A Canonical Approach to Relational Database Design
For Managing the Knowledge of Generalization

João Eduardo Ferreira¹, Larissa P. S. Gomes¹, Shamkant B. Navathe², CaetanoTraina Junior¹

¹ University of São Paulo
Rua do Matão, 1010 - Cidade Universitária
05508-090 São Paulo - SP - Brasil
jef@ime.usp.br, {larissa, caetano1}@icmc.sc.usp.br
² College of Computing,
Georgia Institute of Technology
801 Atlantic Drive, Atlanta
GA 30332-0280- USA
sham@cc.gatech.edu

Abstract. This work proposes a new exhaustive approach to dealing with the
generalization abstraction during the design of relational databases. It proposes
a "set of constraint-attributes" approach at a conceptual level and discusses all
possible ways of mapping these constraints into the relational model. We
briefly point out the implications on physical design of relations as well and
establish guidelines to help decide what the best mapping option to use is,
based on parameters that govern generalization/specialization. The overall
purpose of this work is to increase the semantic value of relational designs that
are sometimes approached in a very ad-hoc manner when it comes to dealing
with the generalization abstraction. A database mapping and design tool has
been developed for this approach at the University of São Paulo, Brazil and has
been applied to many real-life design problems.

1 Introduction

1.1 Context of mapping of the generalization abstraction to the relational model

Based on a study of data models [1][3][5][8][11][12][14][15][16], it is easy to see
that all the models support generalization abstractions which is characterized by the
is-a relationship among the classes. The models are similar in their scope, but use
different notations. After these models show these generalization abstractions the
research stops in the time for the new aspects. Our work flashes back this subject with
new important approach for optimization of semantic mapping to canonical model.
As defined by Elmasri and Navathe [10], a specialization \( E = \{S_1, S_2, \ldots, S_n\} \) is a set of subclasses that has the same super class \( G \), i.e., \( G/S_i \) is a relation of a super class/subclass for \( i = 1,2, \ldots, n \). One can further state that \( G \) is a generalization of the set of subclasses \( \{S_1,S_2,\ldots,S_n\} \).

Elmasri and Navathe [10] as well as Batini et al. [1] define two types of “coverage constraints” on generalizations. They are called as completeness and disjunctedness constraints. A specialization \( E \) is total whenever \( (\bigcup_{i=1}^{n} S_i) = G \); otherwise, it is a partial specialization occurrence. Furthermore, \( E \) is called mutually exclusive (disjointed) whenever \( S_i \cap S_j = \emptyset \) for the entire \( i \neq j \); otherwise, \( E \) is called an overlapped specialization.

The decision of how to define the restrictions of each occurrence of generalization/specialization abstraction depends on real world semantics and on what one wishes to represent in the modeling.

Several studies have presented a systematic approach to the logical design of relational databases [7][13][20] considering the E-R as the conceptual model [6][17][18][21], or considering object-oriented models [4][8]. Among these studies, Elmasri and Navathe [10] propose a mapping approach using the E-R [19] as a conceptual model due to its broad coverage of all the aspects studied in other proposals. They propose four mapping options that take into account the restrictions of participation and superposition of subclasses.

In the first option, the super class and each subclass are mapped onto separate tables. This option can be used for all the restrictions (total or partial participation, overlap or disjunction). With this mapping method, when an inquiry requires the recovery of specific information of a given object, it is necessary to make a search through the keys of all the specific tables. This procedure makes inquiries highly inefficient. In an attempt to overcome this problem, in the case of specialization with disjunction restrictions, Elmasri and Navathe [10] propose that a choice be made of one of the attributes of the super class to specify the condition of membership of a subclass.

Consider the example of a worker super class that specializes in the engineer and laborer subclasses with disjunction and total participation restrictions. In this case, one can define an attribute of the super class (type-of-work) to indicate the subclass for which an object is specialized. Thus, the predicate: type-of-work = ‘engineer’; would indicate that the object worker is specialized in the subclass engineer. This solution, however, is only valid when there is a disjunctive restriction.

In the second mapping option [10] defines a table for each subclass. The super class is eliminated and all its attributes are replicated in the tables that represent the subclasses.

In the third and fourth options, all the subclasses are eliminated and their attributes are represented in the table that maps the super class. The difference between them is that the third option defines a type attribute that will indicate the subclass to which each tuple belongs and is valid only for specialization with disjunction restrictions. In the fourth option, on the other hand, instead of \( t \) there is a set, \( \{t_1, t_2, \ldots, t_m\} \), where \( m \) is the number of subclasses, and each \( t_i \) with \( 1 \leq i \leq m \) is a Boolean attribute that indicates whether or not a tuple belongs to an \( S_i \) subclass. This last option is valid for specialization with overlapping. All the four options work for
total specializations. For partial, option two does not work since it eliminates the super class which must be preserved.

