Transformation-based Framework for the Evaluation and Improvement of Database Schemas

Lemaitre, Jonathan; Hainaut, Jean-Luc

Published in:
CAiSE

Publication date:
2010

Document Version
Early version, also known as pre-print

Transformation-based Framework for the Evaluation and Improvement of Database Schemas

Jonathan Lemaitre and Jean-Luc Hainaut

Laboratory of Database Application Engineering - PReCISE research Center
Faculty of Computer Science, University of Namur
Rue Grandgagnage 21 - B-5000 Namur, Belgium
{jle,jlh}@info.fundp.ac.be
http://www.fundp.ac.be/precise

Abstract. Data schemas are primary artefacts for the development and maintenance of data intensive software systems. As for the application code, one way to improve the quality of the models is to ensure that they comply with best design practices. In this paper, we redefine the process of schema quality evaluation as the identification of specific schema constructs and their comparison with best practices. We provide an overview of a framework based on the use of semantics-preserving transformations as a way to identify, compare and suggest improvement for the most significant best design practices. The validation and the automation of the framework are discussed and some clarifying examples are provided.

Keywords: Database schema, schema evaluation, schema improvement, schema transformation.

1 Context and motivation

Modeling activities are increasingly important in software engineering. According to the Model Driven Engineering (MDE) paradigm, database schemas are considered first-class artefacts. Due to the increasing cost of software design flaws, early evaluation techniques, that are quite common in software engineering, also prove necessary in the domain of database schema design. The use of software metrics have long been adapted to schema evaluation. They provide a synthetic measure of the quality of a schema by comparing some of its global characteristics to those of a set of reference schemas. The authors believe that the next step in schema evaluation should be the precise identification of schema defects according to a set of commonly agreed *best practices* that are to ensure specific requirements such as expressiveness, maintainability, evolutivity, performance, etc. From these defects and their scaling against best practices, it should be possible to measure the quality of a schema and to suggest improvement.

Considering these best practices, the scope of the work described in this paper is the evaluation and the improvement of database schemas for both existing

and under development systems. It relies on the use of semantics preserving transformations and on the identification of schema structures that have been defined to express specific application domain fact types. This analysis is used for evaluating the schema by comparing its structural content to a reference frame and the requirements of the schema context. The improvement activity will modify the schema structure in order to increase its compliance with the context while preserving its semantics.

In section 2, we present the conceptual background of our work, specifically the abstraction/paradigm space, the representation of heterogeneous schemas and the concept of schema transformation. In section 3, we define the concepts at the basis of the framework, namely equivalence classes of constructs, best practices, schema context and construct ranking. In section 4, we illustrate the framework through an example of alternative representations of a common data schema construct. In section 5, we suggest a possible application of the framework for the evaluation and the improvement of schemas. In section 6, we provide some hints about the limits identified so far. Section 7 shows how the framework can alleviate the problem of quality framework validation through field case studies. Sections 8 (Related works) and section 9 (Conclusions) are as usual.

2 Background

In this section we briefly (re)define some basic concepts and techniques that will be used to build our framework.

2.1 Abstraction levels, modeling paradigms and semantics

Database engineering processes generally rely on a hierarchy of abstraction levels. Currently called *Model-Driven Engineering*, multi-level approaches have been described since the seventies, when the terms conceptual, logical and physical were coined by several authors in the database realm. A formal description of the information/data structures of a database, be it in construction or in use, is called a *schema*.

A schema is positioned at a certain level of abstraction, depending on the technical detail it provides. In addition, a schema is expressed in a specification language, based on a definite paradigm (table 1). Entity-relationship (ER) with its many variants, UML class diagrams, relational, object-relational, XML, IMS, standard files and even *schema-less*, are some of them. The database community calls them *models* (e.g., the *relational model*), a term we will use in this paper. There is an agreement on which abstraction level a given paradigm best fits. For instance, the Entity-relationship model is as its best at the conceptual level while the object-relational model should be used at the logical level. This defines a two-dimension space in which an arbitrary schema can be located and evaluated. A

schema will be considered *suitable* if the constructs [1] it is made up of comply with the usual way of doing according to the dimensions that define its position in this space.

Table 1. A part of the abstraction/paradigm space.

Abstraction	Paradigms
Conceptual	Historical ER, rich ER, Merise (Individual model), EER, UML class diagrams, OO, Bachman DS, NIAM, ORM, binary ER, etc.
Logical	Relational, object-relational, OO, hierarchical, network, shallow, standard files, XML DTD, UML class diagrams, XML Schema, binary ER, etc.
Physical	Oracle, DB2, SQL Server, PostgreSQL, MySQL, COBOL files, IMS, IDS2, IDMS, Caché, ZODB, ADO.NET, etc.

