AdaMaP: Automatic Alignment of Relational Data Sources Using Mapping Patterns

Diego Calvanese[1,2], Avigdor Gal[3], Naor Haba[3], Davide Lanti[1](✉), Marco Montali[1], Alessandro Mosca[1], and Roee Shraga[3]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
{calvanese,lanti,montali,mosca}@inf.unibz.it
[2] Umeå University, Umeå, Sweden
[3] Technion – Israel Institute of Technology, Haifa, Israel
avigal@technion.ac.il, {naor-haba,shraga89}@campus.technion.ac.il

Abstract. We propose a method for automatically extracting semantics from data sources. The availability of multiple data sources on the one hand and the lack of proper semantic documentation of such data sources on the other hand call for new strategies in integrating data sources by extracting semantics from the data source itself rather than from its documentation. In this work we focus on relational databases, observing they are created from semantically-rich designs such as ER diagrams, which are often not conveyed together with the database itself. While the relational model may be semantically-poor with respect to ontological models, the original semantically-rich design of the application domain leaves recognizable footprints that can be converted into *ontology mapping patterns*. In this work, we offer an algorithm to automatically detect and map a relational schema to ontology mapping patterns and offer an empirical evaluation using two benchmark datasets.

1 Introduction

Modern industrial processes and business processes require intensive use of large-scale data alignment and integration techniques to combine data from multiple heterogeneous data sources into meaningful and valuable information. Such integration is performed on structured and semi-structured data sets from various sources such as SQL and XML schemata, entity-relationship (ER) diagrams, ontology descriptions, process models, and web forms. Data integration plays a key role in a variety of domains, including data warehouse loading and exchange, aligning ontologies for the Semantic Web, semantic process model matching [16], and business document format merging (*e.g.*, orders and invoices in e-commence) [21]. As an example, consider an application that keeps track of funded project applications, managing the review process through panel meetings.

One of the main challenges of data integration is to create a common semantic understanding from the multiple available data sources. In ontology-based

M. La Rosa et al. (Eds.): CAiSE 2021, LNCS 12751, pp. 193–209, 2021.
https://doi.org/10.1007/978-3-030-79382-1_12

data access (OBDA) and integration [20], this is achieved through two main components: *(i)* an ontology that captures the relevant concepts and relations of the domain of interest at a high level of abstraction, in turn acting as a vehicle for reaching a semantic consensus; and *(ii)* a mapping specification that dictates how the data in relational sources can be used to (virtually) populate the classes and properties of the ontology.

A major impediment towards the adoption of OBDA is that data sources typically lack a proper semantic documentation, which makes it extremely difficult and error-prone to obtain both the ontology and the mapping. Consider, in particular, the case of relational databases, where well-established conceptual modeling principles and methodologies can be employed to design their schemata so as to suitably reflect the application domain at hand. This design phase is centered around the usage of semantically-rich representations such as ER diagrams. However, these representations typically get lost during deployment, since they are not conveyed together with the database itself, or quickly get outdated due to continuous adjustments triggered by changing requirements. This may lead to loss of information regarding concept hierarchies, which are flattened in the corresponding relational schema.

In this work, we aim at reconstructing such lost domain semantics by inspecting relational data sources, without any additional documentation. To do so, we start from the key observation that while the relational model may be semantically-poor with respect to ontological models, the original semantically-rich design of the application domain leaves recognizable footprints that can be converted into the aforementioned ontological patterns. Therefore, we propose to use *ontology mapping patterns* (*mapping patterns* for short) [8], which systematically collect recurring ways of linking relational data sources to ontologies via mapping specifications. A mapping pattern relates a relational schema fragment to a corresponding ontology fragment, establishing the mapping between the two. Mapping patterns, therefore, provide a form of a conceptual middleware that describes a shared set of abstractions that facilitates interoperability.

Specifically, we propose an algorithmic technique called AdaMaP that, given a relational data source, automatically determines how suitable fragments of its schema align with corresponding mapping patterns. Once mapping patterns are suitably instantiated on a given data source, they can be employed for a number of downstream data engineering tasks, *e.g.*, *ontology bootstrapping* [13,17,19,24] and *schema cover* [22].

Given a data source, there are in general multiple, sound ways to identify which patterns are relevant, and how they match. Consequently, to assess the usefulness and efficacy of AdaMaP, we comparatively evaluate the results it produced in two real-world case studies against a set of pattern applications manually identified by a human expert. This shows that most of the time the algorithm and the expert agree, which is particularly significant considering that the mapping patterns turn out to cover a large portion of the data sources at hand.

The contribution of this work is twofold. On a conceptual level, we offer an approach to enrich a relational model with semantics through the identification of the footprints that were left by the conceptual model on which the relations are based. We then offer an algorithmic solution to the mapping problem using mapping patterns. Our empirical evaluation demonstrates the effectiveness of the approach.

The rest of the paper is organized as follows. Section 2 presents the building blocks of our proposed model, namely the OBDA approach and the mapping patterns, and provides the problem definition. Our algorithmic solution, ADaMaP, is described in Sect. 3 followed by an empirical evaluation (Sect. 4). The paper is concluded with related work (Sect. 5) and concluding discussion (Sect. 6). An appendix, offering more in detail discussions and details to support replicability are provided in an online repository.[1]

2 Model

We now detail the building blocks for our proposed method. We begin by presenting the OBDA framework, which we rely on in this work, (Sect. 2.1). Then, Sect. 2.2 provides an overview of mapping patterns, which represent the inherent semantics of a data source. Finally, in Sect. 2.3 we formally define the problem. Throughout, we shall use an example that is based on a database developed by SIRIS Academic S.L., a consultancy company specialized in higher education and research, based on the European CORDIS repository.[2]

2.1 OBDA Framework

In this work, we rely on the OBDA framework of [20]. We use **bold** font to denote tuples, *e.g.*, $\mathbf{x}$, $\mathbf{y}$, treat tuples as sets, and allow the use of set operators on them. An *OBDA specification* is a triple $\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ where $\mathcal{T}$ is an *ontology TBox*, $\mathcal{M}$ is a set of *mappings*, and $\mathcal{S}$ is the schema of a database. The schema of the databse is a pair (Σ, Γ) where the signature Σ is a set of table schemata, and Γ is a set of database constraints, including keys and foreign keys.

