OBJECT IDENTITY IN OMEGA: AN OBJECT-ORIENTED DATABASE SYSTEM FOR MANAGING MULTIMEDIA DATA

Yoshifumi Masunaga

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-88-19 May 1988

Object Identity in OMEGA : An Object-Oriented Database System for Managing Multimedia Data

Yoshifumi Masunaga†

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712-1188

ABSTRACT

OMEGA is a database system for managing multimedia data under development at the University of Library and Information Science. It takes an object-oriented approach, and its data model is called OMEGA. In order to make the object concept and basic object management scheme in OMEGA clear, we focused our investigation on the object identity and related topics. In this paper, it is revealed that three types of object identity exist in OMEGA; the trivial identity, the referential identity, and the arbitrary identity. Based on this result, various types of object identity which exist in the real world are characterized in OMEGA. The contents identity, the identity of complex objects, and the object similarity are such examples. Because object identity and similarity satisfy the condition of the equivalence relation on a set, the quotient class can be defined. By using this, other types of object identity such as the substantial identity and the relational identity are characterized. Moreover the object copy, the object naming, and the comparison between single class and multiclass object management schemes are investigated in this paper.

1. Introduction

In order to increase the productivity of offices and factories as much as possible, it is necessary to integrate various types of data such as text, figures, images, documents, sounds and voices into a single database. Such a database is called a multimedia database where users can interact with multimedia data without being aware of the heterogeneity of media. For example, if one wants to organize a database for remote sensing research support, then not only image data such as LANDSAT and NOAA images, but also figure data such as geographical maps as well as alphanumeric data such as land data, rain data, snow data, sunshine data, industry data, population data, and so on must be integrated into one database.

Due to the following reasons, we are now in a position to develop multimedia databases and their management system [Masu87]. Tremendous progress on

(i) storage media such as optical disk storage systems,

(ii) input/output media such as graphics, multi-windows, facsimile and digitizer, and

(iii) communication media such as optical communication systems and the standardization of communication protocols such as OSI.

---

† The author is on leave from the University of Library and Information Science, Tsukuba-Science-City, Ibaraki 305, Japan.

We can list other reasons such as

(iv) recent progress on semantic data model theory including the object-oriented data model, and

(v) increase of users' demand for multimedia databases.

Research and development of multimedia database management systems were initiated at the beginning of the 1980s. The term multimedia was first seen in the paper [TCE83], and extensive research is reported in [CVL84], [Masu85], [ChHT86], [WoKL86], [Bada87], [Masu87], [WoKi87], [BKW87] and others. Research and development of object-oriented database systems are also very active and very close to the multimedia database field. Extensive research is reported in [CoMa84], [NiTs85], [IEEE85], [Oren86], [LyKe86], [MaDa86], [BuCV86], [MBH86] and others.

In this paper, we first overview an object-oriented multimedia database management system named OMEGA (an abbreviation of Object-oriented Multimedia database Environment for General Application) which is now under development at the University of Library and Information Science. This is done particularly from the data model point of view in section 2. We call the data model of OMEGA the OMEGA model. In section 3, we discuss the meaning of objects in the OMEGA model. The OMEGA objects and the real world objects are clearly distinguished. Particularly, we focus our discussion on object identity and related topics. Three types of object identity exist in OMEGA; the the trivial identity, the referential identity, and the arbitrary identity, which characterize various types of object identity which exist in the real world. The contents identity, the identity of complex objects, and the object similarity are such examples. Because object identity and similarity satisfy the condition of an equivalence relation on a set, the quotient class can be defined. By using this, other types of object identity such as the substantial identity and the relational identity are characterized. Object copy is also discussed. In section 4, basic object management schemes are investigated. Particularly, the object naming and the comparison between single class and multiclass object management schemes are investigated in this section. Section 5 summarizes this paper.

2. Multimedia Data Model OMEGA

Figure 1 shows a concept of multimedia data modeling of the real world. The real world can be recognized and described by acceptors using certain symbol systems which are allowed in a specified media. These representations are called conceptual models. Four types of media are considered: alphanumeric media, figure media, image media, and sound media. The essential point in building a multimedia database is to make clear how to integrate these conceptual models into a single model which is called as a multimedia conceptual model. By multimedia data model we mean a symbol system which can describe any multimedia conceptual model. A multimedia conceptual model is then translated into a logical representation model called as a multimedia database schema by using another symbol system. An instance of the schema is a multimedia database.

OMEGA is a multimedia data model in the above sense. Although it is not explicitly mentioned, an outline of OMEGA model has already been described in [Masu87]. It is an object-oriented data model, and two types of objects can be explicitly supported in it; the Smalltalk-80 objects [GoRo83] and the complex objects [HaLo82]. These two types of objects are necessary to manage multimedia data, or even single media data in offices and factories because real world objects in those fields usually have not only the IS-A relations (i.e., the class hierarchy or the generalization hierarchy) but also the IS-PART-OF relations (i.e., the design hierarchy or the aggregation hierarchy). For example, since an (electric) circuit can be decomposed into functions and connections according to the design hierarchy, two IS-PART-OF relations can be defined between class Circuit and class Function, and class Circuit and class Connection, respectively.

In OMEGA these two types of objects can be treated in a unified manner. The notation of Smalltalk-80 objects will be used to describe objects and classes with extension and modification. The following three concepts are clearly distinguished in our approach.

1. Class template
2. Class
3. Object.

The class template is a schematic definition of a class. It is time variant in the sense of schema evolution [BKKK87]. The parent class is introduced to specify the parent class in the design hierarchy. The referred class is introduced to specify the referred object class for multimedia database integration. The rules are introduced to describe both non-procedural methods and database integrity constraint. We use attribute instead of instance variables. Others are similar to Smalltalk-80. A sample class template is shown in Figure 2.

An object is defined as a package of information, i.e., a list of attribute values, and description of object manipulation (methods) including rules. Thus, when a class template is given, we can list the set of all objects which meet the class template. By class we mean the set of all objects of the class template at a given time. The class varies time to time, i.e., time variant. The class template-class-object relationship is analogous to the schema-relation-tuple relationship in the relational database [Codd70]. Sometimes, the term class will be also used to represent the class template unless it is not confused.

As we mentioned earlier, rules are introduced in our system primarily to describe certain knowledge associated with IS-PART-OF relations. Let us consider an example of rules. Suppose that class EleCirDia (an abbreviation of Electrical Circuit Diagram) has class EleCirDiaFunction and class EleCirDiaConnection as children according to the design hierarchy (see Figure 2). Now suppose that we want to make an object in class EleCirDia secret if any of its components in the design hierarchy is secret. In this case it is possible to find all electric circuit diagrams which are secret if the following rule is specified in the EleCirDia class template, where IS-PART-OF(x, y) represents that x is a part of y. (PROLOG like notation is used here, but it is not essential).

$$\text{secret}(y) \leftarrow \text{IS-PART-OF}(x, y), \text{secret}(x)$$

A similar idea can be seen in [BKW87] as value propagation. A class inherits attributes, messages, methods, and rules from its super classes.

To integrate different conceptual models into one in OMEGA, a method called referred object integration method is proposed in [Masu87]. This is basically a bottom-up approach. In order to make integration as perfect as possible, we introduce a multimedia conceptual model designer to show his/her view of the real world in terms of objects. It in fact is a structural and semantic representation of the designer's knowledge about the real world objects. The proposed object base will be used as a clue to integrate different conceptual models which were constructed in different media into a single database. A simple example of data integration is shown in Figure 3(a), and the corresponding multimedia data model in OMEGA is shown in Figure 3(b).

3. Object Identity in OMEGA

3.1. Objects and Object Creation

Objects are the most fundamental OMEGA elements. It is very important to clearly distinguish objects in OMEGA and objects in the real world. The former is called the OMEGA objects, while the latter is called the real world objects. OMEGA objects can be created by OMEGA users to represent real world objects. For example, suppose that John Smith entered a university. Then in order to represent this fact, a user may create an OMEGA object by sending the following message to class Student.

new:(870100, John Smith, male, Tsukuba-Science-City)

By this message, an OMEGA object "o" could be created which has 870100 as the student identification number, John Smith as name, male as sex, and Tsukuba-Science-City as address. (By the

way Tsukuba-Science-City is a newly developed area in Japan which is 40 miles north of Tokyo where more than fifty Japanese national institutes are located.) The Smalltalk-80 like expression is used only to carry our idea, and is not essential.

In OMEGA, attributes are regarded as functions as in [LyKe86]. For example, suppose class Student has attributes sid (student identification number), name, sex, and address as defined above. Then the following holds.

sid(o) = 870100, name(o) = John Smith, sex(o) = male,
address(o) = Tsukuba-Science-City.

