TITLE OF ENTRY: Object-Role Modeling

BYLINE
Terry Halpin
Neumont University, USA.
www.orm.net

SYNONYMS
Fact-oriented modeling. NIAM.

DEFINITION
Object-Role Modeling (ORM), also known as *fact-oriented modeling*, is a conceptual approach to modeling and querying the information semantics of business domains in terms of the underlying facts of interest, where all facts and rules may be verbalized in language readily understood by non-technical users of those business domains. Unlike Entity-Relationship (ER) modeling and Unified Modeling Language (UML) class diagrams, ORM treats all facts as relationships (unary, binary, ternary etc.). How facts are grouped into structures (e.g. attribute-based entity types, classes, relation schemes, XML schemas) is considered a design level, implementation issue that is irrelevant to the capturing of essential business semantics.

Avoiding attributes in the base model enhances semantic stability, populatability, and natural verbalization, facilitating communication with all stakeholders. For information modeling, fact-oriented graphical notations are typically far more expressive than those provided by other notations. Fact-oriented textual languages are based on formal subsets of native languages, so are easier to understand by business people than technical languages like UML's Object Constraint Language (OCL). Fact-oriented modeling includes procedures for mapping to attribute-based structures, so may also be used to front-end other approaches.

The fact-oriented modeling approach comprises a family of closely related "dialects", known variously as Object-Role Modeling (ORM), Natural Language Information Analysis Method (NIAM), and Fully-Communication Oriented Information Modeling (FCO-IM). While not adopting the ORM graphical notation, the Object-oriented Systems Model (OSM) [4] and the Semantics of Business Vocabulary and business Rules (SBVR) [14] initiative within the Object Management Group (OMG) are close relatives, with their attribute-free philosophy.

HISTORICAL BACKGROUND

In 1973, Falkenberg generalized work by Abrial and Senko on binary relationships to *n*-ary relationships, and excluded attributes at the conceptual level to avoid "fuzzy" distinctions and to simplify schema evolution. Later, Falkenberg proposed the fundamental ORM framework, which he called the "object-role model" [5]. This framework allowed *n*-ary and nested relationships, but depicted roles with arrowed lines. Nijssen adapted this framework by introducing a circle-box notation for objects and roles, and adding a linguistic orientation and design procedure to provide a modeling method called ENALIM (Evolving NAtural Language Information Model) [13]. Nijssen's team of researchers at Control Data in Belgium developed the method further, including van Assche who classified object types into lexical object types (LOTs) and non-lexical object types (NOLOTs). Today, LOTs are commonly called "entity types" and NOLOTs are called "value types". Meersman added subtyping to the approach, and made major contributions to the RIDL query language [12] with Falkenberg and Nijssen. The method was renamed "aN Information Analysis Method" (NIAM). Later, the acronym "NIAM" was given different expansions, and is now known as "Natural language Information Analysis Method".

In the 1980s, Nijssen and Falkenberg worked on the design procedure and moved to the University of Queensland, where the method was further enhanced by Halpin, who provided the first full formalization, including schema equivalence proofs, and made several refinements and extensions. In 1989, Halpin and Nijssen co-authored a book on the approach, followed a year later by Wintraecken's book [16]. Today several books, including major works by Halpin [6], and Bakema, Zwart and van der Lek [1] expound on the approach.

Many researchers contributed to the fact-oriented approach over the years, and there is no space here to list them all. Today various versions exist, but all adhere to the fundamental object-role framework. Habrias developed an object-oriented version called MOON (Normalized Object-Oriented Method). The Predicator Set Model (PSM), developed mainly by ter Hofstede, Proper and van der Weide [9], includes complex object constructors. De Troyer and Meersman developed a version with constructors called Natural Object-Relationship Model (NORM). Halpin developed an extended version simply called ORM, and with Bloesch and others developed an associated query language called ConQuer [2]. Bakema, Zwart, and Van der Lek [1] recast all entity types as nested relationships, to produce Fully Communication Oriented NIAM, which they later modified to Fully Communication Oriented Information Modeling (FCO-IM).

More recently, Meersman and others adapted ORM for ontology modeling, using a framework called DOGMA (Developing Ontology-Grounded Methodology and Applications) (http://www.starlab.vub.ac.be/website/). Nijssen and others extended NIAM to a version called NIAM2007. Halpin and others developed a second generation ORM (ORM 2), whose graphical notation is used in this article.