The ontology $\mathcal{T}$ is formulated in OWL 2 QL [18], whose formal counterpart is the description logic *DL-Lite*$_{\mathcal{R}}$ [7], which notation is adopted in this work. An OWL 2 QL *TBox* $\mathcal{T}$ is a finite set of axioms of the form $B \sqsubseteq C$ or $r_1 \sqsubseteq r_2$, where B, C are *classes* and r_1, r_2 are *object properties*, according to the following grammar (where A is a *class name*, d is a *data property name*, and p is an *object property name*):

$$B \rightarrow A \mid \exists r \mid \exists d \qquad C \rightarrow B \mid \neg B \qquad r \rightarrow p \mid p^-$$

For presentation simplicity we discard datatypes, which are also part of OWL 2 QL.

[1] https://github.com/ontop/ontop-examples/tree/master/caise-2021-patterns.
[2] https://cordis.europa.eu/projects/en.

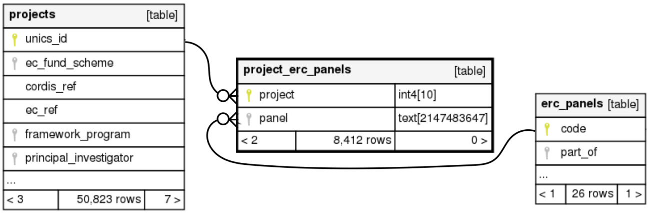


Fig. 1. Fragment of the CORDIS database.

Mappings specify how to populate classes and properties of the ontology with individuals and values, starting from the data in the underlying database. In OBDA, the standard language for mappings is R2RML [9], which we replace here with a more convenient abstract notation, as follows. A *mapping* m is an expression of the form

$$s : Q(\mathbf{x}) \qquad\qquad t : \mathbf{L}(\mathfrak{t}(\mathbf{x}))$$

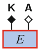
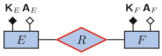
where $Q(\mathbf{x})$ is a SQL query over the database schema Σ, called *source query*, and $\mathbf{L}(\mathfrak{t}(\mathbf{x}))$ is a list of *target atoms* of the form $C(\mathfrak{t}_1(\mathbf{x_1}))$, $p(\mathfrak{t}_1(\mathbf{x_1}), \mathfrak{t}_2(\mathbf{x_2}))$, or $d(\mathfrak{t}_1(\mathbf{x_1}), \mathfrak{t}_2(\mathbf{x_2}))$, where $\mathfrak{t}_1(\mathbf{x_1})$ and $\mathfrak{t}_2(\mathbf{x_2})$ are terms that we call *templates*. In this work we express source queries using *relational algebra* notation, omitting answer variables under the assumption that they coincide with the variables used in the target atoms. Intuitively, a template $\mathfrak{t}(\mathbf{x})$ in a target atom of a mapping corresponds to an *R2RML template*, and is used to generate object *IRIs* (Internationalized Resource Identifiers) or RDF *literals*, starting from database values retrieved by the source query in that mapping.

In our examples, we use the concrete mapping syntax adopted by the OBDA system Ontop [6], in which the source query is expressed in SQL and each target atom is expressed as an *RDF triple pattern with templates*. The answer variables of the source query are indicated in a target atom by enclosing them in curly brackets (`{···}`). A mapping example for the fragment of Fig. 1, expressed in such syntax, is the following:

```
target    :Project-{p_id} a :EC-Project . :Project-{p_id} :cordisRef {cordis_ref} ...
source    SELECT p.unics_id AS p_id, p.cordis_ref AS cordis_ref
          FROM   unics_cordis.projects p
```

The effect of such a mapping, when applied to a database instance $\mathcal{D}$ for Σ, is to instantiate, for each answer tuple returned by the source query, each (RDF) triple pattern with templates in the target to an actual RDF triple. This is done using IRIs and literals that are constructed from the assignments to the answer variables `p_id` and `cordis_ref`, obtained when the source query is evaluated over $\mathcal{D}$.

Table 1. Portion of schema-driven patterns from [8]

E-R DIAGRAM	DB SCHEMA	MAPPING PATTERN	ONTOLOGY
		Schema Entity (SE)	
	$T_E(\underline{\mathsf{K}}, \mathsf{A})$	s: T_E t: $C_E(\mathsf{t}_E(\mathsf{K}))$, $\{d_A(\mathsf{t}_E(\mathsf{K}), A)\}_{A \in \mathsf{K} \cup \mathsf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathsf{K} \cup \mathsf{A}}$
		Schema Relationship (SR)	
	$T_E(\underline{\mathsf{K}_E}, \mathsf{A}_E)$ $T_F(\underline{\mathsf{K}_F}, \mathsf{A}_F)$ $T_R(\underline{\mathsf{K}_{RE}, \mathsf{K}_{RF}})$	s: T_R t: $p_R(\mathsf{t}_E(\mathsf{K}_{RE}), \mathsf{t}_F(\mathsf{K}_{RF}))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$
In case of (_, 1) cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes K_{RE} (resp., K_{RF}).			

2.2 Mapping Patterns

(Ontology) mapping patterns [8] emerge when mapping a database to a *domain ontology*, and explain the link between the conceptualization behind the database design and the domain ontology. To justify our formalization of patterns, we make the following two fundamental observations: *(i)* a conceptual schema may have more than one admissible relational mapping, according to the applied methodology, as well as to considerations about efficiency, performance optimization, and space consumption on the final information system; *(ii)* given the logical schema of a relational database, regardless of its normal form, multiple conceptual schemata can provide (admissible) alternative representations of its domain. By *(i)* and *(ii)*, we explicitly associate a conceptual schema to each database schema in order to disambiguate among possible conceptualizations of the database schema, unlike bootstrapping-oriented approaches, *e.g.*, [13].

