Reference Scheme Reduction on Subtypes in ORM

Andy Carver and Terry Halpin

INTI International University, Malaysia
andy.carver@yahoo.com, t.halpin@live.com

Abstract. Object-Role Modeling (ORM) allows composite reference schemes for object types to be portrayed using either objectification (in the sense of situational nominalization) or coreference (as defined in ORM rather than linguistics). In practical modeling, cases can arise where a subtype of a compositely identified object type has a natural reference scheme that utilizes only some components of the supertype's reference scheme. Using the supertype's reference scheme to verbalize facts for the subtype then leads to redundancy or other irrelevance in the verbalization. Moreover, if such cases are input directly to the ORM's standard relational mapping procedure (Rmap), this can lead to table schemes that are not fully normalized. The paper identifies ways in which such problems can arise, and proposes ways to avoid these problems, partly by extending earlier work on reference scheme reduction, role redirection, and disjunctive reference, illustrating the approach with some practical examples.

1 Introduction

Fact-oriented modeling approaches such as Object-Role Modeling (ORM) [11], Cognition enhanced Natural language Information Analysis Method (CogNIAM) (*http://www.cogniam.eu/*), and Fully Communication Oriented Information Modeling (FCO-IM) [1] capture facts in attribute-free fact types, facilitating validation of conceptual data models with non-technical domain experts by verbalization in natural sentences and use of sample fact populations. Fact-oriented approaches also provide a graphical notation for business constraints that is more expressive than the graphical notation for data modeling in the Unified Modeling Language (UML) or industrial versions of Entity Relationship (ER) modeling. This paper focuses on ORM, discussing recent extensions to its schema transformation and relational mapping procedures.

ORM allows *composite reference schemes* for object types to be portrayed using either *objectification* (in the sense of situational nominalization, where relationships are objectified as states of affairs rather than propositions [7]) or *coreference* (as defined in ORM [11], where two or more relationships combine to provide reference, rather than in linguistics, where coreference involves multiple reference schemes). While this support for composite reference enables ORM to capture many natural identification schemes, it also presents some special challenges if one is to avoid data redundancy. In providing a reference scheme for an entity type, ORM modelers must determine which components are needed to identify entities of that type. But complicating this task is the fact that in ORM there are various ways in which an entity type will by default "inherit", so to speak, a reference scheme (possibly composite).

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 457–466, 2013.

There are at least three ways in which such "inheritance" occurs. First, an entity type that objectifies a binary or longer fact type "inherits", by default, a composite reference scheme based on the fact type it objectifies. Second, composite reference schemes can include referential relationships with one or more other entity types, each with its own reference scheme, perhaps composite, which is then incorporated into the original reference scheme. A third kind is the inheritance of a supertype's reference scheme by its subtype(s).

Reference-scheme inheritance can generate subtler instances of reference-scheme component irrelevance. For even if it is true that the components, and only those, of some entity type *A*'s composite reference scheme are relevant for identifying *A*-type objects, it may also be true, due to some contextual rule(s), that not all of those same components are relevant for identification of entities of type *B*—even if, according to the ORM schema, *B* inherits the complete reference scheme of entity type *A*.

The inclusion of components that are actually irrelevant, in a particular object reference, may result in generation of a database schema that allows data redundancy in some object references. In some cases, it could even allow fact redundancy, and hence lead to a table scheme that is not fully normalized. An earlier ORM paper [3] touched upon reference scheme reduction transforms required for correcting reference-component irrelevance generated via the first two kinds of "inheritance" just mentioned. The current paper considers cases of the third kind, viz. a subtype's inheritance of a supertype's reference scheme.

