# Declarative Specification of Taxonomic Constraint Enforcement in Conceptual Schemas[1]

Dolors Costal, Cristina Gómez and Ernest Teniente

Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemas Informàtics
Jordi Girona 1-3 E08034 Barcelona (Catalonia)
{dolors|cristina|teniente}@lsi.upc.edu

**Abstract.** We propose to declaratively specify policies for the enforcement of taxonomic integrity constraints directly in the structural conceptual schema. These policies depend on the kind of constraint to be enforced (disjointness, covering or specialization) and on the particular event that may cause its violation. Our work eases conceptual modelling since defining taxonomic constraint enforcement declaratively allows omitting its specification from the external events in the behavioural conceptual schema.

## 1. Introduction

An information system maintains a representation of the state of a domain in its information base (IB). The structural conceptual schema defines the structure of the IB while the behavioural schema defines how the IB changes when events occur.

Taxonomies are fundamental constructs of structural schemas [7]. A taxonomy consists of a set of entity types and a set of specialization relationships among them. Usually, taxonomies also include other constraints like disjointness and covering.

Integrity enforcement ensures that constraints are satisfied after each update of the IB. This may be achieved either by integrity checking, which rejects any update leading to an inconsistent state of the IB; or integrity maintenance, aimed at repairing constraint violations by performing additional updates that maintain the IB consistent.

The classical approach to specify how to perform integrity enforcement in conceptual modelling has been to spread the enforcement of those constraints among the various events of the behavioural schema that may violate the constraint. Apart from increasing the difficulty of the conceptual modelling task (since the designer must be able to determine all events that may violate each constraint), this approach has a negative impact on the understandability and modifiability of the resulting conceptual schemas (since integrity enforcement is not localised in a single place).

The approach we propose in this paper overcomes the previous limitations by specifying taxonomic constraint enforcement declaratively (without the need to know which events may violate a given constraint) and in a localised way in the structural conceptual schema. We define a set of predetermined policies for each taxonomic constraint, which depend on the type of change that causes the constraint violation, and that can be easily specified by the designer. As a consequence, the definition of external events becomes simpler because only its intended effect must be considered and taxonomic integrity enforcement actions or conditions may be omitted.

[1] Work supported by the Ministerio de Educación y Ciencia under project TIN 2005-06053

Declarative constraint enforcement has mainly been addressed at a database level. For relational databases, [1,5] support declarative enforcement specification for a broad spectrum of constraints. For object oriented databases, [2] proposes to extend the ODMG Object Model to specify declaratively referential and composite objects constraint enforcement. [4] describes high-level abstractions defining how a database can be made consistent when an update exception occurs. At a conceptual level, [8] studies constraint enforcement for formal oriented specifications, however it does not address the problem of providing declarative policies for integrity enforcement.

Specifying constraint enforcement at a conceptual level has the following advantages: 1) it is specified at the early stages of information systems development, 2) in a technological-independent way, and 3) it allows automatic generation of the integrity enforcement procedures.

## 2. Basic Concepts

Structural conceptual schemas define the structure of the IB and its constraints. We represent by *E(e,t)* the fact entity *e* is instance of entity type *E* at time *t*. Taxonomies of entity types consist of a set of entity types and a set of specialization relationships among them. A specialization is a relationship between two entity types *E'* and *E*, that we denote by *E' IsA E*. A specialization implies a taxonomic specialization constraint between the populations of both types.

Generalization and specialization are different viewpoints of the same *IsA* relationship. We denote by *E Gens* $E_1,...,E_n$ the generalization of entity types $E_1,...,E_n$ to *E*. For instance, in Fig. 1, *Employed Gens Temporary, Permanent*.

Behavioural conceptual schemas define how the IB changes when events occur. External events cause changes in the state of the domain and, consequently, in the IB [6]. These events can be defined precisely in terms of a more basic concept, a structural event. A structural event causes an elementary change in the IB. We use two structural event types, to update the contents of IB taxonomies: entity insertion, denoted by *Insert_E(x,t)*, and entity deletion, denoted by *Delete_E(x,t)*.

