PATROCINADORES

ugr Universidad de Granada

ATC

Sun microsystems

IBM

Bull

Ayuntamiento de Granada

gtj

cibersur

Asociación Española para la Inteligencia Artificial.

Asociación Española de Reconocimiento de Formas y Análisis de Imágenes.

AIPO http://gritva.uall.es8080/aipo
Asociación Interacción Persona-Ordenador.

EUROGRAPHICS http://www.eg.org
Capítulo Español de la European Association for Computer Graphics.

EUSFLAT http://www.eusflat.org/index.htm
European Society for Fuzzy Logic and Technology.

SC of the IEEE CIS
Capítulo Español de la IEEE Computational Intelligence Society.

RADISC
Red Andaluza en Sistemas Complejos.

SEPLN http://www.sepln.org
Sociedad Española para el Procesamiento del Lenguaje Natural.

TIN - MEC
Programa Nacional de I+D en Tecnologías Informáticas. Ministerio de Educación y Ciencia.

CEA-IFAC http://www.cea-ifac.es
Comité Español de Automática de la International Federation of Automatic Control.

ISTANET http://www.istanet.net
Red Andaluza de Tecnología de Sistemas Inteligentes.

W3C http://www.w3c.es
Consorcio World Wide Web. Oficina Española.

RADI-AEB
Red Andaluza de Algoritmos Evolutivos y Bioinspirados.

ISBN: 84-9732-434-X

CEDI 2005 X Jornadas de Ingeniería del Software y Bases de Datos [JISBD'2005]

THOMSON

CEDI 2005

I CONGRESO ESPAÑOL DE INFORMÁTICA

GRANADA DEL 13 AL 16 DE SEPTIEMBRE

X Jornadas de Ingeniería del Software y Bases de Datos

[JISBD'2005]

EDITORES
Ambrosio Toval Álvarez • Juan Hernández Núñez

ACTAS DE LAS

X Jornadas de Ingeniería del Software y Bases de Datos

[JISBD'2005]

EDITORES
Ambrosio Toval Álvarez
Juan Hernández Núñez

JORNADAS ORGANIZADAS POR
Sociedad de Ingeniería del Software y Tecnologías de Desarrollo de Software

THOMSON

Actas de las X Jornadas de Ingeniería
del Software y Bases de Datos [JISBD'2005]

Editores de la serie de Actas del CEDI
Rafael Molina Soriano
Antonio Díaz García
Alberto Prieto Espinosa

Editores de las Actas de las presentes Jornadas
Ambrosio Toval Álvarez
Juan Hernández Núñez

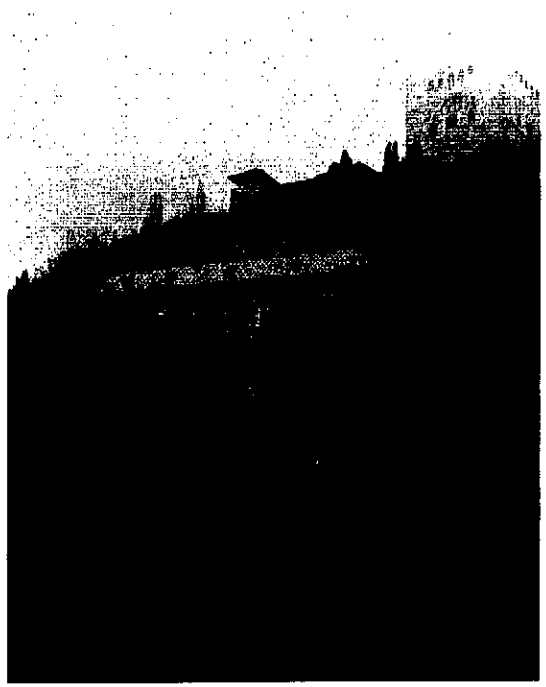
Diseño de Cubiertas
dixi
www.dixi-e.com

Impresión
THOMSON

COPYRIGHT© 2005 International
Thomson Editores Spain
Paraninfo, S.A.
Magallanes 25 · 28015 Madrid España
Tel: 91 446 33 50 · Fax: 91 445 62 18
clientes@paraninfo.es

Impreso en España
Printed in Spain

ISBN: 84-9732-434-X
Depósito legal: SE-4046-2005 European Union
Printed by Publidisa