Formally, a mapping pattern is a quadruple $(\mathcal{C}, \mathcal{S}, \mathcal{M}, \mathcal{O})$, where $\mathcal{C}$ is a conceptual schema, $\mathcal{S}$ is a database schema, $\mathcal{M}$ is a set of mappings, and $\mathcal{O}$ is an ontology. In such mapping pattern, the pair $(\mathcal{C}, \mathcal{S})$ is the *input*, putting into correspondence a conceptual representation to one of its (many) admissible (*i.e.*, formally sound) database schemata. Such variants are due to differences in the applied methodology, considerations about efficiency, performance optimization, and space consumption of the final database. The pair $(\mathcal{M}, \mathcal{O})$, instead, is the output, where the *database schema ontology* $\mathcal{O}$ [25] is the OWL 2 QL encoding of the conceptual schema $\mathcal{C}$, and the set $\mathcal{M}$ of *database schema mappings* provides the link between $\mathcal{S}$ and $\mathcal{O}$. Database schema ontology refers to an ontology whose concepts and properties reflect the constructs of the conceptual schema, mirroring the structure of the relational database.

Table 1 shows two examples of patterns, namely, **Schema Entity** (SE) and **Schema Relationship** (SR). SE is a fundamental pattern that considers a single table T_E with primary key K and other attributes A. The pattern captures how T_E is mapped into a corresponding class C_E. The primary key of T_E is employed to construct the objects that are instances of C_E, using a template t_E specific for that class. Each relevant attribute of T_E is mapped to a data property of C_E.

Example. The `projects` table (Fig. 1) contains ids of projects (attribute `unics_id`), together with their funding scheme, their reference in the CORDIS portal[3], *etc.* It is mapped to the :EC-Project class using `unics_id` to construct its objects. In addition, every attribute in the table is mapped to a corresponding data property.

SR considers three tables T_R, T_E, and T_F, in which the primary key of T_R is partitioned into two parts K_{RE} and K_{RF} that are foreign keys to T_E and T_F, respectively. T_R has no additional attributes. The pattern captures how T_R is mapped to an object property p_R, using the two parts K_{RE} and K_{RF} of the primary key to construct respectively the subject and the object of each triple in p_R.

Example. The table `project_erc_panels` (Fig. 1) connects through two foreign keys the projects to their corresponding ERC panel. Such table is mapped to an :ercPanel object property, for which the ontology asserts that the domain is the class :Project and the range is an additional class :ERC-Panel, which correspond to the `erc_panels` table.

2.3 The Alignment of Data Sources with Mapping Patterns Problem

Let $\mathcal{P}$ be a set of mapping patterns, representing the elementary semantics of an application domain. For the scope of this paper, $\mathcal{P}$ is composed of the patterns proposed by Calvanese *et al.* [8] and illustrated in Sect. 2.2. In addition, recall that $\mathcal{S}$ is a database schema (see Sect. 2.1) composed of $\Sigma = \{T_1, T_2, \ldots, T_n\}$ and Γ, which captures database constraints, including primary and foreign keys. We assume that $\mathcal{S}$ was created from some conceptual model (*e.g.*, the way a relational database schema is created from an ER diagram) and that $\mathcal{P}$ represents the mapping patterns whose inputs are in line with what a designer used when transforming such conceptual model to a database schema. The problem we address in this paper is a reverse engineering one, essentially aligning the tables of Σ with the mapping patterns in $\mathcal{P}$ using Γ.

Formally, we denote by $M(\mathcal{S}, \mathcal{P}) \subseteq \mathcal{S} \times \mathcal{P}$ (M, for short) an alignment between a database schema $\mathcal{S} = (\Sigma, \Gamma)$ and a set $\mathcal{P}$ of mapping patterns. An *alignment* M is a set of correspondences (S, p), each representing an assignment of a schema S to a mapping pattern p whose input database schema can be instantiated to S. Note that a schema S may be involved in more than one correspondence. An alignment consists of a subset of all mapping patterns whose input is in line with the design of the database and we are interested in a *maximal alignment* that represents the full set of such mapping patterns.

Problem 1 (alignment of data sources with mapping patterns). *Let $\mathcal{S} = (\Sigma, \Gamma)$ be a database schema and $\mathcal{P}$ a set of mapping patterns. The* alignment of data sources with mapping patterns *problem aims to find a maximal alignment $M(\mathcal{S}, \mathcal{P})$, such that, for each pair $(S, p) \in M(\mathcal{S}, \mathcal{P})$, the input database schema of p can be instantiated to S.*

[3] https://cordis.europa.eu/projects/en.

3 Extracting Semantics from Data Sources with ADaMaP

We are now ready to describe the proposed ADaMaP algorithm (Sect. 3.1) and discuss some possible usages of the discovered alignment (Sect. 3.2).

3.1 ADaMaP: Automatically Extracting Semantics from Data Sources

We now introduce ADaMaP, an iterative algorithm that automatically aligns a relational data source to mapping patterns (see Sect. 2.2 and a summarization of abbreviations in Table 2). ADaMaP is applied to each table $T \in \Sigma$ separately, aiming to determine the most suitable set of mapping patterns. The utilization of the mapping patterns requires some or all of the following properties in the definition of a relational data source schema design: (1) primary keys, (2) foreign keys, and (3) unique constraints [8]. For the scope of this paper we assume that such properties are well defined and note that in the absence of such properties, discovery methods may be applied, for example, randomness [26] can be used to recover foreign keys.

Given a table T, the inference of ADaMaP is divided into four cases, each targeting a different amount of table relationships with respect to Σ. The table-based inference is illustrated in Fig. 2. We denote T's primary key by K_T and its foreign key(s) by FK_T.

