The Semantics of Reifying N-ary Relationships as Classes

Mohamed Dahchour and Alain Pirotte

Université catholique de Louvain
IAG School of Management, Information Systems Unit (ISYS)
1 Place des Doyens
B-1348 Louvain-la-Neuve, Belgium
dahchour@isys.ucl.ac.be,pirotte@info.ucl.ac.be

Keywords: information modeling, n-ary relationship reification

Abstract: Many data models do not directly support n-ary relationships. In most cases, they are either reduced to some of their binary projections or directly translated into an n-ary "relationship relation" in the relational model. This paper addresses the reification of an n-ary relationship into a new class with n binary relationships and studies the preservation of semantics in the translation. It shows that some semantics may be lost unless some explicit constraints are added to the binary schema.

1 INTRODUCTION

Information modeling focuses on capturing and representing certain aspects of the real world relevant to the functions of an information system. The central constructs in information models are *classes* (or types, entities), representing important things of the application domain, and *relationships* among classes. It is relatively easy to identify classes that appropriately capture real-world objects: they directly correspond to the important concepts naturally manipulated by stakeholders in the application domain. The choice of appropriate relationships to associate classes is comparatively more difficult.

Binary relationships are most frequent in information models. Still, some situations are naturally modeled with ternary and higher-degree relationships. Few models directly support both binary and n-ary relationships (see, e.g., (Dey et al., 1999)). Most often, n-ary relationships are dealt with in one of two ways (Batini et al., 1992; Elmasri and Navathe, 2000; Jones and Song, 1993; Jones and Song, 2000; Ling, 1985; McAllister and Sharpe, 1998; Teorey, 1994; Thalheim, 2000)):

(i) reduce the n-ary relationship to relationships of lower degree. This is not always done correctly, as pointed out in, e.g., (Batini et al., 1992; Elmasri and Navathe, 2000). The literature mostly addresses the conditions under which this decomposition is correct and discusses whether the semantics of the original relationship is preserved in the decomposition.

To illustrate this approach, consider relationship works(Team,Project,Budget) (Figure 1(a)), giving information about teams working on projects with the budgets allotted. Relationship works can be modeled as three binary relationships worksOn(Team,Project), fundedBy(Project,Budget), and usedBy(Team,Budget) (Figure 1(b)), only if a constraint similar to a relational join dependency holds between Team, Budget, and Project in the ternary relationship.

(ii) evacuate the problem into relational modeling by defining a "relationship relation" that directly models the n-ary relationship among n classes as a relation with n attributes, each linking, through a referential integrity constraint, to the primary key of an "entity relation" representing an entity class of the n-ary relationship. The problem with this representation is that the shades of meaning of the original n-ary relationship are blurred by the poorer expressive power of the relational model.

This paper discusses a systematic approach, wholly within the entity-relationship model, to model an n-ary relationship as binary relationships. The approach, which could be called *reification* or *objectification*, permits to take advantage of the richer semantics of the entity-relationship model for re-expressing the semantics of the n-ary relationship. Our presentation is in line with our earlier work on generic rela-

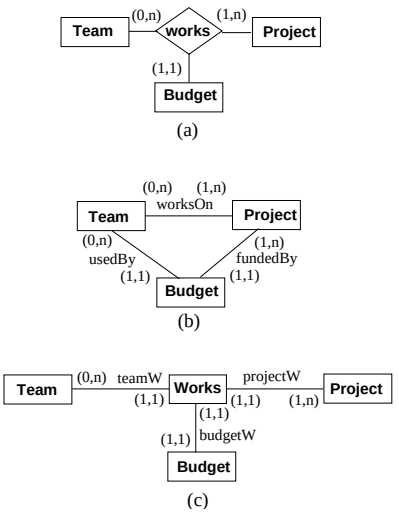

Figure 1: Modeling a ternary relationship with a set of binary relationships.

tionships (see, e.g., (Dahchour, 2001; Dahchour et al., 1999)).