Of course, the domain of an attribute could be set-valued. For example, if attribute "dependent" is defined and suppose that student o has three dependents, then the attribute value could be the set of three objects which represent them. Any numerical value of a character string is again an object in OMEGA.

3.2. Object Identification Number

Whenever an object is created, OMEGA allocates a system-wide unique object identifier. This could be a serial number, or a time stamp, and therefore we call it object identification number.

This means that an attribute named oid is automatically assigned to a class template definition whenever it is defined. Attribute oid is for system use purpose, and not for users. Therefore users are not aware of the existence of attribute oid in the normal use.

3.3. Object Identity

In OMEGA, several OMEGA objects could be created in various classes in order to represent different aspects of the same object in the real world. Therefore one might want to regard those OMEGA objects as identical because they refer to the same real world object. Also, one might want to regard a bunch of objects as identical because of a certain reason. This means that it is necessary to investigate what OMEGA objects are identical in what sense. In this section three types of object identity will be introduced. Before going to details, a general definition of object identity will be given below. We presume that whenever we can say that object o is identical with object o′ in class C, the identity must be determined by using a certain rule which is either defined explicitly in class C or inherited from one of its super classes.

Definition 3.1: Let o and o′ be objects in class C. A rule is said to be an object identity rule if it is defined in the following form:

$$(\forall\, o, o' \in C)\ (P(o, o') \rightarrow o = o'),$$

where P represents a two variable predicate defined on C, and o = o′ represents that object o is identical with object o′. That is, object o and o′ can be said to be identical if they satisfy predicate P. Three types of object identity will be given below.

[1] **Trivial identity**.

Let o and o′ be OMEGA objects. The o is said to be trivially identical with o′ if oid(o) = oid(o′). This could be defined as follows:

$$(\forall\, o, o' \in \text{Object})\ (\text{oid}(o) = \text{oid}(o') \rightarrow o = o'),$$

where Object is the system-defined class which locates on the top of all other classes of OMEGA in the sense of class hierarchy. Since every class in OMEGA can inherit this rule along with super class chain, the trivial identity of objects can be used in any class.

[2] **Referential identity**.

There is another type of object identity in OMEGA which is called referential identity. As we mentioned before, every OMEGA object refers to a certain object in the real world. Therefore, it is assumed that there may exist two objects o and o′ in class C which are not identical in the trivial sense, i.e. oid(o) ≠ oid(o′), but is identical in the referential sense, i.e. both of them refer to the same real world object. In this case, we say that they are referentially identical. A formal definition will be given below.

Definition 3.2: Suppose that class C has an object identity rule p(o, o′) → o = o′[C]. Then, the rule is said to be a referential identity rule of object, if whenever P(o, o′) hold, then o and o′ must refer to the same object in the real world.

We will extend our investigation to cover a more general and useful case in which we can determine the identity of object o and o′ which belong to different classes, say class A and B, respectively. In order to explain our idea, let us consider an example first. Suppose that three classes named Student, ForeignStudent, and MaleStudent are defined in OMEGA, where class Student is a super class of other two classes. (ForeignStudent consists of students who came from abroad, while MaleStudent consists of male students.) Now suppose that a foreign male student (this means that a male student who cam from abroad) is on the university register. Then in order to reflect this fact, objects o and o′ will be created in class ForeignStudent and class MaleStudent respectively. We also assume that class Student has sid, name, sex, and address as attributes, class ForeignStudent has its own attribute nationality in addition to the attributes inherited from Student, and class MaleStudent inherits attributes from Student. Since OMEGA allocates different oid values to o and o′, they are not identical in the trivial identity sense. However, since those two objects represent the same object in the real world, i.e. both refer to the same real world object, we want to say that they are identical in the referential identity sense. To do so, we pay attention to the fact that the values of the attribute sid of o must be identical with that of o′ if both represent the same student. This is true because the same student identification number may not be allocated to more than one student. Therefore we can introduce referential identity rule of objects in class Student, and we can determine the referential identity of object o in class ForeignStudent and o′ in class MaleStudent by using this rule. This is described as follows:

$$(\forall\, o \in \text{ForeignStudent}, \forall\, o' \in \text{MaleStudent})\ (\text{sid}(o) = \text{sid}(o') \rightarrow o = o'\ [\text{Student}])$$

Figure 4 shows the above situation.

The discussion will be formalized below.

Definition 3.3: Class A is said to be an ancestor of class B if there exists a super chain from B to A.

Definition 3.4: Class A and B are said to be of family ties under class C if they have C as a minimal common ancestor, where the term minimal means that any subclass of C can not be a common ancestor of A and B.

Since multiple inheritance may happen in OMEGA, in general more than one minimal common ancestor could be found. It is also assumed that the minimal common ancestor may not be class Object. Such a situation can be illustrated in Figure 5.

Now suppose that o and o′ are objects created in class A and B, respectively, and that A and B are family ties under class C. Since both classes are the subclass of class C, it can be possible to determine whether o and o′ are identical or not by using an object identity rule defined in C.

Definition 3.5: Let o and o′ be objects in class A and B, respectively. Then o is referentially identical with o′ under class C, denoted by o = o′[C], if there exists a referential identity rule of object in a minimal common ancestor C of class A and B by which o and o′ are determined to be referentially identical.

[3] **Arbitrary identity.**

There exists yet another type of object identity in OMEGA which is neither the trivial identity nor the referential identity. This is called the arbitrary identity. The term arbitrary means that one can define object identity in a class arbitrarily. For example, one might want to regard that two OMEGA objects, each of which represents water and ice, respectively, as identical because the substance of water and ice

are identical. Or one might want to regard two companies as identical because both locate in Tokyo. (Therefore the concept of Tokyo company will be established. See section 3.9 for more details.) In this sense, it could be said that the arbitrary identity is based on the result of science, say natural, political, and social sciences, and culture.

Definition 3.6: An object identity rule is called arbitrary identity rule if it is neither the trivial identity rule nor the referential identity rule.

A typical example might be the following case. Suppose that class ChemicalCompund is defined with which an attribute named chemical-formula is associated. Chemists might say that we do not want to distinguish those objects which have the same chemical formula. In order to reflect this wisdom in OMEGA, we can define the following arbitrary identity rule in class ChemicalCompound.

$$(\forall o, o' \in \text{ChemicalCompound})\ (\text{chemical-formula}(o) = \text{chemical-formula}(o') \rightarrow o = o')$$

The above discussion can be expanded to cover a more general and useful case so that the arbitrary identity of objects o and o′ which belong to different classes, say class A and B, respectively, could be determined. For example, suppose that there are three classes in OMEGA which are named Chemical-Compound, LiquidChemicalCompound, and SolidChemicalCompound, where class ChemicalCompound is a super class of the other two. Suppose also that on the other hand there are three real world objects which are alcohol, water, and ice. Then in oder to represent this fact, objects o1 (the representative of alcohol) and o2 (the representative of water) will be created in LiquidChemicalCompound and o3 (the representative of ice) will be created in SolidChemicalCompound class. We also assume that Chemical-Compound class has chemical-formula attribute, and therefore its subclasses have it. Since OMEGA allocates different oid values to o1, o2, and o3, they are not identical with each other in the trivial sense. Moreover they are not identical in the referential sense, because they refer to different objects in the real world. However, as we mentioned before, there might be a situation in which water and ice could be regarded as identical as chemists usually do. Now the identity of object o in LiquidChemicalCompound and object o′ in SolidChemicalCompound can be defined as follows:

$$(\forall o \in \text{LiquidChemicalCompound}, \forall o' \in \text{SolidChemicalCompound})$$
$$(\text{chemical-formula}(o) = \text{chemical-formula}(o') \rightarrow o = o' \text{ [ChemicalCompound]})$$

Figure 6 shows the above situation. A cycle from o2 to o2 indicates that water changes its state to ice when it is cooled and ice becomes water when it is warmed. That is, they are identical in terms of substance, but they differ in terms of state. One might object that the above equation chemical-formula(o) = chemical-formula(o′) is incorrect because the term which indicate the melting heat is not included. If this fact is necessary to be included, then why not add a term of melting heat on the right hand side. Of course we can introduce class GasChemicalCompound similarly.

A formal definition will be given below.

Definition 3.7: Let o and o′ be objects in class A and B , respectively. Then o is arbitrarily identical with o′ under class C, denoted by o = o′ [C], if there exists a minimal common ancestor C of A and B and o is determined to be arbitrarily identical with o′ by using an arbitrary identity rule of object defined in it.

3.4. Absence of Keys

One difference between the object base model OMEGA and the traditional relational data model [Codd70] is the absence of keys in OMEGA. In the relational database, because a relation is a set, the set of attributes (the set of all attributes in the worst case) constitutes a key, i.e. a unique tuple identifier. However, it does not hold in OMEGA. There are many cases in which an object can not be uniquely identified even though all its attribute values are specified.