Whenever a table does not have foreign keys, the corresponding mapping pattern is set to be Schema Entity (SE) as shown in Fig. 2a. Intuitively, this means that in the absense of known relationships, the table should be mapped into an entity set. If a table has a single relationship with a reference table R, ADaMaP applies the inference in Fig. 2b and checks whether the primary key of R, K_R, is the same as the foreign key FK_R. If not, we return to the case of SE. If it is, we check the same condition for the examined table T. If K_T is a foreign key in T, we assign T with a Schema Hierarchy (SH), *i.e.,* recognizing that the entity set corresponding to the table T is a sub-class of the entity set corresponding to R. If not, we check whether the foreign key is a key in T and decide between adding a correspondence of T with SHa (requiring an identifier alignment in case it is) and adding two correspondences to the alignment, (T, SRm) and (T, SE), meaning that T and R should be merged into a single entity.

For a table T with two foreign keys, FK_T^1 referring to table R_1 and FK_T^2 to table R_2 (Fig. 2c), We denote $\mathsf{FK}_T = \mathsf{FK}_T^1 \cup \mathsf{FK}_T^2$. We first check whether $\mathsf{K}_T == \mathsf{FK}_T$. In case of a negative answer, as in most other negative answers in Fig. 2c, we roll back to use the inference in Fig. 2b for each of the foreign keys in T, *i.e.*, as if T has a single foreign key. Then, regardless of the former answer, we check whether the primary keys of R_1 and R_2 are the same as the respective foreign keys FK_{R_1} and FK_{R_2}. The final check in Fig. 2c is conditioned on whether one of the referenced tables contains a foreign key that identifies (id. in the figure) the table. We note here that in case we do not roll back to Fig. 2b, the inference of Fig. 2c may obtain a correspondence between T and one of the following schema relationship patterns: 1) a "simple" relationship

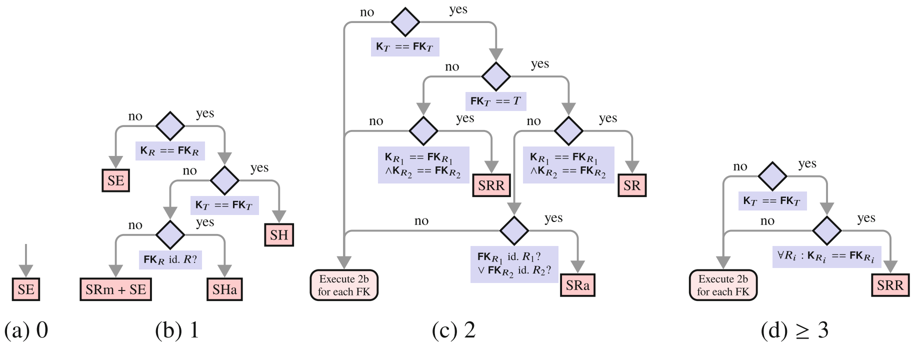


Fig. 2. ADAMAP inference for a table T by the number of foreign keys it contains.

between R_1 and R_2 (SR) if T is only composed of the foreign keys, 2) a reified relationship (SRR), where all the tables are identified by their foreign keys and may have additional attributes, and 3) a relationship that requires an alignment of attributes (SRa) in either R_1 or R_2.

In another special case, a table has three or more references to other tables. The inference of this case, illustrated in Fig. 2d, is quite similar to the path in Fig. 2c resulting in SRR. The idea is that T represents a relationship between multiple tables in DB, each identified by the foreign key to T and may include additional attributes.

Table 2. Patterns abbreviations

Pattern	Abbreviation
Schema entity	SE
Schema relationship	SR
→with identifier alignment	SRa
→with merging	SRm
Schema reified Relationship	SRR
Schema hierarchy	SH
→with identifier alignment	SHa

Finally, by applying the rules illustrated in Fig. 2 for every T in Σ, we obtain an alignment M as a solution to Problem 1. We now illustrate the execution of ADAMAP using the tables in Fig. 1.

Example. Both `projects` and `erc_panels` (see Fig. 1) do not contain a foreign key constraint and thus ADAMAP creates a correspondence between these tables and an SE pattern according to Fig. 2a. `project_erc_panels` has two foreign keys and accordingly we use the inference of Fig. 2c. However, since the union of foreign keys (`project`, `panel`) is not the primary key in `project_erc_panels`, we roll back to the inference in Fig. 2b. In both tables the foreign key refers to a primary key (`unics_id` in `projects` and `code` in `erc_panels`), satisfying the first condition. As for the second condition, we observe that `project` (the foreign key from `projects`) is a primary key in `project_erc_panels`. However, `panel` (the foreign key from `erc_panels`) is not. In the case of `projects` we obtain a correspondence with SH and for `erc_panels`, since `panel` does not identify `project_erc_panels` we resolve a correspondence with two patterns, namely, SRm and SE. The output of ADAMAP is therefore {(`projects`, SE), (`erc_panels`, SE) (`project_erc_panels`, SH), (`project_erc_panels`, SRm), (`project_erc_panels`, SE)}.

3.2 Usage of Aligning Data Sources with Mapping Patterns

ADaMaP provides an automatic approach to enriching a database with semantics using matching patterns. We now describe two possible usages of such a method.

Bootstrapping. In this setting, we only have a conceptual schema of the domain. Using ADaMaP, a database schema can be transformed into a set of patterns. Once the patterns are aligned with the tables in the database schema, they can be used to (semi-)automatically bootstrap a set of mappings, which can then be further refined and extended manually, possibly exploiting again the discovered patterns. These mappings, in turn, may also be applied to bootstrap an ontology, providing the application domain with an additional level of abstraction.

