K91]. The last one assumes equal access weights in both directions.

As mentioned earlier, for the PRODUCT weighting function, the algorithm *PRODUCT_AR* is applicable also for a bidirectional schema. Similarly, the *MINIMUM_AR* is applicable for bidirectional schemas. By applying this algorithm to all nodes as source nodes, we can compute the access relevance for all pairs of nodes in $\min(O(n^3), O(ne \log n))$ time. However, we shall present a more efficient algorithm for the MINIMUM weighting function for bidirectional schemas requiring only $O(n^2)$ time. This algorithm is based on the following theorem.

A *spanning tree* of a graph is a subgraph which is a tree that connects all the nodes of the graph. A maximum–weight spanning tree (MWST) is a spanning tree maximizing the sum of the weights of the edges in the tree.

Theorem 2: Let T be an MWST of an undirected graph G = (V, E). The unique path P in T between a node s and a node t is a most relevant path between s and t in G. (Ommited proof appears in [MGPF92a].)

Our algorithm is based on first finding an MWST of the bidirectional schema. There are famous algorithms of Prim and of Kruskal [AHU83] for this purpose. Prim's algorithm requires

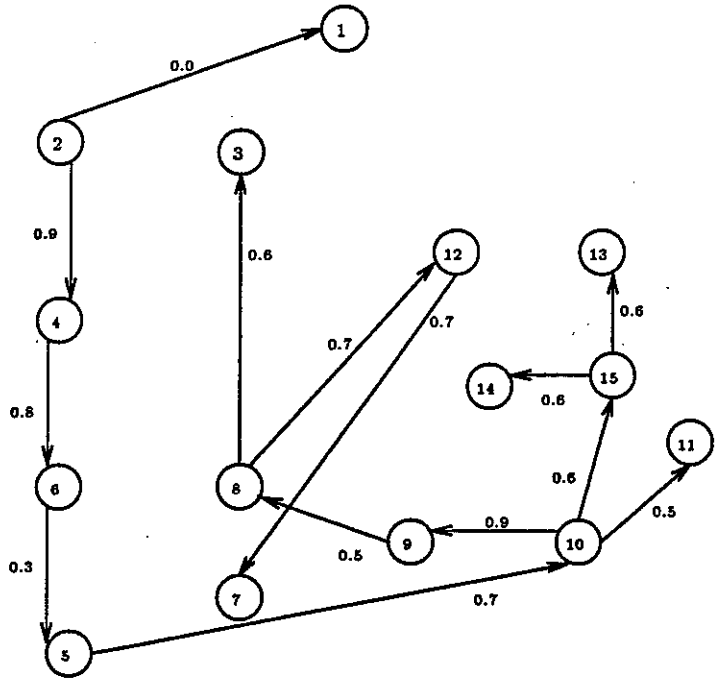

Figure 4: The rooted MWST (rooted at 2)

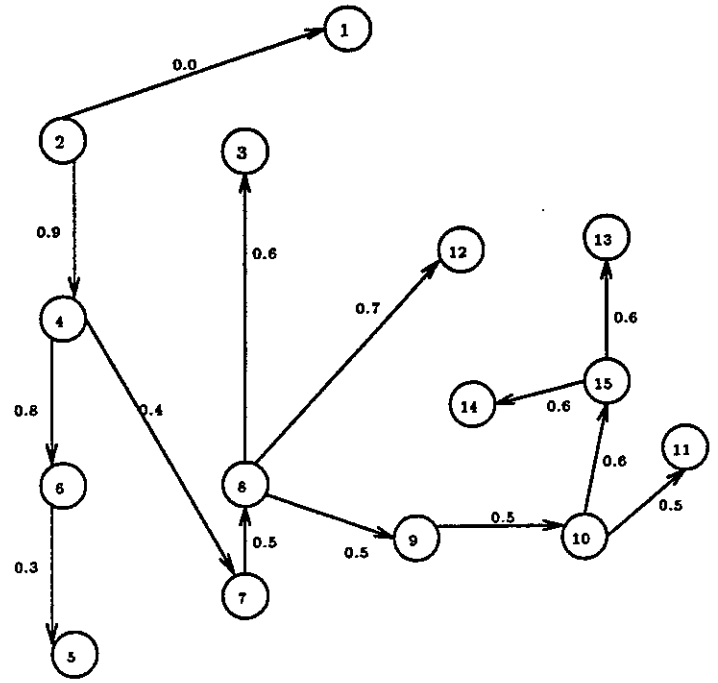

Figure 5: A rooted AR spanning tree (rooted at 2)