To give the flavor of the approach, consider again relationship works(Team,Project,Budget). It can be reified as new class Works with three binary relationships: teamW(Team,Works), projectW(Project,Works), and budgetW(Budget,Works) (see Figure 1(c)).

The rest of the paper is organized as follows. Section 2 presents a common semantics for binary and n-ary relationships. Section 3 presents the reification technique of n-ary relationships, shows how the semantics of an n-ary relationship can be described in terms of the n binary relationships that represent it, and discusses some problems raised by reification. It is shown that some of the original semantics of the n-ary relationship may be lost by reification unless additional constraints are made explicit. Section 4 summarizes and concludes the paper.

2 BINARY AND N-ARY RELATIONSHIPS

This section presents a common semantics for binary and n-ary relationships along several dimensions, including *cardinality, existence dependency, attribute propagation, exclusion,* and *inclusion.* Other dimensions, like *symmetry/asymmetry, recursivity,* and *transitivity* are not discussed here, as they specific to binary relationships. As for notations, binary relationships are drawn as straight lines as in UML, while n-ary relationships are depicted as diamonds. The name of a relationship (resp., class) begins with a lowercase (resp., uppercase) letter.

2.1 Cardinality

The cardinality dimension constrains the number of objects related by a relationship. Let R be a binary relationship associating classes C_1 and C_2, and (c_{min}, c_{max}) be the cardinality at the side of C_1. It reads as follows: each instance of C_1 must participate in at least c_{min} and at most c_{max} links[1] in R at all times[2]. The most frequent cardinalities are: (0,1) (at most one), (1,1) (exactly one), (0,n) or (0,*) (any number, the unconstrained case), and (1,n) (at least one). For example, consider relationship employs(Company(0,n),Person(0,1)). The (0,1) cardinality at the side of Person means that an employee can work for at most one company. The (0,n) cardinality at the side of Company means that each company can be related to any number of employees. Consider now ternary relationship works of Figure 1(a). The (0,n) cardinality at the side of Team means that a team can be associated to any number of (project,budget) pairs. The other cardinalities can be read in a similar way. An interesting discussion about the definition of cardinality, and the related notations and interpretations can be found in (Génova et al., 2001; Castellani et al., 200).

2.2 Existence Dependency

Existence dependency characterizes whether or not an object can exist independently of related objects[3]:

- *dependence* means that the existence of an object of C_1 depends on the existence of objects of C_2 related to C_1 by a binary relationship R. This is known as *mandatory* participation in ER modeling and expressed by a minimum cardinality greater or equal to 1 at the side of C_1 in R.
- *independence* means that the existence of an object of C_1 is independent of the existence of related objects of C_2. This is known as *optional* participation in ER modeling and expressed by a minimum cardinality equal to 0 at the side of C_1 in R.

Existence dependency also specifies how insertion or deletion of one object can influence the existence of connected objects. Let o_1 and o_2 be two objects related by link r. There are three options for deletion operations to maintain the existence dependency:

[1] "Link" is used for "instance of a relationship". Links relate individual objects or entities.

[2] We prefer this version of cardinalities attached to the "source" class of the relationship to that of UML, where they are attached to the "target"class, in particular because this version extends smoothly to n-ary relationships

[3] Existence dependency is sometimes referred to as *referential integrity*, which defines an inclusion relationship among two attributes defined on the same domain in different entities.

- *default deletion:* the deletion of an object implies the deletion of all links referencing it (e.g., deletion of o_1 implies deletion of r);
- *cascade deletion:* the deletion of an object implies the deletion of all links referencing it as well as of those objects themselves (e.g., deletion of o_1 implies deletion of r and o_2);
- *restrict deletion:* the deletion of an object is prohibited if it is referred to by at least one object (e.g., deletion of o_1 is disallowed).

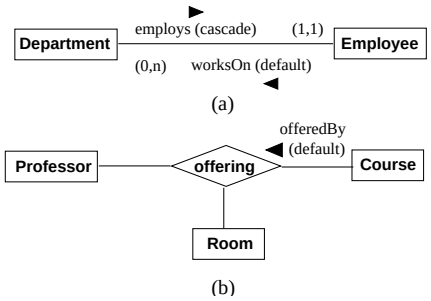

Figure 2: Existence dependency.