1.2 Proposal to Refine the Specialization Semantics

In this work, we propose the definition of generalization/specialization, with special emphasis given to the set of attributes called set of constraint-attributes. This idea of a set of attributes acting as constraints of the generalization abstraction is a new concept that refines the concept of “attribute type” defined by Elmasri and Navathe [10]. We believe that this way the semantic value of the generalization abstraction is increased, presenting an exhaustive set of alternatives for representation in the relational data model. We will present these mapping alternatives and later in the paper supply guidelines of how and when to use each mapping alternative.

In section 2, we attempt to canonically represent the generalization abstraction with a new semantic value for the specialization set of constraint-attributes. Section 3 contains definitions of the mapping options, illustrated as examples, taking into account the possibilities for occurrence of the set of constraint-attributes. In section 4 we present a chart with the types of evaluation and the suitability of options. Finally, section 5 contains our conclusions.

2 A Canonical Data Model for Generalization Abstraction

With the purpose of broadly covering the situations represented by data models, this work defines an object-oriented data model constructed specifically to support generalization abstractions. This model, which will be referred to as Canonical Model for generalization abstractions, is not presented in its complete form, since the other concepts that make up a data model will not be discussed. It is implicitly considered that the semantic concepts and constructors common to object-oriented models are supported. For the purposes of this paper, we just focus on the generalization abstraction.

We will use the term an occurrence of generalization abstraction (OGA) to refer to a specific set of characteristics that define a particular generalization. Basically, the set of characteristics that define an OGA was extracted from the objects models [1] [2][3][4][7][9][10]. From the models studied, the Object Oriented model presented the most complete description of a generalization abstraction, particularly in its treatment of the set of constraint-attributes, which is fundamentally important for this work.

Thus, the basic structure for each OGA is defined as comprising the following elements:

a) a (single) generic type of object– G;

b) a set of specialized types of objects E={E1, E2, ... En};

c) a set of attributes of the generic type XG = {G.Attributei} = K∪AG, where

K={G.AttributeKeyj}, and AG={G.AttributeNonKeyk};

d) a set of attributes for each specialized type AEe = {Ee.Attributei}, 1≤e≤ne;
e) a set of hierarchical representation restrictions for generalization \( R = \{O, D, P, T\} \);

f) a set of physical properties for generalization \( P = \{p_1 \ldots p_p\} \);

g) a predicate for each specialization, based on the set of attributes of the generic type \( P_e = p_e(G.Attribute_i, \ldots) \), \( 1 \leq e \leq n_e \);  

h) a set of rules of initialization of values for each type of specialized object \( I_e = \{E_e.Attribute_i = v_i(G.Attribute_j, \ldots)\} \), \( 1 \leq e \leq n_e \); and

i) a set of constraint-attributes of the OGA, composed of all the attributes of the generic type of this occurrence that participate in the predicate of any specific type, i.e., \( C = \{G.Attribute_i\}, | \exists p_e(G.Attribute_j, \ldots) | 1 \leq e \leq n_e \), where \( G.Attribute_{i} \) preceding at least one predicate \( p_e() \).

Of these elements there are relevant aspects to this work that are described below.

**Predicate for generalization**

A predicate \( p_e \) (Generalization Attribute_i, \ldots) is a function applied over a subset of generic type attributes, which results in a true or false logical value. This function is evaluated for each generic type object, and the values of one or more of the attributes of this object are verified. If the result is true, it means that that object is also of the subtype that corresponds to the evaluated predicate. The function can use the operators \((=, \neq, \geq, >, <, \leq, \in, \notin, \text{etc})\) to construct the predicate expression.

**Set of Constraint-attributes for Generalization**

A set of constraint-attributes \( C \), of the generic type, define at least one predicate of a given OGA. The domain of an attribute contained in the constraint-attribute set defines important data to characterize the OGA. For example, a set of constraint-attributes composed of discrete and finite attributes in which all the predicates correspond to a single <attribute=value> equality characterizes an OGA that is inherently disjoint.

**Physical Properties for Generalization**

The physical properties of the OGA are physical considerations for mapping that must always be followed to ensure data consistency. However, in the implementation of a data structure representing a generalization abstraction, factors that are peculiar to the application and not to the conceptual structure of the abstraction occur that may be important in choosing a particular way of implementing an OGA. In this work, we consider the following factors as properties:

- **\( p_1 \) – High Frequency of access to join subtype and the super type data** – this property takes into account the prediction that access to information in the attributes that correspond to the super type will always, or at least very often, be coupled to information of the respective subtype or types. If this is true, then it is ineffective to keep the two parts of the information separate, since that would require the frequent use of join operation. If the information should preferentially be accessed by either the super type or the subtype, keeping them in smaller, separate tables can speed up the access. Correspondingly \(-p_1\) refers to a low frequency.
$p_2$ – High Frequency of access to join information from several subtypes – this property considers that there is always, or very often, simultaneous access to the information of the attributes of several subtypes. If this property were true, it would be ineffective to keep a table for each subtype, since it would be necessary to make frequent use of join among the tables corresponding to various subtypes. The best option, in this case, would be to keep the information of all the subtypes in a single table, thus making access to the information more efficient. Correspondingly $\neg p_2$ refers to low frequency.

