CSCI585 Classified Interrelated Object Model

Dennis McLeod and Shan Gao
University of Southern California

1. Introduction

CIOM is a simple object model, containing the major components of semantic and object-based database models; it is essentially a modified subset of SDM [1]. With the basic structures, semantic integrity constraints, and operations of SDM, CIOM intends to model the real world and capture the semantics of an application environment.

2. Classes

A class is defined as a collection of objects (information units). This collection is a logically homogeneous set of one type of objects. The objects of a class are said to be the members of the class.

2.1 Class Name

The name of a class is used to identify the class from others. A class name can be any string of symbols that uniquely represents the nature of this class. In CIOM, the class name is written inside of an oval to represent the class.

2.2 Built-in Classes

There are several built-in classes predefined in CIOM. They are often used as the value classes for basic attributes. For example, STRINGS is the class of all strings over the set of alphanumeric characters. NUMBERS is the class of all numeric values of integers, real or complex numbers. BOOLEANS is the class of two values "True" and "False". Other classes such as INTEGERS, DATES, IMAGES, AUDIOS, VIDEOS are defined as well. Their meanings are self-explanatory.

2.3 Member Attributes

Member attributes are the common aspects of members of a class. A member attribute describes each and every member of the class. For example, every person has an age, so "Age" is one common attribute of the class "People". The Dot Notation is used to denote attributes. The above attribute can be written as People**.**Age.

In CIOM, an attribute is drawn as an arrow pointing from one class to another as shown in Figure 1. This is understood as: Class C1 has an attribute A whose value class (set of possible values) is C2.

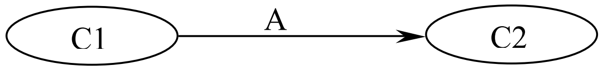


Figure 1. A member attribute

2.3.1 Attribute Names

Each attribute has a name that uniquely identifies that attribute from other attributes of the same class. Similar to a class name, an attribute name can be any string of symbols.

2.3.2 Attribute Value Classes

The value class of an attribute is the set of all possible values that applies to the attribute. This set can be any built-in classes or any user-defined classes as well. For example, the value class of People**.**Age is INTEGERS, while the value class of People**.**Parents is the class People itself.

2.3.3 Inverse Attributes

In a CIOM schema, attributes always have inverses, and are usually shown in pairs. In each pair, each attribute is said to be the inverse of the other. This relation is specified symmetrically. The inverse attribute of attribute A is denoted as A^{-1}. In figure 2, attribute A and A^{-1} are drawn as a pair of inter-linked arrows. C1 is the value class of attribute A^{-1}.

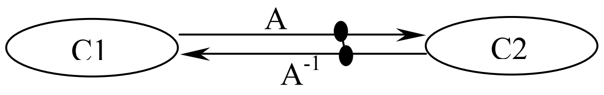
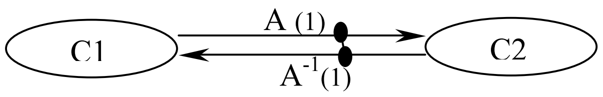


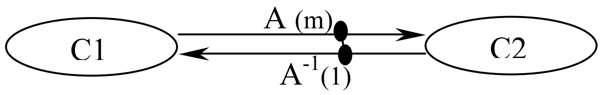
Figure 2. Inverse attributes

2.3.3 Cardinality

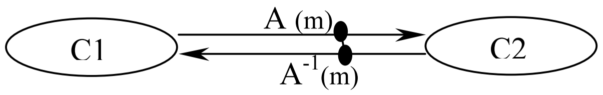
Attributes can be classified into three categories: One-to-One, One-to-Many and Many-to-Many, as shown in Figure 3.



a) One-to-One relationship



b) One-to-Many relationship



c) Many-to-Many relationship

Figure 3. Cardinality

a) A One-to-One relationship means for each member of class C1, its value of attribute A can only be one member of the value class C2. So does the inverse attribute A^{-1}.
b) A One-to-Many relation means for each member of class C1, its value of attribute A can be multiple members of the value class C2. But the value of inverse attribute A^{-1} can only be one member of C1.
c) A Many-to-Many relation means for each member of class C1, its value of attribute A can be multiple members of the value class. So does the inverse attribute A^{-1}.

Examples are: Every person has a unique SSN and the SSN uniquely identifies that person. Every person has one mother, but the mother can have multiple children. Every person can have multiple friends, and every person can be the friend of more than one person.

2.3.4 Non-null Attributes

In CIOM, an attribute can be specified as non-null, abbreviated as "nn". "Null" means no value.

