Verbalizing Business Rules: Part 5

Terry Halpin
Northface University

Business rules should be validated by business domain experts, and hence specified using concepts and languages easily understood by business people. This is the fifth in a series of articles on expressing business rules formally in a high-level, textual language. The first article [3] discussed criteria for a business rules language, and verbalization of simple uniqueness and mandatory constraints on binary associations. The second article [4] examined hyphen-binding, and verbalization of internal uniqueness constraints that span a whole association, or that apply to n-ary associations. The third article [5] covered verbalization of basic external uniqueness constraints. The fourth article [6] considered relational-style verbalization of external uniqueness constraints involving nesting or long join paths, as well as attribute-style verbalization of uniqueness constraints and simple mandatory constraints. This article discusses further aspects of verbalizing mandatory constraints. In particular, it considers verbalization of mandatory constraints on roles of n-ary associations, and disjunctive mandatory constraints (also known as inclusive-or constraints) over sets of roles.

Verbalization of mandatory constraints on roles of n-ary associations

Consider a business that tracks for each sport the current standing of the top-ranked countries in those sports. Two sample reports from this business domain are shown in Figure 1. The domain expert might verbalize the facts on the top row of report (a) thus: Archery is new (it's the first year it's included in the rankings); the US ranks first in archery, and scored 10 points for that. As modelers, we suspect that Rank functionally determines Points in the population, and verify this with the domain expert. We now rephrase the information into elementary facts: the Sport named 'Archery' is new; the Country with code 'US' has the Rank numbered 1 in the Sport named 'Archery'; the Rank numbered 1 earns the Score 10 points.

(a)

Sport	*Rank*	*Country*	*Points*
Archery *	1	US	10
Baseball	1	US	10
	2	JP	5
Cricket	1	AU	10
	1	GB	10
...	...	...	...

* *new*

(b)

Country	
Code	*Name*
AD	Andorra
AE	United Arab Emirates
...	...
ZW	Zimbabwe

Figure 1 Sample reports from a business that tracks leading sport performances.

Similarly, the top row of the reference table shown in report (b) may be verbalized as: the Country with code 'AD' has the name 'Andorra'. If reference schemes are agreed to up front (e.g. Sports are identified by name), these verbalizations may be abbreviated (e.g. the Sport 'Archery' is new).

Once the domain expert agrees with the verbalization, we abstract from the fact instances to the fact types, and add constraints, validating these with the domain expert by verbalizing them in natural language and checking them against sample fact populations. An ORM conceptual schema for this example, together with sample fact populations, is shown in Figure 2. There is one unary fact type, Sport is new, two binary associations Country has CountryName, Rank earns Score, and one ternary association Country has Rank in Sport.

A sample row for rank 3 has been added to illustrate the mandatory and optional nature of the roles played by Rank (a rank's score must be recorded even if no country achieves this rank—this rule was not evident from the original sample, but is obtained when checking constraints with the domain expert). The uniqueness constraint on the ternary association may be verbalized as "**Given any** Country **and** Sport, **that** Country has **at most one** Rank in **that** Sport", or more briefly as "**Each** Country has **at most one** Rank in **each** Sport". This constraint is satisfied by the sample population, where the Country-Sport pairs are unique.