Generalization Schema: A Diagrammatic Notation

The overlap (O) and disjoint (D); partial participation (P) and total participation (T) restrictions are dealt with in the same way as in other data models, as described in section 1.1. We additionally use a new restriction: explicit representation of the set of constraint-attributes (C) and no explicit representation ($\neg$C).

Considering the notation defined in [3], Figure 1 shows a particular diagram to represent the abstraction of generalization. The generic type or super type (G) is represented at the top, together with its K and AG attributes, where K represents the identifier attribute and AG represents the set of attributes of the super type. The rectangles that represent the specifics types, or subtypes (E₁, E₂, ..., Eₘ), are divided into two parts. The upper part contains the definition of the name of the specific object, while the lower part contains the definition of the values of the predicate and the initialization set. The attributes of (E₁, E₂, ..., Eₘ) are represented, respectively, by AE₁, AE₂, ..., AEₘ. The trapeze in the center of the diagram indicates the name of C and the O, D, P and T restrictions.

![Diagram to represent the abstraction of generalization.](image-url)
3 Definition of the Ten Exhaustive Cases of Mapping

The mapping process produces a set of relations from the canonical generalization abstraction, proposed in this work. The mapping is described through a set of ten “mapping options” (MO) for the relational model, in which the objective of each is to meet the needs of a specific situation in an OGA. In this description, each option is characterized by a schema of relations that result from an OGA mapping process.

Using the possible combinations for representation of the generic (G) and specific \{E_i\} information gives one four mapping options, as shown in Elmasri and Navathe [10]. Now, using the (G), the \{E_i\} and the set of constraint attributes (C), with or without a specific table D, gives six additional relevant options to represent the generalization abstraction in the relational model.

Table 1 classifies the ten mapping options in three groups, according to the mapping results:
- G + \{E_i\}: results in one table to deal with the generic type and one for each specific type involved in the OGA;
- \{E_i\}: results in one table for each specific type involved in an OGA, and the generic type information is replicated in each specific type;
- G: results in a single table containing information of the generic type and of all the specific types.

The ten mapping options for generalization defined in this work extend the scope of the proposals available in the literature, providing the designer with a more complete set of mapping possibilities. Each of these options should be made to suit the problem that one wishes to represent in the relational model. It is important to note that these ten mapping options cover all the cases listed in the literature, in addition to presenting six new cases. The six new cases explored in this work correspond to the mapping options 2, 3, 5, 6, 7 and 9.

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
<th>Group</th>
</tr>
</thead>
</table>
| 1       | G = \{K \cup AG\}  
          Ee = \{K \cup AEe\} | \(1 \leq e \leq ne\) |
| 2       | G = \{K \cup C \cup AG\}Ee = \{K \cup AEe\} | \(1 \leq e \leq ne\) |
| 3       | G = \{K \cup AG\}  
          Ee = \{K \cup AEe\}  
          D = \{K \cup C\} | \(1 \leq e \leq ne\) |
| 4       | Ee = \{K \cup AG \cup AEe\} | \(1 \leq e \leq ne\) |
| 5       | Ee = \{K \cup C \cup AG \cup AEe\} | \(1 \leq e \leq ne\) |
| 6       | Ee = \{K \cup AG \cup AEe\}  
          D = \{K \cup C\} | \(1 \leq e \leq ne\) |
| 7       | Ee = \{K \cup AG \cup AEe\}  
          D = \{K \cup C\} | \(1 \leq e \leq ne\) |
| 8       | G = \{K \cup AG \cup \{AE1 \cup AE2 \cup \ldots \cup AEne\}\} | G |
| 9       | G = \{K \cup C \cup AG \cup \{AE1 \cup AE2 \cup \ldots \cup AEne\}\} | G |
The set of attributes that makes up the constraints indicate the subtypes into which an object is specialized, thus optimizing queries to specific subtypes. Moreover, this study takes advantage of the domain of attributes that comprise the \( C \) and define important parameters to characterize an OGA. The implicit or explicit representation of these attributes defines under what situations it is necessary to store them in the base. Option 3 is shown for illustration this work. The options 5, 6, 7 and 9 were omitted because there is no space enough in this paper.

