Applying Analogy to Schema Generation

Antonio L. Furtado, Karin K. Breitman, Marco A. Casanova, Simone D.J. Barbosa

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de S. Vicente, 225
Rio de Janeiro - RJ – BRASIL– CEP 22451-900
{furtado, karin, casanova, simone}@inf.puc-rio.br

Abstract

To support the generation of database schemas of information systems, a five-step design process is proposed that explores the notions of generic and blended spaces and favours the reuse of predefined schemas. The use of generic and blended spaces is essential to achieve the passage from the source space into the target space in such a way that differences and conflicts can be detected and, whenever possible, conciliated. The convenience of working with multiple source schemas to cover distinct aspects of a target schema, as well the possibility of creating schemas at the generic and blended spaces, are also considered.

Keywords: Schema Generation, Analogy, Blending, Lattices, Entity-Relationship Model, Logic Programming

1. INTRODUCTION

Designers of information systems soon learn that reusing their previous experience, and also that of other designers, is a rewarding strategy.

Motivated by this remark, we have been working [2,3] on methods and tools to, starting from some predefined database schema regarded as a source schema, abstract a pattern that captures its structure, which is then repeatedly used to generate one or more target schemas. What makes this strategy viable is the intuitive perception of an analogy between source and target, expressed by saying that the latter is like the former.

Additionally, the source schema should be a typical example among those that are analogously structured, and the terminology of its underlying domain should be familiar even to the less experienced designers. If these requirements are satisfied, it will be possible to instantiate the positions occupied by variables in the pattern, by prompting the designer to indicate which names in the target schema being generated correspond to each name in the example source schema.

In the present paper, we expand our earlier method and introduce a five-step process that takes four spaces into consideration – the source, target, generic and blended spaces, as proposed in [9] for widely different areas. We adopt the familiar Entity-Relationship (ER) model [5] and use the weak entity concept to illustrate the process.

The diagram in figure 1 represents the four spaces and shows how they are articulated in view of the process, whereby, starting from the source, the target is gradually constructed.

![Figure 1: The four-space approach](image-url)
target spaces and denotes the elements that correspond to each other in these two spaces. By contrast, the blended space reflects the join of source and target and inherits all their elements, corresponding or not. Again informally, the blend is the space where one can detect whatever is incomparable or conflicting when putting together source and target, often calling for some creative form of adaptation to be remedied or conciliated \[9,21\]. Goguen [10] formalized blending in category theory.

The text is organized as follows. Section 2 details the five-step process we propose and is the thrust of the paper. Sections 3 and 4 briefly discuss, respectively, the advantages of bringing in a multiplicity of source schemas for designing distinct aspects of a target schema, and the possibility of also creating schemas directly from elements at the generic or blended spaces. Section 5 contains the conclusions.

2. The Five-Step Schema Generation Process

2.1. Example

We adopt a simple example to illustrate the proposed schema generation process. We start with a schema fragment, specifying employees and their dependents, which is probably the most frequently mentioned illustration of the weak entity concept in ER modelling. As a fragment, it only needs the elements relevant to characterize weak entities.

We express schemas with the help of clauses such as those below that introduce two entity classes, employee and dependent:

```
Schema: Emp_Dep
Clauses --
  entity(employee, empno)
  attribute(employee, empno)
  entity(dependent,
    [empno/depno-isdepof-empno, depno])
  attribute(dependent, depno)
  relationship(isdepof,
    dependent/0/n, employee/1/1)
  attribute(isdepof, family_tie)
```

The identifying attribute of employee is empno, whereas dependent, being a weak entity, relies on the identifying relationship isdepof, combined with the discriminating attribute depno. The identifying relationship is 1 to n, being total with respect to dependent and partial with respect to employee; these properties are indicated by associating pairs of minimum and maximum values for the participation of instances of each entity in relationship instances: at least 0 and at most n dependents can be related with exactly one employee. The relationship isdepof has attribute family_tie, with values such as spouse or child. Note that the fragment does not include, as unessential to the characterization of weak entities, certain basic properties of employee, such as those referring to the employment aspect itself.

This schema will be used as the source schema, wherefrom target schemas based on the weak entity concept can be derived, through five consecutive steps, to be described in the sequel.