SCIENTIFIC FUNDAMENTALS

ORM includes graphical and textual *languages* for modeling and querying information at the conceptual level, as well as *procedures* for designing conceptual models, transforming between different conceptual representations, forward engineering ORM schemas to implementation schemas (e.g. relational database schemas, object-oriented schemas, XML schemas, and external schemas) and reverse engineering implementation schemas to ORM schemas.

Attributes are not used as a base construct. Instead, all fact structures are expressed as *fact types* (relationship types). These may be unary (e.g. Person smokes), binary (e.g. Person was born on Date), ternary (e.g. Person visited Country in Year), and so on. This attribute-free nature has several advantages: *semantic stability* (minimize the impact of change caused by the need to record something about an attribute); *natural verbalization* (all facts and rules may be easily verbalized in sentences understandable to the domain expert); *populatability* (sample fact populations may be conveniently provided in fact tables); *null avoidance* (no nulls occur in populations of base fact types, which must be elementary or existential). Although attribute-free diagrams typically consume more space, this apparent disadvantage is easily overcome by using an ORM tool to automatically create attribute-based structures (e.g. ER, UML class, or relational schemas) as views of an ORM schema.

ORM's graphical language is far more expressive for data modeling purposes than that of UML or industrial versions of ER, as illustrated later. The *rich graphical notation* makes it easier to detect and express constraints, and to visually transform schemas into equivalent alternatives.

ORM includes *effective modeling procedures* for constructing and validating models. In step 1a of the Conceptual Schema Design Procedure (CSDP), the domain expert informally verbalizes facts of interest. In step 1b, the modeler formally rephrases the facts in natural yet unambiguous language, using standard reference patterns to ensure that entities are well identified. Verbalized fact instances are abstracted to fact types, which are then populated with sample instances. The constraints on the fact types are verbalized formally, a process that may be automated [8], and these verbalizations are checked with the domain expert, using positive populations to illustrate satisfaction of the constraints as well as counterexamples to illustrate what it means to violate a constraint. This approach to model *validation by verbalization and population* has proved extremely effective in industrial practice, with correct models typically obtained from the outset rather than going through unreliable iterative procedures.

Figure 1 lists the main graphical symbols in the ORM 2 notation [8], numbered for easy reference. An *entity type* (e.g. Person) is depicted as a named, soft rectangle (symbol 1), or alternatively an ellipse or hard rectangle. *Value type* (e.g. PersonName) shapes have dashed lines (symbol 2). Each entity type has a *reference scheme*, indicating how each instance may be mapped via predicates to a combination of one or more values. Injective (1:1 into) reference schemes mapping entities (e.g. countries) to single values (e.g. country codes) may be abbreviated as in symbol 3 by displaying the *reference mode* in parentheses, e.g. Country(.code). The reference mode indicates how values relate to the entities. Values are constants with a known denotation, so require no reference scheme.

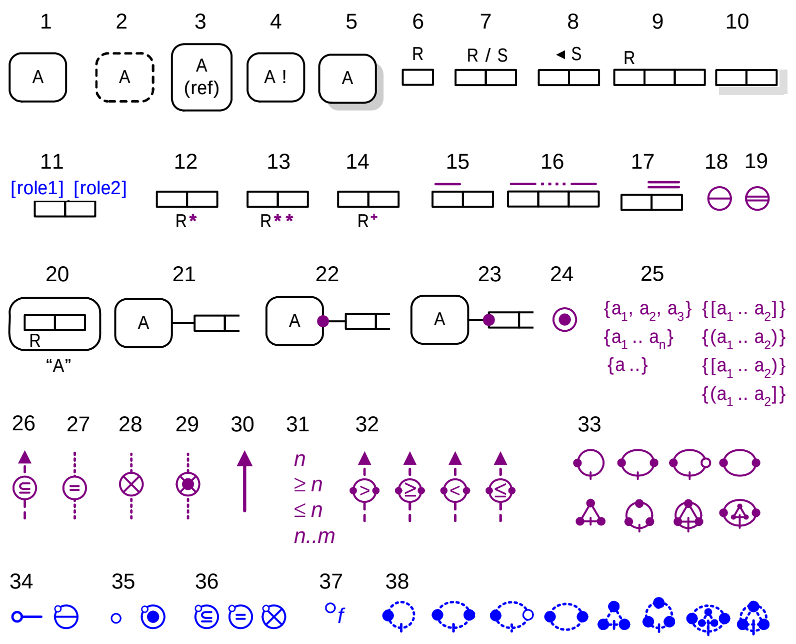


Fig. 1. Main ORM graphic symbols

Relationships used for *preferred reference* are called *existential facts* (e.g. there exists a country that has the country code ‘US’). The other relationships are *elementary facts* (e.g. The country with country code ‘US’ has a population of 301 000 000). The exclamation mark in symbol 4 declares that an object type is *independent* (instances may exist without participating in any elementary facts). Object types displayed in multiple places are shadowed (symbol 5).

