A Close Look at the IFO Data Model

Magdy S. Hanna, Ph.D
Graduate programs in Software
University Of St. Thomas
St. Paul, Minnesota

1. Introduction

The IFO data model was proposed by Abiteboul and Hull [Abiteboul 87] as a formalized semantic database model. It has been claimed by the authors that the model subsumes the Relational model [Codd 70], the Entity-Relationship model [Chen 76], the Functional Data Model [Kerschberg 76] and virtually all of the structured aspects of the Semantic Data Model [Hammer 81], the INSYDE Model [King 85], and the Extended Semantic Hierarchy Model [Brodie 84].

This paper examines the IFO data model as presented in [Abiteboul 87], compares it to other models, and thus concludes that the IFO data model is actually a subset of the Semantic Data Model proposed by Hammer in [Hammer 81]. The paper also shows that the IFO data model has failed to support concepts that are essential to both the E-R model and the Semantic Data Model which are claimed to be subsumed by the IFO model.

Section 2 discusses the three IFO constructs, *objects*, *fragments*, and *relationships*. The mapping of these constructs to constructs in the Semantic Data Model is established as an informal proof of the result that the IFO model is subsumed by the SDM.

Section 3 lists constructs supported by the Entity-Relationship model [Chen 76, Teorey 86] as will as constructs supported by SDM [Hammer 81]that the IFO data model fails to support.

2. IFO Data Model Constructs

This section provide a discussion of the three IFO constructs, *objects*, *fragments*, and *ISA relationships*. The mapping of these constructs to constructs in the Semantic Data Model, as well as other models, is established as an informal proof of the result that the IFO model is subsumed by the SDM. Figures 1 through 6 show different IFO constructs and their mapping to corresponding SDM constructs[1]. In these figures, diagrams are IFO constructs as they appeared in [Abiteboul 87], while text represents equivalent SDM constructs.

2.1. Objects and Object Types

An object type in IFO is a collection of objects having the same characteristics. Object types in IFO correspond to *classes* in SDM, *object types* in the binary relationship model [Nijssen 77, Mark 83], and *entity sets* in E-R model. There are three kinds of *atomic* object types namely *printable*, *abstract*, and *free* object types and two *constructed* object types namely *grouping* and *cartesian product*. These are discussed below.

Printable Object Types

Printable object types are collection of objects having predefined types such as INTEGER, CHARACTER, etc. They serve as the basis for input and output. These correspond to *value sets* in the ER model, *lexical object types* in the binary relationship model, and *name classes* in SDM. In IFO diagrams, printable object types are represented by squares. The *name class* GRADES In figure 1 is a more precise representation to the printable object type.

Abstract Object Types

Abstract object types represent objects in the real world that have no underlying structure. Examples of abstract object types are PERSON, COMPANY, COURSE, etc. These correspond to *nonlexical object types* in the binary relationship model, *entity sets* in the ER model, and *base classes* in SDM. In IFO diagrams, abstract object types are represented by diamonds. In figure 4, the SDM *base class* COURSES corresponds to the IFO abstract object type.

Free Object Types

Free object types, as defined in [Abiteboul 87], represent entities obtained via ISA relationships.

[1] To facilitate the mapping between the two models, examples in this paper were adopted from [Abiteboul 87].

For example, in a university database, PERSONs are abstract objects while STUDENTs are free objects. Free object types are represented in the IFO diagram by a circle. They correspond to *subsets* in the E-R model [Teorey 86], and *subclasses* or *nonbasic classes* in SDM (see section 2.3 and figure 6).

It should be noted that the definition of *free object types* in IFO does not provide clear criteria for the distinction between abstract objects and free objects. For example, DEPARTMENT and COURSE in figure 6, which was adopted from [Abiteboul 87] were considered free objects although they were not involved in any ISA relationship[2].

The IFO model does not specify criteria for forming subtypes (free objects), while in SDM, subclasses can be *attribute-defined*, user-controllable, *existence*, or an *intersection*, *union*, or *difference* of two other subclasses [Hammer 81].

Grouping

Grouping in IFO correspond to the procedure of forming finite set of objects of a given structure type. A grouping type consists of sets of objects and is represented in the IFO diagram by a "star vertex". For example, the *grouping type* CLASS in figure 2 is defined to be sets of STUDENTs. This corresponds to *grouping classes* in the SDM. However, the IFO model, does not specify criteria for constructing grouping types while in SDM, grouping classes can be *expression-defined*, *enumerated*, or *user-controllable*. It should be noted that all the "grouping" examples presented in [Abiteboul 87] involved only free object types (represented by circles). It is not clear whether grouping could also involve abstract types. This observation is also true about the *cartesian product* discussed below.

