Formalization of Database Reverse Engineering

Nonyelum Ndefo(✉)

Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
ndefo@inf.unibz.it

Abstract. During the life cycle of an Information System, the original design of the database may be difficult to acquire. The database schema may have been continuously modified and drifted away semantically from the intended design, or perhaps no conceptual modelling method was employed at all. A conceptual schema offers a much richer description of a domain than the database schema, this makes it important for, among other reasons, the maintenance of semantic consistency of the database at runtime. Database reverse engineering involves the retrieval of domain semantics from an existing set of database schemas into a conceptual schema. Though several research works exist on creating database reverse engineering methodologies, the process in itself has never been properly and fully formalised with an emphasis on its correctness. This paper introduces the ongoing research surrounding database reverse engineering and the goal of rendering a formal reverse engineering framework. The expected formalism will be valuable to database and domain experts as a sound foundation for implementing better reverse engineering tools.

Keywords: Reverse engineering · Relational database · Conceptual modelling · Mapping · Schema transformation

1 Introduction

The concept behind reverse engineering has been used for different causes across computer science disciplines. In the development and maintenance of software systems, reverse engineering tries to reconstruct design information from an implemented system [17]. In data management practices, reverse engineering has been studied in close connection with databases, more specifically with relational databases.

Database Reverse Engineering (DBRE) is a means to recover domain semantics encoded in a specified database and represent them as a high-level conceptual schema that corresponds to the possible design specifications of the database schema [5].

1.1 DBRE Motivation

In general, the reasons to reverse engineer a database may include data migration from legacy systems, data integration, data exchange at conceptual level, semantic acquisition for system maintenance, for domain (re)documentation, and for query answering using the conceptual schema [7].

To fully comprehend the DBRE cause, an understanding of database design procedure is also necessary. Database design, or *forward engineering*, is a process which maps elements in a conceptual schema to elements in a database via a series of tasks such as requirement analysis, entity and relationship identification through dependencies (constraints), key selection, normalisation, etc.

During the database design process, while the specification transitions from the conceptual schema to the logical and physical schemas, it is often found that domain knowledge has been left implicit or lost within the database. This can be attributed to the fact that the relational schema is semantically poorer than most conceptual schemas, and it cannot explicitly express certain domain information, e.g. generalisation and specialisation hierarchies, unions and disjointness of entity sets [12].

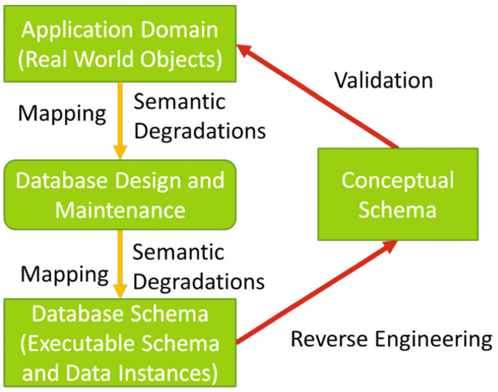
Application Domain (Real World Objects)
Mapping
Semantic Degradations
Database Design and Maintenance
Mapping
Semantic Degradations
Database Schema (Executable Schema and Data Instances)
Validation
Conceptual Schema
Reverse Engineering

Fig. 1. Semantic loss during Database Design [5]

Furthermore during the life cycle of the database system, the structure of the database is prone to changes over the years of its usage and is subject to semantic degradation as described in Fig. 1. Whether this degradation is caused by the poor semantic nature of the relational database, tricky original design or the use of certain optimisation practices e.g. de-normalization of some relations to increase performance, the database users remain at risk of losing the understanding of its contents.

With no concrete knowledge of the initial design strategy used to forward engineer the database system, no detailed documentation through which continuous modifications to the database could have been tracked, and the unavailability of the original database developers, the problem becomes not only non-trivial but also tedious [8]. As one of the main benefits of a conceptual schema is to provide users which an abstract and eminently clear representation of the domain, the DBRE process is required to carefully implement the backward transitioning of the forward engineering process, filling up the perceived semantic gaps in the database schema to construct a conceptual schema.