Schema Cover. The idea of schema cover was first introduced by Saha *et al.* [22], promoting reuse and collaboration among data source providers. Such reuse is based on a repository of information building blocks, referred to as concepts, representative of entities in the domain of discourse (*e.g.*, ERC panels). Schemata are mapped against a set of concepts in a process termed schema cover. The idea is to "cover" a schema and thereby interpret the schema in terms of known concepts. This way, the schema is integrated into an existing body of information and knowledge. For example, consider a network of researchers that represent diverse interdisciplinary research skills and cooperate to submit joint research proposals. The analysis of capabilities does not follow a common format or standard. The aim of schema cover is to allow creating in this case research consortia in response to specific call for proposals.

4 Empirical Evaluation

In this section we provide an empirical evaluation of ADaMaP. Experimental settings are given in Sect. 4.1 and the empirical results are analyzed in Sect. 4.2.

4.1 Experiments Setting

Considered Scenarios. The first aspect to consider is the choice of the experimental scenarios. To assess the feasibility of the approach in practice, we focus on non-trivial and real-world scenarios. Such real-world scenarios should be built around reasonably well-designed database schemata, providing the (primary-key and foreign-key) constraints that are needed by our approach, and any bootstrapping approach in general. We identified two such scenarios, provided in an online repository,[4] and detailed next.

NPD Benchmark (NPD). [15] This scenario is built around the domain of oil and gas extraction, presenting a high number of mappings (>1k). Most mappings were automatically generated, but there are numerous complex manually-written mappings as well. The ontology falls in the OWL 2 QL profile, and consists of

[4] https://github.com/ontop/ontop-examples/tree/master/caise-2021-patterns.

4176 TBox axioms over 343 classes, 142 object properties, and 238 data properties. The database schema consists of 70 tables, 962 columns, and 89 foreign key constraints. The database schema has a structure, but was not designed according to common conceptual modeling practices. In fact, it was automatically generated out of unstructured data as CSV files [14].

CORDIS. This setting is designed around the domain of competitive research projects, provided by SIRIS Academic S.L.[5], a consultancy company specializing in higher education and research. The mappings were manually-written, and they amount to 120. The ontology, expressed in OWL 2 QL, consists of 186 TBox axioms over 24 classes, 24 object properties, and 30 data properties. The database schema is quite well-structured and consists of 19 tables, 6 views, 95 columns, and 20 foreign-key constraints.

Table 3. Coverage analysis

Pattern	#usages	#mappings
SE	13	60
SR	3	3
SRm	3	3
SRR	1	16
Covered Mappings: 89 (out of 120)		

(a)CORDIS Coverage

Pattern	#usages	#mappings
SE	61	454
SRm	74	74
SRR	1	12
SH	3	132
Covered Mappings: 672 (out of 1173)		

(b)NPD Coverage

Database Schema Ontology vs. Domain Ontology. Recall that a mapping pattern puts into correspondence a database schema $\mathcal{S}$, together with its intended conceptualization $\mathcal{C}$, to a pair $(\mathcal{M}, \mathcal{O})$, where $\mathcal{M}$ is a set of database schema mappings and $\mathcal{O}$ is a database schema ontology. Differently from an arbitrary *domain* ontology, $\mathcal{O}$ provides an information-preserving encoding of $\mathcal{C}$, *modulo* the expressivity of OWL 2 QL.

In real-world scenarios, however, it is usually the case that the domain ontology is developed independently from the relational data-source. The NPD and CORDIS scenarios we consider here are no exception to this. This results in a misalignment between the domain ontology and the conceptual schema used for the database, which in turn results in a misalignment between the database schema ontology and the domain ontology. *In our experimental evaluation we shall provide a quantitative measure over this misalignment, for both the* NPD *and* CORDIS *scenarios.*

Applied vs. Discovered Patterns. The conceptual schema of a database, which serves in the design of our patterns, is typically discarded after the design and deployment phases, and therefore it is actually not available as input to AdaMaP. As a result, the algorithm cannot disambiguate all the possible conceptualizations corresponding to the same database schema, but instead chooses one of them (literally, it operates according to the "most-typical" application of a pattern as per [8]).

[5] https://www.sirisacademic.com/wb/.

To check the efficacy of this approach, we have manually analyzed the scenarios and categorized the mappings according to the schema-driven mapping patterns that were actually used in such scenarios.

Evaluation Measures. We use quantitative evaluation measures to measure the differences between ADaMaP's results and the manual analysis. The latter serves as a reference model when comparing the automatically generated and manually extracted reference alignments. M denotes the output of ADaMaP and M^* denotes a manually extracted reference alignment. We use the terms *coordinated positive* and *coordinated negative* to represent agreement between M and M^* on the presence and absence of correspondences, respectively. Disagreements are marked as *discoordinated positive* for correspondences that were identified by the algorithm but not part of the manual alignment and *discoordinated negative* for the opposite situation. We use the well-known precision and recall measures to measure ADaMaP's success in aligning a database schema with a set of mapping patterns, with respect to a manually extracted alignment. Precision (P) measures the ratio of coordinated positive correspondences out of all correspondences assigned by the algorithm. On the other hand, Recall (R) measures the number of coordinated positive correspondences from all the correct correspondences as given in the reference alignment. P and R are formally defined as follows:

$$P_{M^*}(M) = \frac{\mid M \cap M^* \mid}{\mid M \mid}, \qquad R_{M^*}(M) = \frac{\mid M \cap M^* \mid}{\mid M^* \mid} \tag{1}$$

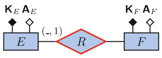
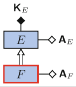
We use precision and recall to define the F1-measure, $F_{M^*}(M)$, calculated as the harmonic mean of $P_{M^*}(M)$ and $R_{M^*}(M)$.

4.2 Results

Coverage Analysis. To analyze to what extent the database schema ontology is aligned with the domain ontology, we check how many mappings in the analyzed scenarios can be explained through the mapping patterns in [8]. A mapping that cannot be justified this way suggest a misalignment between the database schema and the domain ontology.