Cartesian Product

A cartesian product in IFO represent objects as ordered n-tuples of other objects. For example, a MOTOR-BOAT can be defined as a cartesian product of HULL and MOTOR. This construct has been called *aggregation* by Smith and Smith in [Smith 77]and corresponds to constructing *relations* from *attributes* in the relational model, constructing record types from *components* or *items* in the network model, and defining *member attributes* for classes in SDM. Figure 3 shows how cartesian product can be accurately modeled in SDM. In the IFO diagram, cartesian product is represented by a "cross vertex".

2.2. Fragments

Fragments in IFO are used to represent relationships between objects exactly the same way *functions* in the functional data model [Shipman 81] do. It should be noted that fragments in the IFO have limited functionality compared to functions in FDM. Fragments can only be used to model *one-to-one* associations between two objects. *One-to-many* associations are modeled indirectly using fragments and groupings. For example, in figure 4, the fragment ENROLLMENT maps a COURSE to groups of STUDENTs which seems to be rather unnatural. In FDM as well as in all other data models, *one-to-many* associations are directly represented.

Furthermore, the IFO model does not provide a direct way to model *many-to-many* associations. In the functional data model, *many-to-many* associations are represented by two multi-valued functions, one is the *inverse* of the other. The three kinds of associations (1-1, 1-M, and M-N) are directly represented in the E-R model. In the SDM, *one-to-one* and *one-to-many* associations are represented by single-valued attributes and multi-valued attributes, respectively. *Many-to-many* associations are represented by two multi-valued attributes one is the *inverse* of the other. For example, if the association between STUDENTS and COURSES in figure 4 was to be modeled as many-to-many, a member attribute "course-enrolled-in" would be added to the class STUDENTS defined in figure 1 and declared as *multivalued*. Together with the multivalued attribute "students-enrolled" of COURSES, this attribute models the many-to-many association.

Although it was claimed that fragments in IFO make "sharp distinction" between vertices serving the role of domain and vertices serving the role of range, which is not done in FDM, the discussion in [Shipman 81] shows that the distinction is also made in FDM.

Also, it was claimed that *nested functions* (figure 5) is a "marked difference between the FDM and the IFO model". In fact, the same fragment in figure 5 was directly modeled in FDM [Shipman 81] using a multi-argument nested function as follows:

DECLARE GRADE (STUDENT,COURSE(STUDENT))===>STRING

[2] Also, in the example shown in figure 6 in the article, CAR was considered an abstract object and VEHICLE as a free object although, logically, CARs represent a subset of VEHICLEs.

2.3. ISA Relationships

The IFO model has two types of ISA relationship. These are *specialization* (represented by a broad arrow) and *generalization* (represented by a shaded arrow). Specialization is used to model different roles played by objects in an object type. In figure 6, EMPLOYEE and STUDENT are two different roles played by PERSONs. Also, TA is a role played by either a STUDENT or an EMPLOYEE. All functions defined for the supertype as well as its type are inherited downwards by the subtype. This notion of specialization is the same as *specialization hierarchy* in the E-R model [Teorey 86]. It corresponds to the *subclass connection* in SDM. However, the IFO model does not specify criteria for forming subtypes while in SDM, subclasses can be *attribute-defined*, *user-controllable*, *existence*, or *intersection*, *union*, or *difference* of two other subclasses. Moreover, the IFO model did not specify inheritance rules in case of an object type is a subtype of more than one supertype. This notion of multiple inheritance is handled in other work [Banerjee 87].

Generalization, on the other hand, is used to model a situation when distinct preexisting types are combined to form new virtual types. As suggested by Abiteboul, there are three differences between generalization and specialization in IFO.

1. Types forming the generalized type can be specified to be *disjoint* by attaching the label "disjoint" to the virtual type. Subtypes in specialization normally overlap. A more complete form of specialization (subtyping) is supported by the Binary Relationship Model [Nijssen 77, Mark 83] and the NIAM model [Verheyen 82].

2. The virtual type in a generalization is covered by its subtypes while the super type in specialization is not unless it was labeled as "covers".

3. The inheritance of the object type in generalization is from the subtypes to the generalized type while in the case of specialization, the inheritance is from the supertype to subtypes.