A fact type results from applying a logical *predicate* to a sequence of one or more object types. Each predicate comprises a named sequence of one or more *roles* (parts played in the relationship). A predicate is sentence with object holes, one for each role, with each role depicted as a box and played by exactly one object type. Symbol 6 shows a unary predicate (e.g. … smokes), symbols 7 and 8 depict binary predicates (e.g. … loves …), and symbol 9 shows a ternary predicate. Predicates of higher *arity* (number of roles) are allowed. Each predicate has at least one *predicate reading*. ORM uses *mixfix* predicates, so objects may be placed at any position in the predicate (e.g. the fact type Person introduced Person to Person involves the predicate “… introduced … to …”). Mixfix predicates allow natural verbalization of *n*-ary relationships, as well as binary relationships where the verb is not in the infix position (e.g. in Japanese, verbs come at the end). By default, forward readings traverse the predicate from left to right (if displayed horizontally) or top to bottom (if displayed vertically). Other reading directions may be indicated by an arrow-tip (symbol 8). For binary predicates, forward and inverse readings may be separated by a slash (symbol 7). Duplicate predicate shapes are shadowed (symbol 10).

Roles may be given *role names*, displayed in square brackets (symbol 11). An asterisk indicates that the fact type is *derived* from one or more other fact types (symbol 12). If the fact type is derived and stored, a double asterisk is used (symbol 13). Fact types that are semi-derived are marked “+” (symbol 14). *Internal uniqueness constraints*, depicted as bars over one or more roles in a predicate, declare that instances for that role (combination) in the fact type population must be unique (e.g. symbols 15, 16). For example, a uniqueness constraint on the first role of Person was born in Country verbalizes as: Each person was born in at most one Country. If the constrained roles are not contiguous, a dotted line separates the constrained roles (symbol 16). A predicate may have many uniqueness constraints, at most one of which may be declared *preferred* by a double-bar (symbol 17). An *external uniqueness constraint* shown as a circled uniqueness bar (symbol 18) may be applied to two or more roles from different predicates by connecting to them with dotted lines. This indicates that instances of the role combination in the join of those predicates are unique. For example, if a state is identified by combining its state code and country, an external uniqueness constraint is added to the roles played by Statecode and Country in: State has Statecode; State is in Country. Preferred external uniqueness constraints are depicted by a circled double-bar (symbol 19).

To talk about a relationship, one may *objectify* it (i.e. make an object out of it) so that it can play roles. Graphically, the objectified predicate (a.k.a. *nested* predicate) is enclosed in a soft rectangle, with its name in quotes (symbol 20). Roles are connected to their players by a line segment (symbol 21). A *mandatory role constraint* declares that every instance in the population of the role's object type must play that role. This is shown as a large dot placed at the object type end (symbol 22) or the role end (symbol 23). An *inclusive-or* (*disjunctive mandatory)* constraint applied to two or more roles indicates that all instances of the object type population must play at least one of those roles. This is shown by connecting the roles by dotted lines to a circled dot (symbol 24).

To restrict the population of an object type or role, the relevant values may be listed in braces (symbol 25). An ordered range may be declared separating end values by "..". For continuous ranges, a square/ round bracket indicates an end value is included/excluded. For example, "(0..10]" denotes the positive real numbers up to 10. These constraints are called *value constraints*.

Symbols 26-28 denote *set comparison constraints*, which apply only between compatible role sequences. A dotted arrow with a circled subset symbol depicts a *subset constraint*, restricting the population of the first sequence to be a subset of the second (symbol 26). A dotted line with a circled "=" symbol depicts an *equality constraint*, indicating the populations must be equal (symbol 27). A circled "X" (symbol 28) depicts an *exclusion constraint*, indicating the populations are mutually exclusive. Exclusion and equality constraints may be applied between two or more sequences. Combining an inclusive-or and exclusion constraint yields an *exclusive-or constraint* (symbol 29).