Antiguo Hotel Reuma. Granada
Foto realizada para el CEDI2005 por DIXI

X JORNADAS DE INGENIERÍA DEL SOFTWARE Y BASES DE DATOS

COMITÉ EJECUTIVO

Presidente Comité de Programa	Ambrosio TOVAL (Universidad de Murcia)
Secretario Comisión Permanente	Mario PIATTINI (Universidad de Castilla-La Mancha)
Organización y Relaciones con CEDI'2005	Buenaventura CLARES (Universidad de Granada) Patricia PADEREWSKI (Universidad de Granada)
Coordinador de Tutoriales	José L. FERNÁNDEZ (Universidad de Murcia)
Coordinadora de Talleres	Patricia PADEREWSKI (Universidad de Granada)
Coordinador de Demostraciones	Francisco L. GUTIÉRREZ (Universidad de Granada)

PRESIDENCIA DEL COMITÉ PROGRAMA

Ambrosio TOVAL — Universidad de Murcia

COMITÉ DE PROGRAMA

J. ALDANA (U. Malaga)
B. ÁLVAREZ (U. Politécnica Cartagena)
J. ARAUJO (U. Nova de Lisboa)
M. J. ARAMBURU (U. Castellon)
O. BELO (U. do Minho)
P. BOTELLA (U. Politécnica Cataluña)
N. BRISABOA (U. de La Coruña)
C. CALERO (Castilla-La Mancha)
C. CANAL (U. de Málaga)
J.M. CAVERO (U. Rey Juan Carlos)
M. CELMA (U. P.Valencia)
R. CORCHUELO (U. de Sevilla)
Y. CRESPO (U. de Valladolid)
C. DELGADO (U. Carlos III)
O. DÍAZ (U. Politécnica País Vasco)
J. FALCÃO e CUNHA (U. do Porto)
X. FRANCH (U. Politécnica Cataluña)
P. de la FUENTE (U. de Valladolid)
L. FUENTES (U. de Málaga)
M. J. GASPAR da Silva (U. de Lisboa)
J. GÓMEZ (U. de Alicante)
J. HERNÁNDEZ (U. de Extremadura)
J. ITURRIOZ (U. Politécnica País Vasco)
N. JURISTO (U. Politénica de Madrid)
A. LOPES (U. de Lisboa)
H. MADEIRA (U. de Coimbra)
E. MARCOS (U. Rey Juan Carlos)
J.M. MARQUÉS (U. de Valladolid)
E. MENA (U. Zaragoza)
A. M. MORENO (U. Politénica de Madrid)
J. J. MORENO (U. Politécnica Madrid)
J. M. MURILLO (U. de Extremadura)
N.J. NUNES (U. Madeira)
O. PASTOR (U. Politécnica Valencia)
E. PIMENTEL (U. de Málaga)
A.POLO (U. Extremadura)
C. RAMOS (U. do Algarve)
I. RAMOS (U. Politécnica Valencia)
J. RIQUELME (U. Sevilla)
A. RITO (U. Técnica de Lisboa)
M.J. RODRÍGUEZ (U. Granada)
F. RUIZ (Castilla-La Mancha)
J. SAMOS (U. Granada)
F. SÁNCHEZ (U. de Extremadura)
J. SÁNCHEZ (U. Politécnica Valencia)
S. SOUSA BRITO (I. Politécnico Beja)
E. TENIENTE (U. P. Cataluña)
M. TORO (U. de Sevilla)
J. C. TRUJILLO (U. de Alicante)
B. VELA (U. Rey Juan Carlos)
A. VALLECILLO (U. de Málaga)

REVISORES ADICIONALES

Alberto ABELLÓ	Ma. Valeria de CASTRO
Alfredo GOÑI	Manuel RESINAS
Andre L. SANTOS	María del Mar RODÁN
Ángel HERRANZ	María Luisa RODRÍGUEZ
Antonio Cesar GÓMEZ	María-Isabel SÁNCHEZ-SEGURA
Antonio RUIZ	Marta RUIZ
Arantza ILLARRAMENDI	Miguel Ángel LAGUNA
Artur BORONAT	Nathalie MORENO
Bruno MARTINS	Nelson MEDINILLA
Carlos E. CUESTA	Noelia MAYA
Carme QUER	Norberto FERNÁNDEZ
Daniel GOMES	Nuria MEDINA
Enric MAYOL	Oscar DIESTE
Fran J. RUIZ-BERTOL	Pablo FERNÁNDEZ
Francisco COUTO	Paloma CÁCERES
Francisco Luis GUTIÉRREZ	Patricia PADEREWSKI
Ismael Navas DELGADO	Patricio LETELIER
Javier MUÑOZ	Pedro J. CLEMENTE
Jennifer PÉREZ	Pedro J. MUÑOZ
Joan Antoni PASTOR	Pedro SÁNCHEZ
Joao Pedro NETO	Pedro VALDERAS
José Luis GARRIDO	Rafael BERLANGA
José Miguel BLANCO	Raul ROMERO
José Miguel CAÑETE	Sira VEGAS
José Ramon RIOS	Toni URPÍ
Juan Ángel PASTOR	Vicente LUQUE
Juan Manuel VARA	Vicente PELECHANO
Luis SÁNCHEZ	Xavier FERRÉ