It should be noted that "covers" and "disjoint" are the only two constraints mentioned in [Abiteboul 87]. Besides, they are defined only informally and not part of the formal definition of the model. Also, it appears from the above discussion that the distinction between specialization and generalization in IFO, which is claimed to "provide a rich inheritance framework" is rather artificial. It is semanticlly incorrect for functions and object type of the subtype to be inherited by the generalized type. In the example provided in the article, this would mean that the generalized type VEHICLE inherits its type from CAR and MOTOR-BOAT.

In [Teorey 86], generalization represents a case where an entity set (called the generic entity set) is partitioned based on the value of an attribute. The subsets are disjoint and their union makes up the generic entity set. Specialization, on the other hand, represents a case where subsets can overlap and their union need not make up the generic entity set. Furthermore, subsetting is not based on attribute values.

In fact, there is no need to distinguish between generalization and specialization. Only one construct is needed, with constraints such as *covering* and *exclusivity* properly defined. This philosophy has been adopted in the design of UDM [Hanna 89] through *supertype/subtype relationships* with a rich set of constraint. Thus, the two concepts of generalization and specialization are treated uniformly through one construct.

3. What is Missing From the IFO Model

As shown in section 2, constructs in the IFO model can directly map to constructs in SDM. This demonstrates that the IFO model is actually a subset of the SDM, one of the models that were claimed to be subsumed by the IFO model. This section demonstrates that the IFO model even fails to support concepts that are essential to those models that were claimed to be subsumed by the IFO model.

3.1. SDM Concepts Missing From the IFO Model

The following are SDM concepts that are not supported by the IFO model:

(1) Class attributes which applies to a class as a whole. e.g. lowest, highest and average scores defined as a class attribute for the class STUDENTS.

(2) One-to-many associations are not directly representable in IFO.

(3) Many-to-many associations are not supported in IFO.

(4) No facility is provided to uniquely identify objects within object types.

(5) In SDM, a class can be specified not to contain duplicates. The default is that duplicates are allowed unless a unique identifier was specified. This facility is not supported by IFO.

(6) The definition of subclass connections in SDM is more elaborate and its semantics are more well defined than those of *specialization* in IFO. Specialization in IFO has a very vague definition, that is an object type can be a subclass of another object type. No criteria can be specified for subtyping. In contrast, a subclass in SDM can be formed as attribute-defined, user-controllable, intersection, union, or difference of other two subclass or as a result of association with other classes.

(7) The definition of *grouping connection* in SDM is more elaborate than that in IFO. No criteria can be specified to form groupings in IFO. Groupings in SDM can be expression-defined, enumeratable, or user controllable.

(8) In IFO, all names (printable objects) are of type STRING. In SDM, names can be specifies to have certain format or to belong to other name classes.

(9) SDM provides for specifying *value classes* which represent the set of permissible values for attributes. IFO does not provide such facility.

(10) It is possible in SDM to place constraints on the size of multivalued attributes, for example, by specifying the upper and lower bounds.

(11) An attribute in SDM can be specified not to have null value.

(12) An attribute in SDM can be specified to be *not changeable* meaning that once it is assigned a nonnull value it can not be changed except to correct an error.

(13) An attribute in SDM can be specified to be *exhaustive* of its value class.

(14) A multivalued attribute in SDM can be specified to be *nonoverlapping*.

3.2. Entity-Relationship Concepts Missing From the IFO Model

The following concepts are supported by the ER model [Chen 76, Teorey 86] but are not supported by the IFO model:

(1) N-ary associations between object types are not supported by the IFO model. Associations in IFO are strictly unidirectional binary.

(2) Many-to-many associations are not supported by IFO.

(3) Associations in IFO can not have attributes attached to them.

(4) No constraints can be specified for associations in IFO. An example of constraints in the E-R model is specifying the membership of an entity in a relationship to be either optional or mandatory.

(5) No facility is provided by IFO to uniquely identify objects in the database. This feature is provided by almost every data model.

4. Conclusion

It should be noted that only the structural aspects of data modeling are supported by the IFO model. No facilities for specifying constraints on structures are provided. For example, specifying a mapping between objects (functions in IFO) to be one-to-one or one-to-many, many-to-many, mandatory or optional is not part of the IFO model. The structural aspects of data modeling should not be separated from the set of integrity rules and constraints on data structures. Defining an object type to be a subtype of another object type does not make much sense without defining a set of rules to control this subtyping relationship.

It has been demonstrated that the IFO model is indeed a subset of the SDM that was claimed to be subsumed by the IFO model. Furthermore, the IFO model failed to support constructs that are essential to both SDM and the ER model, two models that the were claimed to be subsumed by the IFO model.