For example, suppose that class Bolt is defined in an object base of a part manufacturing company, and that the following three attributes are defined in it (including inherited attributes): part-name, size, and color. Now, suppose that a type A bolt (abbreviated A-bolt) with size 50 and color red is produced.

Then in order to represent this production, an OMEGA object will be created which has (A-bolt, 50, red) as its attribute value list. Moreover, suppose that another bolt with the same standard is produced. Then in order to represent this production, another OMEGA object will be created which has the same attribute list, i.e. (A-bolt, 50, red). Of course this object has different oid value from that of the object created before. However, since the attribute lists of those two objects are the identical, we can not distinguish them even though their attribute lists are given.

In this case, it is supposed that it may not be necessary for users to distinguish this bolt from that bolt of type-A. Class Bolt can be seen to users as a multi set to which they may issue queries and updates. For example, they may want to create a thousand bolts of the same type, or they may want to retrieve any one bolt of a specified type. That is, in class Bolt, key is not necessary to be provided for usual applications.

There are a couple of comments here. In order to create a thousand bolts of type-A with size 100 and color green, we might issue the following message to class Bolt.

new: ((bolt-A, 100, green), 1000)

Although OMEGA allocates different (consecutive) oid values to a thousand bolts, they are irrelevant to users. The following type of query might be needed to support OMEGA users to retrieve any one bolt of type-A.

```
SELECT 1
FROM   Bolt
WHERE  part-name=''A-bolt''
```

3.5. Contents Identity

Users may want to handle contents identity. For example, suppose that you have two documents, one of which, say document D1, is written in English, while the other, say document D2, is written in Japanese. You might want to identify D1 with D2 if the contents of those two documents are identical. This identity, called contents identity, can be supported in OMEGA by defining the following object identity rule in class Document.

$$(\forall\, d, d' \in \text{Document})\ (\text{contents}(d) = \text{contents}(d') \rightarrow d = d')$$

Notice first that ''contents'' is defined as a method in class Document, or it is inherited from its super classes. Second, it should be noticed that this object identity rule is a kind of the arbitrary identity rule which was introduced before. That is, even though two OMEGA objects in class Document do not refer to the identical document in the real world, we can say that they are identical if their contents are identical.

3.6. Identity of Complex Objects

As it was mentioned before, one of the characteristics of OMEGA is to support complex objects explicitly by using IS-PART-OF (binary) relations. A basic consideration on the identity of complex object will be done here.

Suppose we have two complex objects o and o′. Since oid is a system-wide unique object identifier, we can say that o is trivially identical with o′ if oid(o) = oid(o′). Next suppose that o and o′ (in the same class) have different oid values. Here it should be mentioned that the essential difference between a complex object and non-complex object in OMEGA is that the former is created not only to refer to a certain object in the real world but also to represent its design hierarchy structure. Therefore, the design hierarchy structure must also be taken into account when we discuss the identity of complex object. For example, suppose that there is a rectangle R (in the real world) which consists of three squares S1, S2, and S3 put side by side from left to right. This rectangle could be represented in OMEGA

in the following way:

$$r = ((s1, s2), s3) \text{ and } r' = (s1, (s2, s3)),$$

where object r and r′ in class Rectangle represent retangle R, and object s1, s2, and s3 in class Square represent square S1, S2, and S3, respectively. Now what kind of object identity can we introduce between r and r′? Our idea is as follows. Although oid values of r and r′ differ and the design hierarchy trees of r and r′ are also different, we can think of r as referentially identical with r′ because what r refers to and what r′ refers to are the identical real world object, i.e. rectangle R. A little bit more formally, we can say that list ((s1, s2), s3) and (s1, (s2, s3)) are identical in the sense that the latter can be obtained from the former by applying a similar law of the associative law of multiplication, i.e. ((a * B) * c) = (a * (b *c)). Therefore we can define the following referential identity rule of complex object in class Rectangle:

$$(\forall r, r' \in \text{Rectangle})\ (\text{structure}(r) = [\text{associative-law}]\ \text{structure}(r') \rightarrow r = r'),$$

where "structure" is a method defined in class Rectangle (or an inherited one) such that structure(r) = ((s1, s2), s3) and structure(r′) = (s1, (s2, s3)) in this case, and "= [associative-law]" means that the right hand side can be obtained from the left hand side by applying the associative law and vice versa. Of course we can expand our discussion to include the case in which complex object o and o′ belong to class A and B (≠ A) as it was done in section 3.3. The arbitrary identity can be defined in an analogous manner.

3.7. Object Similarity

Object similarity can also be supported in OMEGA. The similarity is very close to the arbitrary identity. But they are totally different from semantic point of view. For example, we say that whales are similar to fishes (in shape), but we usually do not say that whales are identical with fishes.

The similarity is denoted by "~". It is defined in the same manner as the arbitrary identity is defined. For example, if we want to say that all objects in class Bolt which have the same bolt-name, size, and color are similar, then the following similarity rule can be defined in that class.

$$(\forall o, o' \in \text{Bolt})\ (\text{bolt-name}(o) = \text{bolt-name}(o')\ \text{AND}$$
$$\text{size}(o) = \text{size}(o')\ \text{AND}\ \text{color}(o) = \text{color}(o') \rightarrow o \sim o')$$

Mathematical similarity, such as similar triangle, can also be treated in the same manner.

3.8. Quotient Class

We have investigated the trivial, referential and arbitrary identity of object and object similarity. It must be pointed out that an important characteristic is hidden there. That is, it is expected that such an identity and similarity satisfy the condition of equivalence relation on a set, i.e., the reflexive, symmetric, and transitive law. Therefore, we can partition a class using an object identity or similarity and define a quotient class with respect to it.

Suppose that such an equivalence relation, denoted by ≡, is defined on class C. Then we can define a new class which consists of all equivalence classes of C with respect to ≡, which is called the quotient class of C with respect to ≡. It is denoted by C/≡.

By introducing the quotient class, OMEGA becomes very powerful to represent various aspects of the reals world. For example, suppose that there are Seller class and Buyer class. It happens quite often that company A sells item i1 to company B, while B sells item i2 to A. To represent them, object s1 and s2 are created in Seller class to refer to company A and B as sellers, respectively, while object b1 and b2 are created to refer to A and B as buyers, respectively. Since a seller and a buyer are dealers, we want to define class Dealer which consists of the representatives of A and B as dealers. In order to do so, we first define class SellerOrBuyer as the set union of class Seller and class Buyer. Now we introduce the

following object identity rule in it:

$$(\forall o, o' \in \text{SellerOrBuyer})\ (\text{company-name}(o) = \text{company-name}(o') \rightarrow o = o')$$

Then by using the equivalence relation = defined in the above rule, we can define a quotient class SellerOrBuyer/=, where an equivalence class containing s1 and b1 represents company A as a dealer, while an equivalence class containing s2 and b2 represents company B as a dealer. Therefore class Dealer can be defined as the quotient class. This is shown in Figure 7. Another example of a quotient class will be given in the next section.

3.9. Substantial Identity and Relational Identity in the Real World

In the real world two types of object identity exist in the following sense. For example, suppose someone said that "I took the same flight." This statement could be used to represent either one of the following two meanings; either 1) I took the same flight, but I didn't meet you, or 2) I usually took the same flight rather than other flights because it was convenient. The identity in the former case is called the substantial identity, while the latter is called the relational identity.

In OMEGA we can support those two types of identity. Suppose that class AllFlights is defined. The substantial identity could be defined by the following referential identity rule:

$$(\forall o, o' \in \text{AllFlights})\ (\text{flight-number}(o) = \text{flight-number}(o')\ \text{AND}\ \text{date}(o) = \text{date}(o') \rightarrow o = o')$$

On the other hand, the relational identity could be defined in the following way. To do so we first introduce another type of object identity rule in it:

$$(\forall o, o' \in \text{AllFlights})\ (\text{flight-number}(o) = \text{flight-number}(o') \rightarrow o = o')$$

Then, by using this identity we can define a quotient class of AllFlights, i.e. AllFlights/=, which is denoted by Flight. Now we define a referential identity rule in Flight:

$$(\forall o, o' \in \text{Flight})\ (\text{flight-number}(o) = \text{flight-number}(o') \rightarrow o = o')$$

It is easy to see that this identity rule represents the relational identity of flight in the real world.

3.10. Object Copy

Let o be an object of class C. Then we can create a new object, say o', which has the same attribute values of o. We call o' a copy of o. Since o' becomes a new object in OMEGA, a system wide unique object identifier (oid value) is given to it. Depending on the similarity or identity rule defined in class C, o' could be similar, or identical or not with o. Of course, such an operation as creating a thousand copies of a bolt may be necessary. Depending on application, sometimes it may be necessary to keep information about what copies are made from what object.