It is important to note that another way to think of that third kind (viz., the irrelevance of a certain referential component for some subtype's instances' identifications) is that there is a constraint or rule that restricts which objects may play a certain role. This situation is similar to the subtype situation: on the one hand, such a textual constraint would commonly be implemented as a subtype definition in ORM; and on the other hand, we normally introduce subtypes in ORM to declare that some specific role(s) are played only by instances of that subtype. In any case, only when subtypes do host—and thus restrict—some role, is there a chance of this subtyping-based referential irrelevance causing data redundancy. Thus, for our purposes, we may consider all such situations as effectively involving subtyping, even if the "subtype" is only one that is implied by a rule restricting which entities can play a particular role(s).

Subtype-specific referential irrelevance may be due to the referential component's data being *redundant* for objects of that subtype. Examples of this are discussed in Section 2. On the other hand, it could be due to the data's being *derivable* from the rest of the object reference. Examples of this are discussed in Section 3. Finally, it could be because some referential component is *inapplicable* to objects of that subtype. Examples of this are discussed in Section 4. Section 5 concludes the paper by summarizing the main contributions, suggesting some related topics for further research, and listing the references cited.

2 Subtype-Specific Referential Irrelevance Due to Redundancy

Mapping a subtype graph from an ORM schema to a relational schema may involve absorption, separation, or partition [11]. A subtype's non-functional roles (those without a simple uniqueness constraint) always map to a table separate from the

supertype's table. A subtype's functional roles may be absorbed into the supertable, or placed into a separate table, depending on the designer's choice. Referential redundancy of the kind of interest here can occur when a subtype's role is mapped to a separate table from that of its supertype(s), and the subtype inherits from a supertype a compound reference scheme that has at least one component that is not needed to identify instances of the subtype. As we will see shortly, without special precautions this can result in tables that are not fully normalized.

Subtype-specific referential irrelevance because of data *redundancy* could conceivably be due to a restricted uniqueness constraint, whose restriction-rule specifies exactly the same set of objects as does the subtype's definition, but such cases are unlikely to be met in practice. A much more likely possibility is a subtype definition that itself implies the restricted uniqueness constraint—and thus coordinates with it for free, as it were. We restrict our examples to such cases.

2.1 A Simple Example of Subtype-Specific Referential Redundancy

As a simple example of referential redundancy caused by subtyping, consider the ORM schema on the left side of Fig. 1. The subtyping ensures that a Flower-Containment has an additional Color only if the Bouquet-Pattern involved in the Flower-Containment is a Bouquet-Pattern that contains exactly one Floral Species. The restricted uniqueness constraint implied by this rule is seen more easily in the relational schema to which this Rmaps, shown on the right side of Fig. 1.

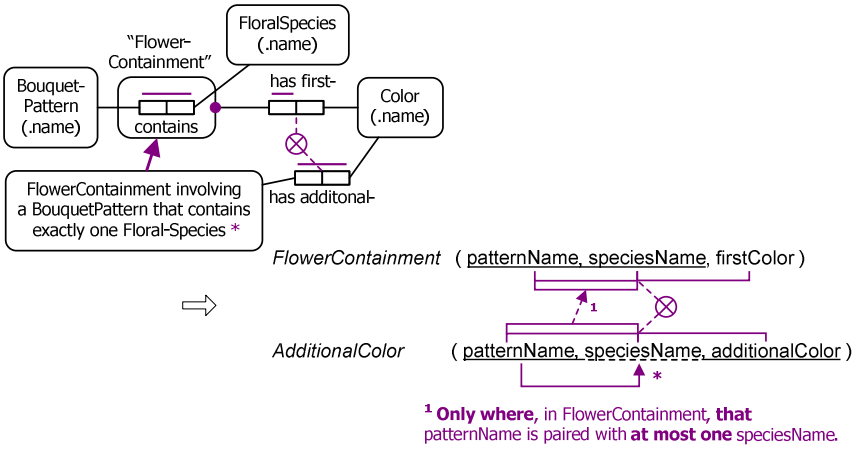

Fig. 1. A simple example that calls for reference-scheme reduction

Here the AdditionalColor table scheme has an extraneous column, speciesName, that is implicitly functionally dependent on just part of the key, so the table scheme is not in second normal form, allowing fact redundancy. This functional dependency (FD) is the restricted uniqueness constraint, implied here by the textual rule to which our

subtype's derivation rule Rmaps, qualifying the foreign key constraint between the two tables. Enforcing this textual rule will thus enforce also the functional dependency, controlling the denormalization.