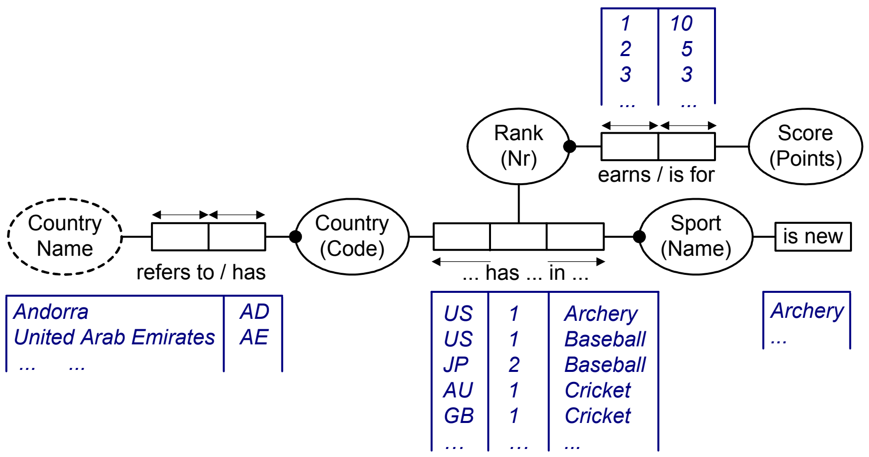

Figure 2 An ORM schema for Figure 1, with sample fact populations.

The uniqueness and mandatory role constraints on the binary fact types are easily verbalized using patterns discussed in earlier articles. Of special interest here is the mandatory constraint on the last role of the ternary fact type. This constraint may be verbalized as follows.

For each Sport, **some** Country has **some** Rank in **that** Sport.

As usual, "**Given any**" may be used instead of "**For each**", and "**at least one**" or "**a**" may be used instead of "**some**". In cases where the same object type plays more than one role in the fact type, subscripts may be used to distinguish the different role players (as discussed in earlier articles).

An ORM diagram shows only one predicate reading for each n-ary association, and this reading is understood to go from one end to the other (left-to-right or top-to-bottom, unless prepended by "<<", which reverses the reading direction). However, predicate readings may be supplied in the textual version of the schema for any permutation of the roles, and these readings may be used in alternative verbalizations. For example, the ternary fact type Country has Rank in Sport in Figure 2 may be given alternative readings such as Rank in Sport is held by Country, or Rank is held by Country in Sport, or Sport has Rank held by Country. If the mandatory constraint applies to the *first* role of such a reading, then the mandatory constraint can be verbalized a little more tidily. For example, the constraint considered earlier may be verbalized thus:

Each Sport has **some** Rank held by **some** Country.

As industrial ER is restricted to binary associations, I'll ignore it for this example. UML does support n-ary associations, but has problems with declaring mandatory constraints on them. For example, suppose we model the example under discussion using the UML class diagram shown in Figure 3. As discussed in earlier articles, "{P} and "{U1}" are non-standard extensions I use to declare preferred identifier and uniqueness constraints on UML attributes.

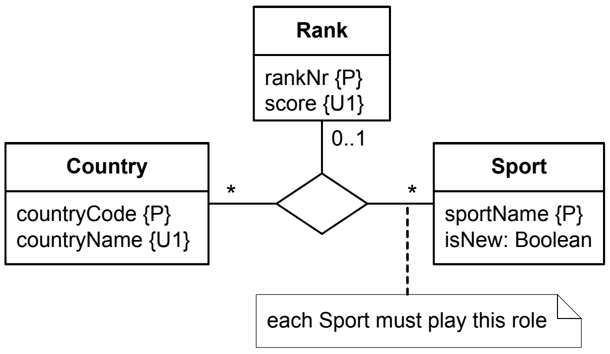

Figure 3 UML schema for Figure 1.

The Country-Sport uniqueness constraint in the ORM ternary is modeled in UML using the 0..1 multiplicity constraint on the role played by Rank. The "*" multiplicities indicate the absence of any other uniqueness constraint. However, the simple mandatory role constraint on Sport cannot be expressed by a multiplicity constraint in UML. It might be thought that this constraint can be expressed by changing the multiplicity on the Country role to 1..*. But this would mean that each Sport-Rank pair formed from the populations of Sport and Rank must be associated with at least one country. But this is not true, since the role played by Rank is optional. For example, the pairs Archery-2 and Archery-3 have no associated country in the sample population.