4. Basic Object Management Scheme in OMEGA

4.1. Object Naming

In OMEGA, a creator is able to name objects which he/she creates. The object name is completely irrelevant to oid, i.e. the object identification number which is given by OMEGA. Since a creator can name objects arbitrarily, the object name does not have unique object identification capability in general, while oid has.

Suppose that there is class Car, and John created a car object to register his car and named it myCar. Also, suppose that George registered his car and named it myCar. Then it is impossible to distinguish which myCar represents John's car. In order to resolve this ambiguity, we assume that OMEGA qualifies the user specified name by adding the creator name. That is, John's car has name John.myCar, while

George's car has name George.myCar.

However, this naming scheme is still unsuitable. For example, suppose that there is class Toy, and John created a toy object to represent his favorite radio control car and named it myCar. Then it becomes again impossible to distinguish whether John.myCar represents his real car or a toy. To resolve this ambiguity, we again qualify the object name by using class name. That is, the above three objects may have the following names:

John.myCar@Car
George.myCar@Car
John.myCar@Toy

Formally, an OMEGA object may have the user define object name which is a triple:

creator-name.user-specified-object-name@class-name

Notice that this naming scheme is still insufficient for unique object identification purpose. A simple example is as follows. Suppose that John has three cars and created three car objects in class Car to represent them. If all of them are named myCar, then it is impossible to distinguish this car from that car by the object name John.myCar@Car. We understand this situation in this way. If John really wants to distinguish each of his cars in OMEGA, he must assign different names to each of them, i.e. myCar1, myCar2, and myCar3 to his first, second and third car, respectively. Then if he says John.myCar2@Car, then his second car object can by uniquely obtained. It is up to users how to name objects so that they have unique object identification capability or not.

4.2. Single Class vs. Multiclass Object Management

When the object management scheme in OMEGA is discussed, object identity must be taken into consideration. Two types of object management schemes are investigated here which are called single class management scheme and multiclass management scheme. Let us take a typical example to explain our investigation.

Suppose that there is a foreign male student on the university register. His attributes are: 870123 as student identification number (sid), Jacky Chen as name, male as sex, Tsukuba-Science-City as address, and Hong Kong as nationality. Also suppose that ForeignStudent, MaleStudent, and Student classes are defined as it was done in section 3.3. Then in order to represent Jacky, object s1 will be created in class ForeignStudent, while object s2 will be created in class MaleStudent. A referential identity rule of object

$$(\forall\, s, s' \in \text{Student})\ (\text{sid}(s) = \text{sid}(s') \rightarrow s = s')$$

will be defined in class Student so that any objects in class Student which refer to the same student in the real world can be regarded as identical in OMEGA. This situation is shown in Figure 8. Notice that when Jacky leaves university, two operations are necessary to reflect this fact. That is, since there are two representatives (i.e. s1 and s2) of Jacky in OMEGA, s1 and s2 must be deleted from ForeignStudent and MaleStudent classes, respectively, to reflect the withdrawal. In general, if there are n representatives of Jacky, n operations are necessary to reflect it. We call this type of object management scheme multiclass object management scheme.

In order to reduce the number of such deletion operations to one, one might define a new class which may be called ForeignMaleStudent. Conceptually, this is defined as an intersection of class ForeignStudent and class MaleStudent. Therefore, the class inherits properties form class ForeignStudent and class MaleStudent. Then, it is clear that creation of one object, say s3, in class ForeignMaleStudent is enough to represent Jacky who is a foreign male student as is shown in Figure 8. Now, notice here that only one operation is enough to reflect his withdrawal; the deletion of s3 from class ForeignMaleStudent. We call this object management scheme single class object management scheme.

Now the problem is as follows. Although we must introduce a new class ForeignMaleStudent, can we say that the single class approach is better than the multiclass approach in terms of the number of operations needed to reflect real world changes? The answer is no because the tables are turned if we meet another situation: Suppose that Jacky succeeded to be naturalized. Since he is not a foreigner any more, this must be reflected in OMEGA. The following alternatives are observed:

1. [Single Class Management Scheme]. Delete object s3 from ForeignMaleStudent class and create a new object, say s4, in MaleStudent class.
2. [Multiclass Management Scheme]. Delete object s1 from ForeignStudent class.

That is, the multiclass approach needs only one operation, while the single class approach needs two operations. Therefore both approaches have merits and demerits. However, the multiclass approach seems attractive because it could be possible to provide a set of very powerful, i.e. semantical object manipulation operations if the multiclass approach is taken under good management of object identity. Let us consider the above example again. By using the object identity rule

$$(\forall s \in \text{ForeignStudent}, \forall s' \in \text{MaleStudent})\ (\text{sid}(s) = \text{sid}(s') \rightarrow s = s'\ [\text{Student}])$$

we can introduce two types of object delete operations:

DELETE s1
FROM ForeignStudent, and

DELETE s1 WITH = IN Student
FROM ForeignStudent,

where = is the above object identity defined in Student. The former intends to delete exactly one object s1 from class ForeignStudent, while the latter intends to delete not only s1 but also every object in any subclass of class Student which is determined identical with s1 by using the object identity rule R. It is clear that the former deletion will be issued when Jacky is naturalized, while the latter will be issued when Jacky leaves university. Because of this reason, OMEGA adopted the multiclass object management scheme.

5. Summary

In this paper we have discussed objects concept and basic object management scheme in OMEGA which is now under development at the University of Library and Information Science. Particularly the concept of object identity was introduced and investigated. Three types of object identity were made clear, which are the trivial identity, the referential identity, and the arbitrary identity. Based on this result, various types of object identity which exist in the real world were characterized in OMEGA. The contents identity, the identity of complex objects, and the object similarity were such examples. Because object identity and similarity satisfy the condition of the equivalence relation on a set, the quotient class was defined. By using this, other types of object identity such as the substantial identity and the relational identity were characterized. Object copy was also discussed. Moreover, the object naming and the comparison between single class and multiclass object management schemes were investigated. It was made clear that multiclass object management scheme could be powerful to represent the real world semantics if object identity is will managed. Design of object manipulation language based on our approach and implementation of OMEGA are left as future works.

Acknowledgements

The author appreciates comments given by Dr. Henry F. Korth of the University of Texas at Austin. This research was partly supported by the Japanese Ministry of Education, Culture, and Science, Grant number 61580017.

References

[BKKK87] Banerjee, J., W. Kim, H-J. Kim, and H. F. Korth, "Semantics and Implementation of Schema Evolution in Object-Oriented Databases," Proc. 1987 ACM-SIGMOD Conference, 311-322.

[BKW87] Banerjee, J., W. Kim, D. Woelk, N. Ballou, H-T, Chou, "A Data Model for Object-Oriented Persistent Databases," ACM Trans. on Office Information Systems, to appear 1987.

[BuCV86] Buchman, A.P., R. S. Carrera, and Bazquez-Galindo, "A Generalized Constraint and Exception Handler for an Object-Oriented CAD-DBMS," Proc. 1986 International Workshop on Object-Oriented Database systems, (Sept. 1986), 38-49.

[ChHT86] Christodoulakis, S., F. Ho, and M. Theodoridou, " The Multimedia Object Presentation Manager of MINOS: A Systematic Approach," Proc. 1986 ACM-SIGMOD Conference, 295-310.

[Codd70] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," Comm. of the ACM, 13.6, (1970), 377-387.

[CoMa84] Copeland, G., and D. Maier, " Making Smalltalk a Database System," Proc. 1984 ACM-SIGMOD Conference, (1984), 316-325.

[CVL84] Christodoulakis, S., J. Vanderbroek, J. Li, T. Li, S. Wan, Y. Wang, M. Papa, and E. Bertino, "Development of a Multimedia Information System for an Office Environment," Proc. 10th International Conference on VLDB, (1984), 261-271.

[GoRo83] Goldberg, A., and D. Robson, Smalltalk-80: Language and its Implementation. Addison-Wesley, (1983).

[HaLo82] Haskin, R.L., and R. A. Lorie, "On Extending the Functions of a Relational Database System," Proc 1982 ACM-SIGMOD Conference, (1982), 201-212.

[IEEE85] Special Issue on Object-Oriented Systems," IEEE Database Engineering, 8.4, (1985).

[LyKe86] Lyngbaek, P., and W. Kent, "A Data Modeling Methodology for the Design and Implementation of Information Systems," Proc 1986 International Workshop on Object-Oriented Database Systems, (1986), 6-16.

[MaDa86] Manola, F., and U. Dayal, "PDM: an Object-Oriented Data Model," Proc. of the International Conference on Object-Oriented Database Systems, (1986), 18-25.

[Masu85] Masunaga, Y., "A Conceptual Design of Multimedia Database Management Systems," Research Report of Univ. of Library and Information Science, 4.1, (December 1985), 9-26, (in Japanese).