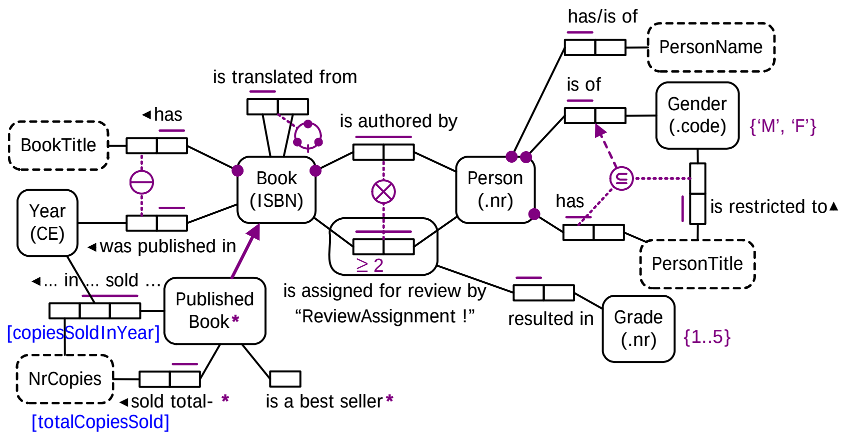
A solid arrow (symbol 30) from one object type to another indicates that the first is a (proper) *subtype* of the other (e.g. Woman is a subtype of Person). Mandatory (circled dot) and exclusion (circled "X") constraints may be displayed between subtypes, but are implied by other constraints if the subtypes have formal definitions. Symbol 31 shows four kinds of *frequency constraint*. Applied to a role sequence, these indicate that instances that play those roles must do so *exactly n* times, *at least n* times, *at most n* times, or *at least n and at most m* times. Symbol 32 shows four varieties of *value-comparison constraint*. The arrow shows the direction in which to apply the circled operator between two instances of the same type (e.g. For each Employee, hiredate > birthdate).

Symbol 33 shows the main kinds of *ring constraint* that may apply to a pair of compatible roles. Read left to right and top row first, these indicate that the binary relation formed by the role population must respectively be irreflexive, asymmetric, antisymmetric, reflexive, intransitive, acyclic, intransitive and acyclic, or intransitive and asymmetric.

The previous constraints are *alethic* (necessary, so can't be violated) and are colored violet. ORM 2 also supports *deontic* rules (obligatory, but can be violated). These are colored blue, and either add an "o" for obligatory, or soften lines to dashed lines. Displayed here are the deontic symbols for uniqueness (symbol 34), mandatory (symbol 35), set-comparison (symbol 36), frequency (symbol 37) and ring (symbol 38) constraints.

Figure 2 shows a sample ORM schema for a book publishing domain. A detailed discussion using the CSDP to develop this schema may be found elsewhere [9]. Each book is identified by an International Standard Book Number (ISBN), each person is identified by a person number, each grade is identified by a grade number in the range 1 through 5, each gender is identified by a code ('M' for male and 'F' for Female), and each year is identified by its common era (CE) number. PublishedBook is a derived subtype determined by the subtype definition shown at the bottom of the figure. ReviewAssignment objectifies the relationship Book is assigned for review by Person, and is independent since an instance of it may exist without playing any other role (one can known about a review assignment before knowing what grade will result from that assignment).

The internal uniqueness constraints (depicted as bars) and mandatory role constraints (solid dots) verbalize as follows: Each Book is translated from at most one Book; Each Book has exactly one BookTitle; Each Book was published in at most one Year; For each Published Book and Year, that PublishedBook in that Year sold at most one NrCopies; Each PublishedBook sold at most one total NrCopies; It is possible that the same Book is authored by more than one Person and that more than one Book is authored by the same Person; Each Book is authored by some Person; It is possible that the same Book is assigned for review by more than one Person and that more than one Book is assigned for review by the same Person; Each ReviewAssignment resulted in at most one Grade; Each Person has exactly one PersonName; Each Person has at most one Gender; Each Person has at most one PersonTitle; Each PersonTitle is restricted to at most one Gender.



* **Each** PublishedBook **is a** Book that was published in **some** Year.
* **For each** PublishedBook, totalCopiesSold= **sum**(copiesSoldInYear).
* PublishedBook is a best seller **iff** PublishedBook sold total NrCopies >= 10000.