Though a paradigm is considered suitable at some, but generally not all, abstraction levels (e.g., binary ER), we sometimes observe some cultural border crossover when a construct of one paradigm is used at an unusual position, in a foreign paradigm or in a non standard abstraction level. We mention three examples. A foreign key, which typically is a relational construct, could be found in an ER schema to avoid spaghetti-like schemas, for instance to reference DATE, LANGUAGE or NOTE general-purpose entities from hundreds of places in the schema. Some developers, quite familiar with XML processing tend to build database conceptual schemas that closely resemble a collection of tree structures. Finally, many current databases result from the straightforward migration from an obsolete technology, so that their schemas exhibit structural idiosyncrasies inherited from this technology.

The position of a schema construct in this space must be further refined by considering whether the intention of the designer has been translated adequately. In other terms, given a construct C, does C best expresses its intended semantics? We will call this property *semantic expressiveness*. A simple example will clarify the point. In modern ER models there is a wide agreement on representing sub-categories $C_{11}, C_{12}, \ldots, C_{1n}$ of a reference category C_1 in the application domain by an *is-a* relation between supertype C_1 and subtypes $C_{11}, C_{12}, \ldots, C_{1n}$. However, there are other ways to express these subcategories, such as by distributing the supertype components among the subtype (downward inheritance), by integrating the subtype components within the supertype (upward inheritance) and by representing is-a relation by one-to-one relationship types (is-a materialization). Therefore, there are often several ways to translate the idea of sub-category, but some can be considered more expressive than others.

To summarize, a schema is positioned at a certain level of abstraction, is expressed in a certain paradigm and is intended to translate the intention of the

[1] A construct is a distinct part of a schema considered as a whole for the purpose of the discussion. Typically it is a data structure (entity type, foreign key, is-a relation, the set of columns of a table, index) or a constraint. We consider in this paper constructs whose structure can be defined formally in a generic way, that is, through a *pattern*.

designer. A construct C of this schema can be evaluated through three questions: Does C naturally belong to this paradigm? Does C *feel comfortable* (so to speak) at this abstraction level? Does it best translate the intention of the designer? Our research consists in converting the answers to these questions into a fine-grained evaluation of a schema and into opportunities for improvement.

2.2 Schema expression: the GER model

Considering the multi-dimensional framework described above, we must be able to express non standard schemas that do not meet the ideal rule *only one paradigm at only one abstraction level*. In addition, a transformation can move a construct across abstraction levels and paradigm boundaries. Therefore, we will base the evaluation framework on a large spectrum data model encompassing all the abstraction levels and paradigms, namely the Generic Entity-relationship model, GER in short [1]. The GER is an extended Entity-Relationship model including, among others, the concepts of entity type, domain, attribute, relationship type, method, inheritance, primary and foreign keys, index, as well as various constraints. It also serves as a generic pivot model between the major database paradigms. In fig. 1, we illustrate the graphical notation of the GER objects used afterward. The graphical notation also supports informal notes (yellow boxes), which will be used to provide information that is not expressed structurally. For instance, a foreign key expressed informally through a note will not be declared in SQL-DDL but will be implemented in the application code. Such constructs are called *implicit constructs* [2]. For instance, in fig. 4, schema (e) includes two explicit foreign keys IdA and IdB to A, denoted by keyword `ref`, while in schema (f), these attributes form implicit foreign keys, defined through two informally noted inclusion constraints. Among both foreign key specifications, the first one clearly is of better quality.

An *operational model* M, that is, a model that is actually used in the design environment, can be described by a *GER submodel*, comprising a subset of the GER constructs together with a set of assembly rules that valid schemas must satisfy. A *M-compliant* schema is a GER schema that includes only constructs allowed in M and that satisfies all the assembly rules of M.