[Masu87] Masunaga, Y., " Multimedia Databases: A Formal Framework," Proc. IEEE Computer Society Symposium on Office Automation, (April 1987), 36-45.

[MBH85] Maryanski, F., J. Bedell, S. Hoelsher, S., Hong, L-A. McDonald, J. Peckham, and D. Stock, "The Data Model Compiler: A Tool for Generating Object-Oriented Database Systems," Proc. of the International Conference on Object-Oriented Database Systems, (1986), 73-84.

[NiTs85] Nierstrasz, O.M., and D.C. Tsichritzis, "An Object-Oriented Environment for OIS Applications," Proc. 11th International Conference on VLDB, (1985), 335-345.

[Oren86] Orenstein, J.A., "Spatial Query Processing in an Object-Oriented Database System," Proc. 1986 ACM-SIGMOD Conference, (1986), 326-336.

[TCE83] Tsichritzis, D., S. Christodoulakis, P. Economopoulos, C. Faioutsos, A. Lee, D. Lee, J. Vanderbroek, and C. Woo, "A Multimedia Office Filing System," Proc. 9th International Conference on VLDB, (1983), 2-7.

[WoKl86] Woelk, D., W. Kim, and W. Luther, "An Object-Oriented Approach to Multimedia Databases," Proc. 1986 ACM-SIGMOD Conference, (1986), 311-325.

[WoKi87] Woelk, D., and W. Kim, ''Multimedia Information Management in an Object-Oriented Database System,'' Proc. 13th International Conference on VLDB, (September 1987), 319-329.

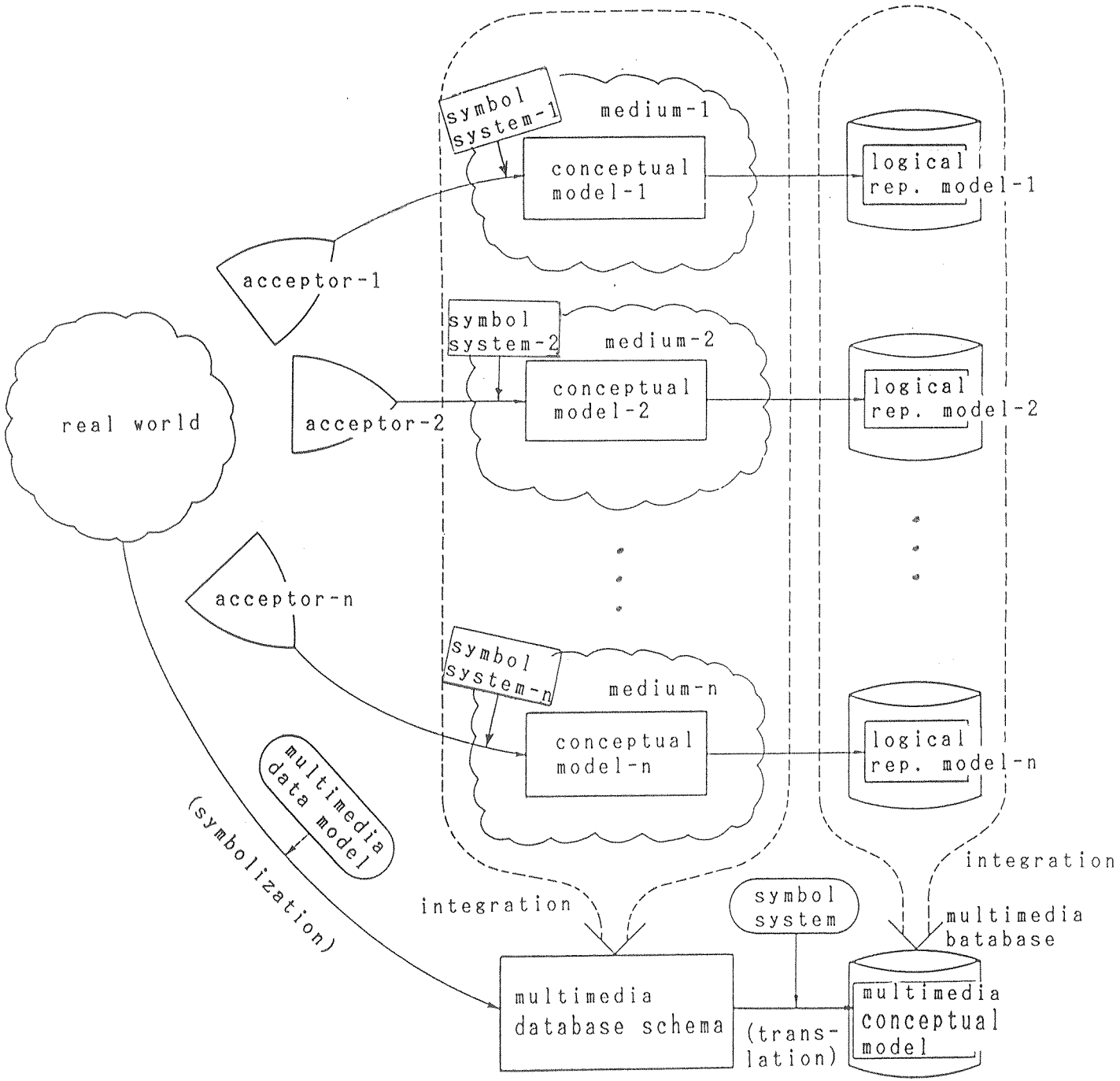

Figure 1. Multimedia Data Modelling.

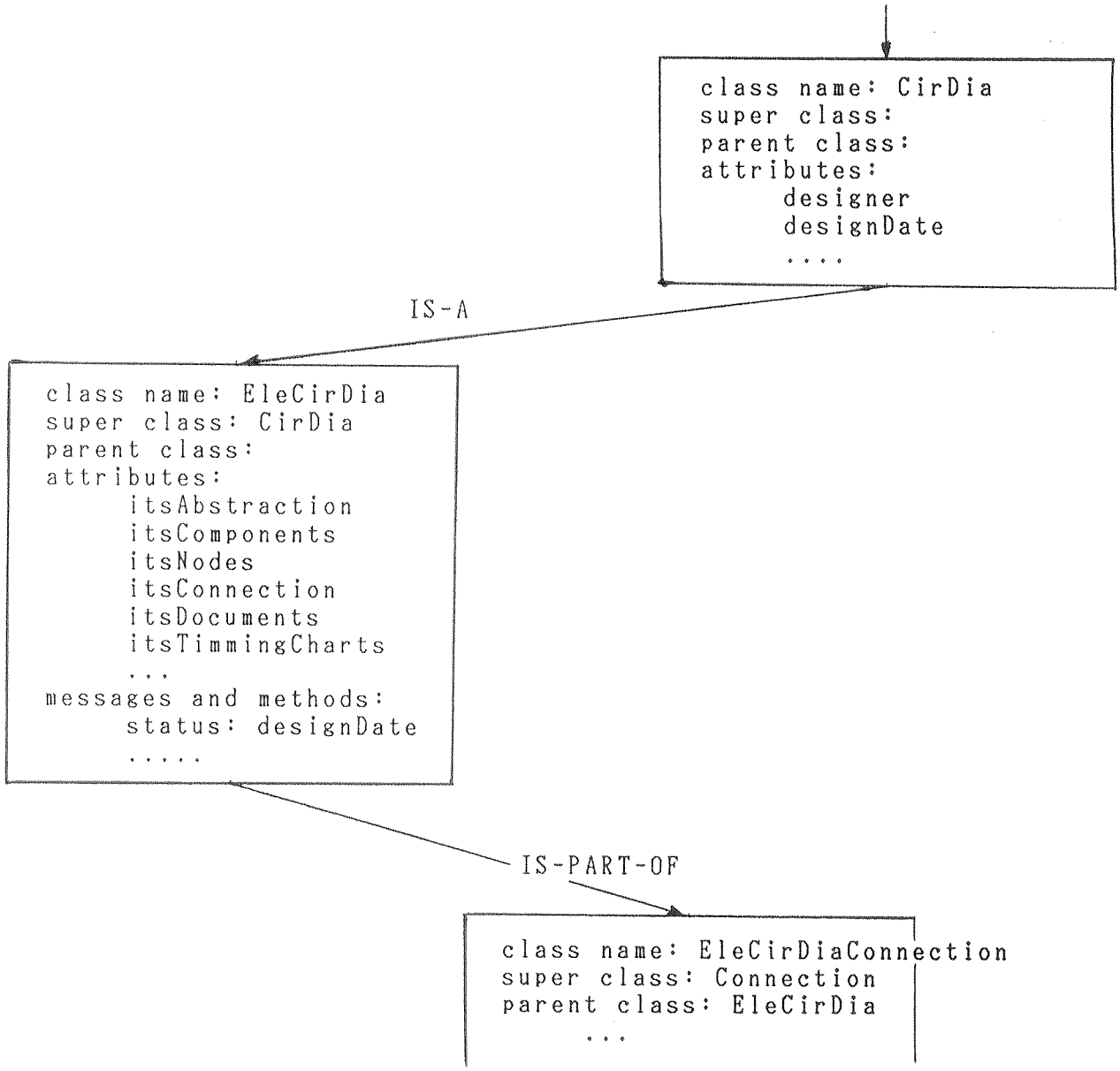

Figure 2. Class Description in OMEGA.