In UML, a multiplicity constraint on a single role indicates the number of instances (playing that role) that are associated (in the association) with each tuple from the Cartesian product of the populations of the other *n*-1 roles in the association (where $n > 1$). Although this works fine for binary associations, UML multiplicity constraints can specify mandatory role constraints in an *n*-ary association if and only if at least *n*-1 of the association roles are mandatory. *If fewer than n-1 roles are mandatory, this cannot be captured by a multiplicity constraint*. The example under discussion is such a case. In Figure 3, the required mandatory constraint is captured informally in a note. Alternatively, this constraint could be declared formally in OCL [9].

Even for *n*-ary associations where *n*-1 or *n* roles are mandatory, UML cannot express these mandatory constraints graphically without asserting a much stronger constraint (each tuple in the unrestricted Cartesian product of the other role populations must play in the association). While this strong constraint does imply the individual mandatory constraints, it is likely to be far too strong in practice except for pathological cases, or for cases where populations of the object types are tightly restricted by value constraints. In some cases, we can cater for fewer than *n*-1 mandatory roles in UML by binarizing the *n*-ary, but as explained elsewhere [2] this leads to other problems, so does not provide a satisfactory solution.

Note that verbalizing any constraint in relational style on an n-ary association is ruled out for UML, as UML has no way of depicting the role order in which any n-ary association reading is to be understood. A verbalization is possible however, so long as we have a name for the association. For example, if we assign the name "Ranking" to the ternary association in Figure 3, the mandatory constraint may now be verbalized thus:

Ranking **is mandatory for** Sport.

Verbalization of inclusive-or (disjunctive mandatory) constraints

Consider the employee report extract shown in Table 1. The "?" mark denotes a null value. Employees are identified by their employee number, and their birthdate must be recorded. In addition, each employee must have a social security number (SSN), or a driver license number (possibly both)—this restriction is an example of an *inclusive-or constraint* (or *disjunctive mandatory constraint*).

Table 1 Extract of Employee details.

EmployeeNr	*BirthDate*	*SSN*	*DriverLicenseNr*
001	01/01/1970	539-01-2345	?
002	11/30/1980	?	170001111
003	01/01/1970	123-45-6789	160101234
…	…	…	…

Inclusive-or constraints often occur in practice, and are supported by ORM's graphical notation. UML and commercial versions of ER include graphical notations for exclusive-or constraints (see later article), but not inclusive-or constraints; in which case, they need to be added textually, either informally as a note or formally in a constraint language such as OCL. Whichever modeling notation is used, inclusive-or constraints should be verbalized in a high level language.

Figure 4(a) models this example in ORM. Here DriverLicense is modeled as an entity type, allowing other information to be recorded about it if desired (e.g. license class, issue date, expiry date). The inclusive-or constraint is depicted by the circled dot, connected to the roles to which it applies. The circled dot is just a mandatory constraint dot, and connecting it to two roles indicates that at least one of these roles must be played by each instance in the population of the Employee entity type. In other words, the logical disjunction of these roles is mandatory for Employee (hence the name "disjunctive mandatory constraint").