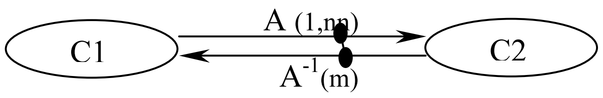


Figure 4. Non-null attribute.

2.3.6 Attribute Mappings

In some cases, in order to retrieve needed information, attributes can be concatenated. For example, in order to get the age of a person's friends, we can use People**.**Friends**.**Age.

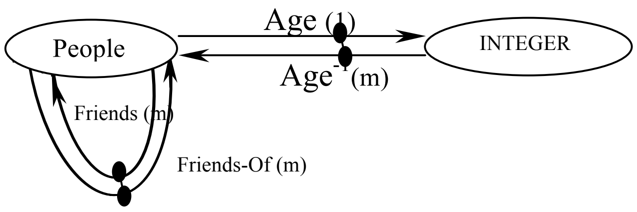


Figure 5. Attribute Mapping

2.4 Class Attributes

CIOM supports aggregate functions by means of class attributes. An attribute can be defined to be the number of members in one class (similar to the count() function in relational SQL). An attribute can also be defined to be the maximum, minimum, average, and sum of the members in one class, similar to max(), min(), avg(), sum() functions in SQL. These attributes are useful when we want to know the aggregate properties of a class, disregarding the detail of its individual members.

3. Subclasses

A subclass means specialization, viz. its membership is a subset of the members of its parent class. For example, Men and Women are two subclasses of class People. In CIOM, a subclass is drawn as a double-line arrow. There are two ways to define a subclass: by operation or by predicate.

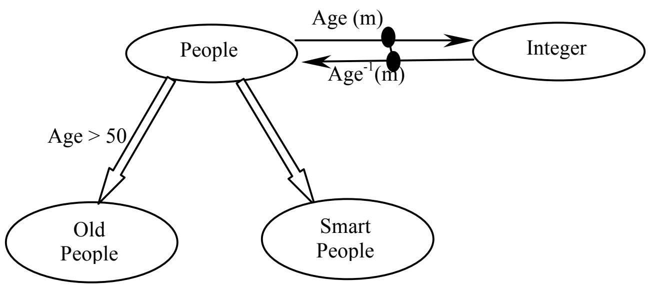


Figure 6. Subclasses

3.1 Predicate Defined Subclasses

For a predicate defined subclass, its membership is determined by some conditions upon one or more attributes. The conditions are called predicates. If a member's attributes values satisfy all conditions, then the member is automatically added to the subclass. In Figure 6, "Old People" is a subclass of class "People" defined by the predicate "Age > 50".

3.2 Operationally Defined Subclasses

For operationally defined subclasses, their membership is not automatically determined by any predicate, but decided by user operations (adding/inserting members). In Figure 6, "Smart People" is a subclass of "People" whose members are explicitly added or removed by user.

3.3. Set-Operators Defined Subclasses

A class can also be defined by set operations: intersection, union, and difference.

3.3.1 Class Intersection

An intersection subclass contains the members in both of the parent classes involved in the intersection. For type compatibility, the Classes involved in this operation must both be subclasses of some common parent, directly or indirectly. The intersection operator is denoted as "∩".

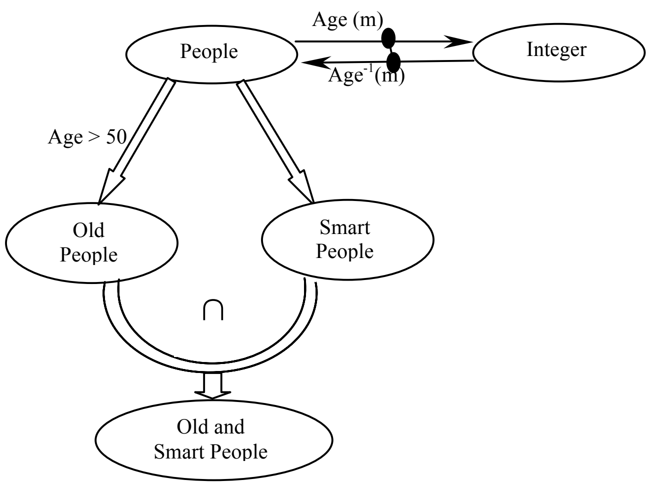


Figure 7. Class Intersection

3.3.2 Class Union

A union subclass contains the members in either of the parent classes involved in the union operation. The union operator is denoted as “U”.