$O(n^2)$ time. As a matter of fact the Prim algorithm can be obtained from our *MINIMUM_AR* algorithm by replacing min(AR[u], W[u, v]) by W[u, v] in line (8).

Theorem 2 shows a strong connection between MWST and maximum access relevance paths for a bidirectional schema. However, we shall show that for a directional schema this is not the situation. We note that an MWST, rooted at s does not necessarily yield the maximum access relevance values. See Figure 4 and Figure 5 showing a rooted MWST and a *rooted AR spanning tree* maximizing the AR values from s to all nodes, respectively. The sum of access weights for the MWST rooted at 2 in Figure 4 is 8.4, while for the AR spanning tree rooted at 2 in Figure 5 it is 7.5. The AR value for the nodes 7, 8, 9, 10, 11, 12, 13, 14, 15, and 3 are all equal to 0.4 for the AR spanning tree; while all the AR values for these nodes is 0.3 for the path in the MWST. Thus, the MINIMUM_AR algorithm of the previous section is different from an algorithm for a rooted MWST of a directed graph. By Theorem 2, the weights of the n–1 edges of the MWST enable us to compute the access relevance for each pair of nodes as the minimum weight along the unique path connecting the pair.

The following algorithm computes the access relevance for all pairs of nodes for a bidirected graph. It stores the access relevance in a matrix ARM[i, j] for each pair (i, j), $i < j$, requiring only $O(n^2)$ time for calculating these $n(n-1)/2$ values. Since Prim's algorithm requires $O(n^2)$ too, this is the complexity of finding all these access relevance values for a bidirectional schema.

```
     Procedure COMPUTE_ARM(IN T: tree; OUT ARM: matrix)
     var
          U: set of nodes;
          u, v, x: node;
     begin
(1)       for i := 1 to n do
(2)            for j := i + 1 to n do
(3)                 ARM[i, j] := 1;
(4)       U := {1};
(5)       while U ≠ V do begin
(6)            let (u, v) be an edge in T such that
                    u is in U and v is in V–U;
(7)            for each node x ∈ U do
(8)                 ARM[x, v] := min(ARM[x, u], W[u, v])
(9)            U := U ∪ {v};
          end
     end;
```

The algorithm *COMPUTE_ARM* first initializes a matrix ARM to 1. This is different compared to the previous algorithms *PRODUCT_AR* and *MINIMUM_AR* because in line (8) we select a minimum value while before we needed a maximum. It then begins with a set U of nodes initialized to {1}. In each iteration it calculates ARM between all nodes $x \in$ U and a new

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2		1	0.3	0.9	0.3	0.8	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
3			1	0.3	0.5	0.3	0.6	0.6	0.5	0.5	0.5	0.6	0.5	0.5	0.5
4				1	0.3	0.8	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
5					1	0.3	0.5	0.5	0.7	0.7	0.5	0.5	0.6	0.6	0.6
6						1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
7							1	0.7	0.5	0.5	0.5	0.7	0.5	0.5	0.5
8								1	0.5	0.5	0.5	0.7	0.5	0.5	0.5
9									1	0.9	0.5	0.5	0.6	0.6	0.6
10										1	0.5	0.5	0.6	0.6	0.6
11											1	0.5	0.5	0.5	0.5
12												1	0.5	0.5	0.5
13													1	0.6	0.6
14														1	0.6
15															1

Figure 6: The triangular ARM Matrix

node v, adjacent to a node $u \in$ U, using ARM[x, u]. For this, the algorithm chooses an edge (u, v) such that $u \in$ U and $v \in$ V–U. Then it computes the access relevance from each node $x \in$ U to v, by choosing the minimum of W[u, v] and ARM[x, v]. This is because the bottleneck edge on the path between x and v in T is either the new edge or the bottleneck edge of the path between x and u in T. Then the algorithm adds the node v to U finishing the iteration. It terminates when U = V. Without loss of generality, we assume that the nodes are renumbered in the order of their traversal. Thus, we compute ARM[i, j] only for $i < j$.