Tables 3(a) and 3(b) report on the number of schema-driven mapping-pattern applications that were manually reported in CORDIS and NPD, respectively, as well as the total number of mappings that are covered by these patterns. For CORDIS, 89 out of 120 mappings (74.16%) can be explained by a schema-driven pattern, whereas for NPD the situation is slightly worse, with only 672 out of 1173 mappings (57.29%). This can be attributed to the fact that the database schema of NPD was not designed according to well-known good practices of conceptual modeling, but was rather automatically generated out of CSV's semi-structured data [14]. The remaining mappings, non explainable through schema-driven patterns, fill the gap between the abstraction levels used in the database schema and the domain ontology.

Table 4. Portion of schema-driven patterns from [8]

E-R DIAGRAM	DB SCHEMA	MAPPING PATTERN	ONTOLOGY
	Schema Relationship with Merging (SRm)		
	$T_F(\underline{\mathsf{K}_F}, \mathsf{A}_F)$ $T_E(\underline{\mathsf{K}_E}, \mathsf{K}_{EF}, \mathsf{A}_E)$	s: T_E t: $p_{EF}(\mathrm{t}_E(\mathsf{K}_E), \mathrm{t}_F(\mathsf{K}_{EF}))$	$\exists p_{EF} \sqsubseteq C_E$ $\exists p_{EF}^- \sqsubseteq C_F$
	Schema Hierarchy (SH)		
	$T_E(\underline{\mathsf{K}_E}, \mathsf{A}_E)$ $T_F(\underline{\mathsf{K}_{FE}}, \mathsf{A}_F)$	s: T_F t: $C_F(\mathrm{t}_E(\mathsf{K}_{FE}))$, $\{d_A(\mathrm{t}_E(\mathsf{K}_{FE}), A)\}_{A \in \mathsf{A}_F}$	$C_F \sqsubseteq C_E$ $\{\exists d_A^- \sqsubseteq C_F\}_{A \in \mathsf{A}_F}$

Mismatches Analysis. The following relates separately to CORDIS and NPD, with reference to Fig. 3.

CORDIS. We observe that the algorithm and the manual analysis disagree on 7 instances, with 5 discoordinated positives and 2 discoordinated negatives. In terms of precision (P), recall (R), and F1-measure (F), ADAMAP obtains the following results:

$$P_{M^*}(M) = R_{M^*}(M) = F_{M^*}(M) = 0.8$$

ADAMAP discovered 80% of the manually assigned correspondences (recall) and 80% of the correspondences assigned by ADAMAP were also assigned manually (precision). Overall, ADAMAP and the manual extraction have 20% of disagreements. All but one disagreement stem from the fact that multiple conceptual schemata can correspond to the same database schema, as observed above. The algorithm cannot determine which of these equally valid choices is actually the one that was adopted by the human designer.

For the table `project_erc_panels` depicted in Fig. 1, the algorithm identifies two applicable patterns, namely SH and SRm from Table 4. The application of SH is justified by the foreign key `project` that coincides with the primary key of table `project_erc_panels`. Under this plausible modeling point of view, projects having an ERC panel are a subclass of projects. The application of *SRm*, instead, is justified by the foreign key `panel`, *i.e.*, the `1-N` relationship between `project_erc_panels` and `erc_panels` has been merged into the former.

However, as introduced in Sect. 2, such table may also match the less typical pattern SR, which is actually the one we observed in the CORDIS scenario. For such reason, the two findings by the algorithm have been categorized as discoordinated positives since the algorithm applied (still suitable) patterns that are different from the one that was chosen in the manual alignment.

```
target    :NUTS2-{nuts_code} a :NUTS2 ; :extendedName {nuts_desc} .
source    SELECT etu.nuts_code AS nuts_code, etu.description AS nuts_desc
          FROM unics_cordis.eu_territorial_units etu
          WHERE etu.nuts_level=2

target    :NUTS3-{nuts_code} a :NUTS3 . :extendedName {nuts_desc} .
source    SELECT etu.nuts_code AS nuts_code, etu.description AS nuts_desc
          FROM unics_cordis.eu_territorial_units etu
          WHERE etu.nuts_level=3
```

Another mismatch, shown above, relates to the table `eu_territorial_units`, which has not been modeled as a separate entity, but rather as a *clustering* of different classes based on the value of attribute `nuts_level`. In these two mappings, two classes are created: `:NUTS2` captures all the *nomenclatures of territorial units for statistics* having *level of division* equal to 2 and `:NUTS3` captures the case where the level of division is 3.

Such clustering cannot be recognized by working at the schema level, but rather requires to inspect the actual data. Consequently, it cannot possibly be discovered by ADaMaP. This calls for an interesting extension of our algorithm, where also this and other forms of *data-driven mapping patterns* [8] are supported.

NPD. For the NPD scenario, we observe that the algorithm and the manual analysis disagree on 35 instances, with 14 discoordinated positives and 21 discoordinated negatives. In terms of precision (P), recall (R), and F1-measure (F), ADaMaP obtains the following results:

$$P_{M^*}(M) = 0.88, R_{M^*}(M) = 0.82,$$
$$F_{M^*}(M) = 0.85$$

Compared to the CORDIS scenario, ADaMaP obtains better results. This is because a portion of mappings for NPD were automatically bootstrapped, which results in the most-typical pattern being applied. Also, in the case of NPD, we observe that ADaMaP showed higher precision than recall, suggesting that ADaMaP may be better in obtaining an agreement with the manual alignment than covering the full scope of correspondences.