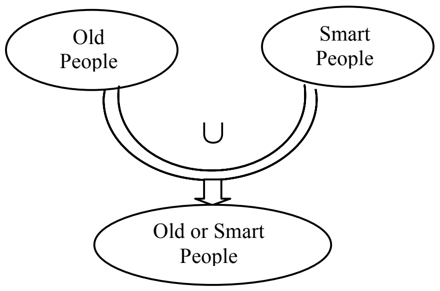


Figure 8. Class Union

3.3.3 Class Difference

A difference subclass contains the members of one parent class that are not in the other class. The difference operator is denoted as “–”. Note the difference operator is not symmetric, i.e., A-B ≠ B-A.

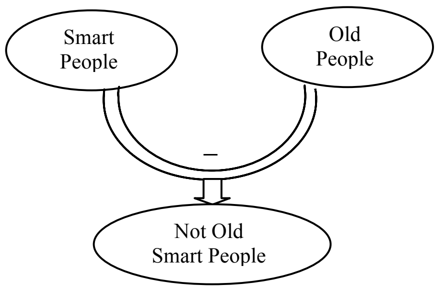


Figure 9. Class Difference

3.4 Attribute Inheritance

A subclass, whether predicate defined or operationally defined, automatically inherits all attributes from its parent class. These attributes need not to be shown on the subclass but they exist implicitly. In Figure 6, Age is an attribute of class People, Old People, and Smart People.

For subclasses defined by set-operators, the attribute inheritance is more complex. A subclass by parents intersection inherits all attributes of both parents. A subclass by parents union inherits only those attributes common to both parents. A subclass generated by parents difference inherits all attributes of the parent at the left side of the operator. This is summarized in table 1.

C3 =	C1 ∩ C2	C1 U C2	C1 – C2
Members(C3) =	Members(C1) ∩ Members(C2)	Members(C1) ∪ Members(C2)	Members(C1) – Members(C2)
Attributes(C3) =	Attributes(C1) ∪ Attributes(C2)	Attributes(C1) ∩ Attributes(C2)	Attributes(C1)

Table 1. Set-Operator Defined Subclasses

3.5 Mutually Exclusive and Collectively Exhaustive Constraints

For two or more subclasses, mutually exclusiveness means that there is no member belongs to more than one of the subclasses. For example, a person cannot be a male and a female at the same time, and then we say the subclasses “male” and “female” are mutually exclusive.

Collectively exhaustiveness means for any member of the parent class, it must belong to at least one of the subclasses. For example, if a person must be either a male or a female, then we say the subclasses “male” and “female” are collectively exhaustive.

4. Grouping Classes

A grouping class defines a class of classes. It is a second order class in that its members are of higher-order object type than those of the underlying classes. We denote grouping classes by triple arrows (see figure 10). The members of a grouping class are viewed as classes themselves.

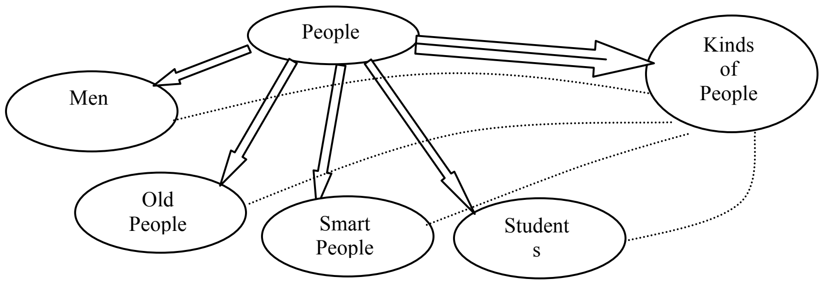


Figure 10. Grouping Class

In Figure 10, “Kinds of People” is a grouping class of “People”. There are four members of it: Men, Old People, Smart People, and Students. The members are themselves classes.

5. An Example CIOM Conceptual Schema

Imagine that you have recently started your own business, acquiring the assets of a failed startup; your new business is an e-commerce company **Future.com**. The major business of **Future.com** is to sell customized computers. All computers have an identifying model number and a price. The model number must be unique, and both pieces of information are mandatory. Each computer has several specified parts (e.g., CPU), other parts are user-configurable (e.g., Scanner). It is necessary to keep track of this information for each customized computer system.

