Uniqueness Constraints on Objectified Associations

Terry Halpin
Northface University

Abstract: Unlike UML and some ER versions, ORM currently allows a fact type to be objectified only if it either has a spanning uniqueness constraint or is a 1:1 binary fact type. This article argues that this restriction should be relaxed, and replaced by a modeling guideline that allows some *n*-ary associations to be objectified even if their longest uniqueness constraint spans *n*-1 roles. The pros and cons of removing this restriction are discussed, and illustrated with examples.

Introduction

In this article, the terms "relationship type", "association", and "fact type" all denote typed predicates (e.g. Person plays Sport). In many business domains, it is perfectly natural to think of certain relationship instances as objects about which we wish to talk. This process of making an object out of a relationship is called *objectification* (also known as reification, or nesting), and is supported by Object-Role Modeling (ORM) [3, 4, 5] and the Unified Modeling Language (UML) [6]. This article assumes basic familiarity with these modeling approaches.

The ORM schema in Figure 1(a) objectifies the fact type Person plays Sport as Play, which is then used as an object type in the fact type Play is at SkillLevel. The latter fact type is said to be *nested*, as it nests another fact type inside it. The exclamation mark "!" appended to "Play" indicates that Play is independent, so instances of Play may exist without participating in other fact types. This is consistent with the optional nature of the first role of Play is at SkillLevel. Figure 1(b) depicts the same example in UML. Here the skill level fact type is represented as an optional attribute on the association class Play.

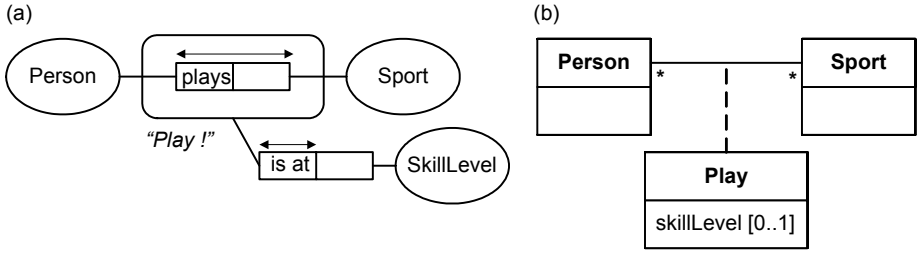

Figure 1 Objectifying an association with a spanning uniqueness constraint in (a) ORM and (b) UML.

Popular, industrial versions of Entity Relationship (ER) Modeling typically provide little or no support for objectification. However, academic versions of ER often provide partial or full support for objectification (e.g. see [1]). As an example of partial support, some ER versions allow objectified relationships to have attributes but not to play in other relationships.

Notice that the objectified association Person plays Sport has a spanning uniqueness constraint. In early versions of ORM, no association could be objectified unless it had a spanning uniqueness constraint. Later, I relaxed that restriction, to allow binary 1:1 associations to be objectified [2]. Currently, ORM allows no other uniqueness constraint patterns on objectified associations. In particular, ORM forbids the following two kinds of associations to be objectified:

(1) An *n*:1 (or 1:*n*) binary association
(2) A ternary or longer association whose longest uniqueness constraint spans exactly n-1 roles.

We exclude from consideration any *n*-ary association whose longest uniqueness constraint spans fewer than *n*-1 roles, because such an association is compound rather than elementary. Compound associations can be split into smaller associations without information loss, so may lead to redundancy problems if used as

primitive associations. Both UML and those versions of ER that support objectification allow cases (1) and (2) to be objectified. The rest of this article discusses whether ORM should be modified to do likewise.

Objectifying *n*:1 (and 1:*n*) associations

The ORM and UML schemas in Figure 2 are based on an academic ER example ([1], p. 34) that objectifies the *n*:1 association Person was born in City. This *n*:1 association may be viewed in the inverse direction as the 1:*n* association City is birthplace of Person, so any comments made here about *n*:1 associations apply equally to 1:*n* associations.

