RODI: Benchmarking Relational-to-Ontology Mapping Generation Quality

Christoph Pinkel [a,*,**], Carsten Binnig [b], Ernesto Jiménez-Ruiz [c], Evgeny Kharlamov [c],
Wolfgang May [d], Andriy Nikolov [a], Martin G. Skjæveland [e], Alessandro Solimando [f,g],
Mohsen Taheriyan [h], Christian Heupel [a] and Ian Horrocks [c]

[a] *fluid Operations AG, Walldorf, Germany*
[b] *Brown University, Providence, RI, USA*
[c] *University of Oxford, United Kingdom*
[d] *Göttingen University, Germany*
[e] *University of Oslo, Norway*
[f] *Università di Genova, Genoa, Italy*
[g] *Inria Saclay & Université Paris-Sud, Orsay, France*
[h] *University of Southern California, Los Angeles, CA, USA*

Abstract Accessing and utilizing enterprise or Web data that is scattered across multiple data sources is an important task for both applications and users. Ontology-based data integration, where an ontology mediates between the raw data and its consumers, is a promising approach to facilitate such scenarios. This approach crucially relies on useful mappings to relate the ontology and the data, the latter being typically stored in relational databases. A number of systems to support the construction of such mappings have recently been developed. A generic and effective benchmark for reliable and comparable evaluation of the practical utility of such systems would make an important contribution to the development of ontology-based data integration systems and their application in practice. We have proposed such a benchmark, called *RODI*. In this paper, we present a new version of *RODI*, which significantly extends our previous benchmark, and we evaluate various systems with it. *RODI* includes test scenarios from the domains of scientific conferences, geographical data, and oil and gas exploration. Scenarios are constituted of databases, ontologies, and queries to test expected results. Systems that compute relational-to-ontology mappings can be evaluated using *RODI* by checking how well they can handle various features of relational schemas and ontologies, and how well the computed mappings work for query answering. Using *RODI*, we conducted a comprehensive evaluation of seven systems.

Keywords: Mappings, Relational databases, RDB2RDF, R2RML, Benchmarking, Bootstrapping

1. Introduction

1.1. Motivation

Accessing and utilizing enterprise or Web data that is scattered across multiple databases is an important task for both applications and users in many scenar-

ios [30,8]. Ontology-based data integration is a promising approach to this task, and recently it has been successfully applied in practice (e.g., [17]). The main idea behind this approach is to employ an ontology to mediate between data consumers and databases. Mappings can then be used to either export data to consumers or to translate (or *rewrite*) consumer queries into queries over the underlying databases on the fly. The latter approach is often referred to as ontology-based data access (OBDA).

Ontology-based data integration crucially depends on usable and useful ontologies and mappings. Ontology

---

development has attracted a lot of attention in the last decade, and ontologies have been developed for various domains including life sciences (e.g., [19]), medicine (e.g., [25]), the energy sector (e.g., [16]), and others. Many of these ontologies are generic enough to be useful as a target ontology in a significant number of ontology-based integration scenarios.

The development of reusable mappings has, however, received much less attention. Moreover, mappings are typically tailored to relate one specific pair of an ontology and one specific database. As a result, mappings typically cannot be as easily reused as ontologies across integration scenarios. Thus, each new integration scenario essentially requires the development of its own mappings. This is a complex and time consuming process. Hence, it calls for automatic or semi-automatic support, i.e., for systems that (semi-) automatically construct useful mappings. In order to address this challenge, a number of systems that generate relational-to-ontology mappings have recently been developed [10,38,14,53,6,45,3].

The quality of such generated relational-to-ontology mappings is usually evaluated using self-designed and therefore potentially biased benchmarks. This situation makes it particularly difficult to compare results across systems. Consequently, there is not enough evidence to select an adequate mapping generation system in ontology-based data integration projects. What matters at the end of the day in practice is whether the generated mappings are usable and useful for the task at hand. We therefore consider mapping quality as mapping utility w.r.t. a query workload posed against the mapped data.[1] This is of particular importance in large-scale industrial projects where support from (semi-)automatic systems is vital (e.g., [17]), In order to help ontology-based data integration finding its way into mainstream practice, there is a need for a generic and effective benchmark that can be used for the reliable evaluation of the *quality* of computed mappings w.r.t. their utility under actual query workloads. *RODI*, our mapping-quality benchmark for *R*elational-to-*O*ntology *D*ata *I*ntegration scenarios, addresses this challenge.

1.2. RODI *Benchmark Approach*

The *RODI* benchmark is composed of *(i)* a software *framework* to test systems that generate mappings between relational schemata [26] and OWL 2 ontologies [2], *(ii)* a *scoring function* to measure the quality of
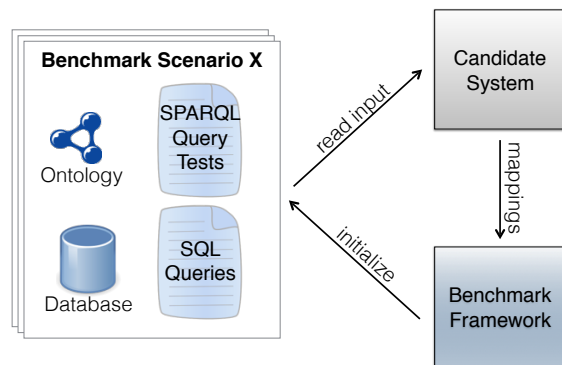

Figure 1. *RODI* benchmark overview

system-generated mappings, *(iii)* different datasets and queries for benchmarking, which we call *benchmark scenarios*, and *(iv)* a *mechanism* to extend the benchmark with additional scenarios. Using *RODI* one can evaluate the *quality* of relational-to-ontology mappings produced by systems for ontology-based data integration from two perspectives: how good the mappings can translate between various particularities of relational schemata and ontologies, and how good they are from the query answering perspective.

To make this possible, *RODI* is designed as an end-to-end benchmark. That is, we consider systems that can produce mappings directly between relational databases and ontologies. Also, we evaluate mappings according to their utility for an actual query workload.

Figure 1 schematically depicts the *RODI* architecture: the benchmark comes with a number of benchmark scenarios. Scenarios are initialized and set up for use by the framework. Candidate systems then read their input from the active scenario and produce mappings, which are evaluated again by our framework.

1.3. Contributions

In this section we summarize the main contributions of the *RODI* benchmark. We note that an earlier version of *RODI* has been introduced in [37]. In this paper we significantly extend our earlier results in several important directions: we extended the systematic analyses of challenges and related approaches; we now cover several new benchmark scenarios as well as additional test categories; we significantly extended the scope of the experimental study and now we cover seven different systems.

In the following we describe the main characteristics of *RODI* and highlight the enhancements with respect to its predecessor [37].

---

[1]*Utility* has also been referred to as *fitness for use* in similar contexts in parts of the literature, e.g., [55].

– *Systematic analyses of challenges and existing approaches in relational-to-ontology mapping generation*: these support and explain the types of challenges systematically tested by *RODI*. This paper contains an updated summary of previous work in addition to a newly added discussion of mapping approaches.

– *Evaluation scenarios*: *RODI* consists of data integration test scenarios from the domains of conferencing, geographical data, and oil and gas exploration. Scenarios are constituted of databases, ontologies, and queries to check expected results. Components of the scenarios are developed in such a way that they capture the key challenges of relational-to-ontology mapping generation. The version of *RODI* presented in this paper contains 18 scenarios in three different domains, as opposed to only 9 scenarios from two domains in the previous version of the benchmark [37]. The newly added scenarios focus on features that are important to test mapping quality under real-world challenges, such as high semantic heterogeneity or complex query workloads in different application domains.

– *The* RODI *framework*: the *RODI* software package, including all scenarios, has been implemented and made available for public download under an open source license.[2] In this paper we describe the new version of the framework in greater detail than before, so readers could thoroughly understand and independently judge *RODI*'s evaluation procedures. Readers should also be able to use the paper as a starting point for applying the benchmark by themselves. To this end, we also added for the first time a description of key implementation details.

– *System Evaluation*: we have used *RODI* to evaluate seven relational-to-ontology mapping systems: BootOX [14], COMA++ [11], IncMap [38], MIRROR [6], the -ontop- bootstrapper [12], D2RQ [3], and Karma [10]. The systems are chosen in a way that they cover the breadth of recent and traditional approaches in (semi-)automatic schema-to-ontology mapping generation. The insights gained from the evaluation allow us to point out specific strengths and weaknesses of individual systems and to propose how they can be improved. Compared to our preliminary experiments from [37],

the study presented in this paper extends not only to twice as many benchmark scenarios as before and adds three additional systems, COMA++, D2RQ and Karma, but it also gives much greater detail on several result aspects, such as a discussion of support for $1{:}n$ and $n{:}1$ mappings, and for the first time it also includes semi-automatic experiments. In total, we present numbers for seven different reporting categories and drill-downs, as compared to only two in our preliminary study. Also, the accompanying discussion adds significant detailed insights over the earlier paper.

In the new version of *RODI*, we have also modified all benchmark scenarios to produce more specific individual scores rather than aggregated values for relevant categories of tests. In addition, we have extended the benchmark framework to allow detailed debugging of the results for each individual test. On that basis we now could point to individual issues and bugs in several systems, some of which have already been addressed by the authors of the evaluated systems.

1.4. Outline

First, we present our analysis of the different types of mapping challenges for relational-to-ontology mapping generation in Section 2. Then, in Section 3 we discuss differences in mapping generation approaches that impact mapping generation, and thus also need to be considered for designing appropriate evaluation approaches. Section 4 presents our benchmark suite and the evaluation procedure. Afterwards, Section 5 discusses some implementation details that should help researchers and practitioners to understand how their systems could be evaluated in our benchmarking suite. Section 6 then presents our evaluation, including a detailed discussion of results. Finally, Section 7 summarizes related work and Section 8 concludes the paper and provides an outlook on future work.

2. Mapping Challenges

In the following we give a summary of our classification of different types of mapping challenges in relational-to-ontology data integration scenarios. For a more detailed discussion, please refer to [37]. As a high-level classification, we use the standard classification for data integration described by Batini et al. [1]: naming conflicts, structural heterogeneity, and semantic heterogeneity. For each of these classes, we list and briefly describe the key challenges that we have identified.

---

[2]Ready-to-use RODI distribution available at: `http://www.cs.ox.ac.uk/isg/tools/RODI/`. Source code available on GitHub: `https://github.com/chrpin/rodi`

2.1. Naming Conflicts

Typically, relational database schemata and ontologies use different conventions to name their artifacts even when they model the same domain based on the same specification and thus should use a similar terminology. The main challenge here is to be able to find similar names despite the different naming patterns. We are particularly interested in differences that are specific to inter-model matching and come on top of other naming differences, which commonly exist in other schema matching cases as well.

2.2. Structural Heterogeneity

The most important differences in relational-to-ontology integration scenarios, compared to other integration scenarios, are structural heterogeneities. Table 1 lists all specific testable relational-to-ontology structural challenges that we have identified.

In brief, there are type conflicts resulting from normalization, denormalization or different modeling of class hierarchies, key conflicts, and dependency conflicts.