Fig. 2. An ORM schema for a book publishing domain

The external uniqueness constraint (circled bar) indicates that the combination of BookTitle and Year applies to at most one Book. The acyclic ring constraint (circle with three dots and a bar) on the book translation predicate indicates that no book can be a translation of itself or any of its ancestor translation sources. The exclusion constraint (circled cross) indicates that no book can be assigned for review by one of its authors. The frequency constraint (≥ 2) indicates that each book that is assigned for review is assigned for review by at least two persons. The subset constraint (circled subset symbol) means that if a person has a title that is restricted to some gender, then the person must be of that gender. The first argument of this subset constraint is a person-gender role pair projected from a join path that performs a conceptual join on PersonTitle. The last two lines at the bottom of the schema declare two derivation rules, one specified in attribute-style using role names and the other in relational style using predicate readings

KEY APPLICATIONS

ORM has been used productively in industry for over 30 years, in all kinds of business domains. *Commercial tools* supporting the fact-oriented approach include Microsoft's Visio for Enterprise Architects, and the FCO-IM tool CaseTalk (www.casetalk.com). CogNIAM, a tool supporting NIAM2007 is under development at PNA Active Media (http://cogniam.com/). *Free ORM tools* include VisioModeler and Infagon (www.mattic.com). Dogma Modeler (www.starlab.vub.ac.be) and T-Lex [15] are academic ORM-based tools for specifying ontologies. NORMA (http://sourceforge.net/projects/orm), an open-source plug-in to Microsoft® Visual Studio, is under development to provide deep support for ORM 2 [3].

FUTURE DIRECTIONS

Research in many countries is actively extending ORM in many areas (e.g. dynamic rules, ontology extensions, language extensions, process modeling). A detailed overview of this research may be found in [9]. General information about ORM, and links to other relevant sites, may be found at www.ORMFoundation.org and www.orm.net.

CROSS REFERENCES

Conceptual schema design, Data model, Entity Relationship (ER) Model, UML.

RECOMMENDED READING

1. Bakema G, Zwart J, van der Lek H (2000) *Fully Communication Oriented Information Modelling*. Ten Hagen Stam, The Netherlands.
2. Bloesch A, Halpin T (1997) Conceptual queries using ConQuer-II. In: *Proc ER'97: 16th Int. Conf. on Conceptual Modeling*, LNCS vol 1331. Springer, Berlin Heidelberg New York, pp 113–126
3. Curland M, Halpin T (2007) Model Driven Development with NORMA. In: *Proc. HICSS-40*, CD-ROM, IEEE Computer Society.
4. Embley D, Kurtz B, Woodfield S (1992) *Object-Oriented Systems Analysis: A Model-Driven Approach*, Prentice Hall, Englewood Cliffs.
5. Falkenberg E (1976) Concepts for modeling information. In: Nijssen GM (ed) *Proc. 1976 IFIP Working Conf. on Modelling in Data Base Management Systems*, North-Holland Publishing, pp 95–109
6. Halpin T (2001) *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
7. Halpin T (2004) Comparing Metamodels for ER, ORM and UML Data Models. In: Siau K (ed) *Advanced Topics in Database Research, vol. 3*, Idea Pub. Group, Hershey, pp 23–44
8. Halpin T, Curland M (2006) Automated Verbalization for ORM 2. In: Meersman R et al. (eds) *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, LNCS vol 4278. Springer, Berlin Heidelberg New York, pp 1181–1190
9. Halpin, T. (2007) Fact-Oriented Modeling: Past, Present and Future. In: Krogstie J, Opdahl A, Brinkkemper S (eds) *Conceptual Modelling in Information Systems Engineering*. Springer, Berlin, pp 19-38
10. Halpin T, Evans K, Hallock P, MacLean W (2003) *Database Modeling with Microsoft® Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
11. ter Hofstede AHM, Proper HA, Weide thP van der (1993) Formal definition of a conceptual language for the description and manipulation of information models, *Information Systems*, vol. 18, no. 7, pp 489–523
12. Meersman R (1982) The RIDL conceptual language, Research report, Int. Centre for Information Analysis Services, Control Data Belgium, Brussels.
13. Nijssen GM (1977) Current issues in conceptual schema concepts. In: Nijssen GM (ed) *Proc. 1977 IFIP Working Conf. on Modelling in Data Base Management Systems*, Nice, France, North-Holland Publishing, pp 31–66
14. OMG 2007, *Semantics of Business Vocabulary and Business Rules (SBVR)*. URL: http://www.omg.org/cgi-bin/doc?dtc/2006-08-05.
15. Trog D, Vereecken J, Christiaens S, De Leenheer P, Meersman R (2006) T-Lex: A Role-Based Ontology Engineering Tool. In: Meersman R et al. (eds) *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, LNCS vol 4278. Springer, Berlin Heidelberg New York, pp 1191-1200
16. Wintraecken J (1990) *The NIAM Information Analysis Method: Theory and Practice*, Kluwer, Deventer, The Netherlands