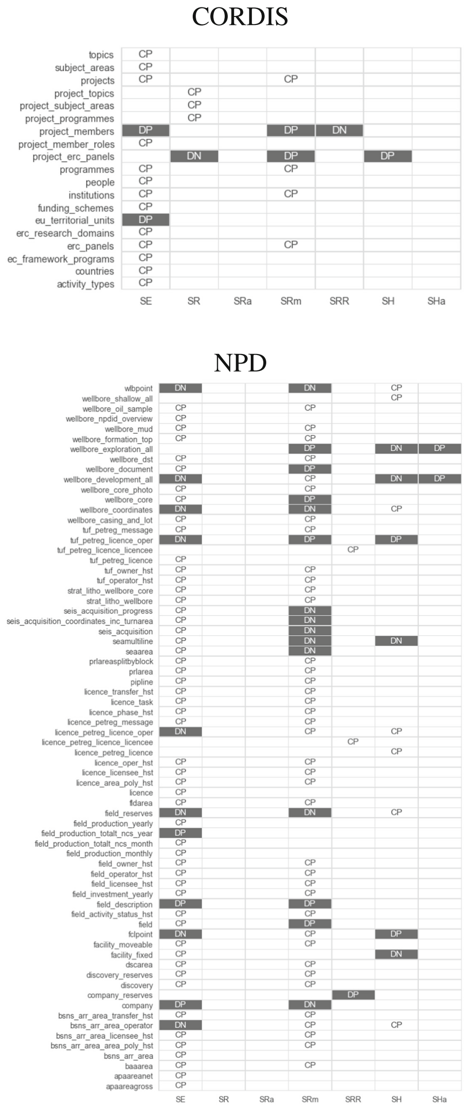


Fig. 3. Algorithm vs Manual analysis

The reasons for the disagreements are totally analogous to those we observed for CORDIS. Something peculiar about this scenario, that we did not observe in CORDIS, is the presence of mistakes both at the level of the database schema and at the level of patterns application. For the former, tables `seaArea`, `seis_acquisition`, `wellbore_core_photo`, and `apaAreaNet` declare non-minimal superkeys[6] as primary keys. Database design theory tells us that this is a conceptual modeling error. Such mistakes in the schema led to "non-conventional" applications of mapping patterns, such as the following:

- `seis_acquisition`: URIs are not built from the primary key of that table, but rather from a proper subset of the primary key that is declared as `UNIQUE`.
- `wellbore_core_photo`: `wellbore_core_photo_id` is `UNIQUE`, strictly contained in the primary key while URIs are built from the (non-minimal) primary key.
- Similar choices to the one above are taken for tables `apaAreaNet` and `wellbore_mud`.

Altogether wrong applications of a pattern are present as well. For instance, for table `seaArea`, the primary key is the pair of attributes `(seaArea_id, seaSurveyName)`. However, attribute `seaArea_id` is declared as `UNIQUE` in the schema. This implies that the primary key is, again, a non-minimal superkey. However, the mapping-designer here has chosen, for building URIs, neither the primary key, nor the unique attribute, but actually the non-key attribute `seaSurveyName`. This breaks the principle of lossless transformation: the `1-1` correspondence between table rows and individuals in the ontology is lost.

5 Related Work

Multiple tools and approaches deal with the problem of extracting an ontology from a relational data source [2,5,11,13,17,19,23,24] have been proposed. The addressed application scenarios span from OBDA and Virtual Knowledge Graph (VKG) systems construction, reverse engineering, data integration, ontology learning, reasoning-based constraints checking, *etc.*. They differ mainly in the ontology languages they support and the required level of automation yet only a few come with a systematic categorization of the mappings that they produce as declarative connection between the data sources and the ontology [2,13,17,23]. The comparison of mapping patterns to alternative categorizations is out of this paper scope (see [8]) and we argue that AdaMaP is agnostic with respect to the mapping patterns nature in place and can, in principle, be fed with any catalog of patterns with a proper formal specification.

The analysis of real-world OBDA scenarios offered in this paper represents an original contribution to the current literature on ontology and mapping extraction from relational data sources, and a novel way to evaluate the performances of an algorithm such as AdaMaP, which is meant to support the identification of suitable and semantic-preserving patterns from relational schemata. To the best of our knowledge, none of the former approaches aim at showing that the mapping patterns (and the ontologies) they produced are sufficiently sound and

[6] Recall that in database theory, a key is a minimal superkey.

complete to reflect the real design choices and conceptual modeling practices that are used by expert designers on real-world scenarios.

The term *ontology mapping patterns*, used in this work to describe semantics, should not be confused with *ontology design patterns* (ODP). In ontology engineering, the latter provides solutions to recurrent modeling issues, and their adoption improves quality in terms of the ontology axioms specification [12]. Ontology mapping patterns stem from observation and categorization of typical relational database structures, their associated constraints and conceptual models.

6 Conclusions

We have introduced ADaMaP, an algorithmic technique that extracts semantics from a relational data source, by automatically identifying how ontology mapping patterns are applied to fragments of its schema. With such identification process each fragment gets projected into a set of ontological axioms, together with mapping rules capturing the schema-to-ontology correspondence. Thanks to ADaMaP, the creation of the ontology and of the mapping rules is no longer completely manual, error-prone effort. The validation of ADaMaP in two real-world case studies confirms that the identified patterns by-and-large agree with those detected by a human expert.

The patterns identified by ADaMaP provide a solid basis that can be manually improved by human experts, overcoming the "blank-page" syndrome when setting up ontology-based data access and integration systems. In addition, the identified patterns can be instrumental in a number of consequent tasks: in data engineering, tackling central problems such as ontology bootstrapping and schema cover, and in process mining, where the increasing focus on artifact-centric [10] and object-centric [1,3] processes requires to reconstruct conceptual data models from event data [4].

As discussed, ADaMaP comes with some limitations that should be tackled. First and foremost, for a given relational schema there are in general many possible combinations of mapping patterns that are, in principle, equally valid. While the current version of ADaMaP returns the "most typical" of such combinations, it would be interesting to allow ADaMaP to incrementally explore multiple possibilities, for example by iteratively generating and recommending alternatives that could then be inspected and further explored by human experts. Second, currently ADaMaP only focuses on the schema of the data source, without exploiting the data stored therein. In a number of situations, determining whether a given mapping pattern can be suitably applied requires to simultaneously inspect the schema, the data, and potential additional constraints that can be inferred from such data. We wish to enrich ADaMaP with data-driven features, allowing it to account not only for the schema-driven mapping patterns considered in this work, but also for the data-driven mapping patterns categorized in [8].