2.2.1. Type Conflicts

Most real-world relational schemata and corresponding ontologies cannot be related by any simple canonical mapping. This is because big differences exist in the way how the same concepts are modeled (i.e., type conflicts). One reason why these differences are so big is that relational schemata often are optimized towards a given workload (e.g., they are normalized for update-intensive workloads or denormalized for read-intensive workloads). Ontologies, on the other side, model a domain on the conceptual level, albeit with different degrees of expressiveness and thus conceptual richness. Another reason is that some modeling elements have no single canonical translation (e.g., class hierarchies in ontologies can be mapped to relational schemata in different ways). In the following, we list the different type conflicts covered by *RODI*:

1. *Normalization artifacts:* Often properties that belong to a class in an ontology are spread over different tables in the relational schema as a consequence of normalization.
2. *Denormalization artifacts:* For read-intensive workloads, tables are often denormalized. Thus, properties of different classes in the ontology might map to attributes in the same table.
3. *Class hierarchies:* Ontologies typically make use of explicit class hierarchies. Relational models

implement class hierarchies implicitly, typically using one of three different common modeling patterns (c.f., [26, Chap. 3]). *(i)* The relational schema materializes several subclasses in the same table and uses additional attributes to indicate the subclass of each individual. With this variant, mapping systems have to resolve n:1 matches, i.e., they need to filter from one single table to extract information about different classes. *(ii)* Use one table per most specific class in the class hierarchy and materialize the inherited attributes in each table separately. Thus, the same property of the ontology must be mapped to several tables, leading to 1:n matches. *(iii)* Use one table for each class in the hierarchy, including the possibly abstract superclasses. Tables then use primary key-foreign key references to indicate the subclass relationship.

2.2.2. Key Conflicts

In ontologies and relational schemata, keys and references are represented differently.

1. *Keys:* Keys in databases are usually implemented using primary keys and unique constraints, while ontologies use IRIs for individuals. The challenge is that integration tools must be able to compose or skolemize appropriate IRIs.
2. *References:* While typically modeled as foreign keys in relational schemata, ontologies use object properties. Moreover, sometimes relational databases do not model foreign key constraints at all.

2.2.3. Dependency Conflicts

These conflicts arise when a group of concepts are related among each other with different dependencies (i.e., 1:1, 1:n, n:m) in the relational schema and the ontology. Relational schemata also often model n:m relationships using an additional connecting table.

2.3. Semantic Heterogeneity

Semantic heterogeneity plays a highly important role for data integration in general. Therefore, we extensively test scenarios that bring significant semantic heterogeneity.

Besides the usual semantic differences between any two conceptual models of the same domain, three additional factors apply in relational-to-ontology data integration: *(i)* the *impedance mismatch* caused by the object-relational gap, i.e., ontologies group information around entities (objects) while relational databases en-

Table 1

Detailed list of specific structural mapping challenges. RDB patterns may correspond to some of the "guiding" ontology axioms. Specific difficulties explain particular hurdles in constructing mappings.

#	Challenge type	RDB pattern	Examples of relevant guiding OWL axioms	Specific difficulty
(1)	Normalization	Weak entity table (depends on other table, e.g., in a part-of relationship)	owl:Class	JOIN to extract full IDs
(2)		$1{:}n$ attribute	owl:DatatypeProperty	JOIN to relate attribute with entity ID
(3)		$1{:}n$ relation	owl:ObjectProperty, owl:InverseFunctionalProperty	JOIN to relate entity IDs
(4)		$n{:}m$ relation	owl:ObjectProperty	3-way JOIN to relate entity IDs
(5)		Indirect $n{:}m$ relation (using additional intermediary tables)	owl:ObjectProperty	k-way JOIN to relate entity IDs
(6)	Denormalization	Correlated entities (in shared table)	owl:Class	Filter condition
(7)		Multi-value	owl:DatatypeProperty, owl:minCardinality [>1], owl:maxCardinality [>1], owl:cardinality [>1]	Handling of duplicate IDs
(8)	Class hierarchies	$1{:}n$ property match	rdfs:subClassOf, owl:unionOf, owl:disjointWith	UNION to assemble redundant properties
(9)		$n{:}1$ class match with type column	rdfs:subClassOf, owl:unionOf	Filter condition
(10)		$n{:}1$ class match without type column	rdfs:subClassOf, owl:unionOf	JOIN condition as implicit filter
(11)	Key conflicts	Plain composite key	owl:Class, owl:hasKey	Technical handling (e.g., Skolemnization)
(12)		Composite key, $n{:}1$ class matching to partial keys	owl:Class, owl:hasKey, rdfs:subClassOf	Choice of correct partial keys
(13)		Missing key (e.g., no UNIQUE constraint on secondary key)	owl:Class, owl:hasKey	Choice of correct non-key attribute as ID
(14)		Missing reference (no foreign key where relevant relation exists)	owl:ObjectProperty, owl:DatatypeProperty	Unconstrained attributes as references
(15)	Dependency conflicts	$1{:}n$ attribute	owl:FunctionalProperty, owl:minCardinality [>1], owl:maxCardinality [>1], owl:cardinality [>1]	Misleading guiding axioms; possible restriction violations
(16)		$1{:}n$ relation	owl:FunctionalProperty, owl:minCardinality [>1], owl:maxCardinality [>1], owl:cardinality [>1]	Misleading guiding axioms; possible restriction violations
(17)		$n{:}m$ relation	owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:minCardinality [>1], owl:maxCardinality [>1], owl:cardinality [>1]	Misleading guiding axioms; possible restriction violations

code them in a series of values that are structured in relations; *(ii)* the impedance mismatch between the closed-world assumption (CWA) in databases and the open-world assumption (OWA) in ontologies; and *(iii)* the difference in semantic expressiveness, i.e., databases may model some concepts or data explicitly where they are derived logically in ontologies.[3]

All of them are inherent to all relational-to-ontology mapping problems.

3. Analysis of Mapping Approaches

Different mapping generation systems make different assumptions and implement different approaches. Thus, a benchmark needs to consider each approach appropriately.

3.1. Differences in Availability and Relevance of Input

Different input may be available to an automatic mapping generator. In relational-to-ontology data integration, the main difference on available input concerns

the target ontology. The ontology could be specified entirely and in detail, or it could still be incomplete (or even missing) when mapping construction starts. Other differences comprise the availability of data or of a query workload.

The case where both the relational database schema and the ontology are completely available could be motivated by different situations. For example, a company may wish to integrate a relational data source into an existing, mature, Semantic Web application. In this case, the target ontology would already be well defined and would also be populated with some A-Box data. In addition, a SPARQL query workload could be known and could be available as additional input to a mapping generator.

On the other side, relational-to-ontology data integration might be motivated by a large-scale industry data integration scenario (e.g., [15,18]). In this scenario, the task at hand is to make complex and confusing schemata easier to understand for experts who write specialized queries. In this case, at the beginning no real ontology is given. At best there might be an initial, incomplete vocabulary.

Essentially, the different scenarios can all be distinguished by the following question: which information is available as input, besides the relational database? We always assume that the relational source database is

---

[3]Some databases may choose to implement calculations with an equivalent effect to some forms of inference, e.g., by using stored procedures for the purpose. We do not consider such programmatic schema extensions.

completely accessible (both schema and data), as this is a fundamental requirement without which relational-to-ontology data integration applications cannot reasonably be motivated. Besides the *availability* of input for mapping generation, there could be additional knowledge about which parts of the input are even *relevant*. For instance, it may be clear that only parts of the ontology that are being used by a certain query workload need to be mapped. If so, this information could also be leveraged by the mapping generation system (e.g., by analyzing the query workload).

It has to be noted that some other and different motivations to work with relational-to-ontology mappings exist as well: for instance, a database schema might be developed or generated to serve as a storage engine for an existing ontology (e.g., [27]). We do not consider these cases but rather think of them as the inverse of what happens for relational-to-ontology mapping generation. Similarly, we consider questions of mapping evolution as a related but separate problem.

For the *RODI* benchmark design we consider different forms of input by means of *RODI*'s composition, offering database, ontology, data and queries, parts of which can be used as additional input to mapping generators.

3.2. Differences in the Mapping Process

Other differences can arise from the process in which mapping generation is approached. These can be either fully-automatic approaches or semi-automatic approaches. Truly semi-automatic approaches are usually iterative [24], as they consist of a sequence of mapping generation steps that get interrupted to allow human feedback, corrections, or other input. Their process is driven by the human perspective rather than by an automatic component. Since we want to better adjust our benchmark to the semi-automatic approaches, we first discuss different ways that are known for the semi-automatic case.

Heyvaert et al. [21] have recently identified four different ways for manual relational-to-ontology mapping creation. Each of those directions inflicts a different interaction paradigm between the system and the user and thus solicits different forms of human input: users can edit mappings based on either the source or target definitions, they can drive the process by providing result examples or could theoretically even edit mappings irrespective of either the source or target in an abstract fashion. Some of us have also earlier identified two fundamentally different user perspectives on mapping generation [9] that drive the process in a different or-

der depending on whether the user feels more at home with the source database or with the target ontology. Moreover, while some approaches consider manual corrections only at the end of the mapping process, more thoroughly semi-automatic approaches allow or even require such input during the process.

In terms of their potential evaluation, iterative approaches of this kind must be considered according to two additional characteristics: First, whether iterative human input is mandatory or generally optional. Second, whether input is only used to improve the mapping as such, or if the systems also exploit it as feedback for their next automated iteration. Systems that solicit input only optionally and do not use it as feedback can be evaluated like non-iterative systems on a fully automatic baseline without limitations. Systems with only optional input that *do* learn from the feedback (if provided), can still be evaluated on the same baseline but may not demonstrate their full potential. Where input is mandatory, systems need to be either steered by an actual human user or at least require simulated human input produced by an oracle.

Next, the kind of human input that a system can process makes a difference for evaluation settings. Most semi-automatic systems either provide suggestions that users can confirm or delete, or they allow users to manually adjust the mapping. An alternative approach is *mapping by example*, where users provide expected results. In addition, however, some systems may require complex or indirect interactions, or simply resort to more unusual forms of input that cannot easily be foreseen.

Each mapping generation system is usually tied to one specific approach and does not allow for much freedom.

We therefore decided that an end-to-end evaluation that allows the use of different types of input is best. Since semi-automatic approaches are becoming more and more relevant, we decided to support them using an automated oracle that simulates user input where possible.