2.3 Schema transformation

In this paper, we will address the multiplicity of representations of a given concept. The most appropriate tool to study this phenomenon is the transformational framework according to which a construct C in a schema can be replaced with another construct C' in a way that preserves some characteristics of C. In particularly, we are interested in *semantics preserving*, or *reversible*, transformations that produce constructs C' that model exactly the same application domain situations as C does. A transformation is reversible iff there exists a function g with inverse g' such that, for each valid instance c of C, g(c) is a valid instance of C' and c = g'(g(c)). Provided we have at our disposal an appropriate set of reversible transformation operators, a fairly large collection of

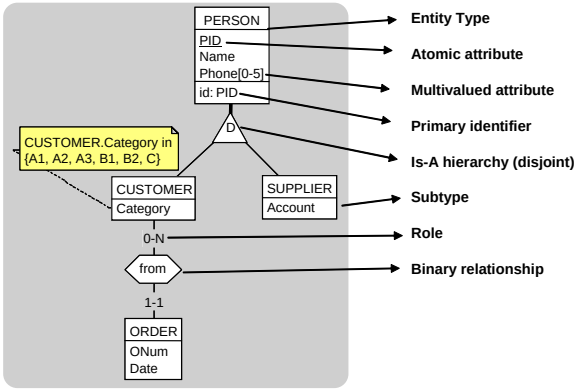

Fig. 1. Sample of GER schema at the conceptual level.

constructs equivalent to C can be generated. The interested reader is referred to reference [1] for a more detailed description of the transformational approach.

3 Definitions

When a designer expresses an application domain fact type in a schema, s/he uses the data model construct that best fits its intention. The construct that most, if not all, skilled designers would choose in this situation is called a *best practice* [2]. Best practices are defined considering the possible **alternative representations** and the **context** in which the schema is used.

3.1 Semantic Equivalence Classes and best practices

Semantic equivalence class of a construct. We consider K, the collection of all the constructs of the GER that are pertinent in some engineering processes and a set of transformations T. Let us also consider a construct C from K and all the equivalent constructs that can be derived through the reversible transformations of T. All these constructs, together with C, form an equivalence class called $ec(C)$. Since only reversible transformations have been applied, $\forall C' \in ec(C), ec(C') = ec(C)$. We now consider the function $sec : K \rightarrow (K \times 2^K)$ which associates to each construct in K its semantic equivalence class (sec). $sec(C)$ is an equivalence classes in which the specific element C has been tagged. We call C the *intention* of this equivalence class. $sec(C)$ provides all the constructs a designer can introduce in a schema to express the semantics (the application domain fact type) of C, hence the name *semantic equivalence class* or *sec*.

[2] Intuitively, a best practice is a common practice among skilled designers.

The concept of best practice. Let us consider a structure comprising a category A together with two of its sub-categories $A1$ and $A2$. It typically translates into an *is-a* relation construct. However, under certain circumstances, other equivalent constructs can be used instead as we have shown it in section 2.1. If we call C the *is-a* relation, $sec(C)$ includes all the constructs that express category/sub-category structures, including C itself. C is the intention of its semantic equivalence class. For a seasoned designer, the *is-a* relation is the preferred translation but others can be used instead, though with a lower preference level. To better describe the notion of preference, we assign to each member C' of $sec(C)$ a preference score expressing the extent to which an expert designer will accept to use C' to express the semantic of C. The higher the score, the better C' will be to express this semantics. The preference score can be defined by a number or, more simply, by a partial order relation ($C" < C'$ if C' is preferred to C'' to express the semantics of C). However, as we will discuss it later on, the preference scoring of *sec* is context dependent. We will call *best practices* of $sec(C)$ the constructs with the highest preference score. It must be noted that, depending on the context, C may not be the best practice in $sec(C)$.

Generation of SEC. The equivalence class of a construct C can be obtained by recursively applying the transformations of T until no new construct can be produced. However, this naive approach can lead to a very large (and, depending on T, possibly infinite) set of constructs of which only a small subset would be of interest. Appropriate meta-rules are necessary to keep the process into reasonable limits. Considering the *is-a* pattern, one can adopt a *regularity of treatment* meta-rule according to which each sub-category of a given category must be expressed in the same way. For example, a construct obtained by applying the upward inheritance transformation to one sub-category and the materialization transformation to another one would be rejected. Another example: when an entity type EA results from the transformation of an attribute A, the attribute(s) of the latter cannot be further transformed through the same transformation (figure 2).

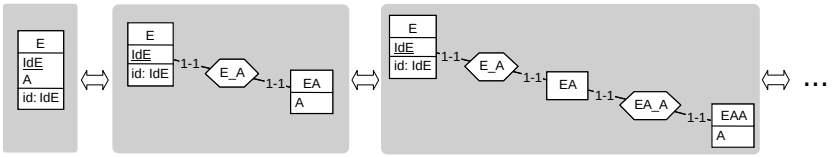

Fig. 2. Infinite transformation of an attribute.

3.2 Context and best practices

The context of a schema S under evaluation is the external conditions that defines its intended use. S has been designed for the abstraction level A, according

to the paradigm P and to meet design criterion D [3]. We call (A, P, D) the *context* of S. Given a construct C that can appear in schema S, a scoring function is assigned to $sec(C)$ for a given context.

