

ADAMAP: Automatic Alignment of Data Sources using Mapping Patterns

Diego Calvanese^{1,2}, Avigdor Gal³, Naor Haba³, Davide Lanti¹,
Marco Montali¹, Alessandro Mosca¹, and Roei Shraga³

¹ Free-University of Bozen-Bolzano, Bolzano, Italy, *lastname@unibz.it*

² Umeå University, Sweden

³ Technion – Israel Institute of Technology, Haifa, Israel

{avigal@, naor-haba@campus., 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. We observe that relational databases 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-commerce) [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 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 centred 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 leads, for example, 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 only inspecting relational data sources, without accessing 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 consequent 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 Section 3 followed by an empirical evaluation (Section 4). The paper is concluded with related work (Section 5) and concluding discussion (Section 6).

2 Model

We now detail the building blocks for our proposed method, namely the OBDA framework (Section 2.1) and mapping patterns (Section 2.2), followed by problem definition (Section 2.3). 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.¹

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 database 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_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.

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

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

where $Q(\mathbf{x})$ is a SQL query over the database schema Σ , called *source query*, and $\mathbf{L}(t(\mathbf{x}))$ is a list of *target atoms* of the form $C(t_1(\mathbf{x}_1))$, $p(t_1(\mathbf{x}_1), t_2(\mathbf{x}_2))$, or $d(t_1(\mathbf{x}_1), t_2(\mathbf{x}_2))$, where $t_1(\mathbf{x}_1)$ and $t_2(\mathbf{x}_2)$ 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 $t(\mathbf{x})$ 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 $\{\dots\}$. A mapping example for the fragment of Figure 1, expressed in such syntax, is the following:

¹ <https://cordis.europa.eu/projects/en>

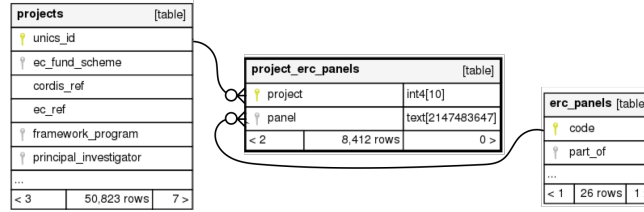


Fig. 1: Fragment of the CORDIS database.

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} .

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. Formally, a mapping pattern is a quadruple (C, S, M, O) where C is a conceptual schema, S is a database schema, M is a set of mappings, and O is an (OWL 2 QL) ontology. In such mapping pattern, the pair (C, 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 (M, O) , instead, is the *output*, where the *database schema ontology* O is the OWL 2 QL encoding of the conceptual schema C , and the set M of *database schema mappings* provides the link between S and O .


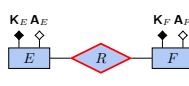
Table 1 shows two examples of patterns, namely, **Schema Entity (SE)** and **Schema Relationship (SR)**. SE is a fundamental pattern that considers a single table T_E with primary key \mathbf{K} and other attributes \mathbf{A} . The pattern captures how T_E is mapped into a corresponding class C_E . 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 (Figure 1) contains ids of projects (attribute `unics_id`), together with their funding scheme, their reference in the CORDIS portal², *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 \mathbf{K}_{RE} and \mathbf{K}_{RF} that are foreign keys to T_E and T_F , respectively. T_R

² <https://cordis.europa.eu/projects/en>

Table 1: Portion of Schema-driven Patterns from [8]

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

has no additional attributes. The pattern captures how T_R is mapped to an object property p_R , using the two parts \mathbf{K}_{RE} and \mathbf{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` (Figure 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 possible mapping patterns representing the elementary semantics of an application domain. For the scope of this paper, \mathcal{P} is composed of the patterns proposed in [8] and illustrated in Section 2.2. In addition, recall that \mathcal{S} is a database schema (see Section 2.1) composed of $\Sigma = \{T_1, T_2, \dots, T_n\}$ 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 Extracting Semantics from Data Sources with ADAMAP

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

3.1 ADAMAP: Automatically Extracting Semantics from Data Sources

ADAMAP is an iterative algorithm, applied to each table $T \in \Sigma$ separately. We assume that the schema design contains a definition of primary keys, foreign keys, and unique constraints. In the absence of such constraints, foreign key discovery [26] may be applied.

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 Figure 2 and its execution guidance is given in the pseudocode of Algorithm 1. We denote T 's primary key by \mathbf{K}_T and its foreign key(s) by \mathbf{FK}_T .