4. *RODI* Benchmark Suite

In the following, we present the details of our *RODI* benchmark: we first give an overview, then we discuss the data sets (relational schemata and ontologies) that can be used, as well as the queries. Finally, we present our scoring function to evaluate the benchmark results.
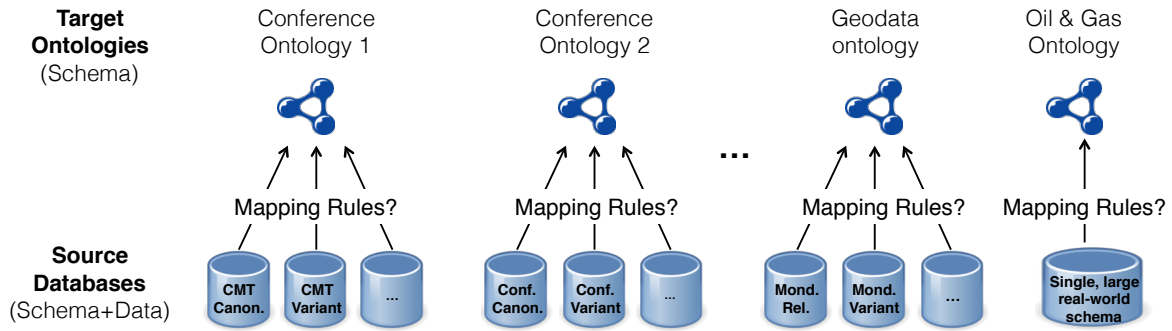
Figure 2. Overview of *RODI* benchmark scenarios

4.1. Overview

Figure 2 gives an overview of the scenarios used in our benchmark. The benchmark ships with data sets from three different application domains: conferences, geodata, and the oil & gas exploration domain. In its basic mode of operation, the benchmark provides one or more target ontologies for each of those domains (T-Box only) together with relational source databases for each ontology (schema and data). For some of the ontologies there are different variants of accompanying relational schemata that systematically vary the types of the targeted mapping challenges.

The benchmark asks the systems to create mapping rules from the different source databases to their corresponding target ontologies. We call each such combination of a database and an ontology a *benchmark scenario*. For evaluation, we provide for each scenario a series of *query pairs* to test a range of mapping challenges as illustrated in Figure 3. Every query pair consists of a SPARQL query ("test query") against the ontology, and a semantically equivalent SQL query ("reference query") against the provided SQL database. The test query runs against RDF data that results from applying the mapping rules of the matching system under consideration. The reference query is directly evaluated by RODI against the SQL database. The results are compared for each query pair and are aggregated in the light of different mapping challenges using our scoring function. For this, all query pairs are tagged with categories, relating them to different mapping challenges.

While challenges that result from different naming or semantic heterogeneity are mostly covered by complete scenarios, we target structural challenges on a more fine-granular level of individual query tests with a dedicated score. To this end, we add a corresponding category tag to query tests that address certain challenges. We target all structural challenges as previously listed in Table 1 in one or more scenarios.
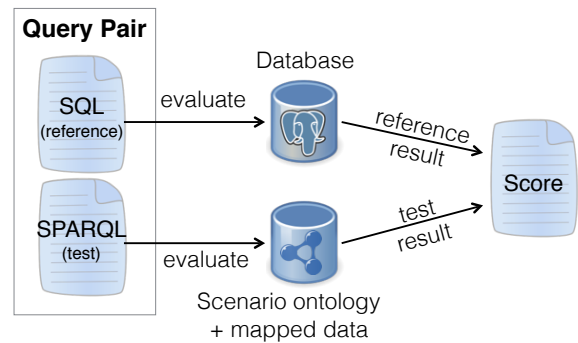

Figure 3. Query pair evaluation

Multi-source integration can be tested as a sequence of different scenarios that share the same target ontology. Although it has to be noted that some specific challenges in multi-source integration, especially conflicts introduced by different sources, may not become visible in sequential tests, this setup covers a wide range of multi-source mapping challenges. We include specialized scenarios for such testing with the conference domain.

In order to be open for other data sets and different domains, our benchmark can be easily extended to include scenarios with real-world ontologies and databases.

While all of our included default scenarios focus on schema-level matching, some cases in the real-world additionally demand data transformations to work fully as expected. These comprise translations between different representations of date and time (e.g., a dedicated date type versus Epoch time stamps), simple numeric unit transformations (e.g., MB vs. GB), unit transformations requiring more complex formulae (e.g., degrees Celsius vs. Fahrenheit), string-based data cleansing (e.g., removing trailing whitespace), string compositions (e.g., concatenating a first and last name), more complex string modifications (e.g., breaking up

a string based on a learned regular expression), table-based name translations (e.g., replacing names using a thesaurus), noise removal (e.g., ignoring erroneous tuples), etc. Our extension mechanism (see Section 4.5) is suited to add dedicated scenarios for testing such conversions, however, we excluded them from our default benchmark for a merely practical reason: To the best of our knowledge no current relational-to-ontology mapping generation system implements any such transformation functionality to date, so there is little practical use for benchmarking it.

In the following we present the data sources (i.e., ontologies and relational schemata) as well as the combinations used as integration scenarios for the benchmark in more details. *RODI* ships with scenarios based on data sources from three different application domains.

4.2. Conference Scenarios

We chose the conference domain as our primary testing domain since *(i)* it is well understood and comprehensible even for non-domain experts, *(ii)* it is complex enough for realistic testing, and *(iii)* it has been successfully used as the domain of choice in other benchmarks before (e.g., [52,23,4]). While we ship several different variants of scenarios for this domain, which vary in size and complexity, they are all built around a core fragment comprising 23 classes and 77 properties.

4.2.1. Ontologies

The conference ontologies in this benchmark are provided by the Ontology Alignment Evaluation Initiative (OAEI) [52,23,4] and were originally developed by the OntoFarm project [20]. We selected three particular ontologies (CMT, SIGKDD, CONFERENCE) based on a number of criteria: variation in size, the presence of cardinalities (especially, functionality of relationships), the coverage of the domain, variations in modeling style, and the expressive power of the ontology language used. Different modeling styles result from the fact that each ontology was modeled by different people based on various views on the domain, e.g., they modeled it according to an existing conference management tool, expert insider knowledge, or according to a conference website. To cover our mapping challenges (Section 2), we selectively modified the ontologies (e.g., we added labels to add interesting lexical matching challenges) as follows: *(i)* we selectively added annotations like labels and comments, as these can help to identify correspondences lexically; *(ii)* we added a few additional datatype properties where they were scarce, as they test other mapping challenges than just classes and object properties; and *(iii)* we fixed a total of seven inconsis-

tencies that we discovered in SIGKDD when adding A-Box facts (e.g., each place with a zip code automatically became a sponsor, which was modeled as a subclass of person).

4.2.2. Relational Schemata

We synthetically derived different relational schemata for each of the ontologies, focusing on different mapping challenges. We provide benchmark scenarios as combinations of those derived schemata with either their ontologies of origin, or, for more advanced testing, paired with any of the other ontologies. First, for each ontology we derived a relational schema using a canonical mapping as described in [27]: The algorithm works by deriving an entity-relationship (ER) model from an OWL ontology. It then translates this ER model into a relational schema according to textbook rules (e.g., [26]). For this paper, we extended this algorithm to cover the full range of expected relational design patterns. Additionally, we extended this algorithm to consider ontology instance data to derive more proper functionalities (rather than just looking at the T-Box as the previous algorithms do). Otherwise, the generated canonical relational schemata would have contained an unrealistically high number of $n{:}m$-relationship tables. The canonical schemata are guaranteed to be in fourth normal form (4NF), fulfilling normalization requirements of standard design practices. Thus, they already include various normalization artifacts as mapping challenges.

From the canonical schema corresponding to each of the ontologies, we systematically created different variants by introducing different aspects on how a real-world schema may differ from the canonical one and thus to test different mapping challenges:

1. *Adjusted Naming:* As described in Section 2.1, ontology designers typically consider other naming schemes than database architects do, even when implementing the same (verbal) specification. Those differences include longer vs. shorter names, "speaking" prefixes, human-readable property IRIs vs. technical abbreviations (e.g., "has-Role" vs. "RID"), camel case vs. underscore tokenization, preferred use of singular vs. plural, and others. For each canonical schema, we automatically generated a variant with identifier names changed in this way.

2. *Restructured Hierarchies:* The most critical structural challenge in terms of difficulty comes with different relational design patterns to model class hierarchies more or less implicitly. As we have discussed in Section 2.2, these changes introduce sig-

nificant structural dissimilarities between source
and target. We automatically derive variants of all
canonical schemata where different hierarchy de-
sign patterns are used. The choice of design pat-
tern in each case is algorithmically determined on
a "best fit" approach considering the number of
specific and shared (inherited) attributes for each
of the classes. For instance, a small number of
sibling classes would be split over several tables
if they mostly used different properties but they
would be rather joined together in a single table
if they would mostly make use of the same set of
properties.

3. *Combined Case:* In the real world, both of the pre-
vious cases (i.e., adjusted naming and hierarchies)
would usually apply at the same time. To find out
how tools cope with such a situation, we also built
scenarios where both are combined.

4. *Removing Foreign Keys:* Although it is considered
as bad style, databases without foreign keys are
not uncommon in real-world applications. This
can be a result of lazy design, or due to legacy
applications (e.g., one popular open source DBMS
introduced plugin-free support for foreign keys
less than five years ago). The mapping challenge
is that mapping tools must find the join paths to
connect tables of different entities. Additionally,
they sometimes even need to guess a join path for
reading attributes of the same entity if its data is
split over several tables as a consequence of nor-
malization. Therefore, we have created one dedi-
cated scenario to test this challenge with the CON-
FERENCE ontology and based it on the schema
variant with restructured hierarchies.

5. *Partial Denormalization:* In many cases, schemata
are partially denormalized to optimize for a cer-
tain read-mostly workload. Denormalization es-
sentially means that correlated (yet separated) in-
formation is jointly stored in the same table and
partially redundant. We provide one such scenario
for the CMT ontology. As denormalization re-
quires conscious design choices, this schema is
the only one that we had to hand-craft. It is based
on the variant with restructured hierarchies.

4.2.3. Integration Scenarios

For each of our three main ontologies, CMT, CON-
FERENCE, and SIGKDD, the benchmark includes five
scenarios (including basic and cross-matching scenar-
ios), each with a different variant of the database
schema (discussed before). Table 2 lists the different
(basic) scenario variants.

Table 2

Basic scenario variants (non-default scenarios are put in parentheses)

	CMT	CONFERENCE	SIGKDD
Canonical	(✓)	(✓)	(✓)
Adjusted Naming	✓	✓	✓
Restructured Hierarchies	✓	✓	✓
Combined Case	(✓)	(✓)	✓
Missing FKs	-	✓	-
Denormalized	✓	-	-

As discussed before, *Canonical* closely mimics the
structure of the original ontology, but the schemata are
normalized and thus the scenario contains the challenge
of normalization artifacts. *Adjusted Naming* adds the
naming conflicts as discussed before. *Restructured hi-
erarchies* tests the critical structural challenge of dif-
ferent relational patterns to model class hierarchies,
which, among others, subsumes the challenge to cor-
rectly build n:1 mappings between classes and tables.
In the *Combined Case*, naming conflicts and restruc-
tured hierarchies are employed and their effects are
tested in combination. This is a more advanced test case.
A special challenge arises from databases with no (or
few) foreign key constraints (*Missing FKs*). In such a
scenario, mapping tools must guess the join paths to
connect tables that correspond to different entity types.
The technical mapping challenge arising from *Denor-
malized* schemata consists in identifying the correct
partial key for each of those correlated entities, and to
identify which attributes and relations belong to which
of the types.

To keep the number of scenarios small for the de-
fault setup, we differentiate between default scenarios
and non-default scenarios. We excluded scenarios with
the most trivial schema versions. In addition, we did
limit the number of combinations for the most complex
schema versions by including only one of each type as a
default scenario. While the default scenarios are manda-
tory to cover all mapping challenges, the non-default
scenarios are optional (i.e., users could decide to run
them in order to gain additional insights). Non-default
scenarios are put in parentheses in Table 2. However,
they are not supposed to be executed in a default run of
the benchmark.