Any model M is a point in the abstraction/paradigm space illustrated in table 1 (e.g., $(conceptual, "binaryER")$, $(logical, "SQL3")$). The members of $sec(C)$ comply with the GER, but, being independent of any abstraction level and paradigm, they may not all comply with M. So, we introduce the concept of *projection* of a set of constructs on a model. Let us consider $K_M \subseteq K$ and model M, such that K_M is the set of constructs that are valid in M. K_M is the projection of K on M. The projection can be applied to semantic equivalence classes. We note $sec_M(C)$ the subset of constructs in sec that are M-compliant, i.e., $sec_M(C) = sec(C) \cap K_M$.

Now, we classify the members of $sec_M(C)$, $(\forall C \in K_M)$ according to the evaluation criterion D. We assign (in a way that will be discussed later) each member of $sec_M(C)$ an order number, or, best, a numeric score in the range $[0-1]$. The first member(s) being the most appropriate according to D and the last one(s) the worst one(s). As we have said above, the intention can be, but need not be, the first member. For instance, in the sec of a foreign key at the conceptual level, *the one-to-many* relationship type will probably be assigned the highest score for $D = Expressivity$. Figure 3 describes a simple sec of unnormalized relation R for the SQL2 logical model. Considering access time optimization criterion, schema **(c)** can be assigned the highest score, since it is optimized for $R * S$ join while being easy to implement through a (non minimal) foreign key [4]. It is followed by schema **(a)**, itself followed by schema **(b)**. Of course, the $Normalization$ criterion would have yielded quite different scores.

Considering construct C, the *bestpractice* for C in a given context (A, P, D) is the member (or members) of $sec(C)$ with the highest score.

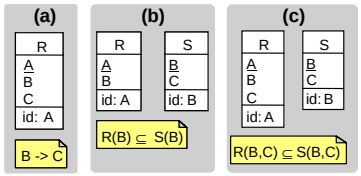

Fig. 3. (a) Relational schema with functional dependency. **(b)** Normalized schema. **(c)** Optimized schema.

[3] For simplicity, we consider that a schema is to meet one design criterion only.
[4] This pattern is known as the *Elementary Key Normal Form*.

4 Illustration

In this section, we illustrate the concepts of semantic equivalence class and schema context.

Figure 4 represents eight typical constructs that translate the category/subcategory structure where subcategories are pairwise disjoint. In these subschemas, `AttX` stands for *a set of attributes of the entity type X*; `AttX[0-1]` means that all these attributes are optional; tag `id` in the 3rd compartment declared a primary key; `coex` means that the attributes must all be null or all not null; `exact-1` means that exactly one attribute must be not null; `ref` declared a foreign key. Since constructs **(b)** to **(h)** can be derived from construct **(a)** through semantics-preserving transformations, this set can be considered the *sec* of any of its members, in particular **(a)**. The constructs **(b)**, **(c)** and **(d)** are common alternative representations of $is-a$ relations that are obtained respectively with the *materialization*, *downward inheritance* and *upward inheritance* transformations. **(e)** and **(f)** are derived from **(b)**. In **(e)**, the primary/foreign keys translate the relationship types of **(b)**. In **(f)**, the foreign keys are not declared but the referential constraints are expressed informally. **(g)** and **(h)** are obtained by the transformation of **(d)**. In ER-like models, **(b)** will be considered the most expressive construct.

Let us now project this set of constructs on some point in the (A,P) space, namely: conceptual extended entity-relationship, logical IMS, logical relational (SQL2) and logical object-relational (SQL3). In table 2, we indicate the results of the projections on these four models. For convenience, we consider that models EER, SQL2 and SQL3 each include a language or a mechanism (OCL-like, check, triggers) to express `coex` and `exact-1` constraints.

Table 2. Suitability of the structures of fig. 4 in 4 projections.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
Conceptual EER	X	X	X	X	-	X	X	X
Logical IMS	-	X	X	-	-	X	-	X
Logical Relational	-	-	X	-	X	X	X	X
Logical Object-relational	X	-	X	-	X	X	X	X

One can observe that nearly all structures of the *sec* are compatible with the EER model. The only exception is the foreign key. The IMS model is the most restrictive, as it imposes a strongly constrained tree-like structure between entity types (called *segment types* in IMS vocabulary). The relational model accepts five constructs. Its restrictions come from unsupported object types: relationship types, is-a relations and complex attributes. Finally the object-relational model is able to represent all *sec* constructs but relationship types and compound attributes.