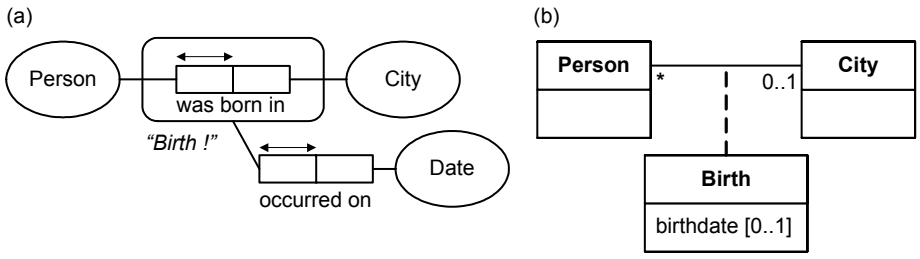

Figure 2 Objectifying an *n*:1 association in (a) ORM and (b) UML.

The Figure 2 example is structurally similar to that of Figure 1 except that the uniqueness constraint on the objectified binary spans just one role, not two. This nesting is currently illegal in ORM, mainly because the nested fact type Birth occurred on Date is compound, rather than elementary. It can be split without information loss into the two simpler fact types Person was born in City and Person was born on Date, as shown in the flattened (un-nested) schemas of Figure 3. Notice that the nesting in Figure 2 allows recording of birth dates only if the birth city is known. In the flattened schemas of Figure 3 this is captured by an ORM subset constraint, or a UML textual constraint (expressed here in the Object Constraint Language (OCL)). Although such a constraint might exist in some business domains, in practice it is far more likely for such *n*-1 associations that no such constraint exists. This is one reason to avoid nesting *n*-1 associations.

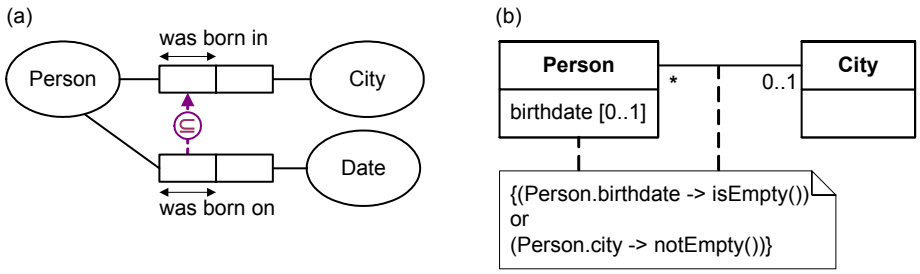

Figure 3 The nested fact type may be split into two binary fact types as shown.

For example, suppose we initially model things using nesting as shown in Figure 2, and populate the model with lots of data. If we later discover that it is possible to know a person's birth date without knowing their birth city, then we need to perform drastic surgery to the schema to cater for this possibility—remove the objectification, and attach the birth date fact type directly to Person. If the schema is also used as the implementation data structure (as it might be for an object-oriented class model), then we also need to modify the data. If instead we transformed the schema to a normalized data structure for implementation, then no changes are needed to the data population. In the latter case, the relational mapping procedure is now more complex, because effectively it must pre-process the schema to remove the nesting before applying a standard mapping procedure such as Rmap [4]. This is another reason to avoid nesting *n*-1 associations. In contrast, the task of evolving the flattened models of Figure 3 to allow recording of a person's birth date without the birth city is trivial: simply remove the subset constraint; no fact types need to be restructured, and no change is needed to the data population.

Although objectification of *n*:1 associations might sometimes be natural from a linguistic perspective, this kind of nesting typically portrays the business domain in an unnecessarily complicated way (why introduce birth cities in order to talk about birth dates?) that is de-normalized (the nesting includes an embedded functional dependency). Given this problem, and the disadvantages discussed earlier, I suggest that as a general rule we continue to treat nesting of *n*:1 associations as illegal in ORM.

There seems to be only one case where we might even consider breaking this rule. This is the case where the uniqueness constraint is likely to change over time to a fully spanning uniqueness constraint. In such cases, it might be argued that we simplify the task of model (schema plus population) evolution by nesting the original *n*:1 association (which may later evolve to an *m*:*n* association). In practice, such cases seem to be rare. Figure 4 depicts a borderline case based on an example I used in an earlier paper [2] to argue against nesting such cases. Although two of the roles have been made mandatory here to show some variations from the earlier cases, this structural change is not central to the discussion.