The rest of this paper is organised as follows: Sect. 2 gives an overview of the DBRE approaches, also highlighting the need for a completely formal framework. In Sect. 3, several related works are reviewed including the state of the art research on DBRE and some studies on schema transformation in database systems. The plan for formalising the entire DBRE process is detailed in Sect. 4, including the choice of conceptual schema and formal language to illustrate the formalism, and important concepts of interest concerning schema transformation. Finally, Sect. 5 provides a summary and additional information about the research.

2 Overview of DBRE Approaches

The DBRE approaches usually perform what is referred to as *semantic acquisition* as part of the first step of the process. This is done by analysing the database schema along with other sources such as the schema's DDL, data instances, and the SQL queries derived from application programs that use the database, in the hope of obtaining a richer description of the domain. These approaches use the mentioned sources (singly or combinatorially) and are able to attain additional semantic information from them, this includes all information about elements, which are basically relations[1] and constraints.

After these sources have been explored and the database schema has been enriched with new-found information, then succeeding steps can occur: the classification and conceptualisation of elements. In the conceptualisation step, each element within the relational schema is transformed into a corresponding element in the conceptual schema by a set mappings.

Here, the term *conceptual schema* is used loosely and considers high-level data models which describe semantics of data in similar ways, for instance, in the Entity Relationship (ER) model with its use of *entities*, which are viewed as *classes* in the Unified Modelling Language (UML) class diagrams, *concepts* in ontology languages, and *objects* in Object Role Modelling (ORM). Even though conceptual schemas in general can have various degrees of expressiveness, it can be proven that at least most of the common elements can be accurately translated among these similar data models. A survey of these approaches shows that a preferred target schema for re-engineering database schemas is the ER model or its upgraded version Enhanced Entity Relationship (EER) [1,6,11,14,15,18,21]. Other choices have been Object Modelling Technique (OMT) [19], ERC+(an extension of the ER model with multivalued objects and multi-instantiation) [3], and First Order Logic (FOL) ontologies [4,13,14]. Table 1 shows a nicely structured overview of certain similarities and contrasts among of the existing approaches.

[1] Relations consist of typed attributes, with each attribute belonging to its own finite attribute domain.

Table 1. Comparison of DBRE approaches

Approach	Source schema	Other sources	Target schema	Formal Verification	Extracted constraints	Automation
Astrova [4]	Relational schema in 3NF	Data instances	Ontology for the Semantic Web	None	Key attributes, Inclusion, Disjointness, Intersection constraints	Semi-automated
Alhajj [1]	Schema in 3NF	Data instances	EER	None	Key attributes, Cardinality, Inclusion constraints	Automated
Andersson [3]	Schema (no assumptions on normal forms)	SQL queries, DDL, Data instances	ERC	None	Key attributes, Inclusion constraints	Non-automated
Chiang et al. [6]	Schema in 3NF	Data instances, DDL	EER	None	Keys, Inclusion constraints	Semi-automated
Petit et al. [18]	Schema in 3NF	Data instances, SQL equi-join queries	EER	None	Key attributes, Inclusion constraints, Disjointness constraints	Semi-automated
Lammari et al. [12]	Schema	DDL, SQL queries, Data instances (if necessary)	EER	None	Key attributes, Inclusion and Intersection constraints	Automated (or Semi-automated, if necessary)
Lin et al. [13]	Schema in 3NF	Data Instances	OWL DL ontology	None	Key attributes, Classes, Data-type Property, Object Property, Cardinality, Inclusion constraints, Disjointness, Equivalent Classes, Covering	Automated
Signore et al. [21]	Schema assumed to be in 3NF	Application program code with embedded SQL	ER Model	None	Key attributes, IS-A hierarchies	Non-automated
Markowitz & Makowsky [15]	Schema in BCNF	None	EER	Proof sketches for mappings	Key attributes, Inclusion constraints	Non-automated
Lubyte & Tessaris [14]	Schema in 3NF	None	DLR-Lite (Description Logic) Ontology	Proof for equivalence preserving mapping	Key attributes, Inclusion, Disjointness, Covering, Mandatory and Optional Participation, Functionality, Role Typing	Automated
Blaha & Premerlani [19]	Schema (no assumptions on normal forms)	Data instances	OMT	None	Key attributes, inclusion constraints, Cardinality	Non-automated
Johannesson [11]	Schema in 3NF		Conceptual schema defined as a pair of language L and a set IC of typing, mapping and generalization constraints	Proof sketch for schema dominance	Key attributes, Inclusion constraints	Non-automated