Note that the subtype-defining rule in our ORM schema is of a peculiar sort: the subtype is defined as that subset of the supertype's objects where, *in the current population of any total table for the supertype's objects,* the objects have the same data value populating a particular component of their reference (viz., the Floral Species component). Thus it is the barest rule one could contrive that would imply the restricted uniqueness constraint. We will shortly consider an example with a more-specific subtyping rule. For our current example, we may remove the redundancy by applying a *role-redirection transformation*, to obtain the ORM schema in Fig. 2(b), which Rmaps to the relational schema in Fig. 2(c). The extraneous column is now absent from the AdditionalColor table; and the qualified subset constraint is now single-column, simplifying its enforcement.

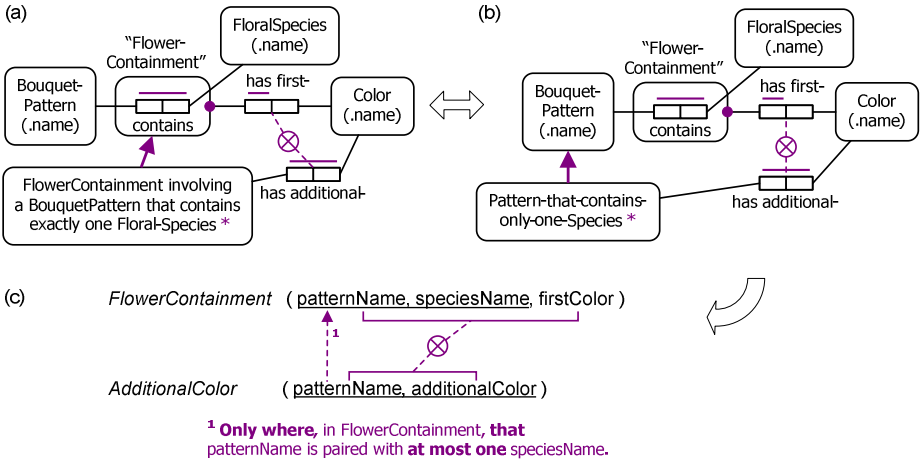

Fig. 2. Using a role-redirection transform to remove the referential redundancy

An abstract pattern for this kind of transformation—which applies only to the case where the nested supertype objectifies a binary, not an *n*-ary—is shown in Fig. 3. The fact type shown with a dashed-line in Fig. 3(a) is a link fact type implied by the objectification. The exact semantics of, and particular constraints shown on, the two rightmost fact types in each schema are there for clarity, but not part of the pattern.

In general, a table scheme that is not in 3rd normal form can also result if a fact type (or another entity type) has at least one non-key role hosted by an entity type that has a redundant component in its reference scheme. Such cases may be dealt with in a similar manner. For our earlier work on role redirection and role reduction transforms in ORM, see [3, 9], and for further aspects on ORM and normalization theory see [4].