We can classify the constructs of the four *sec* resulting from these projections according to a *fitness* criterion combining *expressivity* (specially for EER) and *ease of implementation* (specially for logical models). The goal of such ranking

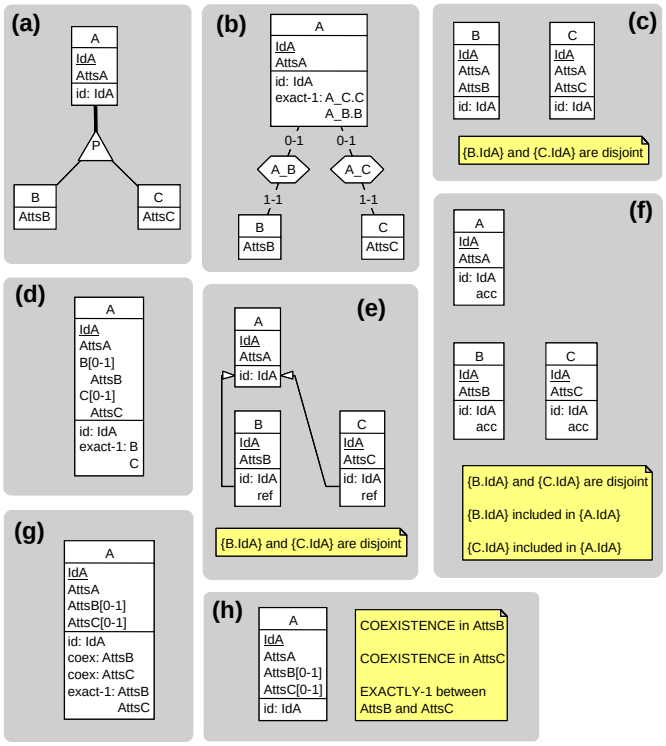

Fig. 4. SEC representing the category/subcategory with a partition constraint.

is to associate a score to the constructs, that will help for the evaluation of the schema and in the choice between alternatives. There are different ways to determine the scores. Among them, we are turning toward the collection of expert opinions. It aims at placing the structure on a quality scale. We apply the *fitness* criterion after a projection on the logical relational model. As a rough guide, we place the remaining structures in a 5 grades ordinal scale (very bad, bad, average, good and very good) as follow: **(h)** < **(f)** < **(c)** < **(g)** < **(e)**. Obviously, the quality rating can be based on a numerical scale, in order to provide a more precise classification.

5 Framework application

An important issue in software engineering is the evaluation of the quality of a system. Software metrics have long been the favourite techniques to evaluate this quality. Though they are not always easy to interpret, metrics are easy to compute and they provide an immediate quantitative result [3–5]. However, when applied to database schemas, they are generally based on counting atomic ob-

jects such as entity types, tables, attributes, etc. independently of their intended meaning.

The approach described in this paper makes it possible to integrate semantic aspects in qualitative and quantitative evaluations. A qualitative evaluation requires a classification based of a simple order relation among the members of each *sec*. Such classification is easy to achieve. However it will neither provide a global score for the schema quality, nor allow one to quantify the difference between two schemas. It allows one to identify construct instances, that are not a *best practice* and that can be improved, together with the possible changes. The quantitative evaluation requires a classification based on a more or less precise numeric scale. The grading will obviously require more effort from experts. Though defining metrics, their computation and their interpretation are beyond the scope of this paper, we will suggest two metrics to develop quantitative evaluations based on this framework.

The first suggested metrics is the **Individual sec evaluation**. It computes the average quality score of a schema S for a particular *sec*, defined by its intention I. We note $sec(I)$ the semantic equivalence class, M the model of S and D the criterion according to which $sec_M(I)$ has been classified. The metrics is defined as follows:

$$IndividualScore_{sec_M}(S, I) = \frac{\sum_{c \in sec_M(I)} \sharp\{inst(c,S)\} \times score_D(c)}{\sum_{c \in sec_M(I)} \sharp\{inst(c,S)\}}$$

where $inst(c, S)$ is the set of all instances of the construct c in the schema S and $score_D(c)$ is the score of c according to criterion D.

