# Verbalizing Business Rules: Part 4

*Terry Halpin*
*Northface University*

Business rules need to be validated by business domain experts, so should be specified using concepts and languages easily understood by business people. This is the fourth in a series of articles on expressing business rules formally in a high-level, textual language. The first article [4] discussed criteria for a business rules language, and verbalization of simple uniqueness and mandatory constraints on binary associations. The second article [5] discussed hyphen-binding, and verbalization of internal uniqueness constraints that span a whole association, or that apply to n-ary associations. The third article [6] discussed verbalization of basic external uniqueness constraints. The current article discusses 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.

## Verbalization of external uniqueness constraints involving nesting

Consider the output report shown in Table 1. This table is used to record which countries play which sports, as well as the ranking assigned (if known). In the sample population, Australia and Great Britain are ranked first and second respectively in cricket, and the United States is ranked first in baseball. We record the fact that New Zealand plays cricket, but we don't yet know its ranking in that sport ("?" denotes a null value).

**Table 1** Sport Rankings of Countries.

| *Country* | *Sport* | *Rank* |
|---|---|---|
| AU | Cricket | 1 |
| GB | Cricket | 2 |
| NZ | Cricket | ? |
| US | Baseball | 1 |

This situation may be schematized in ORM as shown in Figure 1. Here the fact type Country plays Sport is objectified as Play, and this nested object type is then used in the fact type Play achieved Rank. The exclamation mark in "Play !" declares the Play object type to be independent (i.e. instances of it may exist without playing other roles—the achieved role is optional).

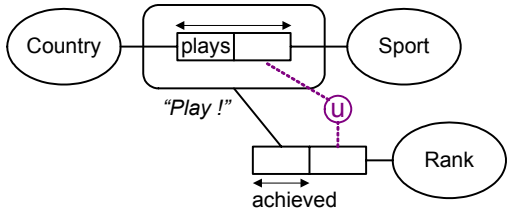


**Figure 1** The external uniqueness constraint forbids ties.

The internal uniqueness constraints indicate that the Play association is many:many, and the achieved association is many:1. The external uniqueness constraint declares a "no-ties" rule, meaning that for any given sport and (non-null) rank, at most one country that plays that sport achieved that rank. For example, the rank of first in cricket can't be assigned to both Australia and Great Britain. In most cases where an external uniqueness constraint involves nesting, the constraint can be verbalized naturally as above. But this assumes the verb phrase in the outer fact type also makes sense when applied to just the target object type (here Country) in the inner fact type, and that the inner fact type includes a reading from that object type. A general verbalization technique that makes no such assumptions can be provided by pre-declaring the *context* in which the constraint is to be interpreted. For example:

**Context**: Country plays Sport **is objectified as** Play.
Play achieved Rank.
**Constraint**: **each** Sport, Rank **combination is associated with at most one** Country.

The UML class diagram shown in Figure 2 models the same situation. UML has no graphic symbol for an external uniqueness constraint, so this constraint is expressed here as an informal comment in a note attached to the model elements relevant to the constraint. Alternatively, it could be expressed formally in OCL [9], but the OCL syntax is currently too technical for reliable validation by business people.

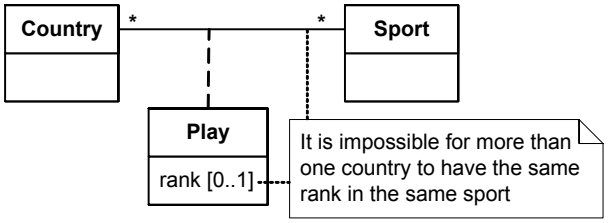


**Figure 2** The no-ties constraint expressed informally as a note in UML.

## Verbalization of external uniqueness constraints involving long join paths

Figure 3 shows an ORM schema fragment for an application dealing with international direct-dial phone numbers for cities. The "<<" indicators reverse the normal downwards/rightwards reading direction of predicates. In navigating ORM schemas, a *conceptual join* or object join occurs when passing through an object type, because the object playing the entry role is the same object playing the exit role [3]. The two roles (colored yellow) spanned by the right-most external uniqueness constraint are projected from a role path that involves three conceptual joins, on Country, State and City. Since all the join roles (shaded green) are mandatory, the joins are all inner joins.