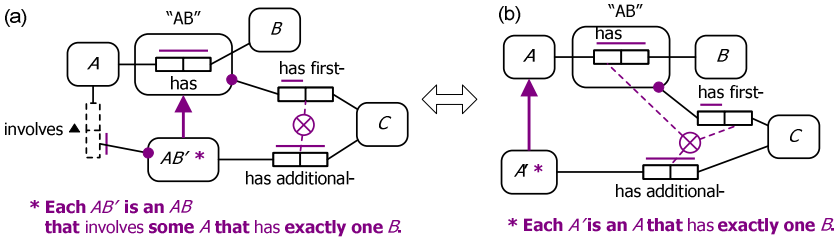

Fig. 3. A role redirection transform pattern

2.2 A Related Case Where the Supertype Is a Nested *n*-ary

A more complicated role redirection transform to remove referential redundancy involving a nested, *n*-ary fact is shown abstractly for the case of $n = 3$ in Fig. 4. The original ORM schema shown in Fig. 4(a) is transformed to the schema in Fig. 4(b). Note the nested binary in the transformed schema, and that its objectifying entity type is non-independent. For convenience, we have added some non-standard ORM graphics on top of the subtype shapes: the ternary fact-type shapes added there depict the restricted uniqueness constraint (spanning only two roles) graphically. Although this illustrates the ternary case, this approach is easily extended for higher arities.

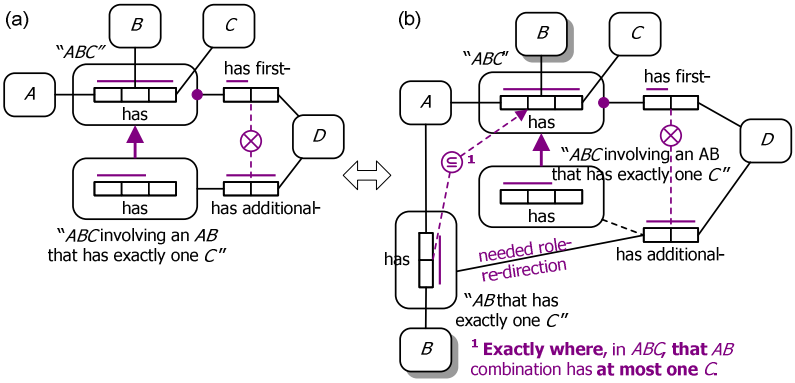

Fig. 4. A role redirection transform involving nesting of an *n*-ary fact type

3 Subtype-Specific Referential Irrelevance Due to Derivability

This section considers cases with a more specific subtyping rule that effectively enables derivation of supertype referential components, which become irrelevant for identifying instances of the subtype. An example containing such a subtyping rule is shown in Fig. 5. In this universe of discourse, we are interested in only three kinds of academy awards: best picture, best actor, and best actress.

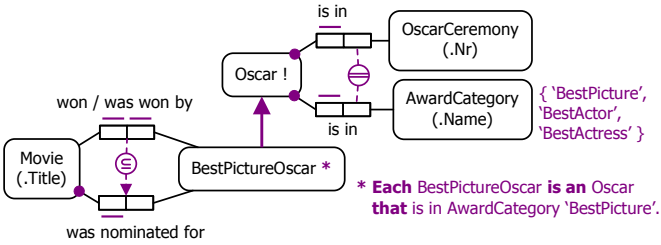

Fig. 5. Another pattern that calls for reference-scheme reduction

Any role hosted by the subtype that Rmaps to a table different from the Oscar table will map to two columns, one for its Oscar ceremony and one for its award category. However the entries in the awardCategory column of this table will all be the same, viz. 'BestPicture'. So, if the subtype's role is non-functional, its award ceremony column is not a key but it functionally determines its award category column, so the table scheme is denormalized. For example, the nomination fact type maps to the table *Movie*(movieTitle, bestPictureNominationOscarCeremony, bestPictureNominationAwardCategory, ...)

This conceptual schema presents a more difficult set of problems than the prior one. If we try verbalizing a sample fact of, say, the fact type BestPictureOscar was won by Movie, we immediately notice some referential irrelevance, even before completing the first object-reference: "The BestPictureOscar that is [an Oscar] in the AwardCategory 'BestPicture' and is in the OscarCeremony ...". Surely a more natural way of verbalizing a sample fact of this type would start out more simply: "The BestPictureOscar that is in the OscarCeremony ...". So while the Fig. 5 scheme might seem a natural way to model this domain, it leads to strange and awkward verbalization.