The second metric is the **Global sec evaluation**. It evaluates the average score of a schema in a particular context using the *sec* of the most significant constructs of M. We note II the set of these significant constructs. We associated a weight to each construct I (and its corresponding *sec*) in order to define their importance. We define normalized weights $weight(I)$ in the range $[0-1]$ so that their sum is equal to 1. The metric is defined as:

$$GlobalScore_M(S) = \sum_{I \in II} weight(I) \times IndividualScore_{sec_{(M)}}(S, I)$$

The evaluation of the schema through the identification of deviations from best practices naturally leads to its improvement according to a definite criterion D. The improvement process takes into account the classification of the *sec* of the most significant constructs (II) in order to maximize the schema quality. Though the specification of a complete improvement method is beyond the objective of this paper, we will sketch a tentative heuristics: for a specific $I \in II$, we transform each instance of I into the instance of $sec_M(I)$ that has the highest score. Figure 5, represents the *sec* of *is-a* relation expressions of fig. 4 projected on the SQL2 model, that is, $sec_{SQL2}(C_{is-a})$. Edges represent standard reversible transformations, acting as improvement paths, and scores translate numerically from 1 to 5 the *very bad* to *very good* scale of section 4. In some cases, the improvement path uses transformations that lead out of sec_M. In this case, the transformation chain has to form a path that goes back in the projected *sec* (e.g., transform **(c)** into **(e)**).

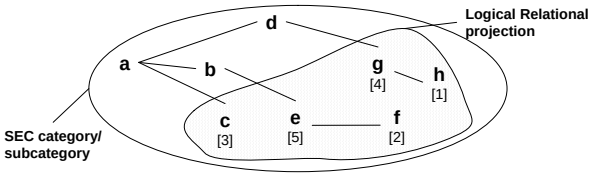

Fig. 5. Transformation associated to a semantic equivalence class.

In the remainder of this section, we apply this approach on a small schema. In fig. 6, we represent 2 equivalent EER schemas. We identify in schema **(a)** two significant constructs which, considering the expressiveness criterion, are of poor quality (they have a low quality score in their respective *sec*). The optional attribute `DiscontinuedDate` is the date from which the product has been discontinued. Its *sec* contains, among others, a representation based on an *is-a* relation, suggesting that the attribute represents a specific category of product. We note C_{Att} the attribute construct and C_{isa} its is-a alternative. The construct comprising entity types `DETAIL`, `DET_ORD` and `DET_PRO`, together with their relationship types, is a complex but valid expression of a single many-to-many relationship type very common in some legacy IMS databases. We note C_{RTs} this complex construct and C_{MM} the *many-to-many* relationship type.

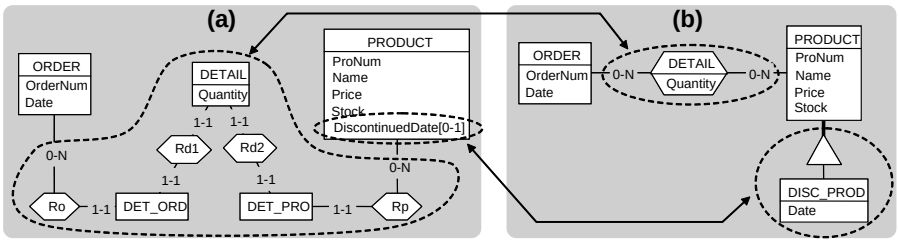

Fig. 6. Evaluation and improvement of a conceptual EER schema.

For the purpose of the example, we consider the same 1 to 5 coarse-grained classification as above. The classification assigns the score 2 to C_{Att} and 5 to C_{isa}. C_{RTs} receives the score 1, while C_{MM} gets a 5. C_{isa} and C_{MM} are the only constructs in their *sec* with the maximum score. These scores allow to compute the *IndividualScore* metric. The results are trivial since they correspond to the score of the only construct of the *sec* present in the schema. As a rough guide, we assign C_{Att} and C_{RTs} respectively scores 0.4 and 0.6. The results are the following: $GlobalScore_{(EER,expressiveness)}(a) = 0.4 \times 2 + 0.6 \times 1 = 1.4$ and $GlobalScore_{(EER,expressiveness)}(b) = 0.4 \times 5 + 0.6 \times 5 = 5$. The expressiveness of **(a)** tends to be very bad while the score of **(b)** is maximum. The improvement process transforms **(a)** into **(b)**.

6 Limits

The framework described in this paper is under evaluation through a collection of case studies. Though is is too early to draw definitive conclusions and to specify precisely its application domain, we have identified some issues that will be addressed in the near future. We will mention two of them.

Interactions are possible between the constructs of two distinct *sec*. We distinguish two types of interactions. In the first one, a construct appears in different SEC. In this case, an in-depth analysis has to be done in order to determine the very purpose of the structure. This can be performed using reverse engineering methods, and particularly the conceptualization step, through which the semantics of a technical construct is elicited. This implies that the identification of some *sec* construct instances may not be fully automated and requires human intervention. This problem is illustrated in the Fig. 7, where the 1-to-1 relationship type can be involved in an category/subcategory relation, a relation between two different concepts (employee-car) or a concept fragmentation.