We also include cross-matching scenarios that re-
quire mappings of schemata to one of the other ontolo-
gies (e.g., mapping a CMT database schema variant to
the SIGKDD ontology). They represent more advanced
data integration scenarios and belong to the default
scenarios.

4.2.4. Data

We provide data to fill both the databases and ontologies. The conference ontologies are originally provided as T-Boxes only, i.e., no A-Box. We first generate data as A-Box facts for the different ontologies, and then transform them into the corresponding relational data using the same process as for translating the T-Box. For the technical process of evaluating generated mappings data is only needed in the relational databases. Hence, generating ontology A-Boxes would not even be necessary for this purpose alone. However, this procedure simplifies data generation since all databases can be automatically derived from the given ontologies as described before. Our conference data generator deterministically produces a scalable amount of synthetic facts around key concepts in the ontologies, such as conferences, papers, authors, reviewers, and others. In total, we generate data for 23 classes, 66 object properties (including inverse properties) and 11 datatype properties (some of which apply to several classes). However, not all of those concepts and properties are supported by every ontology. For each ontology, we only generate facts for the subset of classes and properties that have an equivalent in the relational schema in question.

4.2.5. Queries

For the conference scenarios, all scenarios draw on the same pool of 56 query pairs, accordingly translated for each ontology and schema. The benchmark queries on the conference scenarios are rather simple, using either only one concept and a property of it, or one relationship and two concepts with one property each. So each query tests, and in case of its failure indicates a certain mapping challenge. However, the same query may face different challenges in different scenarios, e.g., a simple 1:1 mapping between a class and table in a canonical scenario can turn into a complicated n:1 mapping problem in a scenario with restructured hierarchies. Also, not all query pairs are applicable on all conference ontologies (and thus, on their derived schemata).

Query pairs are grouped into three basic categories to test the correct mapping of *class instances*, instantiations of *datatype properties* and *object properties*, respectively. Additional categories relate queries to n:1 and n:m mapping problems or prolonged property join paths resulting from normalization artifacts. A specific category exists for the de-normalization challenge.

4.3. Geodata Domain – Mondial Scenarios

As a second application domain, *RODI* ships scenarios in the domain of geographical data.

The Mondial database [33] is a manually curated database containing information about countries, cities, organizations, and geographic features such as waters (with subclasses lakes, rivers, and seas), mountains, and islands. It has been designed as a medium-sized case study (6,000 real-world objects; 16,000 RDF IRIs, 50 properties, 90,000 RDF triples) for several scientific aspects and data models.[4]

Based on Mondial, we have developed a number of benchmark scenarios, which combine the Mondial OWL ontology with a series of different relational schemata. The OWL ontology is quite sophisticated, using many OWL constructs, and providing many potential challenges, e.g.:

- properties whose domain is neither a single class, nor some kind of top class (like usually for the name property) but a union of several classes (e.g., area, which is a property of countries, provinces, lakes, islands etc.).
- multiple properties between the same domain and range, distinguishable by the cardinality: Country.capital is functional with cities as range, while Country.hasCity is 1:n.
- properties that are functional on some subdomain, and n:m on another subdomain (e.g., locatedOnIsland which is functional for mountains and n:m for cities).
- properties that have a named union class as range: the range of City.locatedAt is waters, with rivers, lakes and seas as subclasses.

For the Mondial scenarios, we use a query workload that mainly approximates real-world explorative queries on the data, although limited to queries of low or medium complexity. The queries typically combine more than one concept, or several attributes, or several relationships with a common class. The degree of difficulty in the Mondial scenarios is therefore generally higher than in the conference domain scenarios.

There is only a single default scenario, which is based on the original relational Mondial database (42 tables, 160 columns, 60 foreign keys, and 43,000 tuples). It features a wide range of relational modeling patterns and it also differs from the canonical relational schema in some well-chosen aspects. With these changed aspects it mimicks a "real-life" (legacy) relational database schema:

- classical relational keys/foreign keys built upon literal-valued attributes. Most of them are unary

---

[4] http://www.dbis.informatik.uni-goettingen.de/Mondial/

(usually, the name attribute), but e.g. City(name, country, province) is a weak entity type whose key consists of multiple components.

– E.g., the above-mentioned City.locatedAt $n{:}m$ property cannot be stored in a single $n{:}m$ table since the foreign keys of the range would have to reference several tables for the different subclasses of waters. So another, even slightly denormalized modeling locatedAt(city,province,country,river, lake,sea) has been chosen (where a city located at two rivers and one sea requires two tuples, and the relation has no key since each of the watertype references may be null in some tuples).

In addition, we have designed a systematical series of further scenarios with synthetically modified variants of the canonical relational schema. To keep the number of tested scenarios at bay, we do not consider those additional synthetic variants as part of the default benchmark. Instead, we recommend these as optional tests to dig deeper into specific patterns. These scenarios are similar to the different variants produced in the conference domain, with the additional feature that the database schema and the queries are explicitly designed to test the following crucial elements:

– for all scenarios based on the canonical relational schema, all keys/foreign keys are the IRIs.
– different modeling variants of the class hierarchy wrt. Water/River/Lake/Sea and Mountain/Volcano;
– a variant where all classes mentioned in the ontology, including typically abstract ones like Political-Body, have an own table, each functional property is stored with the most abstract class, and foreign keys also reference the most abstract class;
– different modeling variants of functional properties whose domain is a union of classes like area, population and capital (ranging over cities);
– different modeling variants of City.locatedOnIsland ($1{:}n$) and Mountain.locatedOnIsland ($n{:}m$);
– case-sensitive vs. case-insensitive;
– a variant where all properties are $n{:}m$.

The challenges for the matching systems are not only to produce the appropriate mapping, but also to generate appropriate rules incorporating join paths, which is checked by the design of the benchmark queries. With these scenarios, the behavior of a system can be checked systematically, and even further database variants can easily be designed.

A typical (but already high-end) query is e.g.

```
SELECT ?CN ?MN ?IN
WHERE { ?C a :City; :name ?CN; :locatedOnIsland ?I .
    ?M a :Mountain; :name ?MN; :locatedOnIsland ?I .
    ?I :name ?IN . }
```

where information from the City table (functional: name) must be combined with the $n{:}m$ property City.locatedOnIsland; for Mountain, both properties are functional and might be found in the Mountain table, and a lookup for the Island's name must be made.

4.4. Oil & Gas Domain – NPD FactPages Scenarios

Finally, we include an example of an actual real-world database and ontology, in the oil and gas domain: The Norwegian Petroleum Directorate (NPD) FactPages [47]. Our test set contains a small relational database with a relatively complex structure (70 tables, $\approx$1,000 columns and $\approx$100 foreign keys), and an ontology covering the domain of the database. The database is constructed from a publicly available dataset containing reference data about past and ongoing activities in the Norwegian petroleum industry, such as oil and gas production and exploration. The corresponding ontology contains $\approx$300 classes and $\approx$350 properties.

With this pair of a database and ontology, we have constructed two scenarios that feature a different series of tests on the data: first, there are queries that are built from information needs collected from real users of the FactPages and cover large parts of the dataset. Those queries are highly complex compared to the ones in other scenarios and require a significant number of schema elements to be correctly mapped at the same time to bear any results. The principled benefit of such queries is that they represent actual real-world information needs much more accurately than any simplified query workloads do. A realistic workload can thus be seen as a measure of real-world utility as opposed to simplified queries, which artificially set the bar far too low and thus may convey a false impression of actual utility. Even if today's mapping generation systems may perform poorly on such queries, they will be the most relevant test to pass eventually. We have collected 17 such queries in scenario *npd_user_tests*. In addition, we have generated a large number of small, atomic query tests for baseline testing. These are similar to the ones used with the conference domain, i.e., they test for individual classes or properties to be correctly mapped. A total of 439 such queries have been compiled in the scenario *npd_atomic_tests* to cover all of the non-empty fields in our sample database.

A specific feature resulting from the structure of the FactPages database and ontology are a high number of $1{:}n$ matches, i.e., concepts or properties in the ontology that require a UNION over several relations to return complete results. $1{:}n$ matches as a structural feature

can therefore best be tested in the *npd_atomic_tests* scenario.

4.5. Extension Scenarios

Our benchmark suite is designed to be extensible, i.e., additional scenarios can be easily added. The primary aim of supporting such extensions is to allow domain-specific, real-world mapping challenges to be tested alongside our more default scenarios. Extension scenarios can be added by users of our benchmark without any programming efforts. The creation and addition of scenarios is described in the user documentation of the *RODI* benchmark suite [40].

4.6. Challenge Coverage and Query Category Tags

The scenarios included in *RODI* add up to jointly cover all mapping challenges identified and discussed in the previous sections.

On a per-scenario level, they represent examples from three different application domains. They also have different degrees of complexity. This affects schema size where NPD is the largest, Mondial is in-between and Conference scenarios have different sizes from small to medium. Also, different degrees of query workload complexity are included. Query workloads are modestly complex in conference scenarios and NPD atomic tests, more demanding in Mondial, and most complex with NPD user tests. Moreover, scenarios exhibit different degrees of semantic heterogeneity. There is rather little semantic heterogeneity between conference cases of any ontology and their directly corresponding database schema. Heterogeneity is higher for Mondial and NPD. The highest degree of semantic heterogeneity is however reached for conference scenarios, where we match an ontology to a database schema corresponding to a *different* ontology from the same domain (cross-matching).

On a more fine-grained level, several challenges are tested through a subset of queries. We tag query tests by categories and report separate scores not only for each scenario, but also for each category in each scenario.

Table 3 shows a list of all tags that we use in our default scenarios, a brief description of their purpose, and scenarios that include them. Not all of the category tags correspond to a particular challenge. Some of them also serve to allow drill-downs into basic aspects of the schema, e.g., to separately report on class matches and property matches. However, using tags we can also report on challenges that correspond to some of the category tags in the query tests.

4.7. Evaluation Criteria – Scoring Function

It is our aim to measure the practical usefulness of mappings. We are therefore interested in the utility of query results, rather than comparing mappings directly to a reference mapping set or than measuring precision and recall on all elements of the schemata. This is important because a number of different mappings might effectively produce the same data w.r.t. a specific input database. Also, the mere number of facts is no indicator of their semantic importance for answering queries (e.g., the total number of conference venues is much smaller than the number of all paper submission dates, yet the venues are just as important in a query retrieving information about any of these papers). In addition, in many cases only a subset of the information is relevant in practice and we define our queries on a meaningful subset of information needs.

We therefore observe a score that reflects the utility of the mappings with relation to our query tests as our main measure. Intuitively, this score reports the percentage of successful queries for each scenario.

However, in a number of cases, queries may return correct but incomplete results, or could return a mix of correct and incorrect results. In these cases, we consider per-query accuracy by means of a local per-query F-measure. Technically, our reported overall score for each scenario is the average of F-measures for each query test, rather than a simple percentile of successful queries. To calculate these per-query F-measures, we also need to consider query results that contain IRIs.