As will be noticed, the process takes into due consideration some domain-independent consistency rules inherent in the ER model, such as the following, among others:

1. all entity classes must have identifying properties;
2. relationships can only be defined between defined entity classes;
3. the deletion of an entity instance implies the deletion of all its properties;
4. if a relationship R is total with respect to one of its participating entity classes E, an instance of R cannot be deleted if it is the only one involving a given existing instance of E.

2.2. Step 1 - Generating the Pattern

From the source schema Emp_Dep, the Weak Entity pattern is obtained (Figure 2) by consistently substituting variables for the names of entities, relationships and attributes.

```
Pattern: Weak Entity
Example schema: Emp_Dep
Clauses --
  entity(A, B)
  attribute(A, B)
  entity(C, [B/D-E-B, D])
  attribute(C, D)
```

Figure 2: Generating the pattern

Besides clauses built from those of the source schema, the pattern contains mappings, associating the variables introduced with the corresponding source schema names. Consistent substitution implies that, to give one example, variable A refers to entity employee wherever it occurs in the clauses of the pattern.
2.3. STEP 2 - GENERATING THE TARGET SCHEMA

Suppose the designer wants to specify a Bk_Ed schema, about book editions, and realizes that this too involves the weak entity concept: the editions of a book are comparable to the dependents of an employee, in that to identify an instance of edition, the indication of the book in question is needed, besides the edition number – edno – as discriminating attribute. The generation (Figure 3) is basically done by specializing the clauses of the pattern (belonging to the generic space), but the diagram also refers to the originating source space, to stress that the names in the pattern mappings were extracted from it.

Suppose that the designer judged that the addition and the modification below are necessary:

addition: attribute(book, subject)
modification: isedof – min-1:1

The modification enforces the requirement that a published book must have at least one edition. Then, the Bk_Ed target schema becomes:

```
<table>
<thead>
<tr>
<th>Schema: Bk_Ed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clauses --</td>
</tr>
<tr>
<td>entity(book, isbn)</td>
</tr>
<tr>
<td>attribute(book, isbn)</td>
</tr>
<tr>
<td>attribute(book, subject)</td>
</tr>
<tr>
<td>entity(edition, [isbn/edno-isedof-isbn, edno])</td>
</tr>
<tr>
<td>attribute(edition, edno)</td>
</tr>
<tr>
<td>relationship(isedof, edition/1/n, book/1/1)</td>
</tr>
</tbody>
</table>
```

2.4. STEP 3 - BLENDING THE SOURCE AND TARGET SCHEMAS

The blended space is pictured as a confluence of the source and the target spaces, taking into consideration the correspondences registered in the

```
Figure 3: Generating the target schema
```

```
Figure 4: Blending the source and target schemas
```
generic space (Figure 4).

In the database schema-generation process, elements are obtained by joining each entity and relationship of the source schema with its counterpart in the target schema. To begin with, all information about each entity and relationship, contained in the various clauses of the two schemas, is collected in separate frames, structured as lists of property:value pairs.

Each property of an entity E is represented either by an attribute name, or by a binary relationship name tagged with 1 or 2 to indicate, respectively, whether E is the first or the second participant in the relationship. Since in the present example no restrictions are being imposed on the values, all value positions are filled with an underscore, a usual convention for an anonymous variable.

The properties of a relationship R are similarly represented. They include the identifying attributes of the two participating entities, the minimum and maximum occurrences for the first and for the second participant, and other relationship attributes if any.

The frames extracted from Emp_Dep are:

frame of employee =
[empno:_, isdepof/2:__]
frame of dependent =
[depno:_, isdepof/1:__]
frame of isdepof =
[depno:_, empno:_,
min-1:0, max-1:n, min-2:1, max-2:1, family_tie:_]

and those taken from the Bk_Ed schema are:

frame of book =
[isbn:_, subject:_, isedof/2:__]
frame of edition =
[edno:_, isedof/1:__]
frame of isedof =
[edno:_, isbn:_,
min-1:1, max-1:n, min-2:1, max-2:1]

We shall introduce here a join operation on frames, specifying that, when applied to entity or relationship frames F1 and F2, a frame J results, whose property-value pairs comprise:

a. pairs p1:v1 from F1, for each property p1 not corresponding to any property in F2;
b. pairs p2:v2 from F2, for each property p2 not corresponding to any property in F1;
c. pairs p1:p2:v1,v2, for each two corresponding properties p1 and p2 in F1 and F2, respectively.