An external event consists of a set of structural events which causes composite changes in the IB. In Fig. 1, a *Substitution* external event *{Insert_Employed(Maria,2), Delete_Employed(Pere,2)}* occurring at time *2* has the intended effect of replacing an employee of the company (*Pere*) for another person (*Maria*) previously unemployed.

## 3. Specifying taxonomic constraint enforcement policies

Our proposal for taxonomic constraint enforcement specification consists of providing, for each type of constraint and type of change that may cause its violation, a set of predetermined policies for integrity enforcement. In the following, we give an intuitive explanation of the policies and, finally, we provide an application example which combines several policies. More details of our proposal can be found in [3].

**Disjointness Constraint.** Given a generalization *E Gens* $E_1,...,E_n$, a disjointness constraint indicates that every instance of *E* at time *t* is instance of at most one $E_i$ at *t*. Therefore, the only type of structural event that may induce a disjointness constraint violation is an entity insertion in one of the subtypes of the generalization. This subtype insertion may come directly from the external event definition or, indirectly, from the application of a previous integrity maintenance policy.

Policy *delete-when-subtype-insertion* is an integrity maintenance policy that repairs the external event by adding a deletion of the involved entity from the subtype of the generalization to which it belonged in the previous time instant. The alternative policy *restrict-when-subtype-insertion* is an integrity checking policy and, like all integrity checking policies, consists of rejecting the external event that violates the constraint.

**Covering Constraint.** The covering constraint for a generalization *E Gens* $E_1,...,E_n$ indicates that every instance of the supertype $E$ at time $t$ is instance of at least one subtype $E_i$ at $t$. Then, there are two types of structural events that may induce a covering constraint violation: an entity insertion in the supertype of the generalization or an entity deletion (a set of entity deletions) from one (some) of the subtypes.

Policy *insert-in-*$E_i$*-when-supertype-insertion* repairs a covering violation induced by an insertion in a supertype by performing an insertion of the involved entity in a particular subtype $E_i$. Policy *restrict-when-supertype-insertion* rejects the event.

In case of a covering violation induced by deletions from subtypes, policy *insert-in-*$E_i$*-when-subtype-deletion* adds an insertion of the involved entity in a particular subtype $E_i$. Policy *delete-when-subtype-deletion* repairs this situation by performing a deletion of the involved entity from the supertype of the generalization. Finally, policy *restrict-when-subtype-deletion* rejects the external event.

**Specialization Constraint.** The specialization constraint for a relationship *E' IsA E* indicates that every instance of the subtype *E'* at $t$ is instance of the supertype $E$ at $t$. This constraint may be violated by two types of structural events: an entity insertion in the subtype of the *IsA* relationship or an entity deletion from the supertype.

Policy *insert-when-subtype-insertion* repairs a specialization violation induced by an insertion in the subtype by adding an insertion of the involved entity in the supertype and *restrict-when-subtype-insertion* rejects the external event in that case.

Policy *delete-when-supertype-deletion* repairs a specialization violation induced by a deletion from the supertype by performing a deletion of the involved entity from the subtype and *restrict-when-supertype-deletion* rejects the external event.

**Application to an example.** Consider the taxonomy of Fig. 1.

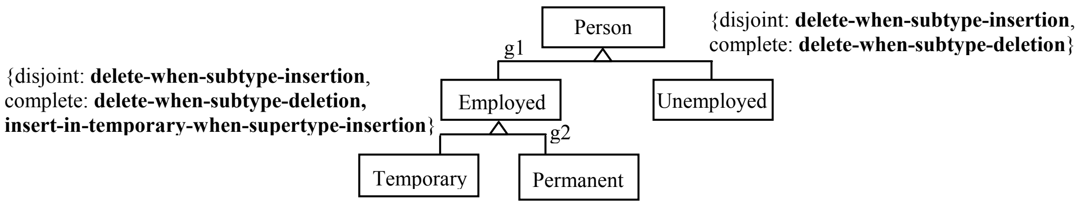
Person
{disjoint: delete-when-subtype-insertion,
complete: delete-when-subtype-deletion}
g1
{disjoint: delete-when-subtype-insertion,
complete: delete-when-subtype-deletion,
insert-in-temporary-when-supertype-insertion}
Employed
Unemployed
g2
Temporary
Permanent