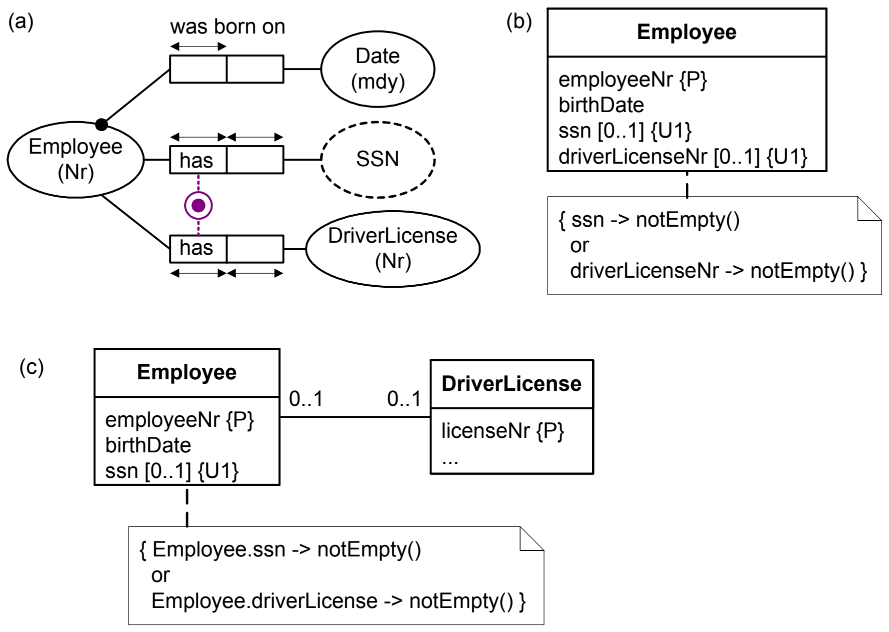

Figure 4 Inclusive-or constraint in ORM and UML.

The usual ORM verbalization of this constraint is:

Each Employee has **some** SSN **or** has **some** DriverLicense.

This can be rendered more naturally by using "**a**" or "**an**" instead of "**some**", and by eliding the second occurrence of "has", to give:

Each Employee has **an** SSN **or a** DriverLicense.

If the predicate names differ however, both must be included, as in the first version. The verbalization would be even better if the full name "SocialSecurityNumber" had been used instead of "SSN" in the model—in this case, "**a**" would be used instead of "**an**" as the rendition of the existential quantifier.

Figure 4(b) models the example in UML, treating the driver license fact type up as an attribute of Employee. In this case, the constraint is formally captured in text as an OCL expression. Figure 4(c) provides an alternative UML model, using a DriverLicense class to allow further details to be stored about driver licenses. Again, an OCL expression is used to capture the constraint formally. Because of the mathematical nature of OCL syntax, a high level verbalization is still required for validating the constraint with non-technical domain experts. In this case, a relational-style verbalization may be used, using the attribute or role names as property names, with a "has" predicate assumed for each. For Figure 4(b) this yields:

Each Employee has **some** ssn **or some** driverLicenseNr.

and for Figure 4(c), we have:

Each Employee has **some** ssn **or some** driverLicense.

Again, "**a**" or "**an**" may be used instead of "**some**". Inclusive-or constraints may apply to a set of two or more roles, so long as those roles are played by compatible object types. For more complex cases involving roles with no starting predicate reading, and possibly from n-ary fact types, a more generic pattern may be used. For example: **each** Employee **plays at least one of the following roles**: Employee has SSN; Employee has DriverLicense.

That completes our coverage of verbalization of uniqueness and mandatory constraints. The next article discusses verbalization of value constraints.

References

1. Halpin, T. A. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
2. Halpin, T.A. 2001, 'Supplementing UML with concepts from ORM', *Unified Modeling Language: Systems Analysis, Design and Development Issues*, eds K. Siau & T. Halpin, Idea Group Publishing, Hershey PA, USA, pp. 168-185.
3. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 1', *Business Rules Journal*, Vol. 4, No. 4 (April 2003), URL: http://www.BRCommunity.com/a2003/b138.html.
4. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 2', *Business Rules Journal*, Vol. 4, No. 6 (June 2003), URL: http://www.BRCommunity.com/a2003/b152.html.
5. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 3', *Business Rules Journal*, Vol. 4, No. 8 (August 2003), URL: http://www.BRCommunity.com/a2003/b163.html.
6. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 4', *Business Rules Journal*, Vol. 4, No. 10 (October 2003), URL: http://www.BRCommunity.com/a2003/b172.html.
7. Halpin, T., Evans, K., Hallock, P. & MacLean, B. 2003, Database Modeling with Microsoft Visio for Enterprise Architects, Morgan Kaufmann, San Francisco.
8. Object Management Group 2003, *UML 2.0 Infrastructure*, URL: http://www.omg.org/uml.
9. Object Management Group 2003, *UML 2.0 Object Constraint Language*, URL: http://www.omg.org/uml.