2.1 Problems with Existing Approaches

More compactly, a DBRE setting should consist of a source schema, a target schema and a well-defined transformation process between the schemas. Closely inspecting these approaches, it is discovered that they are mostly non-automated and involve human interaction not just for a final user validation but often times mid-process. Therefore the mappings induced by the transformations most likely rely on heuristic techniques and informal rules dependent on the expertise and view point of the users. Consequently, each approach, if used on the same database schema may channel towards a relatively diverse target schema, with probable semantic inconsistencies.

The above claim is attributed moreover to the little effort demonstrated in presenting a guaranteed measure for correctness post-transformation, thus rendering the validity of the reverse engineering process questionable.

Take for instance the case of classifying an association between two relations, say *SSN* and *PERSON*. Relation *SSN* holds records of taxation details for each person while relation *PERSON* holds basic information for the each person.

$$\text{PERSON}[\textbf{ssn}, \text{name}, \text{address...}]$$
$$\text{SSN}[\textbf{ssn}, \text{tax\_category}, \text{issuing\_state}, ...]$$

Both relations have the same key *ssn*, used to uniquely identify each tuple in both *PERSON* and *SSN*. If no information about referential integrity is known, based on the set of heuristic rules in [6], this association is justified as a subtype inclusion constraint because the relations share the same key attribute, and is therefore mapped accordingly. Whereas Lin et al. [13] argue that it may not be so that "a *PERSON* is a *SSN*" or vice versa just for this reason and consider a "more natural interpretation" which portrays this association as a one-to-one or one-to-many binary relationship.

Another usually overlooked issue concerns the handling of nullable attributes (null values) within DBRE. Since the relational and conceptual schemas are of different signatures, the interpretation of nullable attributes in the conceptual schema needs to be properly addressed with respect to preserving information. This aspect is yet to be thoroughly treated whether formally or informally.

We believe that providing documented formal descriptions for the process would help in resolving these issues and other related ones. Then we can establish the most logically plausible mappings which rightly preserve source schema semantics in the target schema, and verify that a DBRE transformation can be indeed correct.

However, before delving into our plans to address these issues, we consider it necessary to review relevant literature in the reverse engineering context through which substantial contributions have been produced. In the next section some notes on these notable contributions are presented.

3 Related Work

Since the 1990s, studies concerning DBRE have been carried out, of which the most referenced have been listed in Sect. 2.

In [11], the database schema in 3NF is translated into a conceptual schema represented by a pair consisting of a language and a set of typing, mapping, and generalisation constraints. The methodology investigates object structures and how they can be identified in the database schema based on the correlation between keys and inclusion constraints. The work concludes by defining the notion of *schema dominance* as a measure of correctness for the reverse engineering method. As future work, the paper suggests a potential line of research: *investigating the influence of views on conceptual modelling.* We plan to improve on this work by investigating this schema dominance on different reverse engineering scenarios, after defining mappings (views) of elements in one schema based on the signature of the other schema.

The work of [15] identifies object structures in a database, taking as input the relational schema in Boyce-Codd normal form (BCNF) consisting of key constraints and key-based inclusion constraints. The output of the approach is an EER model. This method argues against the informal mappings presented in other works and introduces some formal mapping descriptions for a procedure which determines convertibility of a BCNF normalised relational schema into the EER model, although BCNF is considered as one of the stricter normal forms in the context of realistic database schemas.