It is important to realize that there is a second way for referential data to be irrelevant: A referential component is irrelevant also if its value is derivable (by some contextual rule applying to this role) from other information in the entity reference. In the present case, the information from which the entity's AwardCategory ("BestPicture") is derivable is in the object (sub)type stated, "BestPictureOscar". The contextual rule that shows its derivability is the very derivation rule for that subtype. The formal-structural reason for this component's irrelevance is thus that the fact type by which the supertype instance is categorized (by the subtype's derivation rule), Oscar is in AwardCategory, is also a component of the supertype's reference scheme. This is exactly why the fact verbalization sounds like a circumlocution.

What is a solution for this kind of irrelevance? A role-redirection transform could be devised; but it would seem an unnatural way to model, because it would relegate to the predicate text the mention of 'the bestPictureOscar", in every fact type in which that subtype currently hosts a role (by redirecting all its roles to the OscarCeremony entity type). The insistent mentioning of an object of this "BestPictureOscar" type in the fact-verbalizations suggests—especially when there are other types of Oscar in the domain—that a BestPictureOscar entity type should host roles in these facts.

However, as discussed in the next section, there are other ways to model this domain that may offer a way to resolve the problem.

4 Subtype-Specific Referential Irrelevance Due to Inapplicability

If we were to base our initial elementary fact types on natural-sounding verbalizations of the facts of interest about Best Picture Oscars and other sorts of Oscars, the verbalizations would, in fact, not include referential components (for the specific sort of Oscar) that were either "redundant" or "derivable"—precisely because they would be irrelevant for unique identification of the entities.

Imagine that, in doing the Conceptual Schema Design Procedure (CSDP) Steps 1 and 2 upon this Oscar domain, our modeling, based on sample-fact verbalizations, led us to the two kinds of Oscar entity types shown in Fig. 6. Imagine also that we wish to record the same unary predicate ("... was fairly judged") for both of these entity types, and report facts of this type, for both of them, in the same report or query result.

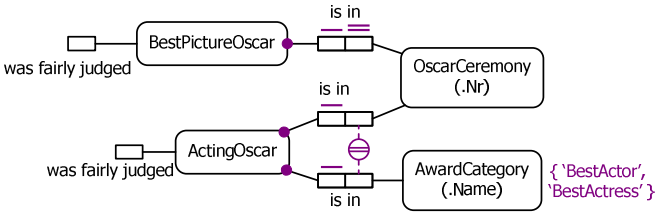

Fig. 6. Distinguishing Oscars with different reference schemes

In this modeling situation, when we came to Step 3 of the CSDP, we might generalize these two "...Oscar" entity types to a common Oscar entity type—even though the two types are mutually exclusive. However, while they have a common referential component, ActingOscar has more such components than does BestPictureOscar. The only way to combine them into one reference scheme would be for the reference scheme for Oscar to be *disjunctive* [12]. We might then end up with a schema at least functionally equivalent to that shown in Fig. 7. Here, the external uniqueness constraint with the small 'o" through its uniqueness bar has outer join semantics [8], so for a given OscarCeremony there is at most one Oscar with no ActingAwardCategory.