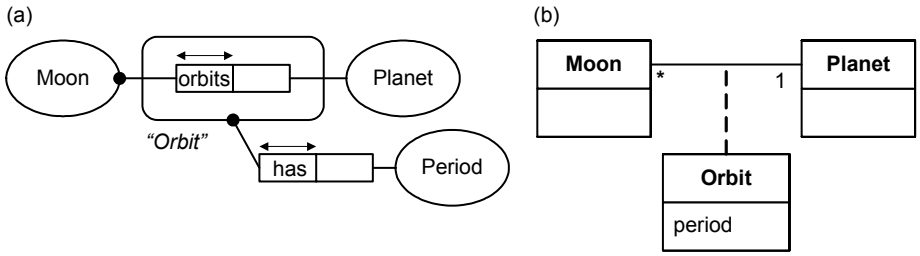

Figure 4 Nested approach to modeling some orbital details in (a) ORM and (b) UML.

The corresponding, flattened schemas for this example are shown in Figure 5. Because each moon orbits only one planet, we lose no information by attaching the orbital period fact type directly to Planet. For clarification purposes, a role name has been added to the ORM schema and a predicate name to the UML schema. It might be argued that the nested version in Figure 4 better represents the close semantic connection between the two fact types (they both relate to orbits). In Figure 5(a), this semantic affinity is formally suggested by the implicit equality constraint between the first roles of the two fact types (implied by the mandatory constraints), and informally suggested by the inclusion of "orbits" within both the predicate readings. A similar comment can be made about the UML schema. This is a minor argument for allowing the nested version.

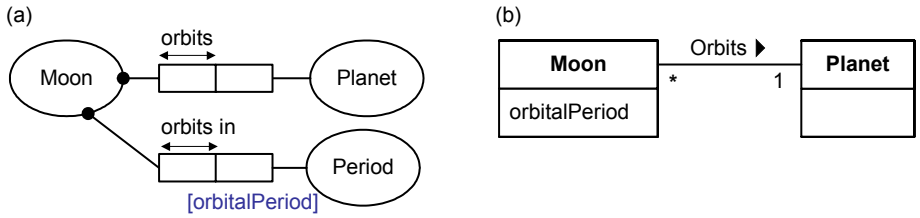

Figure 5 Flattened version of the orbital details example.

The main argument however is based on semantic stability. Suppose that tomorrow we discover in some solar system a moon that orbits two planets, as depicted in Figure 6, where we can sensibly talk of the moon having two orbits (one for each planet), possibly with different orbital periods. To cater for this possibility, we might then weaken the single-role uniqueness constraint on Moon orbits Planet to a spanning uniqueness constraint.

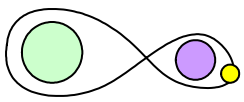

Figure 6 A moon orbiting two planets.

This simple change is all that is required to modify the nested schemas to cater for the new possibility, as shown in Figure 7. Any previous data population is also retained; we simply add the new facts into the same structures. However, because the orbital period is now a function of the orbit only rather than the moon, the flattened version requires a drastic change, since the period fact type must now be restructured to the nested version or some equivalent schema (e.g. absorbed into a ternary fact type).

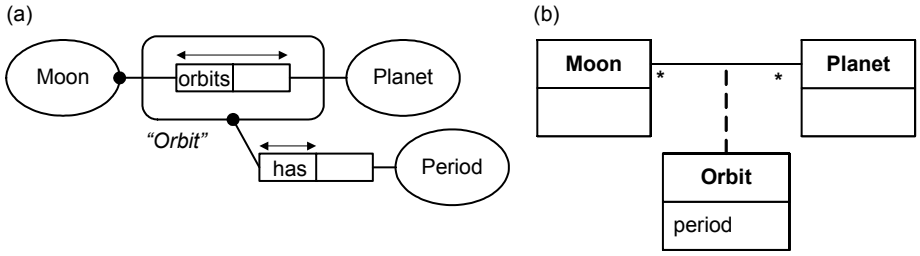

Figure 7 The nested approach is easily adapted by weakening a uniqueness constraint.

While the example above may seem rather improbable, it is easy to imagine more mundane cases where we might decide to relax a uniqueness constraint. In consideration of the above, the following heuristic seems appropriate for nesting of *n*:1 associations.