The most recent research works steer towards reverse engineering a relational database to logic-based ontologies. In [14], an ER model is extracted from an existing relational database, with mappings defined between the extracted schema and the relational database by associating views over the latter as a means to access the underlying data. The approach subtly exploits the Global-as-View (GAV) mappings from data integration. It highlights important points such as obtaining an equivalence preserving schema transformation and uses description logic to express semantics of the schema. Our approach contributes to this by considering in addition reverse mappings similar to the Local-as-View (LAV) to verify correctness DBRE in terms of losslessness.

Astrova [4] focuses on reverse engineering a relational database into an OWL ontology for use on the Semantic Web. The approach confirms semantics by analysing not only the correlation between primary and foreign keys, but also the correlation among data values and attributes. The reverse engineering process is simple and shows some mapping for relations, attributes, relationships, and constraints. However it is not formally presented and there is no verification for information consistency. The work claims to show an extraction of more semantics from the relational database compared to other approaches.

The work in [13] describes a DBRE-based automatic algorithm to extract an OWL-DL ontology with richer semantics from a given relational database. Compared to the others mentioned above, this approach goes a step further by illustrating the correspondences among semantics of the ER model, database schema, and OWL ontology.

In general, the reverse engineering approaches could rely heavily on human intervention, be semi-automated or fully automated. However it is to be noted that in any system, before complete automation can occur for any process there should be a set of formal definitions properly set in place. Though these works have contributed significantly to this field, they will be used as building blocks for rendering a more complete framework.

Since schema transformation is a fundamental part of DBRE, it is useful to study transformation practices involving relational databases.

Hainaut et al. in [9] offer a generic technique for schema transformation for a database, describing schema transformation as consisting of two types of mappings: *structural mapping* and *instance mapping*. The first ensures that elements in the source schema can be replaced by elements in the target schema, and the latter ensures correspondences between instances in both schemas. It also introduces transformation concepts such as *schema reversibility* and *semantics preservation*.

McBrien and Poulovassilis in [16] present a schema transformation approach based on combining both GAV and LAV mapping approaches. The coined name, BAV, meaning *Both-as-View* leverages the benefits of both GAV and LAV views. The concept of reversibility is used to maintain here as a desired aspect of the transformation process.

The work presented in [2] aims to discover semantic matches between two databases along with their already existing conceptual schemas. The approach described here finds the mappings based on the correspondences (or links) between attributes and relations.

Qian, in [20], emphasizes that the correctness of a schema transformation is achieved when the instances and constraints in the source schema are preserved. Three transformation properties are highlighted: instance, constraint, and information preservation. The last is a consequence of the first two. The work defines an information preserving transformation as in terms of containment between the source and target schemas denoted by s and t respectively i.e. to determine if $s \subseteq t \textit{ and } t \subseteq s$. And if these containments hold then s is said to be equivalent to t, denoted by $s \equiv t$.

This section has presented concise descriptions of other research works related to the one proposed. As stated earlier, schema transformation is a pivotal aspect in DBRE, and any plan to produce a useful method should focus primarily on how semantic equivalences are identified between the source and target schemas and how eventual maps are created. This therefore justifies the following section which discusses the details of the proposed formalism.

4 Formalisation Plan of DBRE

By definition, one understands that DBRE consists of three main aspects i.e. the source schema, schema transformation, and the target schema. The approach to formalise DBRE rests on grasping the semantics of each of the individual aspects and how they relate. This section gives brief details on the source and target schemas, and the formalisation plan.

4.1 Source Schema

We will provide various realistic scenarios of database schema as examples. We assume that semantic acquisition has already been carried out on the source schema by analysing the input sources mentioned in Sect. 2, and the schema is now complete with all intended semantics from the domain i.e. all possible information about relations and constraints are known. At this point, the schema is also presumed to have been decomposed into a sufficient normal form, ideally 3NF, so that the conceptual schema elements can be seamlessly identified and classified prior to conceptualisation.