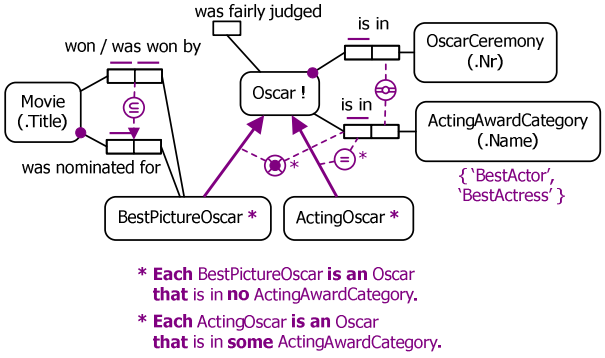

Fig. 7. A disjunctive reference-scheme for Oscar

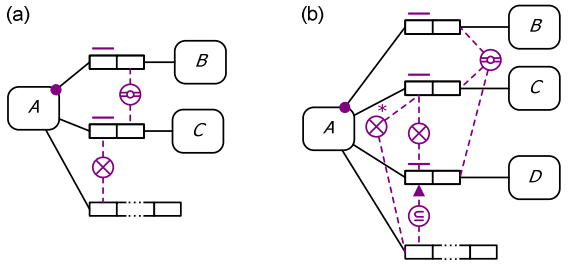

Fig. 8. Two patterns that lead to extraneous, inapplicable relational columns

However, this approach presents us still with a problem of extraneous data—or, to be more precise, extraneous nulls. For we still get an extraneous column—viz. the one for AwardCategory—where any of the BestPictureOscar subtype's roles map to a table different from that to which functional, non-referential (and non-restricted) roles hosted by the supertype map. The only differences will be that such a column will be filled with nulls on every row, instead of the AwardCategory 'BestPicture', and that the column is extraneous not because it is derivable, but because it is *inapplicable*. This is true for any case of disjunctive reference where the restricted role is either explicitly or implicitly spanned by an exclusion constraint spanning also one of the optional referential roles of the entity type. Fig. 8 shows two such schemes.

A role-redirection transform could be devised; but would be unnatural, because the predicate text in every fact type specific to that subtype would include "bestPictureOscar". A better solution for this kind of irrelevance, as well that caused by derivability, is to qualify Step 4 of our Rmap procedure [11] to ensure that when one "unpacks" any surrogate columns produced by Rmapping roles hosted by either explicit or implicit subtypes, where the role maps to a table other than that for any functional, non-referential roles of the supertype, then one unpacks that role only into its *relevant* (i.e. its *applicable, non-derivable, and non-redundant*) referential components.

Another solution is to transform the ORM schema before Rmapping by using one of the strategies for implementing other kinds of disjunctive reference: introduce a simple identifier; concatenate referential components to a simple identifier applying type casting where needed; render the optional referential roles mandatory by using special default values instead of nulls [11, 12]. Of these, introduction of a simple identifier is usually the best option (e.g. OscarId in Fig. 9 removes the problem). The third option (using default values) still leads to derivable irrelevance, unless the Rmap modification suggested above is employed.

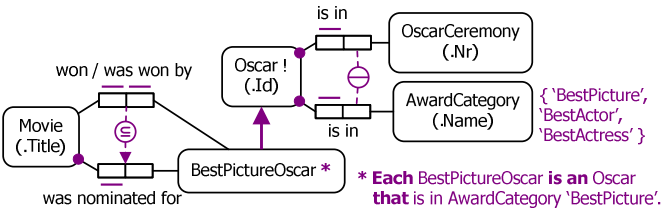

Fig. 9. Introducing a simple identifier

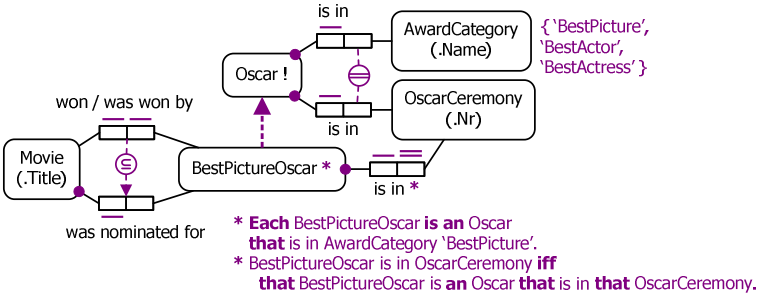

Fig. 10. An alternative approach using a derived reference-scheme

Yet another solution would be to extend ORM to allow preferred reference schemes to be based on derived fact types, as shown in Fig. 10. Here the subtype introduces its own, simpler context-dependent reference scheme which is itself derived using the rule shown. After relational mapping, Oscars in the Oscar supertable are identified compositely, but Oscars in the Movie table are simply identified.