Deletion options can be associated with each class (or role) of a relationship. For example, in Figure 2(a), cascade deletion associated with class Department (or role employs) means that deleting a department entails deleting all its employees. Default deletion associated with class Employee (or role worksOn) states that deleting an employee only implies deleting its link to a department. For the ternary relationship offering in Figure 2(b), default deletion associated with class Course (or role offeredBy) means that deleting a course only implies deleting its links in offering.

2.3 Attribute Propagation

Attributes can propagate through relationships from one class to another. Propagated attributes are sometimes said to be derived. For example, consider relationship publish(Article,Journal), where class Journal has an attribute issueDate. An explicit attribute publicationDate is not necessary in class Article, as the information can be derived from attribute issueDate of Journal via relationship publish. Such propagation mechanisms are expressed in ODMG (Cattell et al., 2000) by the so-called *path expressions*. For example, the publication date of an article article#1 can be expressed by the path expression article#1.journal#1.issueDate which returns the issue date of journal#1 where article#1 appeared.

Attributes may also propagate through ternary relationships. As an example, consider relationship offering(Professor,Course,Room), where class Course has an attribute #Hour giving the total number of hours for a course. The value of an attribute #Hour for class Professor, to represent the total number of hours that a professor devotes to teaching, can be computed as the sum of #Hour values for the courses taught. For example, John.#Hour=sum(c.#Hour) where c belongs to the set of courses taught by John i.e., $\{c_i \in$Course: $\exists r \in$Room offering(John, $c_i, r)\}$.

2.4 Exclusion

We retain both categories of exclusion defined in (Habrias, 1993), called *role* and *relationship* exclusion. Exclusive roles/relationships are noted as a dashed line labeled {xor}.

Role exclusion. Let R_1 and R_2 be two relationships sharing the same class C_0 at one end, and with classes C_1 and C_2, respectively, at the other end. Let $\rho_{\mathsf{C}_0}(\mathsf{R}_i)$ denote the role played by C_0 in relationship R_i and $\pi_{\mathsf{C}_0}(\mathsf{R}_i)$ denote the set of instances of C_0 participating in relationship R_i.

Exclusion between two roles $\rho_{\mathsf{C}_0}(\mathsf{R}_1)$ and $\rho_{\mathsf{C}_0}(\mathsf{R}_2)$ means that the sets $\pi_{\mathsf{C}_0}(\mathsf{R}_1)$ and $\pi_{\mathsf{C}_0}(\mathsf{R}_2)$ are disjoint. Formally, $\rho_{\mathsf{C}_0}(\mathsf{R}_1)$ {xor} $\rho_{\mathsf{C}_0}(\mathsf{R}_2) \equiv \pi_{\mathsf{C}_0}(\mathsf{R}_1) \cap \pi_{\mathsf{C}_0}(\mathsf{R}_2) = \emptyset$. For example, exclusion between roles $\rho_{\mathsf{Apartment}}$(rent) and $\rho_{\mathsf{Apartment}}$(sell) in Figure 3 means that the same apartment cannot be simultaneously rented and sold.

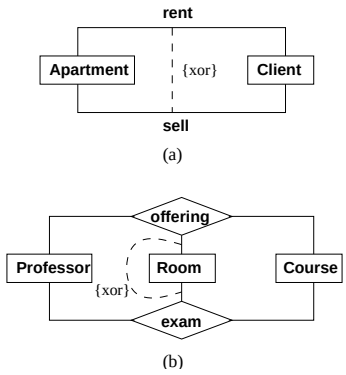

Figure 3: Role exclusion.

An example of role exclusion between two ternary relationships is shown in Figure 3(b): ρ_{Room}(offering) {xor} ρ_{Room}(exam) expresses that a room cannot simultaneously hold a course and an exam.