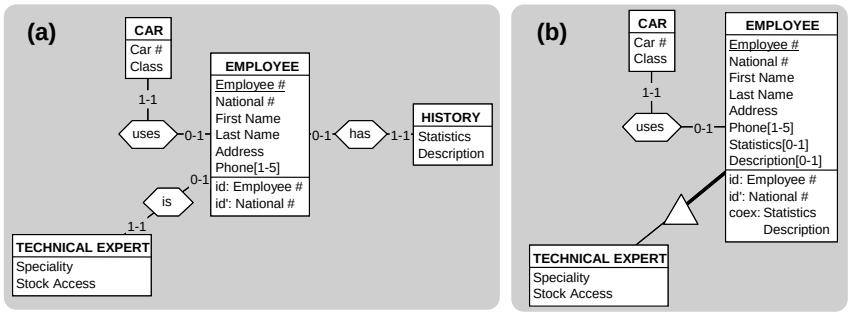

Fig. 7. Relations between constructs of different semantic equivalence classes.

The second type of interactions concerns structures that have a common part or structures included in others. We use the structure **(a)** in the fig. 4 in order to illustrate the problem. The attribute `AttsA` can be transformed into an entity type independently of entity type `A` and the is-a relation. Such transformation is used to extract a concept included in an other. The construct involved in this transformation should only take into account entity type `A` and its internal components (excluding the *is-a* relation) and is included in **(a)**. As in the previous case, such problems may require some interactions with a designer.

As a second example of challenge already identified, it appears that addressing several criteria may lead to a conflicting situation. For example, the *minimality* criterion may contradict "expressivity". *Normalization* versus *optimization* is another popular example.

7 Validation of the framework

The validation and tuning of an evaluation framework is such a complex, costly and time-consuming task that it is seldom carried out according to standard validation methodologies. Ideally, one should collect dozens of real schemas of realistic size, that is, including at least 40-50 entity types or tables, and submit them to dozens of experts in data modeling who are asked to evaluate them according to various quality criteria such as expressivity, readability (for different stakeholders), maintainability, evolvability, time/space performance, DBMS-independence, etc. Considering that good experts are both scarce and very busy, proceeding in this way is quite unrealistic. As a consequence, many proposals are tested on closed systems comprising a teacher in IS design together with his/her students, a procedure sometimes considered unreliable.

The framework described in this paper makes it possible to lower the cost of the validation process.

1. First, the framework relies on the concept of *sec*, the generation of which can be, to a large extent, automated. However, the ordering of their members must be performed by experts. We can formulate two observations. (1) Though K can be an infinite set of patterns (depending on T), practice has shown that only a subset of about 20 constructs is sufficient to built most real schemas. (2) This provides us with about 20 *sec*, most of them comprising from 5 to 10 members. Ordering these sets of constructs is much easier and more deterministic than evaluating complete schemas. According to a first experiment involving four high level industrial experts[5], assigning value scores to one *sec* according to one criterion (e.g. expressiveness) takes about 15 minutes and the results show very little variation among experts. Processing the most useful *sec* for one major criterion requires less than one day per expert, so that the complete parametrization of the framework can be performed in a matter of one or two weeks.
2. Evaluating a schema according to a set of ordered *sec* is an automatic task. Identifying constructs and their intention depends on the (still unknown) quality of the schema, so that a conceptualization step may be necessary. This process is automatic to a large extent[6].
3. It remains to check the validity of the framework. Here, relying on teachers and students makes sense. (Last year) students form a realistic sample of designers of various skills, ranging from desperately inapt to experienced and ingenious. On the other hand, teachers are expected to be expert in evaluating the quality of medium size schemas. Therefore, comparing and aligning academic and automated evaluations allow the tuning of the evaluation framework. These validation and alignment processes still are under investigation.

[5] from the ReveR company, specialized in database reengineering.

[6] the DB-MAIN CASE tool we have been developing since 1993 includes programmable assistants that support this process.