Different mapping generators will typically generate different IRIs for the same entities represented in the relational database, e.g., by choosing different prefixes. F-measures for query results containing IRIs are therefore calculated w.r.t. the degree to which they satisfy *structural equivalence* with a reference result. For practical reasons, we use query results on the original, underlying SQL databases as technical reference during evaluation. Structural equivalence effectively means that if same-as links were established appropriately, then both results would be semantically identical.

Formally, structural equivalence and fitting measures for precision and recall are defined as follows:

Definition 1 (Structural Tuple Set Equivalence) *Let* $V = IRI \cup Lit \cup Blank$ *be the set of all IRIs, literals and blank nodes,* $T = V \times ... \times V$ *the set of all n-tuples of* V. *Then two tuple sets* $t_1, t_2 \in \mathcal{P}(T)$ *are* structurally equivalent *if there is an isomorphism* $\phi : (IRI \cap t_1) \to (IRI \cap t_2)$.

For instance,

Table 3

Category tags used in different default scenarios, with brief description. Relevant challenges where applicable. Note, that some challenges require a check on a combination of tags or will be checked at a scenario-level, not on a query-level.

Tag ID	Meaning	Default Scenarios	Relevant Challenges
class	Matching class	All	-
attrib	Matching datatype property	All	-
link	Matching object property	All	-
1-1	1:1 match	All conference	-
n-1	n:1 match	All restructured conference	6, 9, 10, 12
1-n	1:n match	NPD, some conference	8, 15, 16
union-n	Requires n UNIONs to build match (form of 1:n match with explicit n)	NPD	8, 15, 16
superclass	Test on entities that should comprise sub class entities (form of 1:n)	All conference adjusted naming	8
in-table	Datatype match that finds the data value in the same table that also defines the related entity	All conference	7
other-table	Datatype match that finds the data value in a table other than the one that defines the related entity (requires joins)	All conference	2, 15
path-0	Object property match that finds both related entities in the same table (1:1 or denormalized)	Some conference	7
path-1, path-2	Object property match that finds related entities in two tables that can be directly joined (single JOIN) or joined through one intermediate table (two JOINs). Note: NPD uses join-1, join-2 to denote the same aspect	All conference, NPD	3, 4, 16
path-n	Object property match that requires n JOINs ($n > 2$) to connect the tables that define entities on both sides. Note: NPD uses join-n to denote the same aspect	Some conference, NPD	5
path-X	Additional tag for all queries that are tagged path-n, with any $n > 1$ (denotes multi-hop JOIN of any length)	All conference	4, 5, 17
denorm	Type filtering required due to denormalization	CMT denormalized	6, 7, 12
no-fk	JOINs without leading foreign keys	Conference missing FKs	14

$$\{(\text{urn:p-1}, \text{'John Doe'})\}$$

and

$$\{(\text{http://my\#john}, \text{'John Doe'})\}$$

are structurally equivalent. On this basis, we can easily define the equivalence of query results w.r.t. a mapping target ontology:

Definition 2 (Tuple Set Equivalence w.r.t. Ontology)
Let O be a target ontology of a mapping, $I \subset IRI$ the set of IRIs used in O and $t_1, t_2 \in \mathcal{P}(T)$ result sets of queries q_1 and q_2 evaluated on a superset of O (i.e., over O plus A-Box facts added by a mapping).

Then, $t_1 \sim_O t_2$ (are structurally equivalent w.r.t. O) iff t_1 and t_2 are structurally equivalent and $\forall i \in I: \phi(i) = i$.

With the same example as just before, the two tuples are structurally equivalent, *iff* http://my#john is *not* already defined in the target ontology. Finally, we can define precision and recall:

Definition 3 (Precision and Recall) *Let $t_r \in \mathcal{P}(T)$ be a reference tuple set, $t_t \in \mathcal{P}(T)$ a test tuple set and $t_{rsub}, t_{tsub} \in \mathcal{P}(T)$ be maximal subsets of t_r and t_t, s.t., $t_{rsub} \sim_O t_{tsub}$.*

Then the precision of the test set t_t is $P = \frac{|t_{tsub}|}{|t_t|}$ and recall is $R = \frac{|t_{rsub}|}{|t_r|}$.

Table 4 shows an example with a query test that asks for the names of all authors. Result set A is structurally equivalent to the reference result set, i.e., it has found all authors and did not return anything else, so both precision and recall are 1.0. Result set B is equivalent with only a subset of the reference result (e.g., it did not include those authors who are also reviewers). Here, precision is still 1.0, but recall is only 0.5. In case of result set C, all expected authors are included, but also another person, James. Here, precision is 0.66 but recall is 1.0.

Table 4

Example results from a query pair asking for author names, e.g.,
SQL: SELECT name FROM persons WHERE person_type=2
SPARQL:
SELECT ?name WHERE {?p a :Author; foaf:name ?name}

Reference Result	Result A	Result B	Result C
Jane	Jane	John	Jane
John	John		John
			James

To aggregate results of individual query pairs, a scoring function calculates the averages of per query num-

bers for each scenario and for each challenge category. For instance, we calculate averages of all queries testing 1:n mappings. Thus, for each scenario there is a number of scores that rate performance on different technical challenges. Also, the benchmark can log detailed per-query output for debugging purposes.

4.8. Mapping System Requirements

With *RODI*, we can test mapping generators that work in either one or two stages: that is, they either directly map data from the relational source database to the target ontology in one stage (e.g., [38,11]). Or, they bootstrap their own ontology, which they use as an intermediate mapping target. In this case, to get to the full end-to-end mappings that we can test, the intermediate ontology and the actual target ontology should be integrated via ontology alignment in a second stage. Two-stage systems may either include a dedicated ontology alignment stage (e.g., [14]) or they deliver the first (intermediary) stage only ([12,6,3]). In the latter case, *RODI* can step in to fill the missing second stage with a standard ontology alignment setup [46].

Our tests check the accuracy of SPARQL query results. Queries ask for individuals of a certain type (or their aggregates), properties correlating them, associated values and combinations thereof, sometimes also using additional SPARQL language features such as filters to narrow down the result set. This means that mapped data will be deemed correct if it contains correct RDF triples for all tested cases. For entities, this means that systems need to construct one correctly typed IRI for each entity of a certain type. For object properties, they need to construct triples to correctly relate those typed IRIs, and for datatype properties, they need to assign the correct literal values to each of the entity IRIs using the appropriate predicates. Systems do therefore not strictly need to understand or to produce any OWL axioms in the target ontology. However, our target ontologies are in OWL 2, using different degrees of expressiveness. Axioms in the target ontology can be important as *guidance* to identify suitable correspondences for one-stage systems. Similarly, if two-stage systems *construct* expressive axioms in their intermediate ontology, this may guide the second stage of ontology alignment. For instance, if a predicate is known to be an object property in the target ontology, results will suffer if a mapping generation tool assigns literal values using this property. Also, if a property is known to be functional it might be a better match for an n:1 relation than a non-functional property would be.

5. Framework Implementation

In this section, we discuss some implementation details in order to guide researchers and practitioners to include their system in our benchmarking suite.

### 5.1. Architecture of the Benchmarking Suite

Figure 4 depicts the overall architecture of our benchmarking suite. The framework requires upfront initialization *per scenario*. Artifacts generated or provided during initialization are depicted blue in the figure. After initialization, a mapping tool can access the database (directly or via the framework's API) and the target ontology (via the Sesame API[5] or using SPARQL, or serialized as an RDF file). Finally, it submits generated R2RML[6] mappings in a special folder on the file system, so evaluation can be triggered. As an alternative, mapping tools could also execute mappings themselves and submit final mapped data instead of R2RML. This would be the preferred procedure for tools that do not support R2RML but other mapping languages. More generally, mapping tools that cannot comply with the assisted benchmark workflow can always trigger individual aspects of initialization of evaluation separately. For instance, they could use the framework to setup a test environment, then perform mapping generation, reasoning and mapping execution in their own workflow and finally trigger evaluation.

5.2. Details on the Evaluation Phase

Unless a mapping system under evaluation decides to skip individual steps, i.e., to implement them independently, in the evaluation phase, the benchmark suite will: *(i)* read submitted R2RML mappings and execute them on the database, *(ii)* materialize the resulting A-Box facts in a Sesame repository together with the target ontology (T-Box), *(iii)* optionally apply reasoning through an external OWL API [28] compatible reasoner to infer additional facts that may be requested for evaluation, *(iv)* evaluate all query pairs of the scenario on the repository and on the relational database, and *(v)* produce a detailed evaluation report. Additionally, as mentioned in Section 4.8, *RODI* also provides support to (two-stage) systems that require assistance to align their generated ontology with the target ontology. Information about how individual steps are invoked can be found in *RODI*'s user documentation [40].

---

[5] http://rdf4j.org
[6] http://www.w3.org/TR/r2rml/
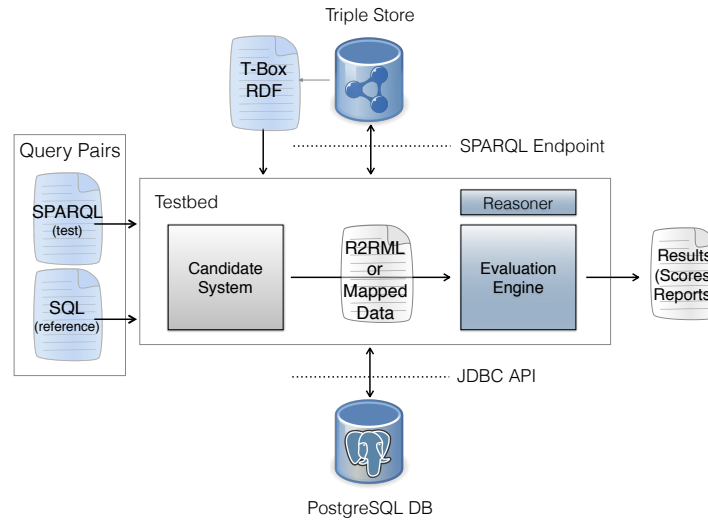
Figure 4. *RODI* framework architecture

We evaluate query results as described in Section 4.7 by attempting to construct an isomorphism ϕ between the query result set and the reference results. Technically, we use the results of the SQL queries from query pairs to calculate the reference result set. For each SQL query in a query pair, we flag attributes that together serve as keys, so keys can be matched with IRIs rather than with literal values. Obviously, literal values need to be exact matches. IRIs always need to match the same unique value from the database in each matching tuple, but this unique value can be different from the IRI as a string.