Relationship exclusion. Let R_1 and R_2 be two relationships sharing the same classes at their ends. Exclusion between relationships R_1 and R_2 means that the set of links of R_1 and the set of links of R_2 are disjoint. Formally, R_1 {xor} $\mathsf{R}_2 \equiv \mathsf{R}_1 \cap \mathsf{R}_2 = \emptyset$. For example, Figure 4(a) shows two exclusive binary relationships borrows and reserves, with the meaning

that a student cannot simultaneously borrow and reserve the same book.

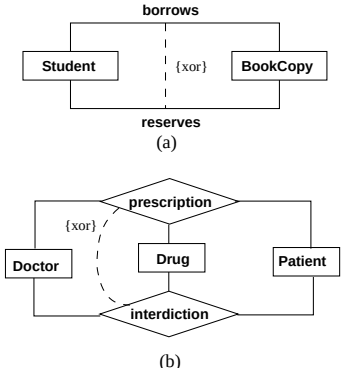

Figure 4: Relationship exclusion.

An example of relationship exclusion between two ternary relationships is depicted in Figure 4(b). It expresses that a doctor does not both prescribe and forbid the same drug to the same patient.

2.5 Inclusion

Like exclusion, the *inclusion* dimension can be defined for both roles and relationships. Inclusive roles/relationships are noted as a dashed arrow labeled {subset}.

Role inclusion. Inclusion of role $\rho_{C_0}(R_1)$ in role $\rho_{C_0}(R_2)$ means that the set of instances of C_0 participating in R_1 is a subset of the set of instances of C_0 participating in R_2. Formally, $\rho_{C_0}(R_1)$ {subset} $\rho_{C_0}(R_2) \equiv \pi_{C_0}(R_1) \subseteq \pi_{C_0}(R_2)$. For example, Figure 5(a) shows inclusion of role $\rho_{Student}$(practices) in role $\rho_{Student}$(registers), with the meaning that a student who practices a sport has necessarily registered for that sport.

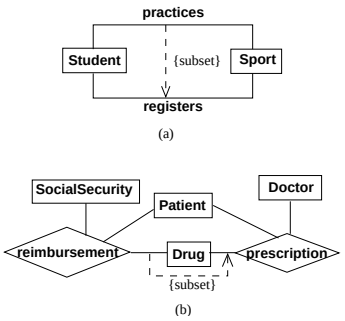

Figure 5: Role inclusion.

An example of role inclusion involving ternary relationships prescription(Doctor,Drug,Patient) and reimbursement (SocialSecurity,Drug,Patient) is shown in Figure 5(b). Inclusion of role ρ_{Drug}(reimbursement) in role ρ_{Drug}(prescription) means that drugs reimbursed by Social Security are necessarily prescription drugs.

Relationship inclusion. Inclusion of R_1 in R_2 means that the set of links of R_1 is a subset of the set of links of R_2. Formally, R_1 {subset} $R_2 \equiv R_1 \subseteq R_2$. For example, Figure 6(a) shows inclusion of relationship published in relationship accepted, meaning that an article published in a journal has necessarily been accepted for publication in that journal.

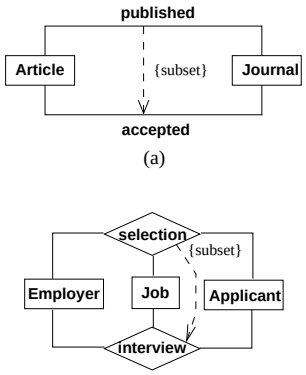

Figure 6: Relationship inclusion.

An example of inclusion between two ternary relationships is shown in Figure 6(b). Relationship selection(Employer,Applicant,Job) gives information about job selection. Relationship interview(Employer,Applicant,Job) gives information about job interviews. Inclusion of selection in interview means that applicant a selected by employer e for job j was necessarily interviewed by e about j.

3 REIFYING N-ARY RELATIONSHIPS

This section first presents the reification technique for n-ary relationships. Then, it shows how the semantics of an n-ary relationship can be described from the semantics of n binary relationships and discusses some problems raised by this reification. A class representing a reified relationship is given the same name as the relationship with its first letter capitalized.