COORDINADOR DE TALLERES

Patricia PADEREWSKI	Universidad de Granada

MIEMBROS COMITÉ DE TALLERES

Mª Visitación HURTADO	Universidad de Granada
Juan Manuel MURILLO	Universidad de Extremadura
José SÁEZ	Universidad de Murcia
Mª Dolores LOZANO	Universidad de Castilla La Mancha
José Hilario CANÓS	Universidad de Valencia
Cecilia DELGADO	Universidad de Granada
Amador DURÁN	Universidad de Sevilla

COORDINADOR DE TUTORIALES

José L. FERNÁNDEZ	Universidad de Murcia

COORDINADOR DE DEMOSTRACIONES

Francisco L. GUTIÉRREZ	Universidad de Granada

SISTEMA AUTOMÁTICO DE REVISIÓN

Quercus Software Engineering Group

Pablo AMAYA	Universidad de Extremadura
Jose M. CONEJERO	Universidad de Extremadura

CONTENIDOS

Finding where to apply object-relational database schema refactorings: an ontology-guided approach

Aline Lúcia Baroni
Universidade Nova de Lisboa
FCT/DI/CITI
QUASAR Research Group
2825-114 Monte da Caparica,
Portugal
alinebaroni@di.fct.unl.pt

Fernando Brito e Abreu
Universidade Nova de Lisboa
FCT/DI/CITI
QUASAR Research Group
2825-114 Monte da Caparica,
Portugal
fba@di.fct.unl.pt

Coral Calero
Universidad de Castilla-La Mancha
Computer Science Department
ALARCOS Research Group
Paseo de La Universidad, 4
13071 Ciudad Real, Spain
coral.calero@uclm.es

Abstract

Less complex object-relational (OR) database schemas are more understandable, changeable, maintainable and reusable. This paper addresses the use of OR metrics to (i) detect complex fragments of OR database schemas, which are candidates to perform refactorings, and (ii) to assess if the application of the selected refactorings has indeed reduced the schema complexity. The schema metrics are formally defined with OCL, upon an ontology of the SQL:2003 standard, expressed as a UML class diagram. This formalization allows automating metrics collection. Although our approach does not point out how to select the most adequate refactoring transformation, it helps designers and developers to pinpoint the spots in the database schema where a refactoring is likely to produce a quality improvement. An illustrating example is included.

1. Introduction

To improve the quality of a released software product, we need to evaluate quantitatively – by means of software measurement - its external and internal characteristics. The former can be used to conceive quality models from the user's point-of-view. The latter can be used in several ways, such as promoting best practices on the design phase, directing verification and validation efforts or re-engineering processes.

A considerable number of studies has shown that software measurement can help enhancing software development products and processes, namely by allowing to detect high complexity parts [15], to forecast and plan testing efforts [12, 19] and to estimate maintenance efforts [6, 20]. In the databases area, metrics have been used to assess schema's quality [3, 7].

One of the most important software quality characteristics is maintainability [9], due to its economical impact. In fact, a considerable share of software development is devoted to software maintenance [14, 18]. Since databases are found on most software systems, namely in the MIS area, the maintainability of database schemas is a relevant research area. Our previous work on this area includes the formal definition of OR database metrics [2]. Our current work is an attempt to reduce database maintenance effort, applying the formalized set of metrics to database schemas in order to guide and assess the result of applying transformations that can improve quality, by reducing schemas complexity.

The research domain that addresses the improvement of software quality, while reducing its complexity through its internal reformulation is referred to as *refactoring* [21] and it has raised a particular interest in the field of object-oriented design and programming. In [21] refactoring is defined as "*the process of changing an object-oriented software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure*".