The validity proof of *COMPUTE_ARM* is straightforward.

To demonstrate the operation of the algorithm *COMPUTE_ARM* we shall use the MWST of Figure 4 assuming that it is bidirected rather than rooted. We shall demonstrate the iteration when nodes 15, 14, 13, 12, 11, 10, 9, 8, and 7 are in U and the next edge to be added is (8, 3). The triangular matrix of Figure 6 shows all AR values calculated by the algorithm *COMPUTE_ARM*. It is clear from this triangular matrix that we need to store only $(n^2/2)$ access relevance values for bidirectional schemas. All the encircled values of Figure 6 show the AR values calculated in this iteration from all nodes in U to node 3. Note that the access relevance of paths between the pairs (9, 3), (10, 3), (11, 3), (13, 3), (14, 3) and (15, 3) is 0.5 due to corresponding values ARM[9, 3], ARM[10, 3], ARM[11, 3], ARM[13, 3], ARM[14, 3], and ARM[15, 3], which are 0.5. The access relevance of paths between a pair (7, 3) (8, 3), (12, 3) is 0.6 due to the access weight of edge (8, 3).

6 Conclusion

In this paper we have discussed the computation of access relevance as a measure of the significance of the (direct or indirect) connections between classes in object-oriented database systems. Those values will serve as a guide for the traversal of an OODB schema by the PMG algorithm [MGPN92] to generate path–methods. We have presented efficient algorithms for two common weighting functions PRODUCT and MINIMUM. For the MINIMUM weighting function, an algorithm for bidirected schemas was presented that is more efficient than the algorithm for the directed schema. Proofs and complexity analyses have been presented.

In an upcoming paper [MGPF92b], we consider the problem of computing access relevance for all the pairs of classes in interoperable object-oriented multidatabases. In this case, the access relevance is already computed for each OODB separately using the above algorithms. We shall present fast online algorithms for finding access relevance for pairs of classes across different OODBs.

References

[AHU83] Aho, A., Hopcroft, J., Ullman, J.D., "Data Structures and Algorithms", Addison-Wesley Publishing Company, Readings, 1983.

[BD86] Bonissone, P.P., Decker, K.S., "Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity", *Machine Intelligence Pattern Recognition 4*, 1986, pp. 217-247.

[BOS91] Butterworth, P., Otis, A., and Stein, J., "The GemStone Object Database Management System", *Communications of the ACM*, Oct. 1991, pp. 64-77.

[FKN91] Fankhauser, P., Kracker, M., Neuhold, E.J., "Semantic vs. Structural Resemblance of Classes", *SIGMOD record, Special issue on 'Semantic Issues in Multidatabase Systems'*, Dec. 1991, pp. 59-63.

[GPN91] Geller, J., Perl, Y., Neuhold, E. J., "Structure and Semantics in OODB class specifications", *SIGMOD record, Special issue on Semantic Issues in Multidatabase Systems*, Dec., 1991, pp. 40-43.

[HGPN92] Halper, M., Geller J., Perl, Y., Neuhold, E. J., "An OODB Graphical Schema Representation", *In the proc. of IDS92, International Workshop on Interfaces to Database Systems, Glasgow, July 1992.*

[K90] Kim, W., "Introduction to object-oriented databases", *The MIT Press*, 1990.

[K91] Kracker, M., "A fuzzy concept network model and its applications", *Technical Report-585, GMD-IPSI*, Germany, Oct. 1991.

[KC86] *Knowledge Craft Manual*, Carnegie Group Inc., 1986.

[KF88] Klir,G.L., Folger,T.A.,"Fuzzy Sets,Uncertainty and Information",*Prentice Hall*, 1988.