3.1 Reification

Let R be an n-ary relationship among n classes C_1, C_2, ..., C_n. Reifying (or objectifying) R consists in

creating a new class, say RClass[4], related by binary relationships $R_1, R_2, \ldots, R_n$ to each of the n original classes $C_1, C_2, \ldots, C_n$, respectively. The cardinality of RClass in R_i ($1 \leq i \leq n$) is (1,1) and that C_i in R_i ($1 \leq i \leq n$) is the same as the cardinality of C_i in R. As an example, relationship prescription in Figure 7(a) is reified as class Prescription involved in three binary relationships as shown in Figure 7(b).

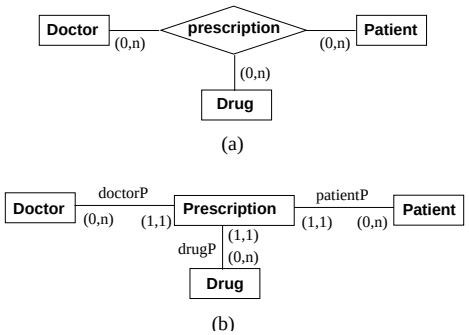

Figure 7: Reification of a ternary relationship.

3.2 Cardinality

For the three binary relationships in Figure 7(b), the cardinality at the side of classes Doctor, Patient, and Drug (i.e., (0,n)) is the same as the cardinality of the corresponding class in the ternary relationship prescription of Figure 7(a). The cardinality at the side of class Prescription is (1,1) for all three binary relationships, meaning that an instance of class Prescription corresponds to exactly one doctor, one patient, and one drug.

3.3 Existence Dependency

For the example in Figure 8(a), consider link offering(p,c,r) relating professor p, course c, and room r. The default mode in Figure 8(a) states that deleting c entails deleting link offering(p,c,r) only, while preserving objects p and r.

After reifying relationship offering in class Offering (see Figure 8(b)), link offering(p,c,r) is represented by three links: professorO(p,o), courseO(c,o), and roomO(r,o) where o is an instance of class Offering. The default mode associated with offeredBy in Figure 8(a) can be expressed by the following modes (see Figure 8(b)) performed in order: (i) cascade, associated with Course in relationship courseO, and (ii) default, associated with Offering in courseO, professorO, and roomO.

[4]In ER modeling, this new class is sometimes called a *weak entity* (Elmasri and Navathe, 2000).

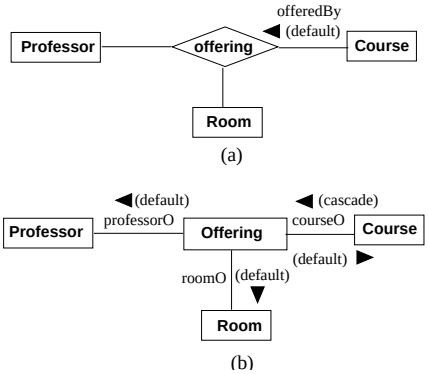

Figure 8: Existence dependency after reification.

Mode (i) ensures that the deletion of C1 implies the deletion of o. Mode (ii) then ensures that, upon deletion of o, links professorO(p,o) and roomO(r,o) are deleted, but p and r are not.

To generalize, consider a ternary relationship R associating classes C_1, C_2, and C_3. Assume that R reifies as class RClass with three binary relationships R_1, R_2, and R_3 to the given classes C_1, C_2, and C_3, respectively. Table 1 shows how the deletion options associated with R can be expressed in terms of the binary relationships that represent it. The deletion option d associated with role r is represented in the table as pair (r,d). The first line of the table corresponds to the example of Figure 8.

R before reification	R after reification
(ρ_{C_i}(R), default)	(ρ_{C_i}(R_i), cascade), (ρ_{RClass}(R_j), default) ($1 \leq j \leq 3$)
(ρ_{C_i}(R), cascade)	(ρ_{C_i}(R_i), cascade), (ρ_{RClass}(R_j), cascade) ($1 \leq j \leq 3$)
(ρ_{C_i}(R), restrict)	(ρ_{C_i}(R_i), restrict), (ρ_{RClass}(R_j), restrict) ($1 \leq j \leq 3$)

Table 1: The existence dependency of a ternary relationship after reification.