In this work, we are particularly interested in refactoring database schemas. A database refactoring is "*a simple change to a database schema that improves its design, while retaining both its behavioral and informational semantics*" [16].

The cornerstone of these definitions is the redistribution of elements that compose a piece of software (classes, methods, attributes, tables, columns, etc) in a way which can facilitate future adaptations and extensions, thus making the evolution of software easier. However, choosing where to apply a refactoring is often based on subjective criteria, rather than on objective ones.

In this paper we propose a metrics-based approach for assisting database schema refactorings. It combines previous research results (OR schema metrics, SQL:2003 ontology and guidelines for refactoring) in the following way: (i) we instantiate the database schema ontology for a given OR schema; (ii) we calculate automatically the values of the OR schema metrics; (iii) with those metrics we identify candidate schema fragments for refactoring; (iv) the refactorings are applied by following the corresponding guidelines; (v) finally, we repeat the first two steps for the refactored schema, in order to ascertain that the transformation has produced the expected result (complexity reduction).

This paper is organized as follows. Section 2 introduces a framework to collect database schema metrics. Section 3 outlines the metrics used to detect refactoring-prone database schema fragments, as well as the ontology proposed for the SQL:2003 standard. Section 4 describes the steps required for applying database refactorings. Section 5 illustrates these steps with a simple example. Section 6 presents our conclusions.

2. A framework for collecting OR database schema metrics

Our approach to formalize metrics definitions (see figure 1) uses OCL (Object Constraint Language) [13], a part of the UML OMG (Object Management Group) standard, and the ontology developed [4] to represent the newest ISO SQL:2003 standard [8]. This approach is an extension of the one we developed previously, concerning the extraction of metrics for object-oriented design models [1, 5].

The SQL:2003 ontology is represented as a UML class diagram, built with the aid of a UML modeling tool (1), where well-formedness rules are enforced by the use of OCL invariants. The OR database schema metrics are expressed as OCL operations (selectors) upon the ontology classes (2). The limitations on the applicability of those metrics are formalized as OCL preconditions upon the corresponding operations. A diagram translator, built upon the UML modeling tool API, converts the metrics and the ontology to a textual format (3) suited for input in an OCL expression evaluator. By means of a SQL:2003 compliant modeling tool, OR database schemas are produced (4). To enable metrics extraction, the SQL:2003 ontology must be instantiated with a concrete example (a OR schema). Using the SQL:2003 modeling tool API, a generator creates the database schema meta-data in textual format to populate the ontology (5). Finally, the OCL expressions evaluator allows collecting the formalized metrics for the concrete example (6).

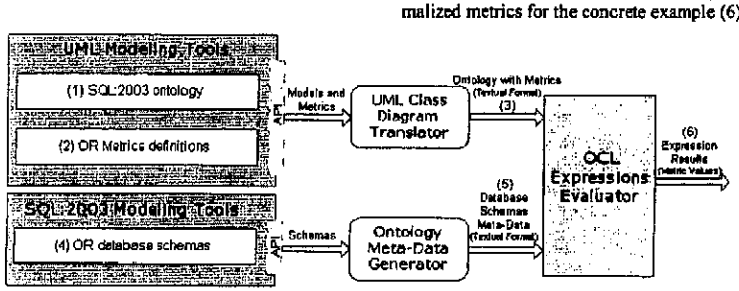


Figure 1. The architecture of our approach

Summarizing, the main benefits of this OCL-based metrics formalization are:

- the metrics definitions are unambiguous, due to its definition upon the ontology (that provides the context) and OCL (that provides both formality and syntax understandability);
- the formalization can be used for automating the metrics collection process.

3. Object-relational database metrics

This section briefly presents both the ontology developed to represent the concepts of the SQL:2003 standard [4], and the metrics we used [3] to detect where to apply refactoring. The metrics were defined for any OR database schemas and are calculated here using the elements of the SQL:2003 ontology.

3.1. An ontology for SQL:2003 OR databases

An ontology representation language should have rich and formal abstractions for specifying the relevant knowledge of the domain of interest. UML class diagrams enriched with well-formedness rules, expressed with OCL, fulfil this purpose. We have used this representation language in our proposed ontology for expressing the knowledge relative to database schema definition as defined in the most recent SQL standard, the SQL:2003 [8]. That knowledge could be found in parts 1 (Framework), 2 (Foundation) and 11 (Information and Definition Schema) of the standard, along with two well-known books on the topic [10, 11].

