IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA ITB10298
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

Chapter XIII

Constraints on Conceptual Join Paths

Terry Halpin, Northface University, USA

ABSTRACT

To ensure that a software system accurately reflects the business domain that it models, the system needs to enforce the business rules (constraints and derivation rules) that apply to that domain. From a conceptual modeling perspective, many application domains involve constraints over one or more conceptual schema paths that include one or more conceptual joins (where the same conceptual object plays roles in two relationships). Popular information modeling approaches typically provide only weak support for such conceptual join constraints. This chapter contrasts how these join constraints are catered for in object-role modeling (ORM), the Unified Modeling Language (UML), the Object-oriented Systems Model (OSM), and some popular versions of entity-relationship (ER) modeling. Three main problems for rich support for join constraints are identified: disambiguation of schema paths, disambiguation of join types, and mapping of join constraints to implementation code. To address these problems, some notational, metamodel, and mapping extensions are proposed.

INTRODUCTION

At the analysis phase of information system development, a conceptual schema should be used to describe the structure of the application domain in a way that is easily understood and validated by the domain expert. Once validated, the conceptual schema may then be mapped to logical/internal/external schemas using automated and/or manual processes. For industrial database applications,

the high-level data modeling is typically performed using a version of the entity-relationship (ER) modeling approach introduced by Chen (1976), such as Information Engineering (IE), Barker ER, or Integration Definition 1 extended (IDEF1X). Recently, object-role modeling (ORM) and the Unified Modeling Language (UML) class diagrams have gained some industrial adoption for information modeling. In addition, modeling techniques from academia, such as the Object-oriented Systems Model (OSM), can be used to construct information models.

Details on IE, Barker ER, and IDEF1X are provided by Finkelstein (1998), Barker (1990), and NIST (1993) respectively. Halpin (1998a, 1998b) offers overviews of ORM. Booch, Rumbaugh, and Jacobson (1999) present a standard coverage of UML. At the time of writing, the latest adopted version of UML is UML 2.0 (OMG, 2003). Embley (1998), and Embley, Kurtz, and Woodfield (1992) cover OSM in detail.

With increasing competition in the marketplace, and the potential costs of bad data, there has been a growing appreciation of the central role that *business rules* play in enforcing data integrity. To ensure that a software system accurately reflects the business domain that it models, the system needs to enforce the business rules (constraints and derivation rules) that apply to that domain. Since such rules need to be validated with domain experts who may have little knowledge of implementation structures, it is best to capture them first in a conceptual schema where they can be readily communicated.

Most popular modeling approaches provide good support for capturing simple business rules, such as cardinality constraints on associations. In practice, however, many application domains involve business rules that are essentially constraints over one or more conceptual schema paths, each of which involves one or more conceptual joins (where the same conceptual object plays a role in two relationships). Although such *conceptual join constraints* often apply to an application domain, they are often omitted from the data model for the domain, partly because popular modeling approaches typically provide only weak support for such constraints. This makes it harder for the modeler to detect the rules and decide whether to include them in the application. It also makes it harder for the developer who must now code the rules instead of benefiting from automated code generation from a high-level rule specification.

This chapter discusses current support for conceptual join constraints in the approaches listed earlier, and suggests ways to improve the support. Three main problems for rich support for join constraints are identified: disambiguation of schema paths, disambiguation of join types, and mapping of join constraints. To address these problems, some notational, metamodel, and mapping extensions are proposed.

The rest of this chapter is structured as follows. The next section deals with uniqueness and frequency constraints on a single join path. The following section