4.2 Target Schema

For the target schema, ORM [10] will be used to describe the semantics extracted from the source schema. Modelling in ORM is fact-based i.e. data is described as elementary facts, which are predicates asserting that an object type participates in a role (relationship). The model is attribute-free, representing the relationship between an object and its attributes also as roles. The example in Fig. 2 illustrates two fact types: one binary and the other ternary. The binary role *seeks* connects the object types *Student* and *Degree*. In the ternary fact, the role *...was awarded...on...* connects the same object types but now with another object type *Date*.

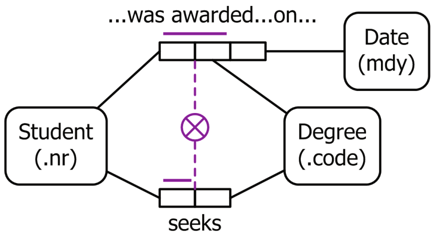

Fig. 2. Example ORM diagram (http://www.orm.net)

Integrity constraints and other constraints can be expressed clearly. Given the same example in Fig. 2, the bars over the roles indicate uniqueness constraints over each entity participating in those roles. The other constraint seen in the diagram is the exclusion constraint (circled 'x') between the roles *seeks* and *...was awarded...on...* indicating disjointness between the instances in these roles.

An advantage of using ORM over the other similar conceptual schemas, besides the fact that it has a standard notation and its ability to model relatively more features, is the possibility to automatically derive natural language verbalisation for its conceptual schemas using the Natural ORM Architect (NORMA) tool within Microsoft Visual Studio development environment. The verbalisation texts are expressed with regards to FORML, the controlled language for ORM and as far we know, an automatic verbalisation feature with such convenient readability is yet to be embedded in any of the other common modelling tools. The following verbalisation texts explain Fig. 2:

- **Student** was awarded **Degree** on **Date**.
- *For each* **Student** and **Degree**, that **Student** was awarded that **Degree** on *at most one* **Date**.
- **Student** seeks **Degree**.
- Each **Student** seeks *at most one* **Degree**.
- It is possible that *more than one* **Student** seeks *the same* **Degree**.
- For each **Student** and **Degree**, *at most one* of the following holds: that **Student** seeks that **Degree**; that **Student** was awarded that **Degree** on *some* **Date**.

This feature is very useful for verification with non-technical users who may have trouble understanding the non-trivial technical terminology in the diagram.

4.3 Schema Transformation

Before discussing the schema transformation aspect, we briefly describe the language intended for expressing the semantics. For the purpose of this research and the nature of the schemas in question, a language which is able to clearly express relational elements is most suitable.

Relational Algebra. Relational algebra is a procedural query language that manipulates instances of relations in a database schema by applying operators in order to also produce as output relational instances. Relational algebra is the backbone of the database query language SQL. Below are the main operators used by the algebra, some of which are inherited from set theory:

$\cap$	intersection	π	projection
$\cup$	union	σ	selection
$-$	minus	$\bowtie$	join
$\times$	cartesian product	$\leftarrow$	rename

We use the formula below to illustrate an example for the syntactic structure of a relational algebra query expression:

$$stellar\_students \leftarrow \pi_{nr,\ age}\ \sigma_{\substack{Degree.code=Student.degree\_code\_FK \\ and\ Degree.final\_score>98}} (Student \times Degree) \tag{1}$$

Apart from query expressions, the procedural system of the algebra can also be employed to describe constraints over relations. Key constraints, inclusion constraints, and functional dependencies are examples of important constraints which can be captured by relational algebra. Constraints can be seen as boolean expressions that are valid i.e. always true. To describe constraints, we depend on the full power of relational algebra, plus assertions between an expression and the empty set $\emptyset$. Take for example the model in Fig. 2. We interpret *seeks* and *...was awarded...on...* simply as binary and ternary relations respectively. To show the exclusion constraint between the two roles, we use the following expressions:

$$Temp \leftarrow \pi_{nr\_FK,\ degree\_code\_FK}(wasAwardedOn)$$
$$Temp \cap Seeks = \emptyset$$

Expressions in relational algebra can also be translated into *relational calculus* which is a variant of first-order logic.