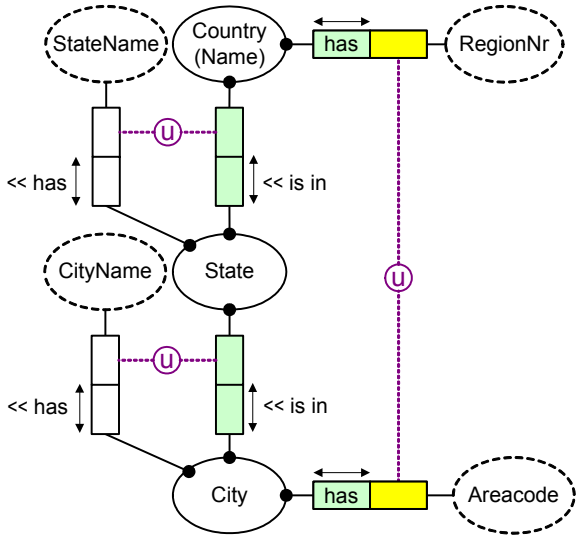


**Figure 3** This model includes an external uniqueness constraint over roles projected from a long join path.

Conceptual joins needn't entail relational joins, since relations may combine multiple fact types. For instance, the ORM schema maps by default to the relational schema shown in Figure 4. At this level, the external uniqueness constraint is enforced by naturally joining the tables on countryName and then ensuring that the regionNr, areaCode bag-projection is unique (a constraint easily coded in SQL). The conceptual joins on State and City come for free at the relational level, since the relevant roles map to the same table.

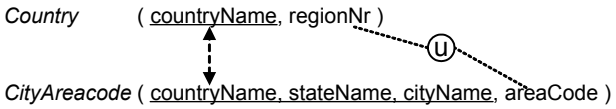


**Figure 4** Relational schema mapped from the conceptual schema in Figure 3.

Here the region number usually identifies a country (e.g. 61 for Australia) but not always (e.g. the United States and Canada both have the region number 1). For the purposes of this example, let's simplify reality a little by assuming that within a given country, the area code identifies a city (e.g. regionNr = 61, areaCode = 7 identifies the Australian city 'Brisbane'). As with the previous example, verbalization of an external uniqueness constraint over a long join path is easier if a context has been pre-declared. In this case, the constraint verbalizes as follows.

**Context**: City has AreaCode
**and** is in State **that** is in Country **that** has RegionNr.
**Constraint**: **each** RegionNr, AreaCode **combination is associated with**
**at most one** City **and at most one** State **and at most one** Country

In this verbalization, the "at most one" constraints on State and Country could be elided because they are implied by simple uniqueness constraints (e.g. **each** City is in **at most one** State), but the verbalization is designed to handle the general case where such additional constraints might not apply. As neither ER nor UML include the notion of join constraints, examples in those notations will not be given.

## Attribute-style Verbalization of Uniqueness and Simple Mandatory Constraints

Figure 5 depicts a simple schema in both ORM and UML notations. Previously, all constraints have been verbalized in *relational-style*, using predicate readings. We now consider how to verbalize uniqueness constraints in *attribute-style* for binary fact types (possibly represented as attributes). In UML models, attribute-style rules may make use of both attribute names and association end names. In ORM, which makes no use of attributes as base constructs, attribute-style rules make use of role names (shown here in square brackets). Here the role name "natives" means persons born in that country.

While the singular or plural nature of the chosen role names or attribute names may help indicate to readers whether certain uniqueness constraints apply, these are merely informal hints, which are potentially unstable (owing to schema changes) and which may be of little use in other natural languages (e.g. Japanese) that often fail to distinguish between singular and plural.

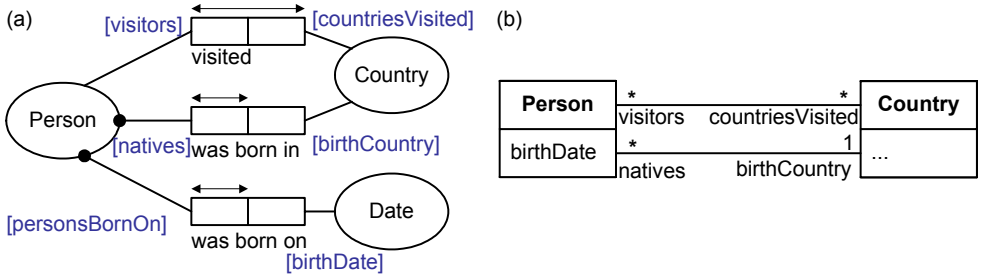


**Figure 5** A simple schema in (a) ORM and (b) UML.

If an object type (e.g. Person) is assigned an attribute (e.g. birthDate), this may be viewed as a *property* of the object type. For any binary association of the form *A r1 r2 B*, where *A* and *B* are object types (not necessarily distinct) and *r1* and *r2* are the roles of the predicate read from *A* to *B*, then *r2* is a *far role* of *A*, and *r1* is a far role of *B*. For example, in the association Person visited Country, countriesVisited is a far role of Person, and visitors is a far role of Country. For attribute-style verbalization, the far roles of an object type may also be treated as properties.

