

A Primer on Object Role Modeling

Stanley D. Blum,
Museum of Vertebrate Zoology
University of California, Berkeley

April 14, 1995

For the last 15 to 20 years, information system developers have found it useful to recognize three kinds of data models or schemas (schemata):

- External schema (user view) -- screens, forms, reports, etc. that make sense to a user;
- Logical schema -- a reconciled combination of user views, in which every data concept in a user view is present (or derivable from more elementary facts) and redundancy is minimized; and
- Physical schema -- the structure of the implemented database.

The originators and proponents of object role modeling¹ assert that another layer, the conceptual schema, should be inserted between the external and logical schemas. The basic notions are that 1) the relationships among data elements can be described according to a relatively simple set of criteria, and 2) the resulting descriptions then determine, in a mathematical sense, a fully normalized logical schema. In other words, a relatively simple algorithm can translate a correctly described conceptual model into a 5th normal form logical model. Full normalization of a logical model not only ensures that it is correctly designed in a mathematical sense, but also brings practical benefits to database development and management.

Object role modeling is a methodology for developing conceptual models. This distinguishes it from the more commonly used entity-relationship (ER) methods, which produce logical models. ER methods do not incorporate the rules and analysis of normalization; the modeler places the output of a normalization analysis (logical table structures) directly into the logical model. In object role modeling, data elements are not combined into tables *a priori*. Instead, descriptions of data element relationships serve as input to a table-building algorithm, and normalization is thereby incorporated into the methodology.

The building blocks of object role modeling are data objects (a data element or a very narrowly defined set of elements) and their interrelationships. Data objects are represented in diagrams by ellipses, and their interrelationships by lines and subdivided boxes that establish connections between objects.

Objects

Object role modeling recognizes four basic kinds of data objects: simple, value, composite, and nested. A **simple object** is one in which real world instances are designated -- uniquely identified -- by a single data element; i.e., a single data element comprises the primary key. Figure 1 shows how people would be represented in an object role model when real-world people are to be designated in the database by an identifying number, *Person_id*, rather than by their names. Here, *Person* is an example of a simple object.

¹ See references at the end of this document.

Object Role Modeling

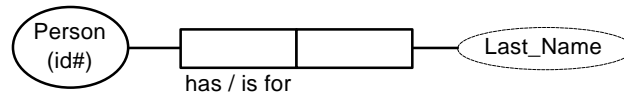


Figure 1. Simple and Value Objects

A **value object** is something like a name (label), number, or date, which represents a simple scalar attribute. Last_Name, in Figure 1, is a value object. A common example used to clarify the nature of value objects concerns the distinction between a person and his or her name². While it is possible to walk up to a person with the last name “Jones”, it is not possible to walk up to the last name “Jones”. “Jones” is just a label. Value objects are represented on model diagrams as dashed ellipses.

A **composite object** is one in which the instances or real-world cases are designated (unambiguously identified) by two or more relationships to other data objects, with appropriate annotation on the relationships, as shown below. (The annotations on the relationships are explained later.) A composite object is analogous to an entity in ER modeling in which the primary key is composed of two or more attributes.

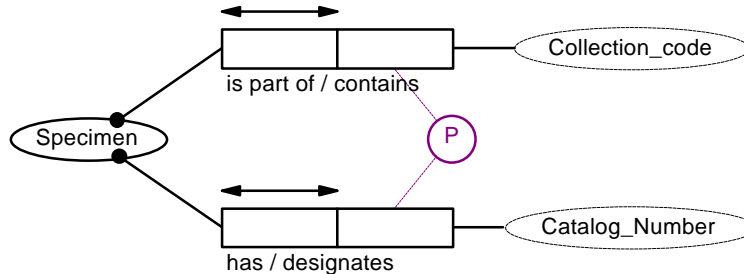


Figure 2. A Composite Object

The example above would be appropriate in an organization that holds more than one collection, each with an independent series of catalog numbers; i.e., within the entire organization, both Collection_code plus Catalog_Number are required to uniquely identify a specimen.

An object whose existence and identity stems from the relationship between two other objects is called a **nested object**. A “circle” (a rectangle with rounded corners) is drawn around the relationship to “objectify” it. A nested object can then participate in relationships with other objects. The example below shows that a Specimen can be cited in many References, a Reference may cite many Specimens, and that a Citation (the discussion of a particular Specimen in a particular Reference) begins on a particular Page_Number. A double ellipse around an object, e.g., the Specimen object below, indicates that it occurs more than once in the model -- either in the same or a different diagram.

² “A Guide to FORML”, Asymetrix Corporation, 1994.

