Verbalizing Business Rules: Part 2

Terry Halpin
Northface University

Regardless of how they are implemented, business rules need to be validated by business domain experts, and thus should be specified at the conceptual level, using concepts and languages (textual or graphical) easily understood by the subject matter experts. This is the second in a series of articles on expressing business rules formally in a high-level, textual language. The previous article [4] identified basic criteria for a business rules language, and discussed how to verbalize simple uniqueness and mandatory constraints on binary associations. The current article shows how to refine some constraint verbalizations by using hyphens to bind adjectives to object terms, and then examines verbalization of internal uniqueness constraints that either span a whole association, or apply to n-ary associations.

Hyphen binding

The ORM diagram in Figure 1 includes three binary associations: Person has first- GivenName; Person has second-GivenName; Person was born on Date. Notice that a *hyphen* "-" is appended to the adjectives "first" and "second". This causes these words to be bound to the following object terms when constraints are verbalized. For example, the mandatory and uniqueness constraints on the association Person has first-GivenName may be verbalized separately thus:

> **Each** Person has **at least one** first GivenName.
> **Each** Person has **at most one** first GivenName.

or in composite form thus: **Each** Person has **exactly one** first GivenName.
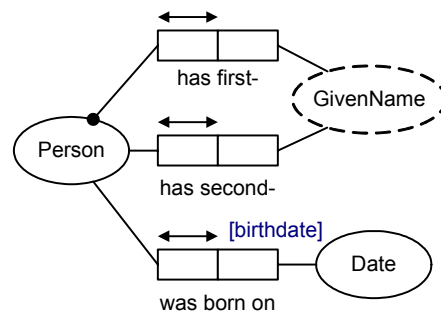

Figure 1   Hyphens may be used to bind adjectives to object terms in verbalizations.

When used as an *input* language, the textual verbalization (with no hyphen) may be parsed to apply the relevant quantifier to the following object term. For example, the composite formulation above may be interpreted as $\forall x$:Person $\exists!y$:GivenName x has first y. However, when generating verbalizations in an *output* language, the predicate must include a hyphen to bind the adjective to the object term, so that the quantifier verbalization is placed *before* the adjective, not after it. For example, if the hyphen is omitted from the first givenname predicate, then the automated verbalization of the mandatory and uniqueness constraints generates the following unsatisfactory formulations: **Each** Person has first **at least one** GivenName; **Each** Person has first **at most one** GivenName. The use of a hyphen thus provides a simple means of generating appropriate verbalizations without needing to provide a deep lexical analysis of the internal predicate structure.

Like UML, ORM allows each role in a predicate to be named. For example, in Figure 1 the role played by Date is named "birthdate". While it is more natural to verbalize this birth fact type as shown (Person was born on Date), hyphen binding gives you the choice to rephrase this in "attribute-style" (e.g. Person has birth-Date).

Spanning Uniqueness Constraints

Figure 2 displays a many:many association with optional roles in ORM [3], Barker ER [1], and UML [5] notations. A cardinality of many (zero or more) is depicted in Barker ER by a crowsfoot, and in UML by "*" (or "0..*"). In Barker ER, this is verbalized as: **Each** Person **may be** a visitor of **one or more** Countries, **and each** Country **may be** visited by **one or more** Persons. In ORM, this is depicted as a uniqueness constraint that spans both the roles. This is reflected in the sample population in Figure 2(a), which shows that although a person may appear twice and a country may appear twice, each combination of person and country may appear only once.
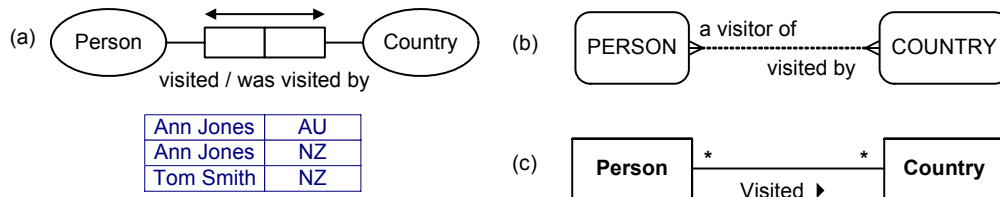

Figure 2   A many-to-many association in (a) ORM (b) Barker ER and (c) UML.