References

Abiteboul, Serge and Hull, Richard. "IFO: A Formal Semantic Database Model" *ACM Transactions on Database Systems* (12:4), 1987 December, pages 525-565.

Banerjee, J., Kim, W., Kim, H.J. and Korth, H.F. "Semantics and Implementation of Schema Evolution in Object-Oriented Databases." *Proceedings SIGMOD*, U. Dayal and I. Traiger, editors, 1987 May, pages 311-322.

Brodie, M. L. and Ridjanovic, D. "On the Design and Specification of Database Transactions," in *On Conceptual Modeling*, Brodie, Mylopoulus, and Schmidt, editors. Springer-Verlag, New York, 1984, 277-306.

Chen, Peter Pin-Shan. "The Entity-Relationship Model -- Toward a Unified View of Data." *ACM Transactions on Database Systems* (1:1), 1976 March, pages 9-36.

Codd, E. F. "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM* (13:6), 1970 June, pages 377-387.

Hammer, M. M. and McLeod, D. J. "Database Description with SDM: A Semantic Database Model." *ACM Transactions on Database Systems* (6:3), 1981 September, pages 351-386.

Hanna, Magdy S. "UDM: A Unified Data Modeling Approach for Logical Database Design." Doctoral dissertation, University of Minnesota, Department of Computer Science, 1989 December.

Kershberg, L. and Pacheco J.E.S. "A Functional Database Model." Technical Report, Pontificia Universidade Catolica do Rio de Janeiro, Brazil, 1976 February.

King, R. and McLeod, D. "A Database Design Methodology and Tool for Information Systems." *ACM transactions on Office Inf. Syst.* (3:1), 1985, pages 2-21.

Mark, Leo. "What is the Binary Relationship Approach." *Entity-Relationship Approach to Software Engineering*, C. G. Davis, S. Jajodia, P. A. Ng and R. T. Yeh, editors, Elsevier Science Publishers B.V. (North-Holland), 1983.

Nijssen, G. M. "Current Issues in Conceptual Schema Concepts." *Architecture and Models in Data Base Systems*, G. M. Nijssen, editor, North-Holland, Amsterdam, 1977, pages 31-65.

Shipman, D. "The Functional Data Model and the Data Language DAPLEX." *ACM Transactions on Database Systems* (6:1), 1981 March, pages 140-173.

Smith, John M. and Smith, Diane C. P. "Database Abstraction: Aggregation and Generalization." *ACM Transactions on Database Systems* (2:2), 1977, pages 105-133.

Teorey, Toby J., Yang, Dongqing, and Fry, James P. "A Logical Design Methodology for Relational Databases Using the Extended Entity-relationship Model." *ACM Computing Surveys* (18:2), 1986 June, pages 197-222.

Verheyen, G. M. A. and Van Bekkum, J. "NIAM: An Information Analysis Method," in *Information System Design Methodology 82*, jointly published by Nijssen Adviesbureau voor Informatica B.V. and Control Data, Rijswijk, The Netherlands, 1982 December, pages 1-53.