For constructing ϕ, we first index all individual IRIs (i.e., IRIs that identify instances of some class) in the query result. Next, we build a corresponding index for keys in the reference set. For both sets we determine binding dependencies across tuples (i.e., re-occurrences of the same IRI or key in different tuples). As a next step, we narrow down match candidates to tuples where all corresponding literal values are exact matches. After this step, we have a set of tuple pairs from the two result set that are candidates for being structurally equivalent, because all their literals are identical. Finally, we check for these candidates to be actually structurally equivalent, i.e., we also check for viable correspondences between keys (in the reference tuples) and IRIs (in the matching result tuples). As discussed, the criterion for a viable match between a key and an IRI is that for each occurence of this particular key and of this particular IRI in any of the tuples, both need to be matched with the same partner. In principle, the same IRIs (and keys) can appear in any number of tuples and in different

positions of each tuple. A simple example for such a query with repeated occurences of the same IRI in several positions of the result tuple could be a request for papers with their authors and reviewers (where each author and reviewer may appear in many different tuples and the same person's IRI may appear as an author in one tuple and as a reviewer in another). All occurences of all IRIs in any tuple need to match the *same* keys in all cases for a structurally equivalent overall match. Hence, we can have transitive n-hop dependencies between n tuples or even cyclic dependencies. The step of finding viable matches accross all tuples therefore corresponds to identifying a maximal common subgraph (MCS) between the dependency graphs of tuples on both sides, i.e., it corresponds to the MCS-isomorphism problem.[7] For efficiency reasons, we approximate the MCS if dependency graphs contain transitive dependencies, breaking them down to fully connected subgraphs. However, it is usually possible to formulate query results to not contain any such transitive dependencies by avoiding inter-dependent IRIs in SPARQL SELECT results in favor of a set of significant literals describing them. All queries shipped with this benchmark are free of transitive dependencies, hence the algorithm is accurate for all delivered scenarios.

Finally, we count tuples that could not be matched in the result and reference set, respectively. Precision is then calculated as $\frac{|res| - |unmatched(res)|}{|res|}$ and recall

---

[7]The MCS isomorphism problem is a well-studied optimization problem and is known to be NP-hard.

as $\frac{|ref| - |unmatched(ref)|}{|ref|}$ in accordance with our precision and recall measures for structural equivalence (Def. 3). Aggregated numbers are calculated per query pair category as the averages of precision and recall of all queries in each category.

6. Benchmark Results

6.1. Evaluated Systems

We have performed an in-depth analysis using *RODI* on a wide range of systems. Those include current contenders in the automatic segment (*BootOX* [14,18,17] and *IncMap* [38,39]), more general-purpose mapping generators (*-ontop-* [12], *MIRROR* [6] and *D2RQ* [3]), as well as a much earlier, yet state-of-the-art system in inter-model matching (*COMA++* [11]). In a specialized semi-automatic series of experiments, we have also evaluated Karma [10,50], which does not support a fully automatic mapping generation mode and works with a sophisticated model of human intervention. As a consequence, it requires a specific experimental setup. Note that *BootOX*, *-ontop-*, *MIRROR* and *D2RQ* are two-stage systems (see Section 4.8), that is, they do not generate mappings targetting the ontology provided in each *RODI* scenario and they generate their own (putative or bootstrapped) ontology instead. *BootOX* includes a built-in ontology alignment system which allows to integrate the bootstrapped ontology with the target (scenario) ontology. In order to be able to evaluate *-ontop-*, *MIRROR* and *D2RQ* with *RODI*, we also aligned their generated ontologies with the target ontology in a similar setup to the one used in *BootOX*.

1. *BootOX* (**B.OX**) is based on the approach called *direct mapping* by the W3C:[8] every table in the database (except for those representing $n{:}m$ relationships) is mapped to one class in the ontology; every data attribute is mapped to one data property; and every foreign key to one object property. Explicit and implicit database constraints from the schema are also used to enrich the bootstrapped ontology with axioms about the classes and properties from these direct mappings. Afterwards, BootOX performs an alignment with the target ontology using the LogMap system [31,13,48].

2. *IncMap* (**IncM.**) maps an available ontology directly to the relational schema. IncMap represents both the ontology and schema uniformly, using a structure-preserving meta-graph for both. It runs in two phases, using lexical and structural matching. We evaluate a current (and yet unpublished) work-in-progress version of IncMap, as opposed to the initial version previously evaluated in [37]. The main difference between the two versions of IncMap are improvements in lexical matching and mapping selection, as well as engineering improvements that increase the mapping quality.

3. *MIRROR* (**MIRR.**) is a tool for generating an ontology and R2RML direct mappings automatically from an RDB schema. MIRROR has been implemented as a module of the RDB2RDF engine morph-RDB [42]. Their output is oblivious of the required target ontology, though, so we perform post-processing with ontology alignment.

4. The *-ontop- Protege Plugin* (**ontop**) is a mapping generator developed for -ontop- [12]. -ontop- is a full-fledged query rewriting system [43] with limited ontology and mapping bootstrapping capabilities. As mentioned above, we need to post-process results with ontology alignment.

5. *COMA++* (**COMA**) has been a contender in the field of schema matching for several years already; it is still widely considered state of the art. In contrast to other systems from the same era, COMA++ is built explicitly also for inter-model matching. To evaluate the system, we had to perform a translation of its output into modern R2RML.

6. *D2RQ platform* (**D2RQ**) is a fully-fledged system to access relational databases as virtual RDF graphs. Since D2RQ relies on its own native language to define mappings and *RODI* only supports standard R2RML mappings, we executed the mappings using D2RQ and provided *RODI* with the materialized data. Just as with MIRROR and -ontop-, a post-process with ontology alignment is required.

7. *Karma* is one of the most prominent modern relational-to-ontology mapping generation systems. It is strictly semi-automatic, i.e., there is no fully automatic baseline that we could use for non-interactive evaluation. In addition, Karma's mode of iterations is designed to take advantage mostly from integrating a series of data sources to the same target ontology. Karma is therefore not well suited for single-scenario evaluations. We therefore only evaluate Karma in a dedicated line of experiments that suit its specifications.

---

[8]http://www.w3.org/TR/rdb-direct-mapping/

6.2. Experimental Setup

We conduct benchmark default experiments as described in Section 4 for all systems except Karma. This includes a selection of nine prototypical scenarios from the conference domain, one from the geodata domain and two from the oil & gas domain, as well as six different cross-matching (conference) scenarios. For all of these main experiments, we observe and report overall *RODI* scores as well as specific selected scores in individual categories.

In addition, we perform two different semi-automatic experiments on selected scenarios for Karma and IncMap, respectively. For Karma, we had to conduct experiments with an actual human in the loop to perform steps that Karma could not automate. With IncMap, we could simulate human feedback by responding to suggestions by taking a response from the benchmark that indicates changes in mapping quality. In both semi-automatic cases, we chiefly observe the number of interactions.

6.3. Default Scenarios: Overall Results

Table 5 shows scores for all systems on all basic default scenarios. At first impression we can observe that all tested systems manage to solve some parts of the scenarios, but with declining success as scenario complexity increases.

For instance, relational schemata in the conference "adjusted naming" scenarios follow modeling patterns from their corresponding ontologies most closely, and all systems without exception perform best in this part of the experiments. Quality drops for all other types of scenarios, i.e., whenever we introduce additional challenges that are specific to the relational-to-ontology modeling gap. The drop in accuracy between *Adjusted Names* and *Restructured hierarchies* settings is mostly due to the n:1 mapping challenge introduced by one of the relational patterns to represent class hierarchies which groups data for several subclasses in a single table. In the most advanced conference cases, systems lose further due to the additional challenges, although to different degrees. Good news is that some of the actively developed current systems could improve their scores compared to previous numbers recorded in January 2015 [37]. A somewhat disappointing general observation, however, is that measured quality is overall still modest compared to results that are known from ontology alignment tasks involving some of the same ontologies (c.f. [52,23,4]). This is disappointing, especially while state-of-the-art ontology alignment software is employed in some of the systems. It could

Table 5

Overall scores in default scenarios (scores based on average of per-test F-measure). Best numbers per scenario in bold print.

Scenario	B.OX	IncM.	ontop	MIRR.	COMA	D2RQ
Conference domain, adjusted naming						
CMT	**0.76**	0.45	0.28	0.28	0.48	0.31
Conference	0.51	**0.53**	0.26	0.27	0.36	0.26
SIGKDD	**0.86**	0.76	0.38	0.30	0.66	0.38
Conference domain, restructured						
CMT	0.41	**0.44**	0.14	0.17	0.38	0.14
Conference	**0.41**	**0.41**	0.13	0.23	0.31	0.21
SIGKDD	**0.52**	0.38	0.21	0.11	0.41	0.28
Conference domain, combined case						
SIGKDD	**0.48**	0.38	0.21	0.11	0.28	0.21
Conference domain, missing FKs						
Conference	0.33	**0.41**	-	0.17	0.21	0.18
Conference domain, denormalized						
CMT	**0.44**	0.40	0.20	0.22	-	0.20
Geodata						
Classic Rel.	**0.13**	0.08	-	-	-	0.06
Oil & gas domain						
User Queries	0.00	0.00	0.00	0.00	-	0.00
Atomic	**0.14**	0.12	0.10	0.00	0.02	0.08

indicate that the specific challenges in relational-to-ontology mapping generation can not convincingly be solved with the same technology that is successful in ontology alignment, but may call for more specialized approaches.

While all of the conference scenarios test a wide range of specific relational-to-ontology mapping challenges, they do so in a highly controlled fashion, on schemata with at best medium size and complexity, and using a largely simplified query workload. For instance, queries in the conference domain scenarios would separately check for mappings of authors, person names, and papers. They would not, however, pose any queries like asking for the names of authors who did participate in at least five different papers. The huge difference here is that, if two out of three of these elements were mapped correctly, the simple, atomic queries would report an average score of 0.66, while the single, more application-like query that correlates the same elements would not retrieve anything, thus resulting in a score of 0.00. None of the systems managed to solve even a single test on this challenge. This kind of real-world queries that mimic an actual application query workload, are precisely what we focus on the remaining default scenarios, which are set in the geodata and oil & gas exploration domains. Consequently, scores are

Table 6

Overall scores in cross-matching scenarios (scores based on average of per-test F-measure). Best numbers per scenario in bold print.

Source	B.OX	IncM.	ontop	MIRR.	COMA	D2RQ
Target ontology: CMT						
Conference	0.20	**0.35**	0.10	0.00	0.00	0.10
SIGKDD	**0.33**	**0.33**	0.19	0.00	0.14	0.19
Target ontology: Conference						
CMT	0.20	**0.34**	0.05	0.00	0.05	0.05
SIGKDD	0.13	**0.30**	0.09	0.00	0.04	0.09
Target ontology: SIGKDD						
CMT	0.51	**0.57**	0.19	0.00	0.24	0.26
Conference	0.24	**0.44**	0.13	0.00	0.09	0.14

lower again in those scenarios. In the geodata scenario, only a minority of query tests could be solved. Detailed debugging showed that the reason for this lies in the more complex nature of queries, most of which go beyond returning simple results of just a single mapped element. In the oil & gas case, the situation becomes even more problematic. Here, the schema and ontology are again a bit more complex than in the geodata scenario, and so is the explorative query workload ("user queries"). None of the systems was able to answer any of these queries correctly after a round of automatic mapping. To retrieve meaningful results, we added a second scenario on the same data, but with a synthetic query workload of atomic queries ("atomic"). On this scenario, results could be computed but overall scores remain low due to the size and complexity of the schema and ontology with a large search space as well as many 1:n matches.

Table 6 showcases results from the most advanced scenarios in the conference domain. All of them are built on the "combined case" scenarios, i.e., they contain a mix of all of the standard relational-to-ontology mapping challenges except for denormalization and lazy modeling of constraints. In addition, they increase the level of semantic heterogeneity by asking for mappings between a schema derived from one ontology to a rather different, independent ontology in the same domain. Scores are generally lower than in the basic conference cases discussed above. Reasonable scores can still be achieved by some systems. Also, the overall trend of performance between the systems mostly remains the same as in the basic scenarios, with a few exceptions. Somewhat surprisingly, COMA loses out more than other contenders. Even more surprising, the performance of BootOX is noticeably low compared to the baseline results from basic scenarios in Table 5. This is unexpected as BootOX essentially applies ontol-

ogy alignment technology that has proven itself in tasks with high semantic heterogeneity [31]. It could, again, be an indicator that out-of-the-box ontology alignment techniques could not take the same leverage that they do when aligning original ontologies.