The standard verbalization in ORM for this spanning uniqueness constraint is shown below. This reflects the absence of a single role uniqueness constraint on the first role, and the absence of a single role uniqueness constraint on the second role.

> **It is possible that the same** Person visited **more than one** Country
> **and that the same** Country was visited by **more than one** Person.

While the above verbalizations clarify the many-to-many nature of the association, they do not explicitly capture the further intention that person-country pairs must be unique. In other words, the population of the association must be a *set* of person-country pairs, not a bag (multi-set). For example, the spanning uniqueness constraint would be violated if we added the pair "Tom Smith, NZ" as a fourth row of the fact table.

This further aspect of the constraint is usually left implicit, because it is assumed that the populations of *all* primitive (non-derived) associations must be sets, not bags. If we wish to make this assumption explicit for any given association, we may add a verbalization to that effect. For our current example, this may be verbalized as follows.

> **Each instance of** Person visited Country **occurs only once.**

As good modeling practice, *any spanning uniqueness constraint should be checked with the domain expert to confirm that a bag is not required*. For example, a person may visit a country more than once. If we wish to record whether a person visited a country, but not how many times they did so, then we may use the fact type Person visited Country without further qualification. If however we wish to know whether a person visited a country many times, we need to modify the model. For example, we might objectify the association Person visited Country as Visit, and then add either a many-to-one, frequency fact type such as Visit occurred NrTimes or a many-to-many, temporal fact type such as Visit began on Date. Figure 3 depicts the situation where for each visit, we record how many times it occurred.
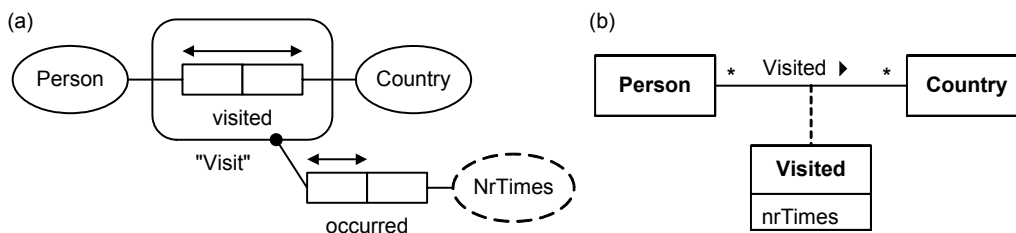

Figure 3   Objectifying an association in (a) ORM and (b) UML.

Clearly it is desirable to use a noun phrase (e.g. "Visit") for the object type created from the association instead of a verb phrase "(e.g. "visited"). This is permitted in ORM, but not UML, which requires the association class to have the same name as the association reading. Industrial versions of ER do not support objectification at all, and hence require Visit to be modeled as a separate entity type that is associated with Person and Country.

If for each visit, we know the number of times or the start dates, then we may also model this situation in ORM or UML using either an m:n:1 ternary association such as Person visited Country NrTimes, or an m:n:p ternary association Person began a visit to Country on Date. Uniqueness constraints on ternary (or longer) associations are discussed in the next section. Other alternatives in ORM include declaring Visit as a co-referenced object type, for example Visit (was by Person, was to Country), using an external uniqueness constraint (see next article in this series). If we wish to use the term "Visit" in the sense of an event that began on just one date, then we may include the start date as part of the identification scheme: Visit (was by Person, was to Country, began on Date). Yet another alternative is to declare Visit as an object type with a simple identifier.