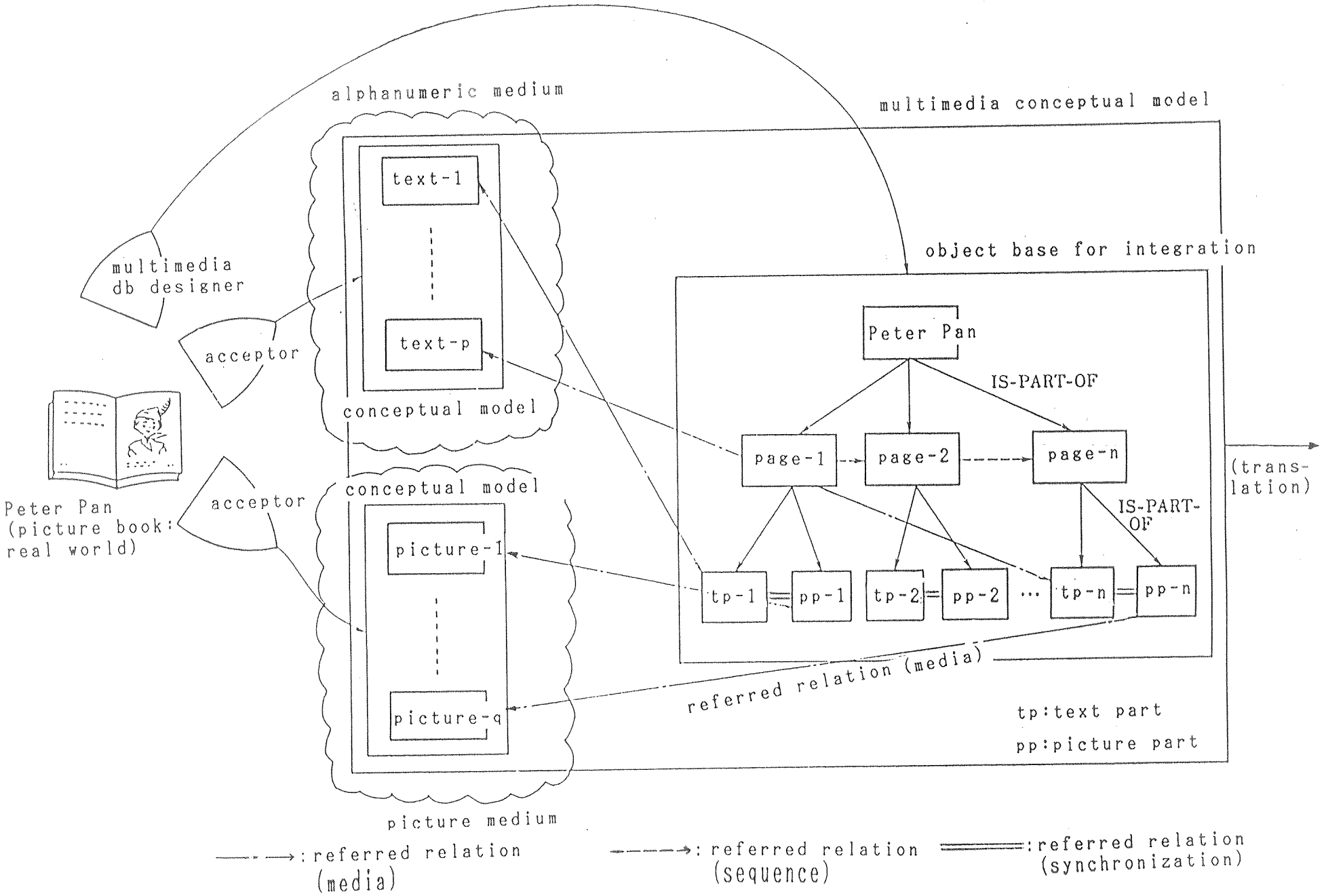

Figure 3 (a). The Referred Object Integration Method (Concept).

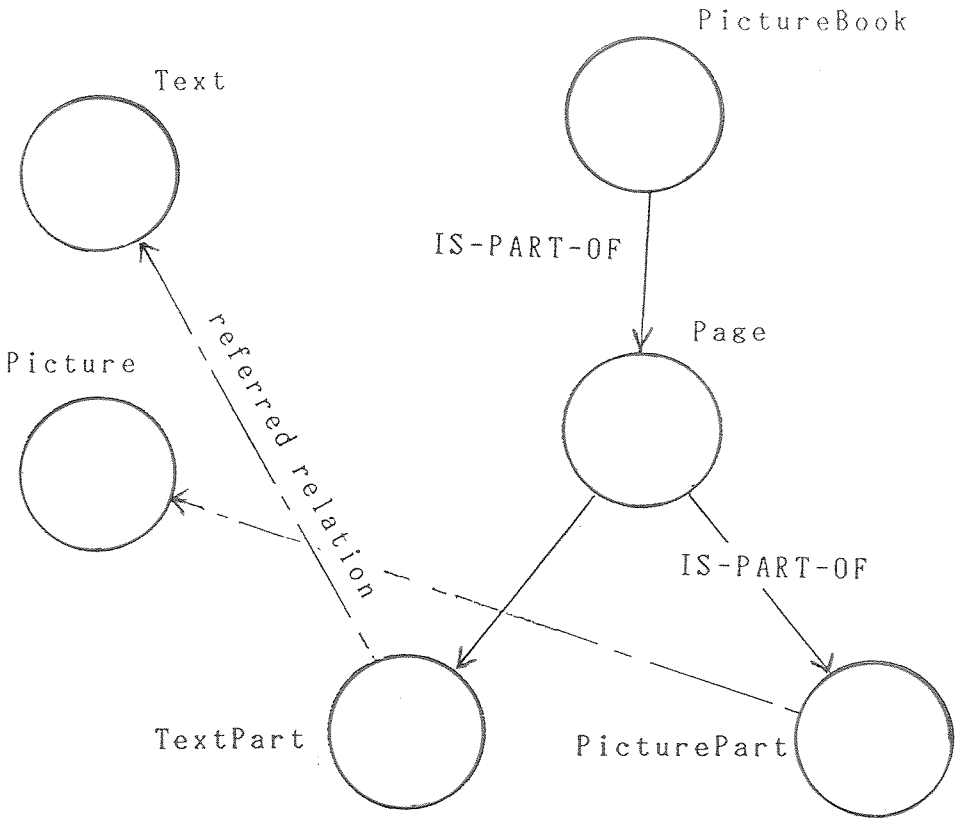

Figure 3 (b). OMEGA Description of Multimedia Conceptual Model of Figure 3 (a).