The big picture shows that the two most specialized and actively developed systems, BootOX and IncMap, are leading the field. Among those two, BootOX is at a clear advantage in scenarios where the inter-model gap between relational schema and ontology is small (e.g., "adjusted naming"). IncMap is gaining ground when more specific inter-model mapping challenges are added. MIRROR, -ontop- and D2RQ generally show weaker results. It has to be noted, though, that these systems have been originally designed and optimized for a somewhat different task than the full end-to-end mapping generation setup tested with *RODI*. MIRROR and -ontop- also fail to execute some of the scenarios due to technical difficulties. For MIRROR in particular, we have encountered a number of so far unresolved difficulties that may also have a detrimental effect on MIRROR scores. COMA keeps up well, given that it is no longer actively developed and improved. Also, while COMA has been constructed to support inter-model matching in general, it has not been explicitly optimized for the specific case of relational-to-ontology matching.

As part of our detailed analysis of the results we could also identify, and partially even fix, a number of technical shortcomings in tested systems. For instance, we encountered issues with MIRROR in certain multi-schema matching cases on PostgreSQL and implemented a solution in exchange with the authors of the system. In another example, IncMap's poor performance in the geodata scenario could in part be explained by its failure to understand the specification of property domains and ranges as a union of several concrete classes. This pattern lead IncMap to skipping such properties altogether. While not yet fixed, the observation points to concrete technical improvements in IncMap. In BootOX, incomplete and unfavorable reasoning settings were detected and fixed.

6.4. Default Scenarios: Drill-down

All systems struggle with correctly identifying properties as Table 7 shows. A further drill-down shows that this is in part due to the challenge of normalization artifacts, with systems struggling to detect any properties that map to multi-hop join paths in the tables. Mapping data to class types appears to be generally easier for all contenders. BootOX is performing best in most cases with all kinds of properties, with IncMap coming

Table 7

Score break-down for queries on different match types with adjusted
naming conference scenarios. 'C' stands for queries on classes, 'D'
for data properties, 'O' for object properties.

Scenario	B.OX			IncM.			ontop			MIRR.			COMA			D2RQ		
	C	**D**	**O**	**C**	**D**	**O**	**C**	**D**	**O**	**C**	**D**	**O**	**C**	**D**	**O**	**C**	**D**	**O**
CMT	**0.92**	**0.73**	**0.50**	0.58	0.46	0.17	0.67	0.00	0.00	0.56	0.00	0.00	0.75	0.46	0.00	0.67	0.00	0.17
Conference	**0.81**	0.27	**0.38**	**0.81**	**0.53**	0.13	0.63	0.00	0.00	0.53	0.00	0.00	0.50	0.40	0.00	0.63	0.00	0.00
SIGKDD	**1.00**	**0.90**	0.25	0.80	0.70	**0.25**	0.73	0.00	0.00	0.46	0.00	0.00	0.80	0.70	0.00	0.73	0.00	0.00

in second. This represents a change over the previous versions of both systems benchmarked earlier, where IncMap was clearly leading on properties [37].

Tables 8 and 9 show the behavior of systems for finding n:1 and 1:n matches between ontology classes and table content, respectively. We highlight the n:1 case in restructured conference scenarios and 1:n matches in the oil & gas scenario as they include the highest number of tests in their respective categories. In both cases results are staggering with all systems failing the large majority of tests. For 1:n matches the situation is slightly better than it is with n:1 matches. This is not particularly surprising in general, as 1:n matches can be composed in mapping rules by adding up several correct 1:1 matches. A correct mapping of n:1 matches between classes and tables, on the other side, usually requires the much more challenging task of *filtering* from the table that holds entities of different types.

6.5. Semi-Automatic, Iterative Scenarios

We have also conducted semi-automatic, iterative experiments on *RODI* scenarios with two different systems, Karma and IncMap. While IncMap was also evaluated in the main line of experiments before on its fully automatic mode, Karma does not support such a baseline mode and always requires human intervention in different forms. This is mainly due to Karma's need for so called *Python transformations*, essentially tiny Python scripts, to skolemize entity IRIs. In contrast to class and property matches, Karma does not learn those transformations. Also, both systems work according to completely different semi-automatic processes. Karma is designed for multi-source integration and learns from human interactions in one scenario to provide suggestions in the *next ones*. IncMap, on the other side, adjusts its suggestions after simple yes/no feedback during *one single* scenario but has no memory between any two scenarios.

For these reasons, a direct experimental comparison between the two systems is not feasible. Instead, we run a separate dedicated experiment for each of them and identify similarities and differences in performance in the following discussion.

With Karma, we ran three experiments, each of which consists of a series of three related scenarios on the same target ontology. This translates to three different source schemata that Karma needs to integrate in a row. As Karma cannot produce any results completely automatically, we conducted this experiment interactively and recorded the number of human interactions needed to complete the mapping for each of the data sources. Figure 5 shows that in all cases the total number of required interactions drops for later data sources over previous ones. The drop in manual class matches and property matches is made possible by type learning. Python transformations remain approximately constant across subsequent data sources as no learning support and suggestions are available for these transformations.

Due to the manual input, mappings resulting from Karma's semi-automatic process are generally of high quality and did mostly reach scores close to 1.0 (cf. Table 10).

For IncMap, we ran a series of regular single-scenario tests, but in an incremental, semi-automatic setup [36]. That is, for each of the scenarios, we simulated human feedback in the form of choosing a suggestion from shortlists of three suggestions, each. To simulate this kind of feedback we simply used the benchmark as an oracle to identify the best pick. We observed how the score achieved by IncMap's mappings changes after a number of iterations, i.e., we report a score at k human interactions [35].

Table 11 shows those the scores for three conference domain scenarios, before feedback (@0), and after 6, 12, or 24 interactions, respectively. It is clearly visible that scores increase with ongoing feedback. From the first few rounds of feedback, the system profits most. After that, gains are moderate.

Note that these changes in score are based on feedback during several iterations on the same scenarios. It would be most interesting to see an evaluation of a system that combines the approaches of Karma and IncMap. From the results available from these two sys-

Table 8

Score break-down for queries that test n:1 matches in restructured conference domain scenarios. 1:1 and n:1 stands for queries involving 1:1 or n:1 mappings among classes and tables, respectively.

Scenario	B.OX		IncM.		ontop		MIRR.		COMA		D2RQ	
	1:1	n:1	1:1	n:1	1:1	n:1	1:1	n:1	1:1	n:1	1:1	n:1
CMT	**0.86**	0.00	0.79	0.00	0.57	0.00	0.00	0.00	0.58	0.00	0.57	0.00
Conference	0.78	0.00	**0.89**	0.00	0.56	0.00	0.00	0.00	0.56	0.00	0.67	0.00
SIGKDD	**1.00**	0.00	0.86	0.00	0.86	0.00	0.00	0.00	0.86	0.00	0.86	0.00

Table 9

Score break-down for queries that require 1:n class matches on the Oil & Gas atomic tests scenario.

Scenario	B.OX			IncM.			ontop			MIRR.			COMA			D2RQ		
	1:1	1:2	1:3	1:1	1:2	1:3	1:1	1:2	1:3	1:1	1:2	1:3	1:1	1:2	1:3	1:1	1:2	1:3
Oil & Gas Atomic	0.17	**0.11**	**0.07**	**0.20**	0.01	0.03	0.10	0.09	**0.07**	0.00	0.00	0.00	0.03	0.00	0.00	0.11	0.00	**0.07**

(a) Target Ontology CMT   (b) Target Ontology Conference   (c) Target Ontology SIGKDD
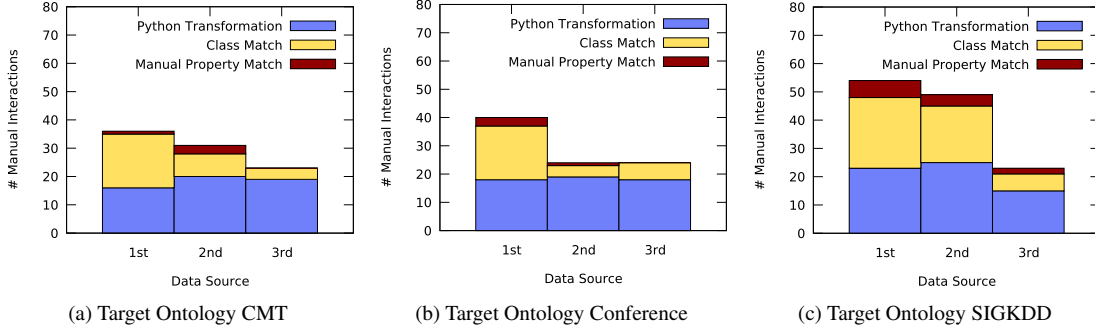
Figure 5. Karma multi-source integration counting human interactions

Table 10

Semi-automatic Karma mappings: generally very high scores thanks to human input.

Series	1st	2nd	3rd
To CMT	0.97	0.85	0.99
To Conference	0.90	1.00	1.00
To SIGKDD	1.00	0.99	1.00

Table 11

Impact of incremental mapping: scores for IncMap after *k* interactions in adjusted naming scenarios.

Scenario	@0	@6	@12	@24
CMT	0.45	0.73	0.92	0.96
Conference	0.53	0.61	0.68	0.77
SIGKDD	0.76	0.85	1.00	1.00

tems so far, it becomes clear that either approach has its own benefits. A direct comparison is not possible, though, as both follow a fairly different kind of process (multi-source vs. single-source) and also request differ-

ent forms of human input (e.g., Python transformations in Karma).

7. Related Work

Mappings between ontologies are usually evaluated only on the basis of their underlying correspondences (usually referred to as ontology *alignments*). The Ontology Alignment Evaluation Initiative (OAEI) [52,23,4] provides tests and benchmarks of those alignments that can be considered a de-facto standard. Mappings between relational databases are typically not evaluated by a common benchmark. Instead, authors typically compare their tools to one or more of the industry standard systems (e.g., [22,11]) in a scenario of their own choice. A novel TPC benchmark [41] was recently created to close this gap. However, no results are reported so far on the TPC-DI website. To the best of our knowledge, no benchmark to measure specifically the quality of inter-model relational-to-ontology mappings was available before the original release of *RODI* [37].

Similarly, evaluations of relational-to-ontology mapping generating systems were based on one or several data sets deemed appropriate by the authors and are therefore not comparable. In one of the most comprehensive evaluations so far, QODI [53] was evaluated on several real-world data sets, though some of the reference mappings were rather simple. IncMap [38] was first evaluated on a choice of real-world mapping problems based on data from two different domains. Such domain-specific mapping problems could be easily integrated in our benchmark through our extension mechanism.