- *Do not objectify an n:1 association unless it may weaken to an m:n association at a later time.*

Objectifying *n*-ary associations whose longest uniqueness constraint spans *n*-1 roles

In an earlier paper [2], I argued that we should allow nesting of 1:1 associations to avoid making arbitrary (and perhaps politically incorrect) preferences at the conceptual level. For example, if we were not to objectify the current marriage fact type shown in Figure 8, we would need to introduce Husband and Wife subtypes of Person and then make an arbitrary choice as to which of these subtypes should have the marriage date fact type attached.

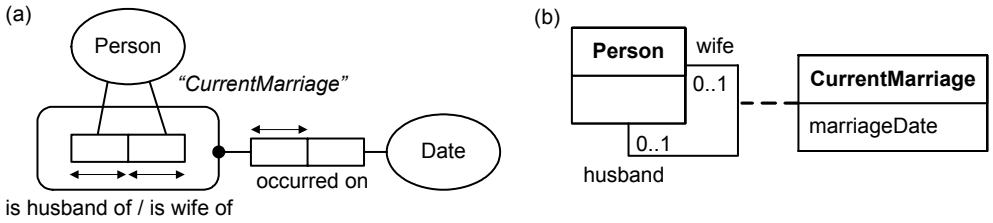

Figure 8 Nesting of 1:1 associations in allowed in both (a) ORM and (b) UML.

The approach I am about to introduce for *n*-ary fact types ($n > 2$) may be regarded as a logical extension of the rule for the 1:1 case. Consider the example shown in Figure 9. The objectified, ternary association in Figure 9(a) has two overlapping uniqueness constraints, each spanning *n*-1 (in this case 2) roles. The constraint on the first two roles of the ternary indicates that within any given sport any given athlete has at most one current rank. The constraint on the last two roles indicates that within any given sport any given rank is currently assigned to at most one athlete (i.e. no ties are allowed). For each current ranking of an athlete in a given sport, we record the date on which this ranking was awarded, and optionally the achievements on which this award was based.

Figure 9(b) depicts the same business domain in UML, where the ternary association is objectified as an association class, the award date fact type is modeled as an attribute of this class, and the achievement fact type is modeled as a separate association.

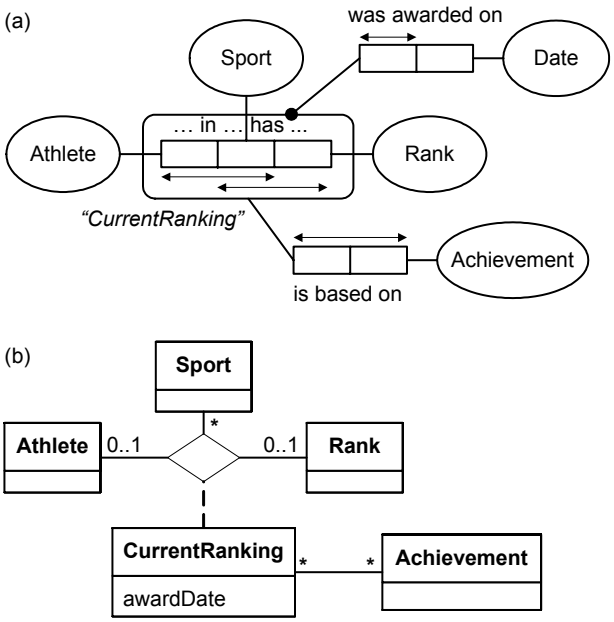

Figure 9 Objectifying a ternary association with overlapping uniqueness constraints.

The schema in Figure 9(a) is currently illegal in ORM, because the objectified relationship is not 1:1 and it has no spanning uniqueness constraint. To transform it into a legal ORM schema, there are two options based on which uniqueness constraint in the ternary is chosen for the objectification. The option shown in Figure 10 uses the uniqueness constraint spanning the roles played by Athlete and Sport in the ternary as the basis for objectification. This model is legal, but slightly more complex because the second uniqueness constraint on the former ternary is now specified as an external uniqueness constraint.

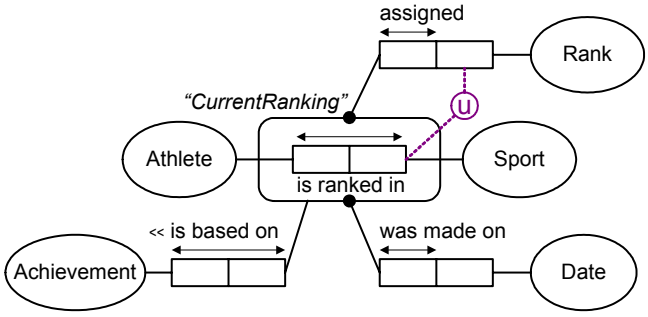

Figure 10 Objectifying a binary based on the first uniqueness constraint in the former ternary.