Whenever a table does not have foreign keys, the corresponding mapping pattern is set to be Schema Entity (SE) as shown in Figure 2a. If a table has a single relationship with a reference table R , ADAMAP applies the inference in Figure 2b and checks whether the primary key of R , \mathbf{K}_R , is the same as the foreign key \mathbf{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 \mathbf{K}_T is a foreign key in T , we assign T with a Schema Hierarchy (SH). If not, we check whether the foreign key is a key in T and decide between adding a correspondence of T with SHa (in case it is) and adding two correspondences to the alignment, (T, SRm) and (T, SE) .

For a table T with two foreign keys, \mathbf{FK}_T^1 referring to table R_1 and \mathbf{FK}_T^2 to table R_2 (Figure 2c), We denote $\mathbf{FK}_T = \mathbf{FK}_T^1 \cup \mathbf{FK}_T^2$. We first check whether $\mathbf{K}_T == \mathbf{FK}_T$. In case of a negative answer, as in most other negative answers in Figure 2c, we roll back to use the inference in Figure 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 \mathbf{FK}_{R_1} and \mathbf{FK}_{R_2} . The final check in Figure 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 Figure 2b, the inference of Figure 2c may obtain a correspondence between T and one of the following schema relationship patterns: 1) a “simple” relationship 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 is illustrated in Figure 2d and is quite similar to the path in Figure 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.

Algorithm 1 ADAMAP Inference

Input: a schema Σ and a set of patterns \mathcal{P}

Output: an alignment M

$M \leftarrow \emptyset$

for $T \in \Sigma$ **do**

apply Figure 2 with T to obtain a set of correspondences m

$M \leftarrow M \cup m$

end for

Return: M

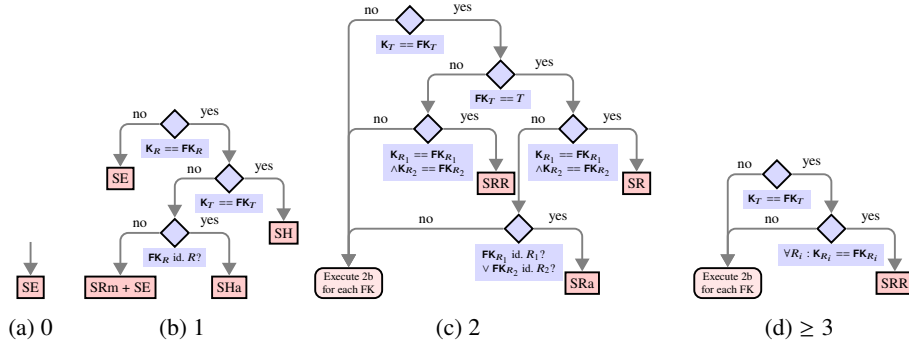


Fig. 2: ADAMAP inference for a table T by the number of foreign keys it contains.

Example. Recalling Figure 1, both `projects` and `erc_panels` do not contain a foreign key constraint and thus ADAMAP creates a correspondence between these tables and an SE pattern according to Figure 2a. `project_erc_panels` has two foreign keys and accordingly we use the inference of Figure 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 Figure 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 method.

Bootstrapping. In our 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 Section 4.1 and the empirical results are analyzed in Section 4.2.

4.1 Experiments Setting

Considered Scenarios. The first aspect to consider is the choice of the experimental scenarios. In order to assess the feasibility of the approach in practice, we need to 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, detailed next.

NPD Benchmark (NPD). [15] This scenario is built around the domain of oil and gas extraction. It presents the highest number of mappings (>1k). The majority of these 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 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 some structure, however it was not designed according to common conceptual modeling practices. In fact, it was automatically generated out of unstructured data in the form of CSV files [14].

CORDIS. This setting is designed around the domain of competitive research projects, provided by SIRIS Academic S.L.³, 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.

Database Schema Ontology vs. Domain Ontology. Our approach based on mapping patterns puts into correspondence a database schema, together with its intended conceptualization, to a pair “(OBDA mappings, database schema ontology)”. Following [25], with *database schema ontology* we refer here to an ontology whose concepts and properties reflect the constructs of the conceptual schema, mirroring the structure of the relational database. In our approach, the database schema ontology provides an

³ <https://www.sirisacademic.com/wb/>

information-preserving encoding of the conceptual schema input to each pattern, *modulo* the expressivity of the OWL 2 QL language.