A number of papers discuss different quality aspects of relational-to-ontology mapping generation in a more general way. Console and Lenzerini have devised a series of theoretical OBDA data quality checks w.r.t. consistency [5]. As such, these could also be used to judge mapping quality to a certain degree. However, the focus of this work is clearly different. Also, the approach is agnostic of actual requirements and expectations and only considers consistency of data in itself. A more multi-dimensional approach has been proposed by Westphal et al. [54]. Their proposals do not include a single, globally comparable scoring measure, but rather a collection of different measures, which could be sampled and combined as suitable or applicable in different scenarios. Tarasowa et al. propose a similarly generic approach to quality measurement on relational-to-ontology mappings [51]. Dimou et al. [7] have proposed unit testing as a generic and domain-independent quality measure for relational-to-ontology mappings. Impraliou et al. [29] present a benchmark composed by a series of synthetic queries to measure the correctness and completeness of relational-to-ontology query rewriting. The presence of complete and correct mappings is a prerequisite to their approach. Mora and Corcho discuss issues and possible solutions to benchmark the query rewriting step in OBDA systems [34]. Mappings are supposed to be given as immutable input. The NPD benchmark [32] measures performance of OBDA query evaluation. Neither of these papers, however, address the issue of systematically measuring mapping quality.

A comprehensive overview of relational-to-ontology efforts, including related approaches of automatic mapping generation, can be found in the two surveys [44,49].

8. Conclusion

We have presented a novel benchmark suite *RODI* that allows to test the quality of system-generated relational-to-ontology mappings. The prime application area of *RODI* is ontology-based data integration. *RODI* tests a wide range of data mapping challenges that are specific to relational-to-ontology mappings, and which we identified in this paper.

Using *RODI* we have conducted a thorough evaluation of seven prominent relational-to-ontology mapping generation systems from different research groups. We have identified strengths and weaknesses for each of the systems and in some cases could even point to specific erroneous behavior. We have communicated our observations to the authors of BootOX, IncMap, MIRROR, D2RQ and -ontop- and most of them already used our feedback to improve their systems and the quality of the computed mappings. Overall, the systems demonstrate that they can cope well with relatively simple mapping challenges. However, all tested tools perform poorly on most of the more advanced challenges that come close to actual real-world problems. Thus, further research is needed to address these challenges.

Future work includes repeated evaluations of a growing number of relational-to-ontology mapping generation systems. It would be particularly interesting to evaluate semi-automatic tools in a more comprehensive way, and to directly compare different tools under identical settings. Additionally, we expect several of the tested systems to address issues pointed by our evaluation with *RODI*. Another avenue of future work includes the extension of the benchmark suite, e.g., by adding scenarios from other application domains relevant for ontology-based data integration.

References

[1] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[2] Bernardo Cuenca Grau et al. OWL 2: The Next Step for OWL. *J. Web Sem.*, 6(4), 2008.

[3] Christian Bizer and Andy Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In *3rd International Semantic Web Conference (ISWC2). Poster track.*, 2004.

[4] Michelle Cheatham et al. Results of the Ontology Alignment Evaluation Initiative 2015. In *Proceedings of the 10th International Workshop on Ontology Matching*, pages 60–115, 2015.

[5] Marco Console and Maurizio Lenzerini. Data Quality in Ontology-Based Data Access: The Case of Consistency. In *AAAI*, 2014.

[6] Luciano F. de Medeiros, Freddy Priyatna, and Oscar Corcho. MIRROR: Automatic R2RML Mapping Generation from Relational Databases. In *ICWE*, 2015.

[7] Anastasia Dimou, Dimitris Kontokostas, Markus Freudenberg, Ruben Verborgh, Jens Lehmann, Erik Mannens, Sebastian Hellmann, and Rik Van de Walle. Assessing and Refining Mappingsto RDF to Improve Dataset Quality. In *ISWC*, 2015.

[8] Xin Luna Dong and Divesh Srivastava. Big Data Integration. *PVLDB*, 6(11):1188–1189, 2013.

[9] Christoph Pinkel et al. How to Best Find a Partner? An Evaluation of Editing Approaches to Construct R2RML Mappings. In *ESWC*, 2014.

[10] Craig A. Knoblock et al. Semi-Automatically Mapping Structured Sources into the Semantic Web. In *ESWC*, 2012.

[11] David Aumueller et al. Schema and Ontology Matching with COMA++. In *SIGMOD*, 2005.

[12] Diego Calvanese et al. Ontop: Answering SPARQL Queries over Relational Databases. *Semantic Web Journal*, 2016.

[13] Ernesto Jiménez-Ruiz et al. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *ECAI*, 2012.

[14] Ernesto Jiménez-Ruiz et al. BOOTOX: Practical Mapping of RDBs to OWL 2. In *ISWC*, 2015.

[15] Evgeny Kharlamov et al. Optique 1.0: Semantic Access to Big Data – The Case of Norwegian Petroleum Directorate's FactPages. In *ISWC (Posters & Demos)*, 2013.

[16] Evgeny Kharlamov et al. How Semantic Technologies Can Enhance Data Access at Siemens Energy. In *ISWC*, 2014.

[17] Evgeny Kharlamov et al. Ontology Based Access to Exploration Data at Statoil. In *ISWC*, 2015.

[18] Martin Giese et al. Optique – Zooming In on Big Data Access. *IEEE Computer*, 48(3), 2015.

[19] Michael Bada et al. A Short Study on the Success of the Gene Ontology. *J. Web Sem.*, 1(2), 2004.

[20] Ondrej Svab et al. OntoFarm: Towards an Experimental Collection of Parallel Ontologies. In *ISWC (Posters & Demos)*, 2005.

[21] Pieter Heyvaert et al. Towards Approaches for Generating RDF Mapping Definitions. In *ISWC (Posters & Demos)*, 2015.

[22] Ronald Fagin et al. Clio: Schema Mapping Creation and Data Exchange. In *Conceptual Modeling: Foundations and Applications*. Springer, 2009.

[23] Zlatan Dragisic et al. Results of the Ontology Alignment Evaluation Initiative 2014. In *OM*, 2014.

[24] Sean M. Falconer and Natalya Fridman Noy. Interactive Techniques to Support Ontology Matching. In *Schema Matching and Mapping*. Springer, 2011.

[25] Fred Freitas and Stefan Schulz. Survey of current terminologies and ontologies in biology and medicine. *RECIIS – Elect. J. Commun. Inf. Innov. Health*, 3:7–18, 2009.

[26] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems – The Complete Book*. Prentice Hall, 2nd edition, 2008.

[27] Thomas Hornung and Wolfgang May. Experiences from a TBox Reasoning Application: Deriving a Relational Model by OWL Schema Analysis. In *OWLED*, 2013.

[28] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1), 2011.

[29] Martha Impraliou, Giorgos Stoilos, and Bernardo Cuenca Grau. Benchmarking Ontology-based Query Rewriting Systems. In *AAAI*, 2013.

[30] Jennie Duggan et al. The BigDAWG Polystore System. *SIGMOD Record*, 44(2), 2015.

[31] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-Based and Scalable Ontology Matching. In *ISWC*, 2011.

[32] Davide Lanti, Martin Rezk, Mindaugas Slusnys, Guohui Xiao, , and Diego Calvanese. The NPD Benchmark for OBDA Systems. In *SSWS*, 2014.

[33] Wolfgang May. Information Extraction and Integration with FLORID: The MONDIAL Case Study. Technical report, Universität Freiburg, Institut für Informatik, 1999.

[34] Jose Mora and Oscar Corcho. Towards a Systematic Benchmarking of Ontology-Based Query Rewriting Systems. In *ISWC*, 2014.

[35] Heiko Paulheim, Sven Hertling, and Dominique Ritze. Towards Evaluating Interactive Ontology Matching Tools. In *ESWC*, 2013.

[36] Christoph Pinkel. Interactive Pay as You Go Relational-to-Ontology Mapping. In *ISWC, Part II*, 2013.

[37] Christoph Pinkel, Carsten Binnig, Ernesto Jiménez-Ruiz, Wolfgang May, Dominique Ritze, Martin G. Skjæveland, Alessandro Solimando, and Evgeny Kharlamov. RODI: A benchmark for automatic mapping generation in relational-to-ontology data integration. In *12th European Semantic Web Conference (ESWC)*, pages 21–37, 2015.

[38] Christoph Pinkel, Carsten Binnig, Evgeny Kharlamov, and Peter Haase. IncMap: Pay-as-you-go Matching of Relational Schemata to OWL Ontologies. In *OM*, 2013.

[39] Christoph Pinkel, Carsten Binnig, Evgeny Kharlamov, and Peter Haase. Pay as you go Matching of Relational Schemata to OWL Ontologies with IncMap. In *ISWC (Posters & Demos)*, 2013.

[40] Christoph Pinkel and Ernesto Jiménez-Ruiz. RODI: Technical User Manual. `http://www.cs.ox.ac.uk/isg/tools/RODI/manual.pdf`.

[41] Meikel Poess, Tilmann Rabl, and Brian Caufield. TPC-DI: the first industry benchmark for data integration. *PVLDB*, 7(13):1367–1378, 2014.

[42] Freddy Priyatna, Oscar Corcho, and Juan Sequeda. Formalisation and Experiences of R2RML-based SPARQL to SQL Query Translation Using Morph. In *WWW*, 2014.

[43] Mariano Rodriguez-Muro and Martín Rezk. Efficient SPARQL-to-SQL with R2RML mappings. *J. Web Sem.*, 33, 2015.

[44] Juan Sequeda, Syed Hamid Tirmizi, Óscar Corcho, and Daniel P. Miranker. Survey of Directly Mapping SQL Databases to the Semantic Web. *Knowledge Eng. Review*, 26(4), 2011.

[45] Juan F. Sequeda and Daniel Miranker. Ultrawrap Mapper: A Semi-Automatic Relational-Database-to-RDF (RDB2RDF) Mapping Tool. In *ISWC (Posters & Demos)*, 2015.

[46] Pavel Shvaiko and Jérôme Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1), 2013.

[47] Martin G. Skjæveland, Espen H. Lian, and Ian Horrocks. Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data. In *ISWC*, 2013.

[48] A. Solimando, Ernesto Jiménez-Ruiz, and G. Guerrini. Detecting and Correcting Conservativity Principle Violations in Ontology-to-Ontology Mappings. In *ISWC*, 2014.

[49] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing Relational Databases into the Semantic Web: A Survey. *Semantic Web*, 3(2), 2012.

[50] Mohsen Taheriyan, Craig Knoblock, Pedro Szekely, and José Luis Ambite. Learning the Semantics of Structured Data Sources. *J. of Web Semantics*, 2016.

[51] Darya Tarasowa, Christoph Lange, and Sören Auer. Measuring the Quality of Relational-to-RDF Mappings. In *KESW*, 2015.

[52] The Ontology Alignment Evaluation Initiative (OAEI). `http://oaei.ontologymatching.org`.

[53] Aibo Tian, Juan F. Sequeda, and Daniel P. Miranker. QODI: Query as Context in Automatic Data Integration. In *ISWC*, 2013.

[54] Patrick Westphal, Claus Stadler, and Jens Lehmann. Quality Assurance of RDB2RDF Mappings. Technical report, University of Leipzig, 2014.

[55] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality Assessment for Linked Data: A Survey. *Sem. Web J.*, 7(1), 2015.