[KKS92] Kifer, M., Kim, W., Sagiv, Y., "Querying Object-Oriented Databases", *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, San Diego, California, June 2-5, 1992, pp. 393-402.

[KN89] Kracker, M., Neuhold, E. J., "Schema Independent Query Formulation", *Proceedings of the 8th Int. Conf. on Entity-Relationships Approach*, F. H. Lochovsky (Ed.), Toronto, Canada, 1989, pp. 233-247.

[KNBDF91] Klas W., Neuhold E. J., Bahlke, R., Drosten K., Fankhauser P., Kaul M., Muth P., Oheimer M., Rakow T., Turau V., "VML Design Specification Document", *Technical Report, GMD-IPSI,* Germany, 1991.

[L85] Litwin, W., "Implicit Joins in the Multidatabase System MRDSM", *IEEE-COMPSAC*, 1985, pp. 495-504.

[LLOW91] Lamb, C., Landis, G., Orenstein, J., Weinreb, D., "The Objectstore Database System", *Communications of the ACM*, Oct. 1991, pp. 50-63.

[LVZ92] Lanzelotte, R.S.G., Valduriez, P., Zait, M., "Optimization of Object-Oriented Recursive Queries using Cost-Controlled Strategies," *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, San Diego, California, June 2-5, 1992, pp. 256-265.

[M86] Motro, A., "Constructing Queries from Tokens", ACM SIGMOD, 1986, pp. 120-131.

[M91] Martin, R. "ONTOS Overview", *Proceedings of Executive Briefing on Object-Oriented Database Management*, San Fransisco, Sep. 1991.

[MGPF92a] Mehta, A., Geller, J., Perl, Y., Fankhauser, P., "Algorithms for Access Relevance to Support Path–Method Generation in OODBs", *NJIT T.R. CIS 92–10.*

[MGPF92b] Mehta, A., Geller, J., Perl, Y., Fankhauser, P., "Computing Access Relevance to Support Path–Method Generation for Interoperable Multi-OODBs", *NJIT T.R. CIS 92–09, In the proceedings of RIDE–IMS '93, Research Issues in Data Engineering–Interoperability in Multidatabase Systems, Vienna, Austria, April 18–20, 1993.*

[MGPN92] Mehta, A., Geller, J., Perl, Y., Neuhold, E.J., "The OODB Path–Method Generator (PMG) using Precomputed Access Relevance", *NJIT T.R. CIS 92–08, Submitted for Publication.*

[MU83] Maier,D., Ullman, J.D., "Maximal objects and the semantic of universal relation databases.", *ACM Transactions on Database Systems*, Vol. 8., No. 1, 1983, pp. 1-14

[NPGT89] Neuhold, E. J., Perl, Y., Geller, J., Turau, V., "Separating Structural and Semantic Elements in Object-Oriented Knowledge Bases", *Advanced Database System Symposium*, Kyoto, Japan, 1989, pp. 67-74.

[NPGT90] Neuhold, E. J., Perl, Y., Geller, J., Turau, V., "A Theoretical Underlying Dual Model for Knowledge Based Systems", *The first international conference on Systems Integration*, Morristown, NJ, 1990, pp. 96-103.

[NPGT91] Neuhold, E. J., Perl, Y., Geller, J., Turau, V., "The Dual Model for Object-Oriented Databases", NJIT, *T.R. CIS-91-30.*

[NS88] Neuhold, E.J., Schrefl, M, "Dynamic Derivation of Personalized Views", F. Bancilhon, D. DeWitt (eds.), *Proceedings of the 14th International Conference on Very Large Databases, VLDB '88*, Los Angeles, CA, Aug. 29-Sep.1, 1988, pp. 183-194.

[SS61] Schweizer, B., and Sklar, A., "Associative Functions and Statistical Triangle Inequalities", *Publications Mathematicae Debrecen*, Vol. 8, 1961, pp. 169-186.

[Z65] Zadeh, L.A., "Fuzzy Sets", *Information and Control*, Vol. 8, 1965, pp. 228-353.