3.4 Attribute Propagation

Consider the propagation of attribute #Hour in relationship offering of Figure 9(a). After the reification of offering as shown in Figure 9(b), the value of #Hour for John can be computed as: John.#Hour=sum(c.#Hour) where $c \in \{c_i \in$Course: $\exists o \in$Offering, professorO(John,o) and courseO(c_i,o)$\}$.

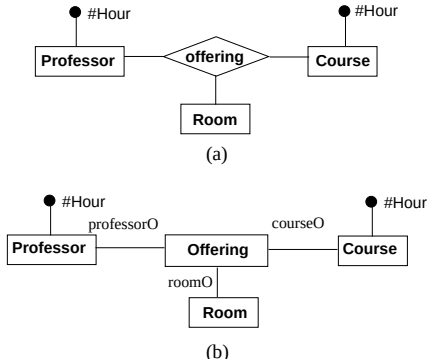

Figure 9: Attribute propagation after reification.

3.5 Exclusion

Role exclusion. When both ternary relationships offering and exam of Figure 3(b) are reified as shown in Figure 10, the example of role exclusion in Figure 3(b) can be expressed in terms of exclusion between roles of two binary relationships roomO and roomE, namely, $\rho_{\mathsf{Room}}(\mathsf{roomO})$ {xor} $\rho_{\mathsf{Room}}(\mathsf{roomE})$ (i.e., a room cannot simultaneously hold a course and an exam).

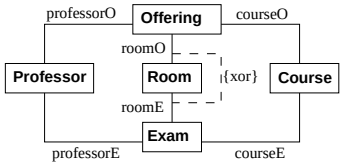

Figure 10: Role exclusion of Figure 3(b) after reification.

Relationship exclusion. Unlike role exclusion, relationship exclusion for n-ary relationships cannot be expressed in terms of exclusion between binary relationships.

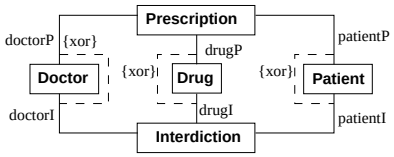

Figure 11: Relationship exclusion of Figure 4(b) is lost after reification.

Figure 11 shows the reification of both ternary relationships prescription and interdiction of Figure 4(b) as classes Prescription and Interdiction, respectively.

The example of relationship exclusion in Figure 4(b) (i.e., a doctor does not both prescribe and forbid the same drug to the same patient) cannot be expressed in terms of the three role exclusion constraints shown in Figure 11.

To accurately express the example of exclusion in Figure 4(b), the following additional constraint is required:

$\forall do$:Doctor, pa:Patient, dr:Drug
[$\exists pr$:Prescription (doctorP(do,pr) $\wedge$
drugP(dr,pr) $\wedge$ patientP(pa,pr))] $\equiv$
[$\neg$ ($\exists i$:Interdiction (doctorI(do,i) $\wedge$
drugI(dr,i) $\wedge$ patientI(pa,i)))]

3.6 Inclusion

Role inclusion. After reifying both ternary relationships reimbursement and prescription of Figure 5(b) as shown in Figure 12, the role inclusion in Figure 5(b) can be expressed in terms of inclusion of role $\rho_{\mathsf{Drug}}(\mathsf{drugR})$ in role $\rho_{\mathsf{Drug}}(\mathsf{drugP})$.

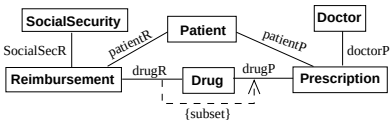

Figure 12: Role inclusion of Figure 5(b) after reification.

Relationship inclusion. Like relationship exclusion, relationship inclusion in Figure 6(b) cannot be expressed by the conjunction of the three role-inclusion constraints shown in Figure 13.