The alternative shown in Figure 11 uses the uniqueness constraint spanning the roles played by Rank and Sport in the ternary as the basis for objectification. Structurally, there is no formal feature in the schema of Figure 9(a) that determines which of the schemas in Figure 10 and Figure 11 is preferable. Pragmatically, we would often prefer the schema in Figure 10 if in the global schema Athlete played other roles, and Rank played no other roles. In such a case, the UML schema might depict the Rank fact type using an attribute. Suppose however that Rank does play other roles, and that at most one other role is played by Athlete. For example, we might normally use numbers (1, 2, 3,…) to identify ranks but also record rank names (e.g. "Top", "Runner-up", "Third", …). In this case, there seems to be no formal way of deciding which of the schemas in Figure 10 and Figure 11is preferable.

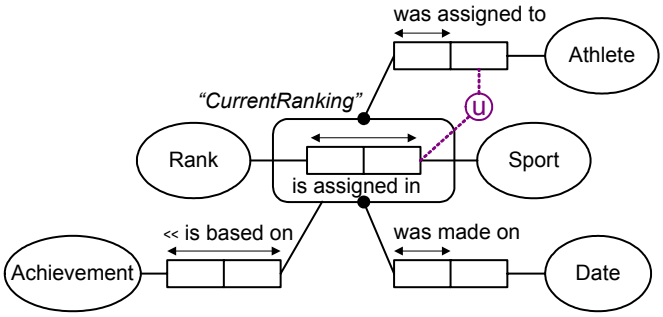

Figure 11 Objectifying a binary based on the second uniqueness constraint in the former ternary.

One can imagine other cases where the decision of which binary nesting to choose is arbitrary. For example, replace Rank by Prize, where the same prize (e.g. a car or trip) may be awarded in many sports, and many details (monetary value etc.) may be recorded for prizes. In such cases, it seems reasonable to allow schemas like that in Figure 9(a), simply to avoid forcing the modeler to make an arbitrary choice of which uniqueness constraint to use as the basis for nesting. The 1:1 case considered earlier may now be seen as a special case of a predicate with multiple uniqueness constraints offering alternative candidates for nesting.

The other case where it makes sense to allow objectification of *n*-ary associations that have no spanning uniqueness constraint is where it is possible that the uniqueness constraint pattern on the association may change to a spanning uniqueness constraint at a later time. The arguments for this allowance are analogous to the arguments presented in the previous section for binary associations.

Conclusion

The foregoing discussions may be summarized by the following modeling heuristic:

> *Do not objectify an association with no spanning uniqueness constraint unless*
> *its uniqueness constraint pattern may weaken to a spanning uniqueness constraint later*
> *or*
> *it has at least two uniqueness constraints, and there is no obvious choice as to which of these is the best basis for objectification.*

Objectification of associations with no spanning uniqueness constraint in cases not covered by these exceptions is likely to be bad modeling. A modeling tool could assist by reminding the modeler of the above heuristic whenever the modeler objectifies a fact type with no spanning uniqueness constraint. If you have any constructive feedback on this article, please e-mail me at: thalpin@comcast.net.

References

1. Batini, C., Ceri, S. & Navathe, S. B. 1992, *Conceptual Database Design*, Benjamin/Cummings, Redwood City.
2. Halpin, T.A. 1993, 'What is an elementary fact?', *Proc. First NIAM-ISDM Conference*, eds G. M. Nijssen & J. Sharp, Utrecht, (Sep), 11 pp. Online at http://www.orm.net/pdf/ElemFact.pdf.
3. Halpin, T. A. 1998 (revised 2001), 'Object Role Modeling: an overview'. Online at http://www.orm.net/pdf/ORMwhitePaper.pdf.
4. Halpin, T.A. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann Publishers, San Francisco.
5. Halpin, T.A., Evans, K, Hallock, P. & MacLean, W. 2003, *Database Modeling with Microsoft® Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
6. Object Management Group 2003, UML 2.0 Specification (see Infrastructure and Superstructure documents). Online at http://www.omg.org.