Uniqueness Constraints on n-ary Associations

While industrial ER is restricted to binary associations, both ORM and UML support n-ary associations ($n > 2$). For example, Figure 4 depicts the ternary fact type Room at HourSlot is booked for Course. The ORM model includes two uniqueness constraints, each spanning two roles. The UML model includes equivalent multiplicity constraints.
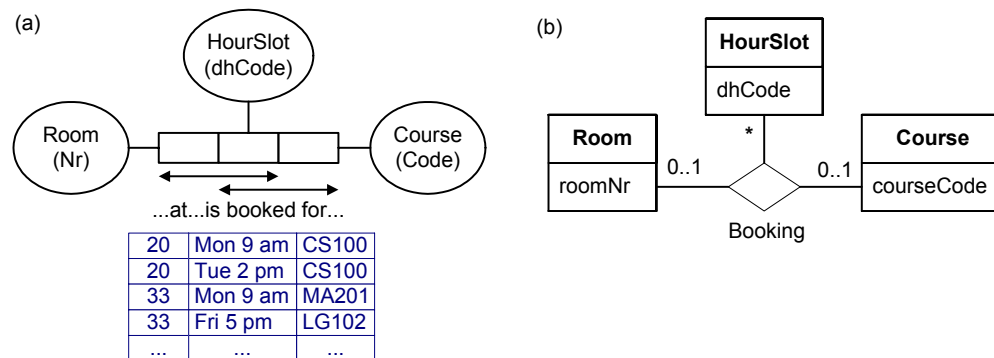

Figure 4   Uniqueness constraints on a ternary association in (a) ORM and (b) UML

Consider first the ORM uniqueness constraint spanning the first two roles of the association. Recall that each role may be associated with a column in the instance table for the fact type. Hence this constraint implies that each Room-HourSlot pair that appears in this table must occur there only once. This is consistent with the sample fact population provided in Figure 4(a). This constraint may be verbalized in positive form thus:

> **Given any** Room **and** HourSlot
>   **that** Room at **that** HourSlot is booked for **at most one** Course

This verbalization may be formalized using the at-most-one quantifier ($\exists$?) thus: $\forall x$:Room $\forall y$:HourSlot $\exists?z$:Course x at y is booked for z. In English verbalizations, the pronoun "**that**" is often used to demarcate the relevant object variable. The constraint may also be verbalized in negative form thus:

> **It is impossible that the same** Room at **the same** HourSlot is booked for **more than one** Course

As explained in the previous article, the negative form is best used in conjunction with counter-examples to check with the domain expert whether the constraint actually applies. For example, if we add the entry "(33, Fri 5 pm, CS200)" as a fifth row to the fact instance table, the combination of rows 4 and 5 present a concrete example of the same room (33) at the same hour slot (5 p.m. Friday) being booked for two different courses (LG102 and CS200). Is this situation possible or not? The negative verbalization stresses that this is impossible.

3

Now consider the uniqueness constraint over the roles played by HourSlot and Course. This verbalizes in positive and negative form as shown below.

+ve: **Given any** HourSlot **and** Course
        at most one Room at **that** HourSlot is booked for **that** Course

−ve: **It is impossible that more than one** Room at **the same** HourSlot is booked for **the same** Course

This second uniqueness constraint is highly debatable, and would not apply in some universities. To test it with a counterexample, we might add the entry "(50, Fri 5 pm, LG102)" as a fifth row to the fact instance table, so that the combination of rows 4 and 5 present a concrete example of two different rooms (33 and 50) at the same hour slot (5 p.m. Friday) being booked for the same course (LG102). If the domain expert says this is possible, we should remove the constraint.

If an n-ary association is elementary, then any uniqueness constraint on it must span at least n-1 roles. If desired, the absence of a uniqueness constraint over some combination of n-1 roles may be explicitly verbalized using default form. For example, the absence of a uniqueness constraint that spans the first and last roles of the association may be verbalized as shown below. This possibility is illustrated by the first two rows of the instance table in Figure 4.

It is possible that the same Room at **more than one** HourSlot is booked for **the same** Course

Now that we have discussed the relevant patterns for declaring a uniqueness constraint over n-1 roles of an n-ary association, we have just one case left to consider: a uniqueness constraint spanning all n roles of the association. Figure 5 shows an example of an n-ary association with such a spanning uniqueness constraint.
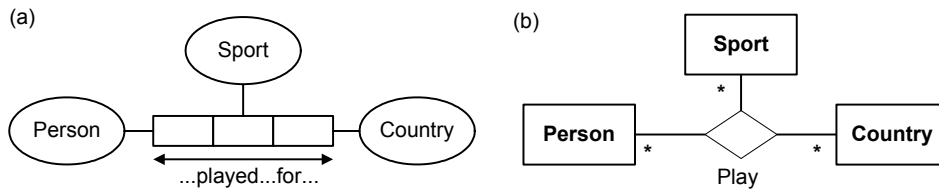

Figure 5   Uniqueness constraint spanning a ternary association in (a) ORM and (b) UML.