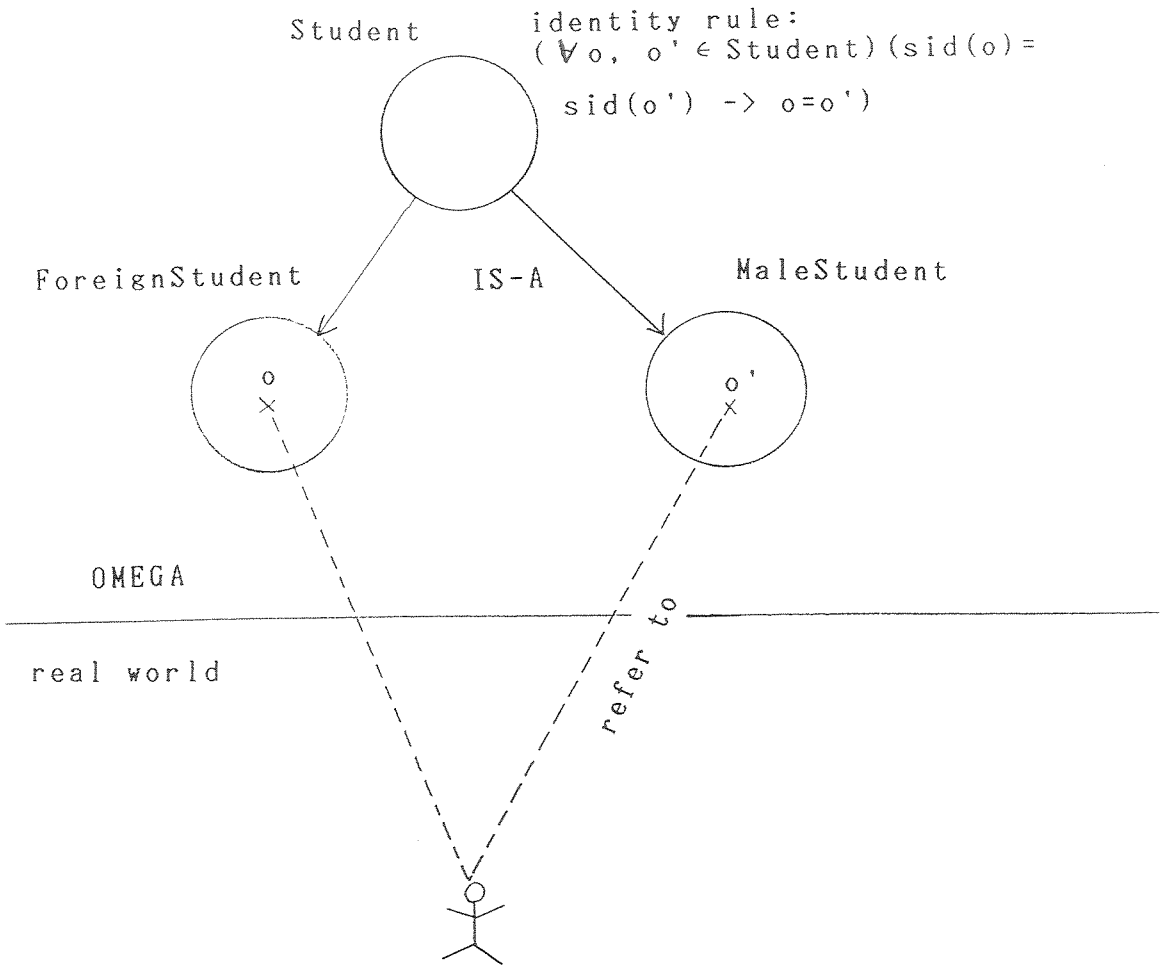

Figure 4. The Referential Identity.

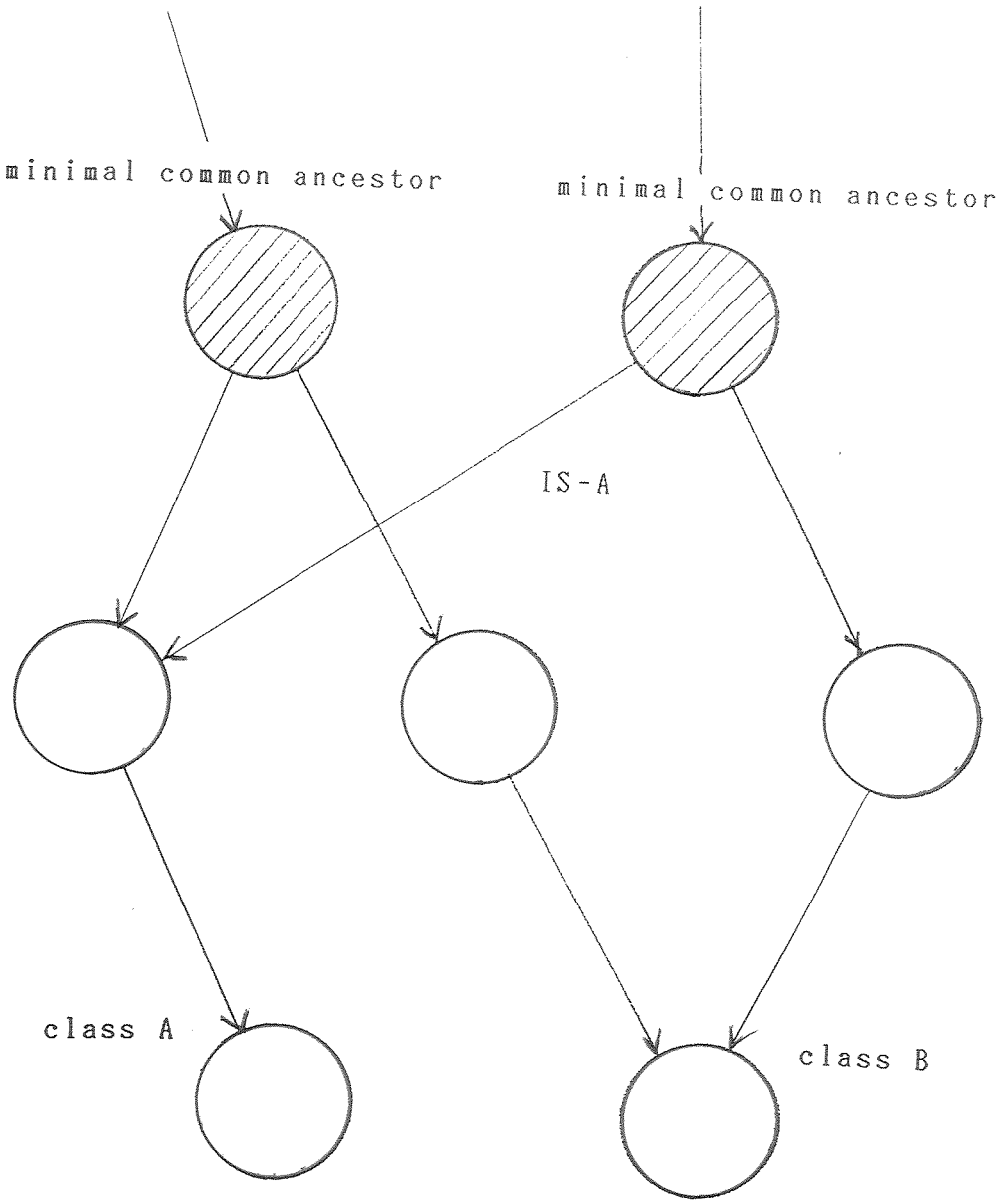

Figure 5. Minimal Common Ancestors of class A and class B.

identity rule:

(∀o, o'∈ ChemicalCompound)(chemical-formula(o)=

chemical-formula(o') -> o=o')

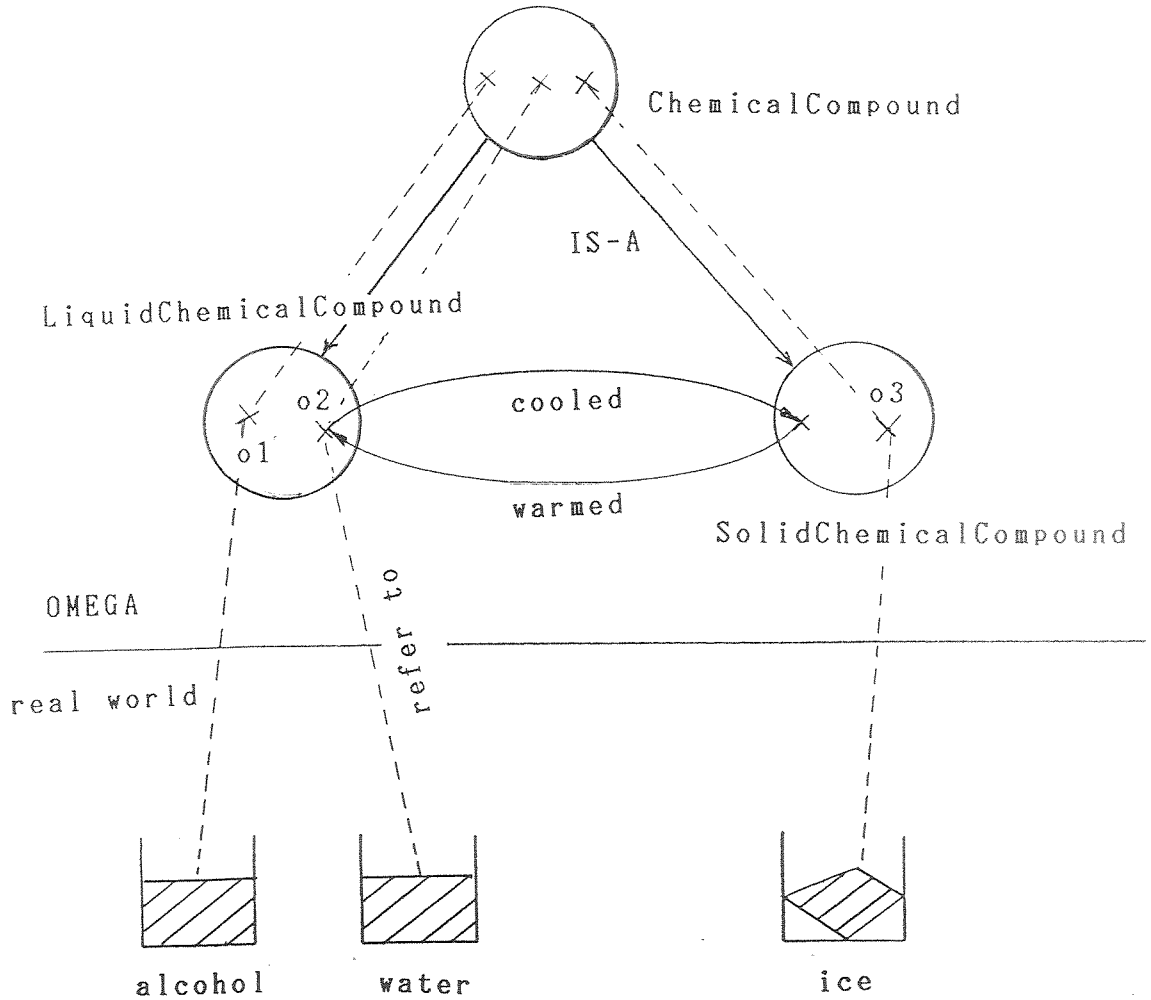

Figure 6. The Arbitrary Identity.