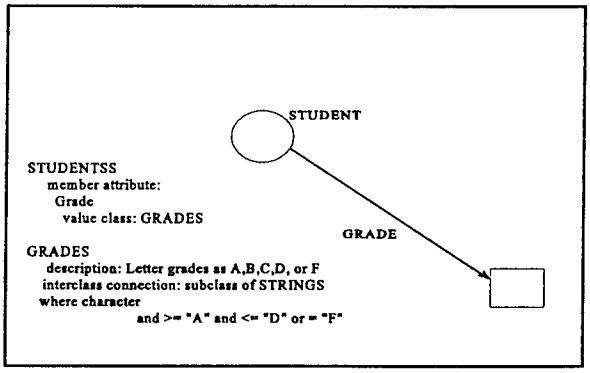


Figure 1: One-to-one Association

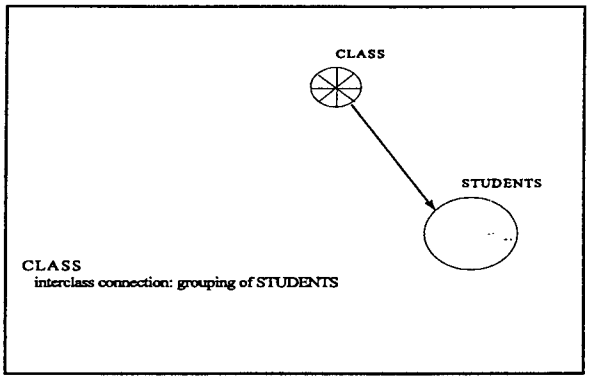
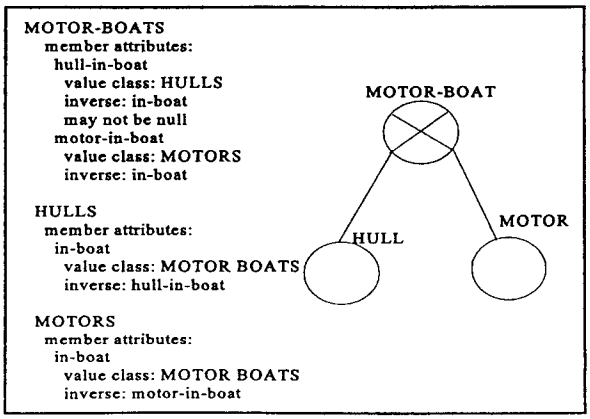


Figure 2: Grouping



Fugure 3: Cartesian Product

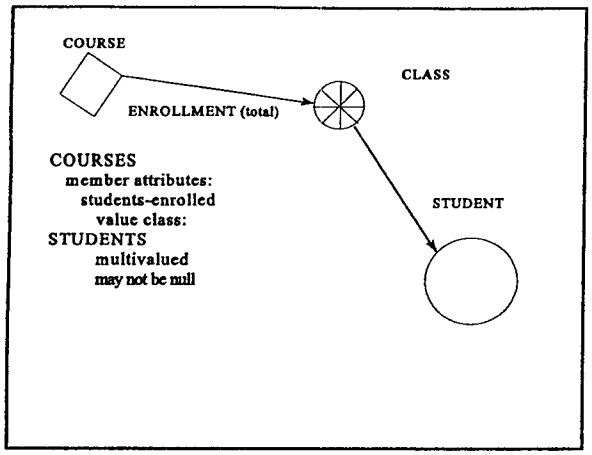


Figure 4: One-to-many Associations

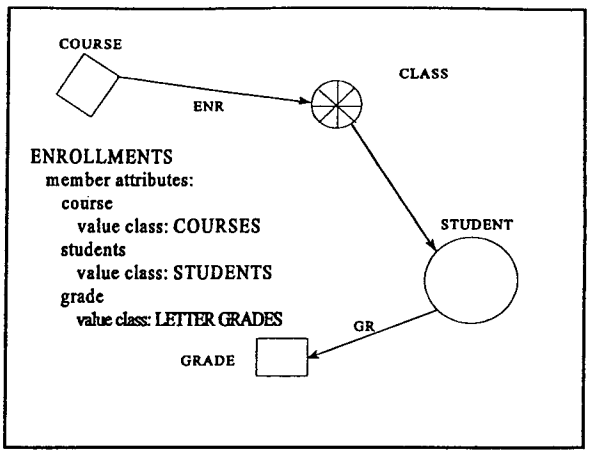


Figure 5: Nested Functions

PERSONS
member attributes:
idnum
value class: subclass of STRINGS

STUDENTS

interclass connection: subclass of PERSONS where specified
member attributes:
major
value class: DEPARTMENTS

EMPLOYEES
interclass connection: subclass of PERSONS where specified
member attributes:
supervisor
value class: SUPERVISORS

DEPARTMENTS
description: might have more connections and attributes

SUPERVISORS
interclass connections: subclass of EMPLOYEES where specified

TA
interclass connections: subclass of PERSONS where specified
and (is in STUDENTS or in EMPLOYEES)
member attributes
ta-for
value class: COURSES

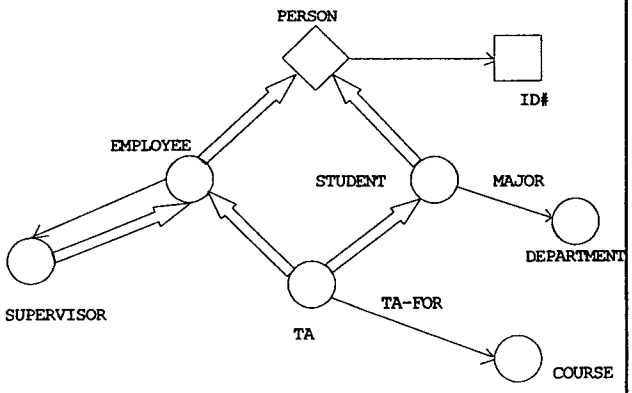


Figure 6: Specialization