In real-world scenarios, however, it is usually the case that a 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. We start from 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, more than one conceptual schema can be designed which provide (admissible) alternative representations of its domain. By (i) and (ii), [8] explicitly associate a conceptual schema to each database schema in order to disambiguate among other possible conceptualizations of this latter (unlike bootstrapping-oriented approaches that can be found in the literature).

The conceptual schema of a database is typically discarded after the design and deployment phases, and therefore it is not available as input to ADAMAP. As a result, our 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]).

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:

$$P_{M^*}(M) = \frac{|M \cap M^*|}{|M|}, R_{M^*}(M) = \frac{|M \cap M^*|}{|M^*|} \quad (1)$$

Pattern	#usages	#mappings	Pattern	#usages	#mappings
SE	13	60	SE	61	454
SR	3	3	SRm	74	74
SRm	3	3	SRR	1	12
SRR	1	16	SH	3	132
Covered Mappings 89 (out of 120)			Covered Mappings 672 (out of 1173)		
(a) CORDIS Coverage			(b) NPD Coverage		

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 2a and 2b 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.

Mismatches Analysis. The following relates separately to CORDIS and NPD.

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.

Table 3: Portion of Schema-driven Patterns from [8]

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

For the table `project_erc_panels` depicted in Figure 1, the algorithm identifies two applicable patterns, namely SH and SRm from Table 3. 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 Section 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 dis-coordinated 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 dis-coordinated positives and 21 dis-coordinated 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.

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 superkeys⁴ 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]. 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 few of them 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.

⁴ Recall that in database theory, a key is a minimal superkey.

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 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

topics	CP						
subject_areas	CP						
projects	CP			CP			
project_logics		CP					
project_subject_areas		CP					
project_programmes		CP					
project_members	DP			DP	DN		
project_member_roles	CP						
project_erc_panels		DN		DP		DP	
programmes	CP						
people	CP						
institutions	CP			CP			
funding_schemes	CP						
eu_territorial_units	CP						
erc_research_domains	CP						
erc_panels	CP			CP			
ec_framework_programs	CP						
countries	CP						
activity_types	CP						

(a) CORDIS

wilport	DN			DN		CP	
wilbore_shallow_all				CP			
wilbore_oil_sample	CP			CP			
wilbore_npidi_overview	CP			CP			
wilbore_murd	CP			CP			
wilbore_formation_top	CP			CP			
wilbore_exploration_all				DP		DN	DP
wilbore_gh	CP			CP			
wilbore_document	CP			CP			
wilbore_development_all	DN			CP		DN	DP
wilbore_core_photo	CP			CP			
wilbore_core	CP			CP			
wilbore_coordinates	DN			DN		CP	
wilbore_saving_and_hist	CP			CP			
lf_petreg_message	CP			CP			
lf_petreg_licence_sper	DN			DP		CP	
lf_petreg_licence_licence						CP	
lf_petreg_licence	CP			CP			
lf_owner_hist	CP			CP			
lf_operator_hist	CP			CP			
strat_litho_wilbore_core	CP			CP			
strat_litho_wilbore	CP			CP			
seis_acquisition_progress	CP			DN			
seis_acquisition_coordinates_inc_burmaria	CP			DN			
seis_acquisition	CP			DN			
seismultiline	CP			DN		DN	
sqarea	CP			DN			
prifareasatiblock	CP			DN			
prifarea	CP			DN			
pipline	CP			CP			
licence_transfer_hist	CP			CP			
licence_task	CP			CP			
licence_phasic_hist	CP			CP			
licence_petreg_message	CP			CP			
licence_petreg_licence_sper	DN			CP		CP	
licence_petreg_licence_licence						CP	
licence_petreg_licence	CP			CP			
licence_oper_hist	CP			CP			
licence_licence_hist	CP			CP			
licence_area_poly_hist	CP			CP			
licarea	CP			CP			
fldarea	CP			CP			
field_reserves	DN			DN		CP	
field_production_yearly	CP			CP			
field_production_total_ncs_year	DP			CP			
field_production_total_ncs_month	CP			CP			
field_production_monthly	CP			CP			
field_owner_hist	CP			CP			
field_operator_hist	CP			CP			
field_licence_hist	CP			CP			
field_investment_yearly	CP			CP			
field_description	CP			CP			
field_activity_status_hist	CP			CP			
field	CP			CP			
fdcore	DN			CP		DP	
facility_moveable	CP			CP			
facility_filed	CP			CP		DN	
fdarea	CP			CP			
discovery_reserves	CP			CP			
discovery	CP			CP			
company_reserves	CP			CP		DP	
company	CP			DN			
bens_area_transfer_hist	CP			CP			
bens_area_operator	DN			CP		CP	
bens_area_licence_hist	CP			CP			
bens_area_area_poly_hist	CP			CP			
bens_area	CP			CP			
basarea	CP			CP			
aparea	CP			CP			
apareaagross	CP			CP			

(b) NPD

Fig. 3: Algorithm vs Manual Analysis

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].