There are two basic ways that property names may be used in verbalization. The first way is to assume an underlying association of the form *ObjectType has Property* (e.g. Person has birthDate, Person has countriesVisited), and then use relational-style verbalization of the constraints in conjunction with quantifiers (e.g. **zero or more, at least one, at most one, exactly one**). As usual, the phrase "**exactly one**" is shorthand for "**at least one and at most one**". Using this technique, the presence or absence of uniqueness and mandatory constraints for Figure 5 may now be verbalized in attribute-style as follows.

**Each** Person has **zero or more** countriesVisited.
**Each** Person has **exactly one** birthCountry.

**Each** Person has **exactly one** birthDate.
**Each** Country has **zero or more** visitors.
**Each** Country has **zero or more** natives.
**Each** Date has **zero or more** personsBornOn.

In UML, Date would normally be considered a data type, on which the attribute birthDate is defined, and an attribute name such as "personsBornOn" would not be supplied. However the above "constraint" stated explicitly for Date is implicit in UML.

The second way to verbalize constraints in attribute-style is to pre-declare the object type as the context (using either "**For each**" or "**Context**:"), and then constrain each of its properties using predefined unary predicates (e.g. **is unique**). Using this second technique, the presence or absence of uniqueness and mandatory constraints for this example may be verbalized as follows.

**For each** Person
    countriesVisited **is multi-valued**
    birthCountry **is unique**
    birthDate **is unique**
    countriesVisited **is optional**
    birthCountry **is mandatory**
    birthDate **is mandatory**

**For each** Country
    visitors **is multi-valued**
    natives **is multi-valued**
    visitors **is optional**
    natives **is optional**

If desired, the constraints from the viewpoint of Date may be verbalized in ORM explicitly as:

**For each Date**
    personsBornOn **is multi-valued**
    personsBornOn **is optional**

Attribute-style verbalization for simple internal constraints is often less natural than relational-style. However attribute-style notation is useful for providing compact verbalization of some complex constraints, as well as derivation rules that rely on mathematical computation (see later articles).

To illustrate the utility of attribute-style for some constraints, consider Figure 6. The external uniqueness constraint was verbalized in relational-style in [6] as follows:

**Given any** $Date_1$ **and** $Date_2$
    **there is at most one** Period **that** began on $Date_1$
                **and** ended on $Date_2$.

In this example, we have added role names in the ORM schema for startDate and endDate. If desired, these names could be used in placed of "$Date_1$" and "$Date_2$" in the relational-style verbalization. The external uniqueness constraint tells us that each combination of startDate and endDate applies to at most one Period. The attributes corresponding to these roles appear in the extended UML class diagram, where the constraint under consideration is expressed by appending the non-standard notation "{U1}" after the attributes to indicate these are subject to the same uniqueness constraint.

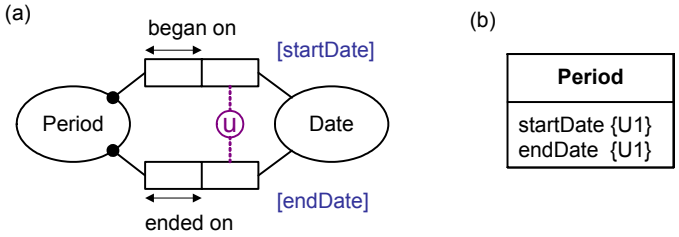


**Figure 6** An external uniqueness constraint in (a) ORM and (b) an extended version of UML.

Regardless of which notation is used, the external uniqueness constraint may be verbalized in attribute-style as follows.

**At most one** Period has **the same combination of** startDate **and** endDate.

That completes our basic coverage of verbalization of uniqueness and simple mandatory constraints. The next article discusses advanced 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.

*References*


1. Barker, R. 1990, *CASE*Method: Tasks and Deliverables*, Addison-Wesley, Wokingham, England.
2. Halpin, T. A. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
3. Halpin, T. A. 2002, 'Join Constraints', *Proc. EMMSAD'02: 7th Int. IFIP WG8.1 Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Toronto (June, 2002), available online at http://www.orm.net/pdf/JoinConstraints.pdf.
4. 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.
5. 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.
6. 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.
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.