Value v1,v2 in item c is obtained by, in turn, joining the two values v1 and v2, according to the following criterion: if the values are identical constants, or at least one of them is a variable, v1,v2 is the result of their unification [13]; otherwise the result is a term formed by the two values prefixed by an asterisk to indicate that they are in conflict.

The frames characterizing the blended space, obtained by joining the frames taken from the source and the target schemas, are shown below. Non-corresponding properties and conflicting values are stressed (in italic, boldface; the symbol “∪” denotes the join of two frames):

Femployee ∪ Fbook =
[empno-isbn:_,
isdepof/2-isedof/2:__,
subject:__]
Fdependent ∪ Fedition =
[depno-edno:_,
isdepof/1-isedof/1:_]
Fisedepof ∪ Fisedof =
[depno-edno:_,
empno-isbn:_,
min-1:*[0,1],
max-1:n,
min-2:1,
max-2:1,
family_tie:_]

A disclaimer is in order here. We have considered only one simple type of conflict. If the designer is allowed to perform arbitrary modifications to the target schema initially obtained by instantiating the pattern variables (cf. step 2), other types of conflict may occur, calling for the specification of appropriate criteria to handle them. As noted in [9], blending is, in general, a particularly complex task, requiring a great deal of creativity from the part of the designer, who may have to devise ad hoc ways to achieve consistency. Moreover, conflicts detected through blending may affect the design of application-oriented operations on the generated schemas (a topic briefly addressed in section 2.7).

2.5. STEP 4 - REVISIING THE TARGET (AND SOURCE) SCHEMAS

The resulting blended space can be reinjected into the derived target space, and even into the originating source space, if the designer admits the possibility of

Figure 5: Revising the target (and source) schemas
also reconsidering it (Figure 5).

In our example, a convenient way to call the designer's attention to what was not used from the source schema is to display together, in frame format, the entire list of current properties of each entity and relationship in the target schema, expanded as the result of blending. Such frames are directly obtained from the blend frames by reducing the paired names assigned to corresponding properties to their original names in the target space, while, naturally, keeping the names of the source space properties until now disregarded:

```
frame of book =
[isbn:_, isedof/2:_, subject:_]
frame of edition =
[edno:_, isedof/1:_]
frame of isedof =
[edno:_, isbn:_, min-1:1, max-1:n, min-2:1, max-2:1, family_tie:_]
```

Surely, the designer may or may not judge appropriate to reconsider what was initially left out, in this case the relationship attribute family_tie. Would there be different "ties" between edition and book? Ironically, the remark that "so-and-so is a revised edition of his father" is not uncommon, a playful but expressive metaphorical connection between the domain of human beings, underlying employee, and the domain of books, which would bring to mind that an edition may be classified as revised, corrected, expanded, abridged, and also simply as regular, which are some of the possible values for a new ed_type attribute for the isedof relationship.

The reconsideration of a source schema, such as Emp_Dep, for expansion is more rarely desirable, especially if one wishes to keep it as a fragment containing only the features necessary to characterize weak entities. But in the event that the designer wants to examine the possibility, the blend frames can be alternatively renamed as follows:

```
frame of employee =
[empno:_, isdepof/2:_, subject:_]
frame of dependent =
[depno:_, isdepof/1:_]
frame of isdepof =
[depno:_, empno:_, min-1:0, max-1:n, min-2:1, max-2:1, family_tie:_]
```

What can be the "subject" of an employee? The subject of a book can be some fictional genre, but it can also be a professional field, such as engineering, or accounting, which may suggest a new attribute profession for the employee entity, with possible values including engineer and accountant, among others.

A further reduction of Emp_Dep to suppress the family_tie attribute is more likely to happen. This would become advisable if the attribute is systematically disregarded, even at this revision step, in a long series of target schemas generations. Reconsidering a source schema, and consequently the pattern abstracted from it (as covered in step 5) is a case of double-loop learning [1]: the continuing use of a model providing clues for its correction and refinement.

### 2.6. Step 5 - Revising the pattern

Since the generic space is often intended as a help to generate a plurality of target spaces, conflicts located at the blended space, as well as changes made at the source space from suggestions motivated by observing the blend, may entail the reconsideration of the generic space (Figure 6).