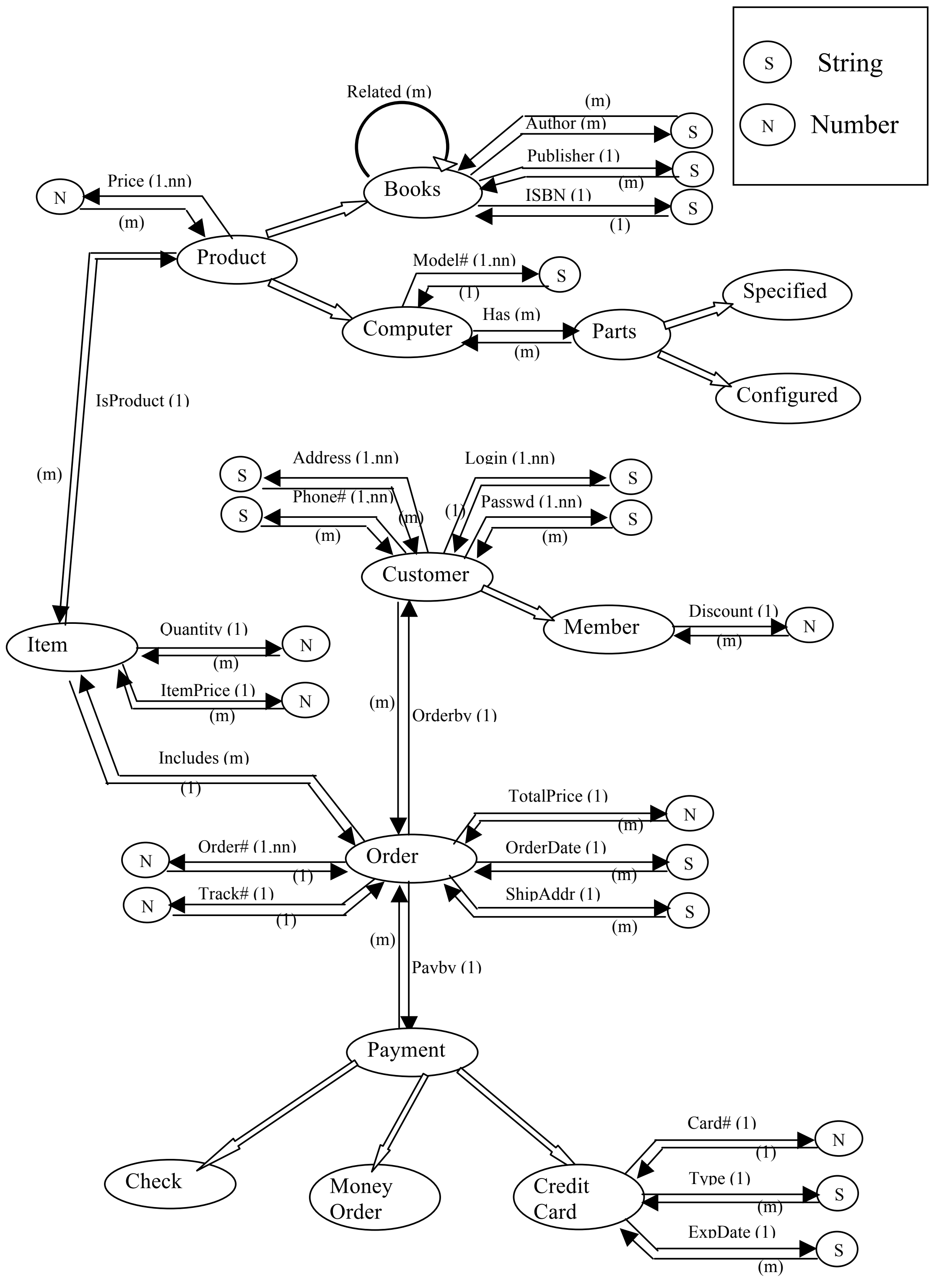
Future.com also sells computer books. The book data includes its author(s), publisher, and ISBN number. You like the idea of providing some guides to customers. (e.g., Customers who bought book A usually bought book B and C also). Thus for each book, you need to keep a list of other books which are frequently sold together with this book.

Future.com keeps track of each customer’s address, phone number, unique login name, and password. Once again, all information is mandatory. All members can buy products at a fixed discount.

A **Future.com** shopper uses a shopping cart to keep all items. The shopping cart itself does not constitute a verified order; it is just a tentative version. Therefore, **Future.com** does not want to keep any shopping cart information in its database. However, if a client checks out, the information in the shopping cart is entered into the database and a tracking number is assigned, which can be used to check the status of the order.

This order information consists of a unique order number, customer, items (computers/books), a quantity per item, the item prices, the total price and order date. It also keeps information of the payment method, and shipping address. The payment method could be a credit card, check, or money order. If a shopper uses a credit card, **Future.com** will request a card number, card type, and expiration date.

Below is one possible CIOM conceptual schema to capture the information specified above (note that for simplicity, attribute and inverse pairs are indicated by placing them in close proximity):



Reference:

[1] Hammer, M., and McLeod, D. "Database Description with SDM: A Semantic Database Model". ACM TODS 6(3): 351-386 (1981).