Object Role Modeling

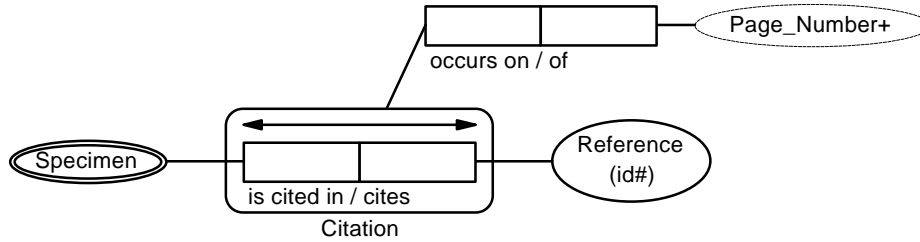


Figure 3. A Nested Object

Relationships and Constraints

Object role modeling has a rich language to describe relationships among data objects. The whole notion of relationship, however, is slightly different from that used in ER modeling. In ER modeling, one thinks of instances in one entity being related to instances in another entity. In object role modeling, an object contributes instances to a role in a relationship. A relationship between objects is formed by juxtaposing roles from those objects, a subset of valid instances. In all of the examples above, the relationships are binary (involve two roles), but the language supports descriptions of relationships that are unary, binary, ternary, and quaternary; i.e., that involve one, two, three, or four roles. Quaternary relationships, however, are exceedingly rare. In virtually every case where a quaternary relationship seems appropriate, the same information can be handled by two or more binary or ternary relationships.

Role boxes can be thought of as columns that hold examples of the objects participating in a relationship, i.e., the data values that designate object instances. The methodology does not require that roles be populated with examples, but it does require that relationships be characterized in ways that are most clearly seen when the roles are populated.

Uniqueness constraints. The most basic characteristic of a relationship is its set of “uniqueness constraints”, which embody the real-world combination rules for the relationship. For example, the following assertions could be taken as combination rules for the Person--Last_Name relationship:

- A Person can have only one Last_Name.
- A Last_Name can be used by more than one Person.

The following table is sufficient to demonstrate these rules.

OBJECT	Person	Last_Name
KEY	Person_id	Last_Name
EXAMPLES	1	Smith
	2	Jones
	3	Smith

Values in the Person_id column are unique, while values in the Last_Name column can be repeated. A relationship’s uniqueness constraints are indicated with one or more double headed arrows over each role, and/or combination of roles, that must be unique in a valid population. The uniqueness constraints for the Person--Last_Name relationship are shown in Figure 4.

Object Role Modeling

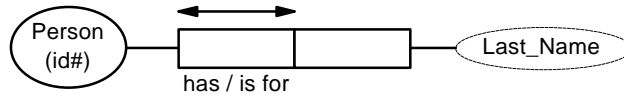


Figure 4. Uniqueness Constraint on a Single Role

The Specimen--Reference relationship above (Figure 3) has the following combination rules:

- A Specimen can be cited by many references.
- A Reference can cite many Specimens.

An example population for this relationship is shown in the table below. Note that the data columns,

OBJECT	Specimen	Specimen	Reference
KEY	Collection_cod e	Catalog_Number	Reference_id
EXAMPLES	Mam	123	1
	Mam	123	2
	Bird	345	2

Collection_code and Catalog_Number, are both required to identify a Specimen because it was defined as a composite object in Figure 2.

In this relationship, a single Specimen can occur in more than one row, a single Reference can occur in more than one row, but the combination of Specimen and Reference can occur only once -- it must be unique. The uniqueness constraint for this relationship therefore spans both the Specimen and Reference roles.

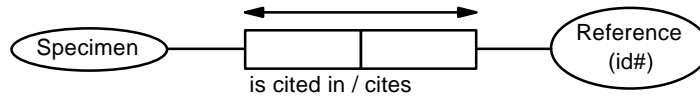


Figure 5. Uniqueness Constraint Spanning Two Roles

Mandatory roles. Another kind of rule that can be specified about a relationship is whether or not every object instance must participate in the relationship. For example, the assertion could be made that “every Person must have a Last_Name”. In other words, every instance of the Person object must participate in the Person--Last_Name relationship. A solid black dot on the object margin, where it joins to a relationship, indicates that every instances of the object must participate in the relationship (populate the role) to be valid.

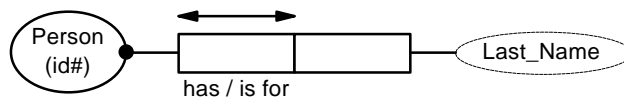


Figure 6. A Mandatory Role

Interpredicate Constraints I: external uniqueness constraints. Interpredicate constraints concern roles in two or more relationships. They are useful for expressing a wide variety of “business rules” in a model. Figure 2 illustrates an external uniqueness constraint in which the roles (values) of two objects

Object Role Modeling

are used to designate or identify the instances of a third. Two kinds of relationship annotations are required to establish a consistent identification scheme. The object being identified must be a mandatory participant in its relationships to the identifying objects, and an interpredicate constraint specifies that the combinations of values donated by the identifying objects must be unique. In Figure 2, every Specimen must have both a Collection_code and Catalog_Number to be valid. In addition, every combination of Collection_code and Catalog_Number, as applied to actual specimens, must be unique. A “P” in a circle, connected to the roles played by Collection_code and Catalog_Number, indicates that these two objects form the primary key of the Specimen object.