In our example, the blend mirrors the fact that an identifying relationship must be total with respect to the weak entity, but no such requirement is imposed with respect to the entity on which it relies for identification. So the conflict registered in the property:value pair min-1:* (0,1) of the frame resulting from the join of Fisdepof with Fisedof should motivate the insertion of a hotspot [19] in the Weak Entity pattern, i.e., a place where the specification becomes flexible.

The adopted notation, using a question mark as prefix, will signal that the designer should be queried about the min-1 property of the relationship denoted by variable E, and that the value supplied must be chosen as 0 or 1.

Moreover, if at step 4 a new attribute such as profession is added to the source target, or if the family_tie relationship attribute is removed from it, the pattern must be modified accordingly, so that it will continue to reflect the Emp_Dep schema.

If all these modifications occur, after deleting the lines

```
attribute(E, F)
F:family_tie
```

![Figure 6: Revising the generic space](image-url)
and adding or modifying three lines (in boldface), the pattern would become:

Pattern: Weak Entity
Example schema: Emp_Dep
Clauses --
  entity(A, B)
  attribute(A, B)
  attribute(A, G)
  entity(C, [B/D-E-B, D])
  attribute(C, D)
  relationship(E, C/?(0,1)/n, A/1/1)
Mappings --
  A:employee
  B:empno
  G:profession
  C:dependent
  D:depon
  E:isdepof

2.7. TOWARDS THE DESIGN OF OPERATIONS

In [6] we added, both to schemas and patterns, clauses defining operations in terms of their pre- and post-conditions [8].

Without going into details, we now give one example of the repercussion of conflicts detected at the blending stage on the design of operations. Suppose that an operation named end_coverage has been defined over the source schema, allowing to remove a child C of an employee E from the list of dependents of E, if the birth_year of C (an additional attribute of dependent) precedes a currently determined limit. Note that indicating the deletion of the literal dependent([E,C]) should cause the deletion of all properties of the entity instance C, in view of ER rule 3. On the other hand, note that the repeated execution of end_coverage is allowed, legitimately, to leave an employee with no dependents.

end_coverage(C,E)
pre-cond: dependent([E,C]),
  family_tie([E,C],child),
  birth_year([E,C],Y),
  Y < b_ylimint.
post-cond: ¬dependent([E,C]).

Also suppose that, during step 2 of the interactive process, the designer reacted favourably when prompted to introduce an operation corresponding to end_coverage, with the purpose to analogously discard editions whose year of publication, ed_year (again a new attribute, corresponding to birth_year), came before a currently designated year. In the context of library management, this is a well documented practice, known as weeding library collections [20].

A conservative librarian would very likely demand that systematic discarding be restricted to regular editions, a requirement that can be easily expressed if attribute ed_type has been supplied as a counterpart to family_tie, as considered earlier.

However, straightforward renaming and the replacement of child by regular is not sufficient here to avoid a conflict of the generated weed operation with specific characteristics of the target schema registered when blending, namely, the totality property of isedof with respect to book, combined here with ER rule 4. One solution to the conflict is illustrated in the version of weed shown below, which can be repeatedly applied to discard any number of non-special editions, provided that the book itself remains – by keeping its newest edition – to adopt a usual criterion.

weed(E,B)
pre-cond: edition([B,E]),
  ed_type([B,E],regular),
  ed_year([B,E],Y),
  edition([B,En]),
  ed_year([B,En],Yn),
  Yn > Y,
post-cond: ¬edition([B,E]).

Further refined versions may specify different values of ed ylimint for different subjects, in view of constantly updated studies to determine the period of obsolescence for publications belonging to each so-called Dewey class [14].

3. COVERING DIFFERENT ASPECTS THROUGH MULTIPLE SOURCE SCHEMAS

Patterns to model the same concept can be obtained from different source schemas. We chose the Emp_Dep example to construct the Weak Entity pattern, but other examples could be selected, from which a family of versions of the pattern would be obtained and made available to designers. Originating from source schemas featuring different sets of names, the mapping section of each version would differ from that of the others.

