Implementing Specialisation Hierarchies within Relational Databases

Adrian Hargreaves
School of Information Technology
Whitireia New Zealand
adrian.hargreaves@whitireia.ac.nz

Abstract

The extended entity relationship model (EERM) attempts to capture more semantic detail than is otherwise possible in the standard entity relationship model. In particular, the specialisation hierarchy, which is conceptually similar to the Object Oriented notion of inheritance, is a construct used in EERM for representing entity supertypes and subtypes. Grouping entities into various types based on the generic and unique characteristics they contain, affords the benefit of avoiding null values in the resulting table structures. The approach also allows relationships that are unique to a particular subtype to be modelled. However, it is the author's experience that while many course texts provide examples of how these specialisation hierarchies can be logically modelled, little is said on how they can be implemented within relational databases. A technique that has been used with second year IT degree students to map specialisation hierarchies to the internal model of a relational DBMS will be presented.

Keywords

EERD, Specialisation Hierarchies, Internal Model

This poster paper appeared at the 2nd annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2011) incorporating the *24th Annual Conference of the National Advisory Committee on Computing Qualifications*, Rotorua, New Zealand, July 6-8. Samuel Mann and Michael Verhaart (Eds).

Introduction

To illustrate a technique that may be used to map specialisation hierarchies to an internal model, a simple example has been provided. The example involves an entity supertype (called Employees) that contains generic attributes relating to all employees within an organisation. The entity subtypes (called Administrators and Trainers in the example) contain unique attributes particular to each subtype. The specialisation hierarchy shown in Figure 1 document these entity super/subtypes and includes other details relevant to this example. The letter 'd' contained in the circle (called a *Category* symbol) denotes that a *Disjoint* constraint applies. This indicates that an employee cannot be both and administrator and a trainer at the same time and vice versa. The double line beneath the Category symbol indicates that a *Complete* constraint applies. This means that each employee occurrence in the Employee supertype will also always be a member of one of the entity subtypes i.e. the cardinality on the relationship from supertype to each subtype is 1:[0..1]. To determine which subtype the supertype occurrence is related to, a subtype discriminator is employed in the form of an attribute in each entity (EmpType in Figure 1). The value of this attribute is used to distinguish between each subtype. This value is documented in the logical model on the connector above each subtype ('A' – Administrator, 'T' – Trainer in the example illustrated in Figure 1).

Using the details contained in Figure 1 and the introduction above, a technique to map specialisation hierarchies to an internal model is now explained.

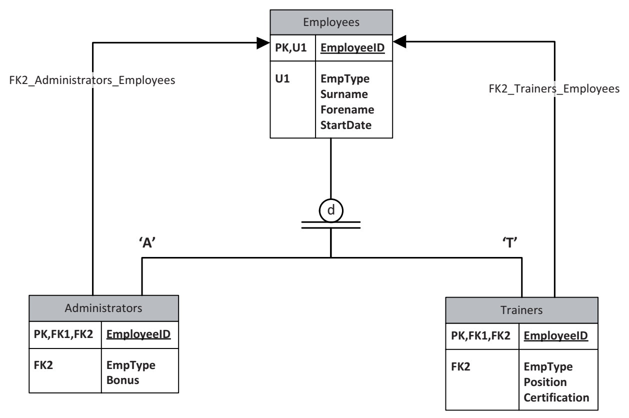

Figure 1: Employees Specialisation Hierarchy

Mapping Technique

1) Specify a common primary key for the entity supertype and each entity subtype (EmployeeID in the example).
2) Specify a subtype discriminator attribute for the entity supertype and each entity subtype (EmpType in the example).
3) Define a foreign key in each entity subtype that references the primary key in the supertype, thereby establishing a 1:1 relationship between the entity super/subtypes (shown as FK1 in each entity subtype).
4) Specify a uniqueness constraint in the entity supertype comprising of both the primary key and subtype discriminator (U1 in the example).
5) Define a foreign key in each entity subtype that references the uniqueness constraint defined in the entity supertype (FK2_Administrators_Employees and FK2_Trainers_Employees in the example). This ensures that the primary key/subtype discriminator combination provided in the subtype actually exists in the supertype.
6) Create a check constraint in the entity supertype against the subtype discriminator attribute such that only valid values (i.e. 'A' or 'T' in this example) are allowed.
7) Create a check constraint in each entity subtype against the subtype discriminator attribute such that only the valid value (i.e. 'A' in Administrators, 'T' in Trainers) is allowed.

   This check constraint in combination with each foreign key reference to the uniqueness constraint in the supertype ensures that the subtypes are only populated with valid data.

Conclusion

The mapping technique described provides a relatively simple approach, appropriate to second year IT degree students, for implementing specialisation hierarchies in relational databases. Although the technique is only directly applicable to the form of specialisation hierarchies described in the example (i.e. Disjoint and Complete), the technique can be extended to other forms with slight modifications to the check constraints defined in the entity super/subtypes.

References

Rob, P. & Coronel, C. (2009). *Database Systems: Design, Implementation, and Management (8th Ed.).* Boston, Massachusetts: Course Technology.

Dewson, R. (2008). *Beginning SQL Server 2008 for Developers*. New York: Apress

Quatrani, T. (1998). *Visual Modelling with Rational Rose and UML*. Reading, Massachusetts: Addison Wesley Longman.