This ontology was divided into two parts. One contains the aspects related to data types (Figure 2) and the other, the information about the SQL schema objects (Figure 3). A complete description of the ontology is given in [4].

3.2. OR metrics formalized under the proposed ontology

For illustration purposes, the formalization of some of the metrics originally defined on [3] is shown. Some auxiliary functions were used in the formal definition of the metrics, but they are not included here due to space limitations. The formalization effort can be found in [2][1].

Table Size (TS)

The Table Size (TS) is defined as the sum of the Total Size of the Simple Columns (TSSC) and the Total Size of the Complex Columns (TSCC) in the table:

$$TS = TSSC + TSCC$$

```
BaseTable:: TS(): Real
= if self.isKindOf
  (TypedTable)
  then
    self.structuredType.
         hierarchySize()
  else
    self.TSSC() +
    self.TSSC()
endif
```

We consider that all simple columns have a size equal to one, so that the TSSC metric is equal to the Number of Simple Columns (NSC):

$$TSSC = NSC$$

```
BaseTable:: TSSC():
Integer
= self.allSimpleColumns()
  -> size()
```

TSCC is defined as the sum of each Complex Column Size (CCS):

$$TSCC = \sum_{i=1}^{NCC} CCS_i$$

```
BaseTable:: TSCC(): Real
self.allComplexColumns()
-> collect(elem:Column |
elem.CCS()) -> sum
```

NCC is the Number of Complex Columns in the table.

[1] Some auxiliary functions are:

```
Column:: CCS(): Real = self.SHC() / self.NCU()
Column:: SHC(): Real = self.dataType.oclAsType
  (StructuredType).SC() +
Column:: NCU(): Integer =self.dataType.oclAsType
      (StructuredType).columnsNumberUsingThis().
    self.dataType.oclAsType
BaseTable:: involvedClasses():
  Set(StructuredType)= self.complexColumnTypes() ->
  union(self.allComplexColumns() -> collect(c: Column
      | c.columnType().oclAsType(StructuredType).
      allDependencies())-> flatten) -> asset
```

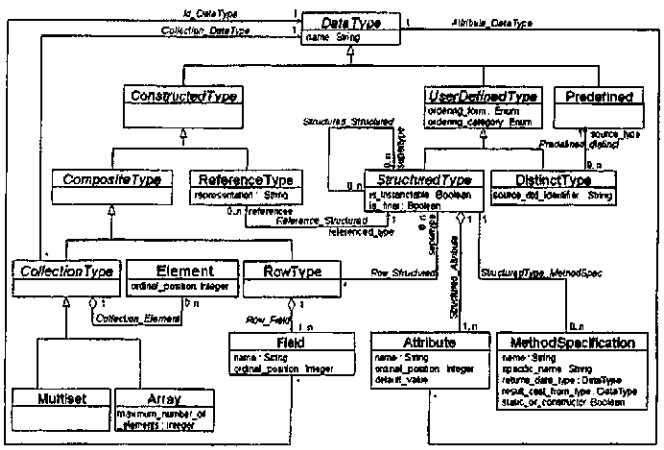


Figure 2. Data types sub-ontology

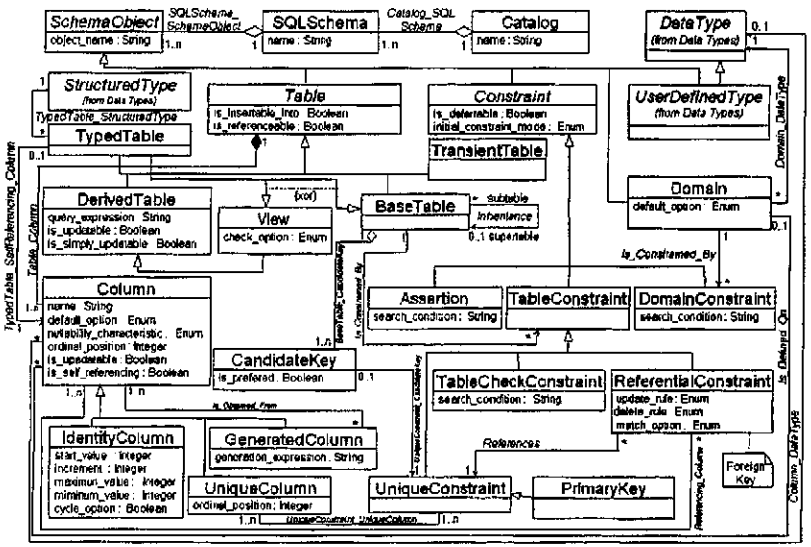


Figure 3. Schema objects sub-ontology

Number of Involved Classes (NIC)