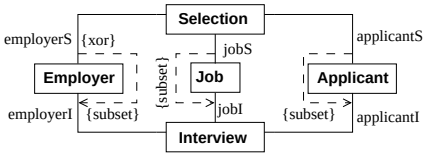

Figure 13: Relationship exclusion of Figure 4(b) is lost after reification.

To accurately express the example of inclusion constraint in Figure 6(b), the following additional constraint is needed:

$\forall e$:Employer, a:Applicant, j:Job
($\exists s$:Selection (employerS(e,s) $\wedge$ jobS(j,s) $\wedge$
applicantS(a,s))) $\Longrightarrow$
($\exists i$:Interview (employerI(e,i) $\wedge$ jobI(j,i) $\wedge$
applicantI(a,i)))

4 CONCLUSION

This paper discussed the reification of n-ary relationships as new classes with n binary relationships. We

first defined a common semantics for both binary and n-ary relationships along several dimensions, including cardinality, existence dependency, attribute propagation, exclusion, and inclusion. We then presented the reification approach, which consists in transforming an n-ary relationship into a new class with n new binary relationships. We then discussed some problems raised by this transformation. Although reification is often assumed to fully preserve the original semantics of the reified relationships, we showed that this is only true in part. The semantics of cardinality, existence dependency, attribute propagation, role exclusion, and role inclusion are indeed preserved, but relationship exclusion and inclusion are lost. We proposed additional constraints to fully recapture them.

REFERENCES

Batini, C., Ceri, S., and Navathe, S. (1992). *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings.

Castellani, X., Habrias, H., and Perrin, P. (200). A synthesis on the definitions and the notations of cardinalities of relationships. *Journal of Object-Oriented Programming*, pages 32–35.

Cattell, R., Barry, D., Berler, M., and Eastman, J., editors (2000). *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann.

Dahchour, M. (2001). *Integrating Generic Relationships into Object Models Using Metaclasses*. PhD thesis, Département d'ingénierie informatique, Université catholique de Louvain, Belgium.

Dahchour, M., Pirotte, A., and Zimányi, E. (1999). Materialization and its metaclass implementation. Technical Report YEROOS TR-99/01, IAG-QANT, Université catholique de Louvain, Belgium. To be published in IEEE Transactions on Knowledge and Data Engineering.

Dey, D., Storey, V., and Barron, T. (1999). Improving database design through the analysis of relationships. *ACM Trans. on Database Systems*, 24(4):453–486.

Elmasri, R. and Navathe, S. (2000). *Fundamentals of Database Systems*. Addison-Wesley, 3rd edition.

Génova, G., Llorens, J., and Martínez, P. (2001). Semantics of the minimum multiplicity in ternary associations in UML. In Gogolla, M. and Kobryn, C., editors, *Proc. of the 4th Int. Conf. on the Unified Modeling Language, UML'01*, LNCS 2185, Toronto, Ontario, Canada. Springer-Verlag.

Habrias, H. (1993). *Introduction à la spécification*. Masson.

Jones, T. and Song, I. (1993). Binary imposition rules and ternary decomposition. In *Proc. of the Conf. on Information Science and Technology, InfoScience'93*, pages 267–274, Seoul, Korea.

Jones, T. and Song, I. (2000). Binary equivalents of ternary relationships in entity-relationship modeling: A logical decomposition approach. *Journal of Database Management*, pages 12–19.

Ling, T. (1985). A normal form for entity-relationship diagrams. In Chen, P., editor, *Proc. of the 4th Int. Conf. on the Entity-Relationship Approach, ER'85*, pages 24–35, Chicago, Illinois.

McAllister, A. and Sharpe, D. (1998). An approach for decomposing n-ary data relationships. *Software - Practice and Experience*, 28(1):125–154.

Teorey, T. (1994). *Database Modeling and Design: The Fundamental Principles*. Morgan Kaufmann, 2nd edition.

Thalheim, B. (2000). *Entity-Relationship Modeling. Foundations of Database Technology*. Springer-Verlag.