Transformation. The transformation aspect of DBRE represents the core of the entire process, and therefore requires the profoundest consideration. Here, we summarise the plan to formalise this aspect.

A semantically enriched source schema s is a pair (Σ_s, λ_s), where Σ_s is the signature of the schema consisting of a set of relations, and λ_s is a set of constraints in s. The target schema t is a pair (Σ_t, λ_t) where Σ_t is the signature of the conceptual schema consisting of objects connected by roles, and λ_t is a set of constraints in t. Conventionally, a DBRE schema transformation yields a set of mappings μ for each relation r_s such that $r_s \in s$ and is transformed into a corresponding object o_t such that $o \in t$, without violating λ_s.

After conceptualisation of each relational term into a corresponding conceptual term, a mechanism is required to check the correctness of the transformation. Our research work particularly contributes to this by expanding the transformation process. We propose including inverse mappings as a means to check correctness, therefore defining mappings not only in the direction $s \rightarrow t$ but also from $t \rightarrow s$, for every element in both schemas.

More formally, we describe the semantics of the mapping below:

The mapping μ is a function symbol such that $\mu : r_s \rightarrow v_t$, where v_t is an associated view in t i.e. a relational algebraic expression for the corresponding o_t which associates with r_s. Then we say that μ^{-1} is the inverse of μ such that $\mu^{-1} : o_t \rightarrow v_s$, where v_s is an associated view in s i.e. a relational algebraic expression for the corresponding r_s which associates with o_t. Therefore we may also state that μ is a bijection. In μ, each r_s is mapped to a v_t expression in a LAV-like manner. In μ^{-1}, the reverse holds, where each object o_t is mapped to a v_s expression, as would GAV mappings.

Though s and t describe the same domain, they are projected differently. Defining views over them will require decomposing and joining of these projections. For this reason we need to ensure that both μ and μ^{-1} can be carried out in such a way that the semantics of the data and their constraints are not lost between them. This leads to investigating the schema transformation property known as *losslessness*.

We say that μ and μ^{-1} together are lossless if and only if $\lambda_s \models \lambda_t[o^i_t/v_s(o^i_t)]$ and $\lambda_t \models \lambda_s[r^i_s/v_s(r^i_s)]$. More verbosely, this means that λ_s entails λ_t following the substitution of each o_t according to v_s, the defined view for o_t over the language of s, and λ_t entails λ_s following the substitution of each r_s according to v_t, the defined view for r_s over the language of t.

If this losslessness property is always true, then we can assert that $\mu^{-1}(\mu(r_s)) = r_s$, for each r_s. This means that $\mu^{-1} \circ \mu$ which maps from $r_s \rightarrow o_t \rightarrow r_s$ constitutes the identity function id_{r_s} for all r_s, and $\mu \circ \mu^{-1}$ which maps from $o_t \rightarrow r_s \rightarrow o_t$ constitutes the identity function id_{o_t} for all o_t.

Intuitively, with the evidence of a lossless transformation we can then affirm that the DBRE process is correct and complete, then a possibility opens up for building queries over the conceptual schema itself.

5 Summary

The main contribution of this research will be the formalisation of the entire DBRE process, with focus on ensuring correctness on the conventional methods.

The submission of this paper marks the conclusion of the first 6 months of the ongoing PhD research work. The past months have been dedicated to literature study of which we have been able to produce substantially. Having covered the significant grounds for this research work, the immediate next step is to demonstrate DBRE in action, covering various natural scenarios, with the affiliated proofs to support its correctness.