References

1. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: Proc. of the 17th Int. Conf. on Software Engineering and Formal Methods (SEFM). Lecture Notes in Computer Science, vol. 11724, pp. 3–25. Springer (2019)
2. Arenas, M., Bertails, A., Prud’hommeaux, E., Sequeda, J.: A direct mapping of relational data to RDF. W3C Recommendation, World Wide Web Consortium (Sep 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: Proc. of the 17th Int. Conf. on Business Process Management (BPM). Lecture Notes in Computer Science, vol. 11675, pp. 139–156. Springer (2019)
4. Bano, D., Weske, M.: Discovering data models from event logs. In: Proc. of the 39th Int. Conf. on Conceptual Modeling (ER). Lecture Notes in Computer Science, vol. 12400, pp. 62–76. Springer (2020)
5. Bizer, C., Seaborne, A.: D2RQ: Treating non-RDF databases as virtual RDF graphs. In: Proc. of the 3rd Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 3298. Springer (2004)
6. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: 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. of Automated Reasoning* **39**(3) (2007)
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>, available at <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 (Sep 2012)

10. Fahland, D.: Artifact-centric process mining. In: *Encyclopedia of Big Data Technologies*. Springer (2019)
11. Gupta, S., Szekely, P., Knoblock, C.A., Goel, A., Taheriyani, M., Muslea, M.: Karma: A system for mapping structured sources into the Semantic Web. In: *Proc. of the Extended Semantic Web Conf. (ESWC)* (2012)
12. Hitzler, P., Gangemi, A., Janowicz, K.: *Ontology engineering with ontology design patterns: foundations and applications*, vol. 25. IOS Press (2016)
13. Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: BootOX: Practical mapping of RDBs to OWL 2. In: *Proc. of the 14th Int. Semantic Web Conf. (ISWC)*. *Lecture Notes in Computer Science*, vol. 9367, pp. 113–132. Springer (2015)
14. Kharlamov, E., Giese, M., Jiménez-Ruiz, E., Skjæveland, M.G., Soylu, A., Zheleznyakov, D., Bagoši, T., Console, M., Haase, P., Horrocks, I., Marciuska, S., Pinkel, C., Rodriguez-Muro, M., Ruzzi, M., Santarelli, V., Savo, D.F., Sengupta, K., Schmidt, M., Thorstensen, E., Trame, J., Waaler, A.: Optique 1.0: Semantic access to Big Data: The case of Norwegian Petroleum Directorate’s FactPages. In: *Proc. of the 12th Int. Semantic Web Conf., Posters & Demonstrations Track (ISWC)*. *CEUR Workshop Proceedings*, <http://ceur-ws.org/>, vol. 1035, pp. 65–68 (2013)
15. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD benchmark: Reality check for OBDA systems. In: *Proc. of the 18th Int. Conf. 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: *International Conference on Business Process Management*. pp. 319–334. Springer (2012)
17. de Medeiros, L.F., Priyatna, F., Corcho, O.: MIRROR: Automatic R2RML mapping generation from relational databases. In: *Proc. of the 15th Int. Conf. on Web Engineering (ICWE)*. pp. 326–343. Springer (2015)
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 (Dec 2012)
19. Pinkel, C., Binnig, C., Kharlamov, E., Haase, P.: Incmap: pay as you go matching of relational schemata to OWL ontologies. In: *Proc. WS on Ontology Matching*. *CEUR Workshop Proceedings*, <http://ceur-ws.org/>, vol. 1111 (2013)
20. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. *J. on Data Semantics* **10**, 133–173 (2008)
21. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. *VLDBJ* (2001)
22. Saha, B., Stanoi, I., Clarkson, K.L.: Schema covering: a step towards enabling reuse in information integration. In: *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. pp. 285–296. IEEE (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 - Volume 929*. pp. 97–108. WOP’12, 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: *Proc. of the 14th Int. Semantic Web Conf., Posters & Demonstrations Track (ISWC)*. *CEUR Workshop Proceedings*, <http://ceur-ws.org/>, vol. 1486 (2015)
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. *Proceedings of the VLDB Endowment* **3**(1-2), 805–814 (2010)