5 Conclusion

This paper examined cases where a subtype of a compositely identified object type has a natural reference scheme that utilizes only some components of the supertype's reference scheme. As well as leading to redundancy in verbalizing some subtype facts at the conceptual level, such cases can map to relational schemas involving table schemes that are not fully normalized unless special precautions are taken.

We identified ways in which such problems can arise, and proposed ways to avoid these problems, partly by extending earlier work on reference scheme reduction, role redirection, and disjunctive reference. We used the term "redundancy" to refer to referential-component irrelevance of a very restricted sort, not based on the component's derivability or inapplicability. In the very broadest sense, of course, the term "redundancy" could include the latter two cases. But they seem different enough conceptually, from this most restricted case, that they deserve their own categories. Also, we found that whereas role-redirection was a feasible and seemingly appropriate solution for this most restricted type of "redundancy", it did not seem appropriate for cases of component derivability or inapplicability. For all three kinds of referential irrelevance for a subtype, however, we found that an extension to RMap Step 4 would be an effective solution for avoiding redundancy in the relational schema. Often, the most natural way of modelling a supertype's reference scheme is disjunctive reference; for which are available also other remedies for any relational-level redundancy.

Future plans in this area of research include implementing an extended version of Rmap in the NORMA tool [5] to deal properly with such cases, including introduction of surrogate identifiers as a mapping choice, and investigating further cases of disjunctive reference in which supertypes that are partitioned into subtypes with different reference schemes are allowed to abstract reference schemes from their subtypes.

References

1. Bakema, G., Zwart, J., van der Lek, H.: Fully Communication Oriented Information Modelling. Ten Hagen Stam (2000)
2. Carver, A.: How To Avoid Redundant Object-References. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 770–779. Springer, Heidelberg (2008)
3. Carver, A.: Roles in ORM: A Suggested Semantics. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM-WS 2011. LNCS, vol. 7046, pp. 360–369. Springer, Heidelberg (2011)
4. Carver, A., Halpin, T.: Atomicity and Semantic Normalization. International Journal of Information System Modeling and Design 1(2), 23–39 (2010)
5. Curland, M., Halpin, T.: The NORMA Software Tool for ORM 2. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 190–204. Springer, Heidelberg (2011)
6. Halpin, T.: What is an Elementary Fact? In: Nijssen, G.M., Sharp, J. (eds.) Proc. 1st NIAM-ISDM Conf. (1993)
7. Halpin, T.: Objectification of Relationships. In: Siau, K. (ed.) Advanced Topics in Database Research, vol. 5, pp. 106–123. Idea Publishing Group, Hershey (2006)
8. Halpin, T.: Modeling of Reference Schemes. In: Nurcan, S., Proper, H.A., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T., Bider, I. (eds.) BPMDS 2013 and EMMSAD 2013. LNBIP, vol. 147, pp. 308–323. Springer, Heidelberg (2013)
9. Halpin, T., Carver, A., Owen, K.M.: Reduction Transformations in ORM. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 699–708. Springer, Heidelberg (2007)
10. Halpin, T., Curland, M.: Recent Enhancements to ORM. In: Demey, Y.T., Panetto, H. (eds.) OTM 2013 Workshops. LNCS, vol. 8186, pp. 467–476. Springer, Heidelberg (2013)
11. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann Publishers (2008)
12. Halpin, T., Ritson, R.: Fact-Oriented Modelling and Null Values. In: Srinivasan, B., Zeleznikov, J. (eds.) Research and Practical Issues in Databases. World Scientific, Singapore (1992)
13. Levinson, S.C.: Pragmatics. Cambridge University Press, Cambridge (1983)