References

1. Alhajj, R.: Extracting the extended entity-relationship model from a legacy relational database. Inform. Syst. **28**(6), 597–618 (2003)
2. An, Y., Borgida, A., Miller, R.J., Mylopoulos, J.: A semantic approach to discovering schema mapping expressions. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 206–215, April 2007
3. Andersson, M.: Extracting an entity relationship schema from a relational database through reverse engineering. In: Loucopoulos, P. (ed.) ER 1994. LNCS, vol. 881, pp. 403–419. Springer, Heidelberg (1994). doi:10.1007/3-540-58786-1_93
4. Astrova, I.: Reverse engineering of relational databases to ontologies. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 327–341. Springer, Heidelberg (2004). doi:10.1007/978-3-540-25956-5_23
5. Chiang, R.H.L., Barron, T.M.: Quality issues in database reverse engineering: an overview. In: Proceedings for Operating Research and the Management Sciences, pp. 185–189, June 1995
6. Chiang, R.H.L., Barron, T.M., Storey, V.C.: Reverse engineering of relational databases: extraction of an eer model from a relational database. Data Knowl. Eng. **12**(2), 107–142 (1994)
7. Hainaut, J.-L.: Introduction to database reverse engineering. LIBD Lecture Notes (2002)
8. Hainaut, J.-L.: Research in database engineering at the university of namur. ACM SIGMOD Rec. **32**(4), 124–128 (2003)
9. Hainaut, J.-L., Tonneau, C., Joris, M., Chandelon, M.: Schema transformation techniques for database reverse engineering. In: Proceedings of the 12th International Conference on ER Approach, pp. 353–372. Springer, Arlington-Dallas (1993)

10. Halpin, T.: Object-role modeling: principles and benefits. Int. J. Inform. Syst. Model. Des. **1**(1), 33–57 (2010)
11. Johannesson, P.: A method for transforming relational schemas into conceptual schemas. In: Proceedings of the 10th International Conference Data Engineering, 1994, pp. 190–201. IEEE (1994)
12. Lammari, N., Comyn-Wattiau, I., Akoka, J.: Extracting generalization hierarchies from relational databases: a reverse engineering approach. Data Knowl. Eng. **63**(2), 568–589 (2007)
13. Lin, L., Zhuoming, X., Ding, Y.: Owl ontology extraction from relational databases via database reverse engineering. JSW **8**(11), 2749–2760 (2013)
14. Lubyte, L., Tessaris, S.: Automatic extraction of ontologies wrapping relational data sources. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 128–142. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03573-9_10
15. Markowitz, V.M., Makowsky, J.A.: Identifying extended entity-relationship object structures in relational schemas. IEEE Trans. Softw. Eng. **16**(8), 777–790 (1990)
16. McBrien, P., Poulovassilis, A.: Data integration by bi-directional schema transformation rules. In: Proceedings of the 19th International Conference on Data Engineering, 2003, pp. 227–238. IEEE (2003)
17. A Müller, H., Jahnke, J.H., Smith, D.B., Storey, M.-A., Tilley, S.R., Wong, K.: Reverse engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, pp. 47–60. ACM (2000)
18. Petit, J.-M., Toumani, F., Kouloumdjian, J.: Relational database reverse engineering: a method based on query analysis. Int. J. Coop. Inform. Syst. **4**(02n03), 287–316 (1995)
19. Premerlani, W.J., Blaha, M.R.: An approach for reverse engineering of relational databases. In: Proceedings Working Conference on Reverse Engineering, pp. 151–160, May 1993
20. Qian, X.: Correct schema transformations. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 114–128. Springer, Heidelberg (1996). doi:10.1007/BFb0014146
21. Signore, O., Loffredo, M., Gregori, M., Cima, M.: Reconstruction of ER schema from database applications: a cognitive approach. In: Loucopoulos, P. (ed.) ER 1994. LNCS, vol. 881, pp. 387–402. Springer, Heidelberg (1994). doi:10.1007/3-540-58786-1_92