The short verbalization of this constraint is shown below, indicating that the population must be a set, not a bag.

Each instance of Person played Sport for Country **occurs only once.**

When validating this constraint with domain experts, it is often safer to use the longer, default form that spells out the absence of uniqueness constraints on each combination of n-1 roles. For an n-ary association, there are n such combinations. For each such possibility, exactly one object type occurrence is preceded by "**more than one**" with the other object type occurrences preceded by "**the same**". In this ternary example, there are three possibilities that need to be checked with the domain expert, as shown in the verbalization below.

It is possible that the same Person played **the same** Sport for **more than one** Country
and that the same Person played **more than one** Sport for **the same** Country
and that more than one Person played **the same** Sport for **the same** Country.

This explicitly declares the absence of a 2-role uniqueness constraint over: (1) the first two roles, (2) the first and third roles, and (3) the second and third roles of the fact type.

If the same object type plays more than one role in an n-ary association, the use of the pronoun "**that**" in verbalizing a constraint may lead to ambiguity. For example, Figure 6(a) shows the quaternary association Person judged Person to have SkillLevel in Language. In UML, this would typically be modeled using a ternary association class with a skill level attribute, as shown in Figure 6(b).
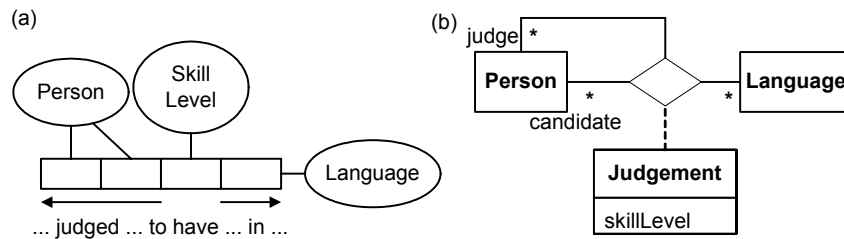
4

Figure 6   An *n*-ary association with more than one role played by the same object type.

The uniqueness constraint in Figure 6(a) spans two roles played by Person and one role played by Language. Verbalizing this constraint in positive form using the pronoun "**that**" leads to the following awkward formulation:

> **Given any** Person, Person **and** Language
>   **that** Person judged **that** Person to have **at most one** SkillLevel in **that** Language**.**

This verbalization is confusing, because it is unclear which person is intended by the separate uses of "that Person". In such cases, it is convenient to append numbered *subscripts* to distinguish the object variables [2]. The constraint may now be verbalized clearly in positive form, as follows.

> **Given any** $Person_1$, $Person_2$ **and** Language
> $Person_1$ judged $Person_2$ to have **at most one** SkillLevel in **that** Language**.**

The verbalization in negative form, while somewhat stilted, is still unambiguous.

> **It is impossible that the same** Person1 judged **the same** Person2 to have **more than one** SkillLevel in **the same** Language**.**

That concludes our discussion of internal uniqueness constraints. The next article will discuss external uniqueness constraints, which apply uniqueness restrictions to combinations of roles projected from different associations.

References

1. Barker, R. 1990, *CASE*Method: Tasks and Deliverables*, Addison-Wesley, Wokingham, England.
2. Bloesch, A. C. & Halpin. T. A. 1997, 'Conceptual queries using ConQuer-II', *Proc. 16th International Conference on Conceptual Modeling ER'97* (Los Angeles), Springer LNCS 1331 (Nov.) 113-126. Available online at http://www.orm.net/pdf/ER97-final.pdf.
3. Halpin, T.A. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
4. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 1', *Business Rules Journal*,
5. Object Management Group 2001, Unified Modeling Language (UML) Specification - Version 1.4, available online at http://www.omg.org/technology/documents/formal/uml.htm.