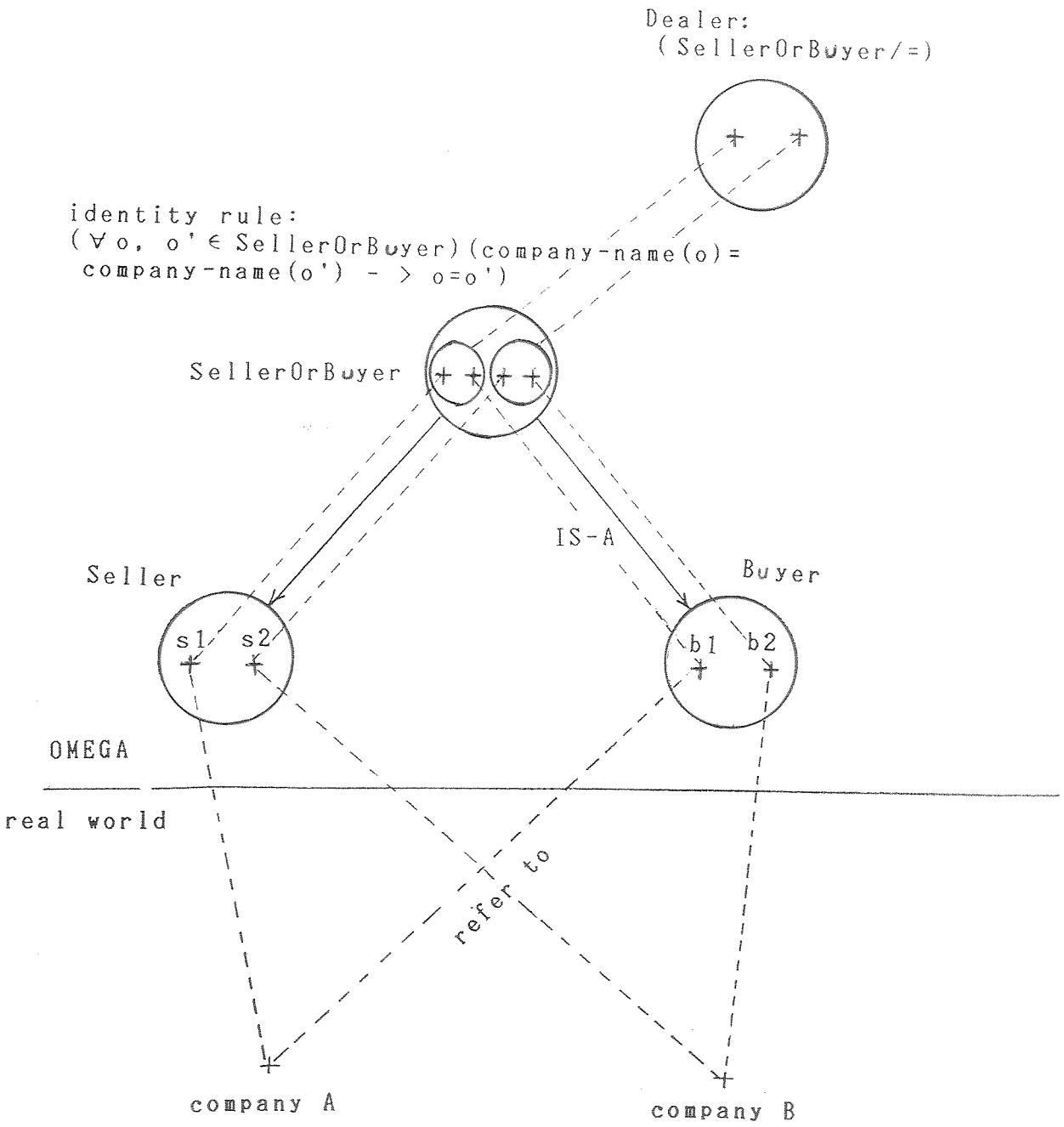

Figure 7. Quotient Class.

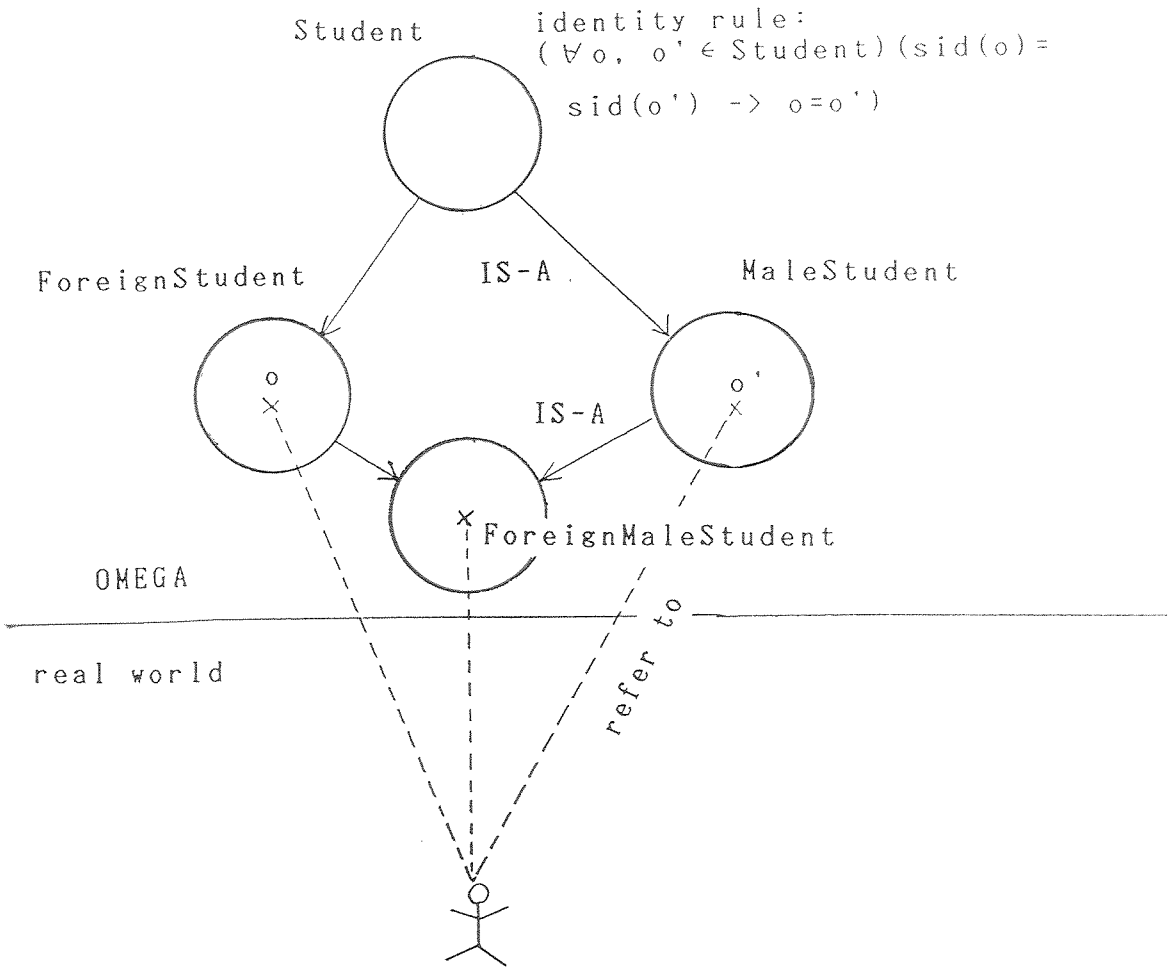

Figure 8. Single Class vs. Multiclass Object Management.