The Number of Involved Classes (NIC) measures the number of classes that compose the types of the complex columns of a table, using the generalization and aggregation relationships.

```
BaseTable:: NIC(): Integer
= self.involvedClasses() -> size
```

Number of Shared Classes (NSC)

The Number of Shared Classes (NSC) measures the number of involved classes for a table that are used by other tables.

```
BaseTable:: NSC(): Integer
= self.involvedClasses()
  -> select(elem: StructuredType|
            elem.isShared())
     -> size
```

4. Refactoring OR database schemas

Although the initial design of a software system might have been carefully produced, during its lifetime it is likely that it will suffer continuous evolution. Over subsequent versions there is a risk that the design becomes more complex and less maintainable, reducing the overall system quality. The same may happen with database schemas. In this case, database schema refactorings can be used to mitigate this quality problem.

The refactoring process consists of a number of distinct steps [17], summarized below:

1. identify where the refactoring(s) should take place;
2. determine which refactoring(s) should be applied to the identified software fragment;
3. guarantee that the identified refactoring(s) preserves behavior;
4. apply the refactoring(s);
5. assess the effect of the refactoring(s) on software quality characteristics (e.g. understandability, maintainability);
6. maintain the consistency among the refactored software artifact and other artifacts (for example, if code is refactored, it should be consistent with the documentation, with the original requirements or with a test battery).

For performing step 1, we evaluate a database schema with the suggested metrics, to identify tables with higher complexity, which hampers their understandability and maintainability. We call these *refactoring-prone tables*. The metrics presented above, along with others proposed in [3], are used to assess several aspects of complexity, such as size and coupling.

The appropriate refactoring(s) to apply to the refactoring-prone tables (step 2) are then chosen, with the aid of a *database refactorings catalog* [16]. Step 3 is beyond the scope of this paper. After applying the refactoring (step 4), OR database metrics are collected again upon the refactored tables (step 5) and the results are compared with those collected during step 1. If these metrics do not grant evidence that the complexity of the refactored-prone tables has indeed dropped down, then a rollback to step 2 is advised. Otherwise, in case of complexity reduction, the changes should be propagated to all interested parties (technical team) and should be kept consistent with other deliverables (step 6).

5. An Example

In this section we illustrate, with an example, the stepwise application of our proposed approach to database schema refactoring. Figure 4 represents an OR database schema mapped to the ontology entities presented in section 3. This mapping is performed by instantiating the ontology, herein represented as a UML object diagram. The instantiation depicts one *SQLSchema* (*video and music*) composed of three tables (*customers*, *movie_titles* and *movie_stars*). Each of these tables has several columns, either simple or complex ones. The metrics values for the tables in this schema are shown in Table 1.

	customers	movie_stars	movie_titles
NIC	5	5	0
NSC	3	3	0
TSSC	6	5	0
TSCC	3	3	3
TS	9	8	3