8 Related Work

In the context of data schemas, very few authors seem to have explored the use of reversible transformations to deal with schema quality. Among them, Codd proposed the concept of relational normalization [6]. The normalization process relies on the use of transformations in order to eliminate problematic functional dependencies. Compared with our framework, it deals with a *no redundancies* criterion. An early synthesis of the existing normal forms was proposed by Kent [7]. Another proposal was made by Assenova and Johanesson [8], who used reversible transformations to increase the understandability of the conceptual models, a criterion they decomposed into smaller quality criteria. In their work, a qualitative quality indicator is associated to transformations for each criterion. In our framework, we choose to relate the quality to the structure itself and develop more precise indicators. Burton and Weber [9] realized a study on the use of attributes in relationship types and its impact on schema clarity. Their observations were based on equivalent schemas. A similar work was carried out by Gemino and Wand [10] on the use of mandatory properties and subtypes on ER schemas. Both proposals deal with schema quality and propose solutions to increase it, but none highlights the use of reversible transformations. Outside the context of data schemas, Bouhours et al. proposal focuses on the transformation of software architecture according to quality requirements [11]. Their transformations consist in applying design patterns that best satisfy the requirements (another name for design criteria). Finally, Kurtev [12] uses the concept of transformation space for dealing with schema quality. Such space represents a transformation by its initial and resulting structures and allows to link it with quality indicator. However, studied objects are atomic, while we consider semantically richer constructs.

9 Conclusion

The framework described in this paper intends to improve the precision and the automation of the evaluation of a database schema according to a definite criterion. Built on the transformational paradigm, it provides a sound and rigorous basis to develop evaluation strategies (including metrics-based ones) and improvement techniques. In particular, it makes explicit and implements the idea that a designer chooses, among a collection of candidate constructs (semantic equivalence class), which best fits its intention, that is, the fact type from the application domain. A defect in a schema occurs when this choice does not prove to be optimal. The framework makes it possible to identify this collection and the best choice, called *best practice*.

The framework also shows the importance of the three components of the context of a schema: the level of abstraction, the paradigm (that both form the data model) and the design criterion.

At the present time, we are parameterizing and validating the framework through practical case studies and with the help of a community of expert designers. We intend to compare the results of our framework with synthetic metrics-based approaches. In addition, we are developing a tool, built on the DB-MAIN platform, to identify significant patterns in a schema, to associate with each of them a quality score according to a definite criterion and to suggest improvement transformations.

References

1. Hainaut, J.L.: The transformational approach to database engineering. In Lämmel, R., Saraiva, J., Visser, J., eds.: Generative and Transformational Techniques in Software Engineering. Volume 4143 of LNCS., Springer (2006) 95–143
2. Cleve, A., Lemaitre, J., Hainaut, J.L., Mouchet, C., Henrard, J.: The role of implicit schema constructs in data quality. In: Proc. of the International Workshop on Quality in Databases and Management of Uncertain Data, Auckland, New Zealand. (2008) 33–40
3. Genero, M., Piattini, M., Manso, M.E.: Finding "early" indicators of uml class diagrams understandability and modifiability. In: ISESE, IEEE Computer Society (2004) 207–216
4. Manso, M.E., Genero, M., Piattini, M.: No-redundant metrics for uml class diagram structural complexity. In: CAiSE. Volume 2681 of Lecture Notes in Computer Science., Springer (2003) 127–142
5. Si-Said Cherfi, S., Akoka, J., Comyn-Wattiau, I.: Perceived vs. measured quality of conceptual schemas: An experimental comparison. In: ER (Tutorials, Posters, Panels & Industrial Contributions), Australian Computer Society (2007) 185–190
6. Codd, E.F.: Normalized data structure: A brief tutorial. In: SIGFIDET Workshop, ACM (1971) 1–17
7. Kent, W.: A simple guide to five normal forms in relational database theory. Commun. ACM **26**(2) (1983) 120–125
8. Assenova, P., Johannesson, P.: Improving quality in conceptual modelling by the use of schema transformations. In: ER '96: Proc. of the 15th International Conference on Conceptual Modeling, London, UK, Springer-Verlag (1996) 277–291
9. Burton-Jones, A., Weber, R.: Understanding relationships with attributes in entity-relationship diagrams. In: ICIS '99: Proc. of the 20th international conference on Information Systems, Atlanta, GA, USA (1999) 214–228
10. Gemino, A., Wand, Y.: Complexity and clarity in conceptual modeling: comparison of mandatory and optional properties. Data Knowl. Eng. **55**(3) (2005) 301–326
11. Bouhours, C., Leblanc, H., Percebois, C.: Alternative models for a design review activity. In: Proc. of the 2nd workshop on Quality in Modeling, Nashville, TN (USA), Springer (2007) 65–79
12. Kurtev, I.: Adaptability of model transformations. PhD thesis, University of Twente, Enschede (2005)