A similar annotation, but with a “U” in the circle instead of a “P”, specifies that combinations of the indicated roles must be unique, but that the combinations are not used as the primary key of the common object. For example, if the population of people to be stored in the database were small, it might be reasonable to create a unique index on First_Name and Last_Name. On the other hand, if the population of people were large, two people with the same first and last names might be an eventuality, and a unique index would be overly restrictive. In this case an “I” could be placed in the circle to indicate that a non-unique index (duplicates allowed) is to be created for the combined values of these roles.

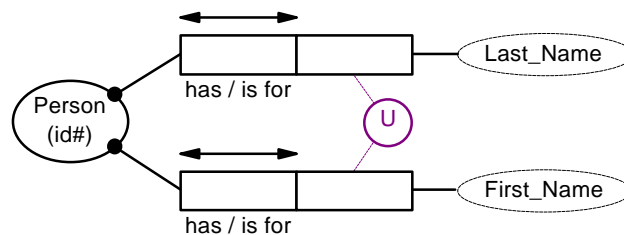


Figure 7. An Interpredicate Uniqueness Constraint

Interpredicate Constraints II: participation constraints. It is also possible to specify exclusion, equality and subset constraints between relationships that involve the same object. All of the definitions that follow concern cases where the same object participates in two or more relationships. An exclusion constraint is used to specify that an instance of the object can participate in only one role at a time (not two or more roles simultaneously). An equality constraint is used to specify that if an instance occurs in one role it must also occur in the other(s). A subset constraint is used to specify that an instance must participate in one relationship in order to participate in the other, but not vice versa. The following hypothetical rules about shipments, shipping methods, carriers, people and payment methods illustrate these different constraints (Figure 8).

Exclusion: A Shipment can either be shipped via a commercial Carrier, or hand carried by a Person (not both). A circled “X” with dashed lines connecting roles indicates exclusion; e.g.: if Shipment #4 is sent with UPS, Shipment #4 cannot also be hand carried; if Shipment #5 is hand carried by Person #99, Shipment #5 cannot also be shipped with UPS.)

Equality: For every Shipment sent with a commercial Carrier, a Shipping_Method must be specified. A bi-directional dashed arrow between roles indicates the equality constraint; e.g., if Shipment #4 is sent with UPS, then a Shipping_Method must also be recorded.

Subset: Only shipments that are sent with a commercial Carrier can be sent COD, but not every Shipment sent with a commercial Carrier is sent COD. A uni-directional

Object Role Modeling

dashed arrow points from the subset role to the superset role; e.g., for Shipment #4 to have been sent COD, it must have been sent with a commercial Carrier.

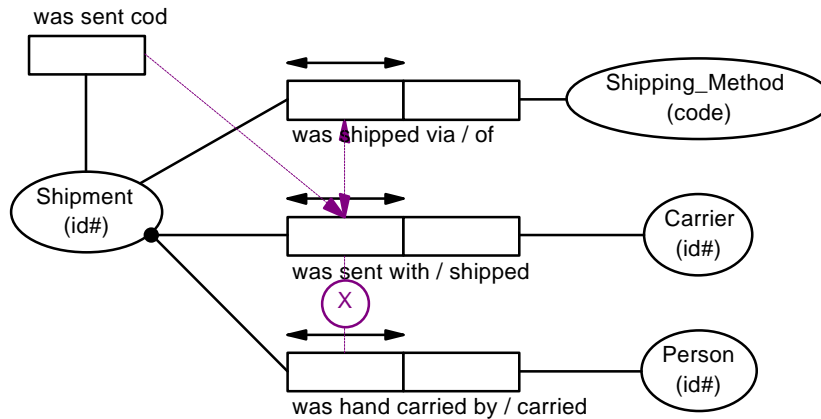


Figure 8. Subset, Equality, and Exclusion Constraints

Note, the “was sent COD” role represents a unary relationship. This is one way to represent a logical (yes/no) attribute. The role-box is not intended to contain the values “yes” or “no”, but rather a subset of object instances. “Yes” or “no” for a given instance is indicated by the presence of its identifying value in the role.

Note also that the relationships “Shipment was sent with Carrier” and “Shipment was hand carried by Person” both connect to the same mandatory dot on the Shipment object. The shared connection indicates that every Shipment must participate in at least one of the relationships.