Table 1. Metrics values for the example

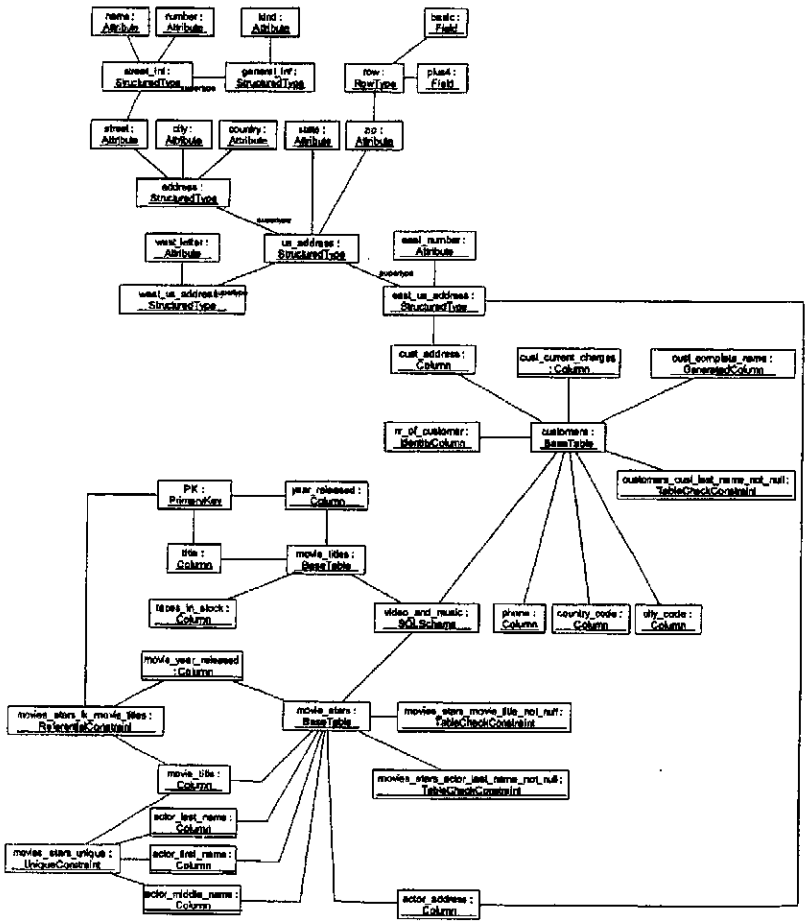


Figure 4. Example of database schema mapped to the SQL:2003 ontology

The table size metric (TS) for *customers* and *movie_stars* has a much higher value compared with *movie_titles*. Besides, those tables present high coupling values, as shown by NIC and NSC metrics. Therefore, tables *customers* and *movie_stars* are potential candidates for performing a refactoring.

We applied the following structural refactorings, suggested in [16] to the table *customers*, in the complex column *cust_address*.

1. To the zip fields (*basic* and *plus4*): *combine columns representing a single concept.*
2. To *address*: *remove structured type*, considering customers of only one country.
3. To the schema object *name*, associated with *street_inf*: *move attribute* to *us_address*.
4. To the schema object *number*, associated with *street_inf*: *move attribute* to *us_address*.
5. To the schema object *name*, associated with *us_address*: *rename attribute* to *street_name*.
6. To the schema object *number*, associated with *us_address*: *rename attribute* to *street_number*.
7. To *street_inf*: *remove structured type.*
8. To *street*: *remove attribute.*
9. To *general_inf*, in order to make it represent the attribute *kind*: *replace structured type.*
10. To *kind*, associated with *general_inf*: *remove redundant attribute.*

Note that we have generalized the applicability of the original structural refactoring, intended for *Tables* and *Columns*, to *Structural Types* and *Attributes*, which are also ontology entities.

The resulting transformations for the table Customers are presented in Figure 5. The other tables remain as originally defined. The selected refactorings depend upon the experience of the database developers, which are responsible for verifying that the schema semantics is not violated. For instance, the elimination of the structured type in step 2 can only be performed if the user requirements allow having only one country.

After the refactorings, the metrics are again collected. The resulting values, contained in Table 2, compared to those in Table 1, show a reduction both in coupling (NIC and NSC values) and table size, for *customers* and *movie_stars*.

	customers	movie_stars	movie_titles
NIC	2	2	0
NSC	2	2	0
TSSC	6	5	0
TSCC	2.25	2.25	3
TS	8.25	7.25	3

Table 2. Metric values after refactoring

One important remark is that if a table has a high value for TS and this is due to a high value for TSCC, then refactorings can be employed to reduce the complexity of the schema, considering the reduction of NIC and NSC. If both the latter have high values, the table is highly coupled, and then, some refactoring-prone tables can be identified. In most cases, the table size should not be considered alone, because refactorings can even increase the value of this metric. It was observed, in our experiments, that when the TS value is high and also the NIC result is high, refactorings can considerably reduce the schema complexity.

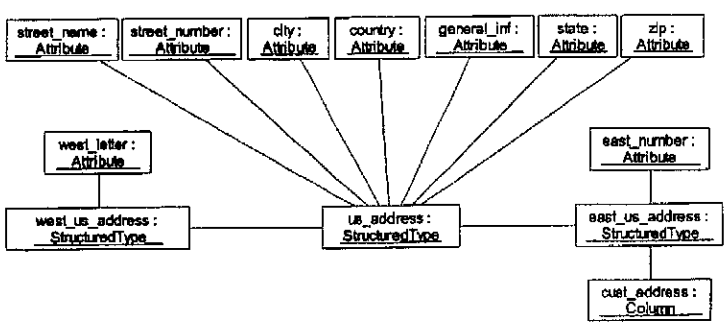


Figure 5. Refactoring the table "customers"

6. Conclusions and future work

Using OCL upon an ontology for OR database schemas, we formalized, in a previous work [16], a set of complexity metrics. This formalization solves the problem of ill-definitions and eases the metrics collection process.