Acknowledgements. This research has been partially supported by the EU H2020 project INODE (grant agreement No 863410), by the Italian PRIN project HOPE, by the European Regional Development Fund (ERDF) Investment for Growth and Jobs Programme 2014–2020 through the project IDEE (FESR1133), and by the Free University of Bozen-Bolzano through the projects QUADRO, KGID, GeoVKG, and STyLoLa. Diego also acknowledges the support of the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and Gal the support of the Benjamin and Florence Free Chair. We thank Kurt Stockinger for his feedback.

References

1. Aalst, W.M.P.: Object-centric process mining: dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (eds.) SEFM 2019. LNCS, vol. 11724, pp. 3–25. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30446-1_1
2. Arenas, M., Bertails, A., Prud'hommeaux, E., Sequeda, J.: A direct mapping of relational data to RDF. W3C Recommendation, World Wide Web Consortium (September 2012)
3. Artale, A., Kovtunova, A., Montali, M., van der Aalst, W.M.P.: Modeling and reasoning over declarative data-aware processes with object-centric behavioral constraints. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 139–156. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_11
4. Bano, D., Weske, M.: Discovering data models from event logs. In: Dobbie, G., Frank, U., Kappel, G., Liddle, S.W., Mayr, H.C. (eds.) ER 2020. LNCS, vol. 12400, pp. 62–76. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62522-1_5
5. Bizer, C., Seaborne, A.: D2RQ: treating non-RDF databases as virtual RDF graphs. In: Proceedings of the 3rd International Semantic Web Conference (ISWC). LNCS, vol. 3298. Springer, Heidelberg (2004)
6. Calvanese, D., et al.: Ontop: answering SPARQL queries over relational databases. Semantic Web J. **8**(3), 471–487 (2017)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J. Automat. Reasoning **39**(3), 85–429 (2007) https://doi.org/10.1007/s10817-007-9078-x
8. Calvanese, D., Gal, A., Lanti, D., Montali, M., Mosca, A., Shraga, R.: Mapping patterns for virtual knowledge graphs. CoRR Technical Report arXiv:2012.01917, arXiv.org e-Print archive (2020), https://arxiv.org/abs/2012.01917
9. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C Recommendation, World Wide Web Consortium (September 2012)
10. Fahland, D.: Artifact-centric process mining. In: Encyclopedia of Big Data Technologies. Springer, Heidelberg (2019) https://doi.org/10.1007/978-3-319-77525-8_93
11. Gupta, S., Szekely, P., Knoblock, C.A., Goel, A., Taheriyan, M., Muslea, M.: Karma: a system for mapping structured sources into the Semantic Web. In: Proceedings of the Extended Semantic Web Conference (ESWC) (2012)
12. Hitzler, P., Gangemi, A., Janowicz, K.: Ontology Engineering with Ontology Design Patterns: Foundations and Applications, vol. 25. IOS Press, Amsterdam (2016)

13. Jiménez-Ruiz, E., et al.: BootOX: practical mapping of RDBs to OWL 2. In: ArenasArenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 113–132. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25010-6_7
14. Kharlamov, E., et al.: Optique 1.0: Semantic access to Big Data: The case of Norwegian Petroleum Directorate's FactPages. In: Proceedings of the 12th Int. Semantic Web Conference, Posters & Demonstrations Track (ISWC). CEUR Workshop Proceedings, vol. 1035, pp. 65–68 (2013) http://ceur-ws.org/
15. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD benchmark: Reality check for OBDA systems. In: Proceedings of the 18th International Conference on Extending Database Technology (EDBT), pp. 617–628 (2015)
16. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 319–334. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32885-5_25
17. de Medeiros, L.F., Priyatna, F., Corcho, O.: MIRROR: automatic R2RML mapping generation from relational databases. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 326–343. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19890-3_21
18. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, World Wide Web Consortium (December 2012)
19. Pinkel, C., Binnig, C., Kharlamov, E., Haase, P.: IncMap: pay as you go matching of relational schemata to OWL ontologies. In: Proceedings of WS on Ontology Matching. CEUR Workshop Proceedings, vol. 1111 (2013) http://ceur-ws.org/
20. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Seman. **10**, 133–173 (2008)
21. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**, 334–350 (2001) https://doi.org/10.1007/s007780100057
22. Saha, B., Stanoi, I., Clarkson, K.L.: Schema covering: a step towards enabling reuse in information integration. In: Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE), pp. 285–296. IEEE Computer Society (2010)
23. Sequeda, J., Priyatna, F., Villazón-Terrazas, B.: Relational database to RDF mapping patterns. In: Proceedings of the 3rd International Conference on Ontology Patterns, WOP 2012, vol. 929. pp. 97–108. CEUR-WS.org, Aachen, DEU (2012)
24. Sequeda, J.F., Miranker, D.P.: Ultrawrap mapper: a semi-automatic relational database to RDF (RDB2RDF) mapping tool. In: Proceedings of the 14th International Semantic Web Conference, Posters & Demonstrations Track (ISWC). CEUR Workshop Proceedings, vol. 1486 (2015) http://ceur-ws.org/
25. Spanos, D.E., Stavrou, P., Mitrou, N.: Bringing relational databases into the semantic web: a survey. Semantic Web J. **3**(2), 169–209 (2012)
26. Zhang, M., Hadjieleftheriou, M., Ooi, B.C., Procopiuc, C.M., Srivastava, D.: On multi-column foreign key discovery. Proc. VLDB Endowment **3**(1–2), 805–814 (2010)