Miscellaneous constraints. Both roles in a relationship can be played by the same object. This kind of structure is typically used to represent things like org-charts, part assemblies, genealogies, etc.; i.e., a hierarchy or network. Four kinds of constraints apply only to these “ring” relationships, and are thus termed ring constraints. They specify which instances can be related to each other.

Intransitive: if (A, B) and (B, C); then not (A, C)

Irreflexive: not (A, A)

Asymmetric: if (A, B); then not (B, A)

Acyclic: if (A, B) and (B, C); then not (C, A). *Note the order of excluded relationship-case here (C, A), is different from that in the intransitive constraint.*

The appropriate ring constraint to apply in a simple genealogy is the acyclic constraint. This establishes a consistent polarity or direction (ancestor / descendant) throughout the hierarchy.

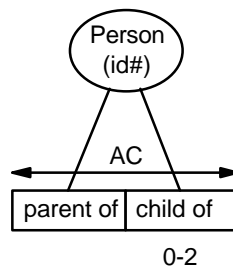


Figure 9. Ring and Frequency Constrains

Frequency constraints. In Figure 9, the sex-related roles of the parents (mother versus father) are not distinguished, and thus a Person can have more than one parent (as opposed to only one father and one mother). “More than one parent”, however, implies many parents, which does not rule out three or four. “More than one” is not a sufficient depiction of reality or what should be allowed in the database. A frequency constraint of “0-2” is therefore placed on the child role to indicate that a person may have at most two parents. In other words, a Person can appear in the child role a maximum of two times. The minimum value of 0 indicates that the tree is finite. If the Person object is not empty, then at any given moment it must contain at least one person who is not the child of anyone.)

Frequency constraints may be applied to any role, not just those that are part of a recursive (ring) relationship.

Subtypes

It is common in to encounter situations in which two or more objects participate in similar or identical roles with another set of objects, but each also participates in its own set of relationships. In entity-relationship terminology, the situation would be described as two or more entities sharing attributes (fields), but each also possessing unique attributes. These are cases in which initial inspection leads the modeler to recognize two or more objects and then wonder whether in fact they might represent the “same thing”. There are also cases in which initial inspection leads the modeler to recognize a broadly defined object and then wonder whether the single object should be split into distinct objects. In either case, it may be appropriate to create a supertype-subtype hierarchy the “ambiguous” objects. If a model were to incorporate detailed descriptions of zoological specimens, for example, the following three facts might hold: every Specimen was collected from a Locality; only Arthropod_Specimens have an Antennae_Segment_count; and only Vertebrate_Specimens have a Vertebral_count.

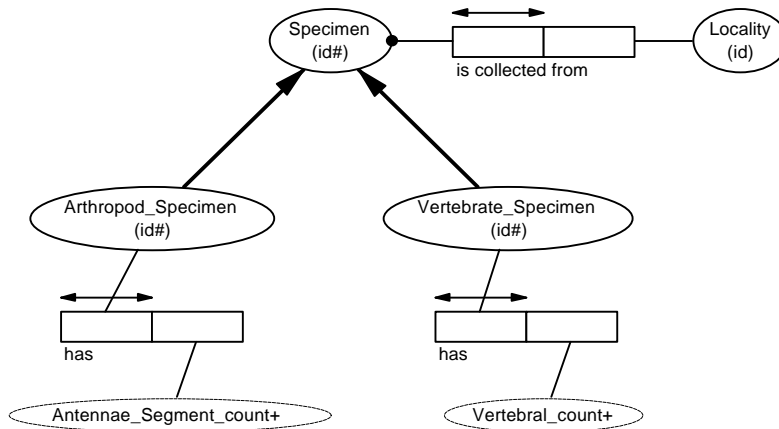


Figure 10. A Subtype Hierarchy

Object role modeling supports subtyping as follows. A solid arrow is drawn from the subtype to the supertype. The super- and subtypes must be identified by a compatible data structure (e.g., an integer, or a string of the same length). Subtypes “inherit” all relationships from their supertypes (i.e., the capability to participate in those relationships). Disinheritance of a supertype’s relationships is not supported. A subtype can have more than one supertype, but the hierarchy must have a single root supertype.

Object Role Modeling

References

More detailed expositions of object role modeling can be found in:

Nijssen, G. M. and T. A. Halpin. 1989.

Conceptual Schema and Relational Database Design: A Fact Oriented Approach. xiv + 342 pp. Prentice Hall of Australia Pty Ltd. Sydney. ISBN 0-13-167263-0. (*Currently out of print, but new edition forthcoming.*)

Asymetrix Corporation. 1994.

InfoModeler: A Guide to FORML. xiv + 144 pp. Asymetrix Corporation, 110-110th Avenue NE, Suite 700, Bellevue, WA 98004. (*This reference is included as documentation of the software product "InfoModeler".*)