**Option 3:** \( G = \{K \cup AG\} \), \( E_i = \{K \cup AE_i\} \) \( 1 \leq i \leq m \) and \( D = \{K \cup C\} \)

To illustrate this mapping option, consider a company in which the employees can work as either systems analysts or as programmers. The employee who does an outstanding job receives an additional position: that of supervisor of his department.

Figure 2a represents this example by means of an OGA in which the Employee can be specialized according to \( C\) job-type in Supervisor, Analyst and Programmer. The restrictions defined for this OGA are partial participation (P) and overlapping subtypes (O).

In option 3, a \( G \) relationship is created for the generic type, \( E_i \) relationships for each specific type, with \( 1 \leq i \leq m \), where \( m \) is the number of subtypes, and a \( D \) (domain) relationship. The \( G \) relationship has a set of identifying attributes (K) as its primary key and a set of attributes of the generic type (GA). Each \( E_i \) relationship has a set of identifying attributes (K) as its primary key, as well as a set of attributes of the specific type that is being mapped (AE\(_i\)). The \( D \) relationship has a set of identifying attributes (K) and the \( C \) that defines the specialization, both as primary key. The primary key of the generic type (K) is propagated to the \( E_i \) and \( D \) relationships.
In this example, the $C$ is determined by the value of one single-valued attribute whose domain is $\text{dom}(\text{Attributes of job-types}) = \{\text{programming, documentation, modeler, DBA, coordination, management}\}$. To verify the subtype to which a given object of the generic type has been specialized, one defines:

1. **PredicateProgrammer** $\in \{\text{programming, documentation}\}$;
2. **PredicateAnalyst** $\in \{\text{DBA, modeler}\}$;
3. **PredicateSupervisor** $\in \{\text{coordination, management}\}$.

Thus, as in the case of MO2, this mapping option provides a syntactic structure to map the $C$. However, in the MO2, this attribute was represented in the table that maps the super type, together with AG, being restricted to the OGAs in which there is a disjunction.

In the mapping that results from the MO3, Figure 2_b, a D_Employee relation is defined to map the $C$ and, because $K$ and $C$ are keys to the relation, this MO can be used in OGAs in which there is overlap.

Table D allows one to check all the functions of an employee. Thus, a query to table D_Employee supplies the information that the attributions of a given Employee are DBA, programming and documentation. An evaluation of predicates (1), (2), and (3) reveals that this Employee is a Supervisor and a Programmer, the specific information of which can be checked in the corresponding tables.

The importance of table D_Employee is evident for the two following reasons:

1. all the job-types of an employee can be consulted;
2. one can determine the specific table or tables to which a given employee has been specialized.

Thus, this MO may be used for any of the two ways in which the $C$ can be represented. This option is good when the domain $C$ is represented by a set of attributes that are multi-valued.

There is association between of the ten mapping options proposed together with explanatory captions for each item evaluated according property. It is possible to show when the ten options are best to use according to types of properties. Two types of properties have been defined: hierarchical and physical. These properties classify many situations of design and developing a database. It is out of this paper scope to show and analyze these properties according to the ten options.
5 Implementation of a Tool

We have implemented a design tool that supports the above mapping scheme and design framework. By taking into consideration the characteristic of each generalization abstraction, the users can choose the most appropriate mapping option to build the physical database of the relational model. The present set of tools on the commercial scene. For a list of current commercial tools, see Chapter 16 in [10]. None of these commercial tools (e.g., ERWin, Designer 2000, Modelmart, ER Studio etc.) give the user any control in the choice of design as far as generalization is concerned. The mapping scheme has been used in an application in Brazil to control the design which involved two million of instances.

6 Conclusion

This work has presented a model for clearly representing the knowledge of generalization abstraction during relational design. We defined the generalization abstraction by proposing a scheme that focuses on set of constraint-attributes. We then presented a series of ten mapping options for dealing exhaustively with all possible situations. The mapping options provide a well-defined and exhaustive scheme for going from conceptual schemas to logical designs of relations and eventually physical design of the relational schemas.

Using the mapping options presented herein, a new semantic value can be observed for the set of constraint-attributes that better characterizes and delimits the hierarchy of specialization in the conceptual, logical and physical design of relational databases.

The guidelines that are given here and the parameters that are shown for evaluating the mapping options offer clear guidelines for the mapping of specialization hierarchies into relational database designs and eventually, physical database modes.

Our future research includes investigations of the characteristics of active rules to control data consistency in the data generalization hierarchies. We call them rules for consistency of data integrity in the generalization hierarchies. We are also defining some formal properties of our canonical model of generalization abstraction. Another aspect to be investigated is the use of the mapping options presented here to determine an optimal choice of data fragments in the methodology of distribution and control of data in distributed databases systems.
References