In this paper we addressed the problem of pinpointing where to apply refactorings and assess if their application was successful, regarding database schema complexity reduction. The preliminary results presented herein are encouraging on the potential of our approach to assist the database schemas refactoring process.

As future work, we plan to research the use of other OR database metrics [3] for the same aim. We also plan to build more elaborate examples for empirically validating the OR schema metrics as indicators of refactoring-prone tables. The validation work will use experts' opinion in controlled experiments, by comparing how their proposed refactoring-prone tables match with those identified through the metrics.

Acknowledge

This work was partly funded by the Portuguese Foundation of Science and Technology (FCT / MCES) and by project CALIPO (TIC2003-07804-C05-03) from the MECD (Spain).

References

[1] A. L. Baroni, *Formal Definitions of Object-Oriented Design Metrics*. Master Thesis, École Nationale Supérieure dos Techniques Industrielles e des Mines de Nantes, France, August, 2002.

[2] A.L. Baroni, C. Calero, F. Ruiz and F. Brito e Abreu. "Formalizing Object-Relational Structural Metrics". *Proceedings of the 5th Portuguese Association of Information Systems Conference*. November, 2004.

[3] C. Calero, *A Set of Metrics for Maintainability of Relational, Object-Relational and Active Database Systems*. PhD Thesis, Universidad de Castilla-La Mancha, March, 2001.

[4] C. Calero, F. Ruiz, A. L. Baroni, F. Brito e Abreu and M. Piattini, "An Ontological-Approach to Describe the SQL:2003 Object-Relational Features". *Submitted to the Computer Standards and Interfaces Journal*. April, 2005.

[5] F. Brito e Abreu. *Using OCL to Formalize Object-Oriented Metrics Definitions*. Technical Report ES007/2001. INESC, Portugal, June, 2001.

[6] F. Brito e Abreu and W. Melo. "Evaluating the Impact of Object-Oriented Metrics on Quality". *Proceedings of the 3rd International Software Metrics Symposium*, Berlin, Germany, March 1996.

[7] H. M. Sneed and O. Foshag, "Measuring Legacy Database Structures". *Proceeding of the European Software Measurement Conference* (FESMA 98), Antwerp, May, 1998.

[8] ISO / IEC 9075 Standard, *Information Technology - Database Languages - SQL:2003*, International Organization for Standardization, 2003.

[9] ISO / IEC TR 9126 Standard, *Software Engineering - Product Quality*. International Organization for Standardization, Geneva, 2003.

[10] J. Melton, *Advanced SQL: 1999 - Understanding Object-Relational and Other Advanced Features*, Morgan Kaufmann Publishers, 2002.

[11] J. Melton and A. R. Simon, *SQL: 1999 - Understanding Relational Language Components*, Academic Press, 2002.

[12] M. H. Tang, M. H. Chen, e M. Kao, "Investigating Test Effectiveness on Object-Oriented Software - A Case Study". *Proceeding of the Twelfth Annual International Software Quality Week*, 1999.

[13] OMG, *UML 2.0 OCL Specification*. Final Adopted Specification., October, 2003.

[14] P. Grubb and A. A. Takang. *Software Maintenance: Concepts and Practice*. World Scientific Publishing Company. ISBN: 9812384251, 2003.

[15] S. Henry e C. Selig, "Predicting Source-Code Complexity at the Design Stage". *IEEE Software*, 1990.

[16] S. W. Ambler, *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. John Wiley and Sons. ISBN: 0471202835, 2003.

[17] T. Mens and T. Tourwé, "A Survey of Software Refactoring". *IEEE Transactions on Software Engineering*, vol.30, number 2, February, 2004.

[18] T. M. Pigoski. *Practical Software Maintenance: Best Practices for Managing your Software Investment*. John Wiley and Sons. 0471170011, 1997.

[19] W. Harrison, "Using Software Metrics to Allocate Testing Resources". *Journal of Management Information Systems*, 1988.

[20] W. Li e S. Henry, "Object-Oriented Metrics that Predict Maintainability". *Journal of Systems and Software*, vol. 23, pp. 111-122, 1993.

[21] W.F. Opdyke, Refactoring: A Program Restructuring Aid in Designing Object-Oriented Applications Framework, PhD Thesis, Univ. of Illinois at Urbana-Champaign, 1992.