More importantly, not all clauses might be identical, which reflects permissible structural variations, according to which the versions could be classified. A designer would then have a chance to choose the version appearing more congenial to the case on hand. For instance, a schema Prod Comp, treating the components of products as another example of Weak Entity, would come equipped with operations such as repair and replace as alternative ways to handle a component found to be
defective. Thanks to the availability of such operations, Prod_Comp would seem a better source than Emp_Dep for generating a schema Bk_Vol dealing with volumes of books, inevitably susceptible to damage in the everyday functioning of a library environment.

Repeating the pattern generation process with a second version is another advantage of keeping several examples around, since this provides a means to check the result. Assume, for instance, that a version of Weak Entity is available wherein the identifying relationship is total with respect to both participating entities. If the designer of Bk_Ed had not noted at step 2 (see section 2.3) the need to correct the specification of isedof, blending it with the schema generated from this second version of the pattern would reveal the conflict.

But the application of more than one source must also be considered along a separate line of reasoning. Early studies on analogy and metaphor [15] already argued in favour of the use of multiple sources to provide a fuller characterization of a target possessing many properties, which might however be grouped into a manageable number of meaningful clusters. Morgan [18] used a set of eight metaphors to explore the concept of organization from the viewpoints of different competing theories.

We worked with Emp_Dep as source schema to characterize a structural feature of the Bk_Ed schema, namely the reliance on an identifying relationship to designate instances of weak entities. Many other sources can be brought in to suggest other types of properties and operations; integrity constraints, expressed e.g. in first-order logic notation, could also be added. Here we previously treated books as library items, but clearly they can also be seen as products, merchandises, objects of intellectual property, etc.

On the other hand, the name of the source schema used to derive a certain set of properties of a concept serves to designate a distinct aspect of the concept. Following the orientation prescribed in [11], when performing a problem-solving algorithm of exponential or high polynomial complexity, one can establish that only the properties of the involved entities that have been derived from the one (or the few) designated source(s) will be considered, thereby reducing the computational effort.

4. Categorizations from the Generic and the Blended Spaces

Whereas the patterns at the generic space are preserved to help in the future creation of any number of target schemas, the frames composed at the blended space are only used in connection with a specific source-target pair, and can in principle be discarded after the generation process terminates.

Yet both the generic and the blended spaces, whose role is no more than auxiliary in the derivation of targets from sources, can give rise to new full-fledged conceptual spaces, through a process sometimes called categorization [9]. This is more easily accomplished when generic and blend represent the confluence of spaces associated with the same underlying domain.

Entities employee and student provide an example of this situation, since both have human beings as underlying domain. As a convenience, their corresponding properties can be identically named, so that they can more appropriately be called common properties, to be factored out to characterize a person entity – in a sense, a materialization of the generic space. Both the common and the exclusive properties of employee and student are, in turn, inherited by the trainee entity, which materializes the blended space. In [3] we represented these four entity classes as nodes of the lattice induced by is-a links, and showed that, their properties being so specified, the meet and the join of the frames of employee and student yield, respectively, the frames of person and trainee.

When different underlying domains are involved, categorization can still be envisaged. The resulting blend is then populated with hybrid entities, which may either appear realistic or fantastic, depending on the context. Conflating persons, objects or events is a powerful literary practice, and, surprisingly, offers sometimes intuitive clues to solve problems, as in the Buddhist monk riddle expounded in [11]. A blend conflating persons and books, for instance, might make sense in a cartoon universe, as a Digital Storytelling application aiming to teach children how to use the facilities of a library. Apart from Information Systems, on which the present paper concentrates, and Digital Storytelling, other Computer Science areas such as Software Engineering have drawn significantly from the notions of analogy [4] and blending [12].

5. Concluding Remarks

We were able to run experiments employing the current version of the five-step process, with the help of an interactive logic programming tool. Also, although simple, the weak entity example helped us
gain a better understanding of design by analogy and blending.

Much work remains to be done, especially to extend the process as described in section 2, in order to cope with an ampler variety of conflicts, and to develop semi-automatic algorithms or heuristics to recommend adequate strategies for handling the different situations that may arise in practice.

The topics broadly sketched in sections 3 and 4 should also be included as objectives for future research, aiming at their integration in a more comprehensive treatment of the schema generation problem.

ACKNOWLEDGMENTS
This work is partially supported by CNPq under grants 550250/2005-0 and 311794/2006-8.

REFERENCES