**Fig. 1 Example of taxonomic constraint enforcement policies**

In the following, we show how to repair the *Substitution* external event according to the policies specified in the taxonomy of Fig. 1. Assume an IB at time *1* containing: *{Person(Pere,1), Person(Maria,1), Employed(Pere,1), Permanent(Pere,1), Unemployed(Maria,1)}*. The event induces the following violations: 1) the insertion of *Maria* as employed induces the violation of the covering constraint *g2*. Then, according to its declared policy, *Insert_Temporary(Maria,2)* is included in the external event. 2) The insertion of *Maria* as employed also induces the violation of the disjointness constraint *g1*. Due to its policy, *Delete_Unemployed(Maria,2)* is added. 3) The deletion of *Pere* as employee induces the violation of the covering constraint *g1*. According to the *delete-when-subtype-deletion* policy, *Delete_Person(Pere,2)* is included in the external event. 4) The deletion of *Pere* as employee also induces the violation of the specialization constraint *Permanent IsA Employed*. A default policy, *delete-when-supertype-deletion*, is applied adding *Delete_Permanent(Pere,2)*. Summing up, the resulting external event is: *{Insert_Employed(Maria,2), Delete_Employed(Pere,2), Insert_Temporary(Maria,2), Delete_Unemployed(Maria, 2)*, *Delete_Person(Pere,2), Delete_Permanent(Pere,2)}*.

This example illustrates that the designer may omit the needed repair actions because they can be obtained from the declared enforcement policies.

## 4. Conclusions and future work

We have proposed a set of predetermined policies for the enforcement of taxonomic integrity constraints. These policies are declaratively specified in the structural part of the conceptual schema by stating the kind of enforcement to apply when a certain type of structural event violates a taxonomic constraint.

Our approach facilitates conceptual modelling since taxonomic constraint enforcement must not be spread along all the events that may violate an integrity constraint. Moreover, it has a positive impact on the understandability and modifiability of conceptual schemas. In this way, we overcome the limitations of previous proposals for the specification of taxonomic constraint enforcement.

Further work may include the enforcement of other integrity constraints. We could also define a UML Profile for these policies and incorporate them in a CASE tool.

## References

1. Baralis, E.; Ceri, S.; Paraboschi, S. “Declarative Specification of Constraint Maintenance”, In Proc. ER 1994, LNCS 881, pp. 205-222.
2. Bertino, E.; Guerrini, G. “Extending the ODMG Object Model with Composite Objects”, In Proc. of ACM SIGPLAN, 1998, pp. 259-270.
3. Costal, D.; Gómez, C.; Teniente, E. “Declarative Taxonomic Constraint Enforcement in Conceptual Schemas (Extended Version)”, Technical Report UPC, LSI-05-8-R.
4. Etzion, O.; Dahav, B. ”Patterns of Self-stabilization in Database Consistency Maintenance”, In DKE 1998, pp. 299-319.
5. Gertz, M. “Specifying Reactive Integrity Control for Active Databases”, In Proc. of the IEEE RIDE-ADS, 1994, pp.62-70.
6. Olivé, A. "An Introduction to Conceptual Modeling of Information Systems". In Advance Database Technology and Design,(Piattini,M; Diaz,O. Eds.), Artech House, 2000, pp. 25-57.
7. Olivé, A.; Teniente, E. “Derived Types and Taxonomic Constraints in Conceptual Modeling”, In Information Systems, vol. 27, no. 6, Sept. 2002, pp.365-389.
8. Schewe, K.D.; Thalheim, B. “Towards a Theory of Consistency Enforcement”, In Acta Informatica, vol. 36, 1999, pp. 97-141.